SIEMENS

SIMATIC

WinCC Advanced V13.0 SP1

System Manual

Online help printout

12/2014

System overview of STEP 7 and WinCC 1

What's new in WinCC Advanced? 2

What's new in STEP 7 Basic? 3

Readme 4

Installation 5

Migrating projects and programs 6

First steps 7

Introduction to the TIA Portal 8

Editing projects 9

Editing devices and networks 10

Programming the PLC 11

Visualize processes 12

Using technology functions 13

Using online and diagnostics functions 14

Using Team Engineering 15

Hardware documentation 16

Legal information

Warning notice system

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

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

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

Caution
indicates that minor personal injury can result if proper precautions are not taken.

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

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

Qualified Personnel

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

Proper use of Siemens products

Note the following:

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 06/2015 Subject to change

Copyright © Siemens AG 2014.
All rights reserved

Table of contents

1 System overview of STEP 7 and WinCC....33

1.1 Scaling of STEP 7 and WinCC in the TIA Portal....33

1.2 Options for STEP 7 Engineering System....34

1.3 Options for WinCC Engineering and Runtime systems....34

2 What's new in WinCC Advanced?....37

2.1 What's new in WinCC V13 SP1?....37

3 What's new in STEP 7 Basic?....39

3.1 What's new in STEP 7 Basic?....39

4 Readme....43

4.1 Notes on the TIA Portal....43
4.1.1 General notes....43
4.1.2 Notes on libraries....45
4.1.3 Notes on memory cards....45
4.1.4 Notes on the hardware configuration....46
4.1.5 Notes on instructions....47
4.1.6 Notes on online and diagnostics....47

4.2 WinCC Comfort/Advanced....48
4.2.1 Security information....48
4.2.2 News....53
4.2.3 Notes on use....53
4.2.4 Migration....56
4.2.5 Engineering System....60
4.2.5.1 Screens and Screen Objects....60
4.2.5.2 Tags and connections....66
4.2.5.3 Alarm system and alarm displays....68
4.2.5.4 System functions and scripts....69
4.2.5.5 Reports....73
4.2.5.6 Recipes....73
4.2.5.7 User administration....74
4.2.5.8 Communication....75
4.2.6 System-wide functions....78
4.2.7 Compiling and loading....78
4.2.8 Runtime....82
4.2.8.1 Notes on operation in Runtime....82
4.2.8.2 Notes on operation of panels in Runtime....84
4.2.8.3 Notes on operation of Runtime Advanced....85
4.2.9 HMI devices....86
4.2.9.1 Notes on HMI devices....86

4.3 STEP 7 Basic....89
4.3.1 Security information....89
4.3.2 Notes on use....91

4.3.3 Editing devices and networks....92
4.3.3.1 General information on devices and networks....92
4.3.3.2 Use of modules on the S7-1200....92
4.3.3.3 Replacing ET 200S positioning modules....93
4.3.3.4 CP 343-2 on SIMATIC S7 Embedded Controller EC31-RTX....94
4.3.3.5 F-CM AS-i Safety ST for ET 200SP....94
4.3.3.6 S7 routing via IE/PB Link....94
4.3.3.7 Notes on online and diagnostics....95
4.3.3.8 Network components....95
4.3.4 Programming a PLC....98
4.3.4.1 General notes on PLC programming....98
4.3.4.2 Compatibility of PLC programs from V12 SP1 or V13....100
4.3.4.3 Instructions....103
4.3.4.4 Testing the user program....106
4.3.5 Inter Project Engineering (IPE)....107
4.3.5.1 Notes on IPE....107
4.3.6 Technological functions....108
4.3.6.1 Notes on technological functions....108

5 Installation....111

5.1 Notes on the installation....111

5.2 System requirements for installation....112
5.2.1 Notes on licenses....112
5.2.2 Notes on the system requirements....113
5.2.3 System requirements STEP 7 Basic....114
5.2.3.1 Licensing of STEP 7....114
5.2.3.2 Handling licenses and license keys....116
5.2.3.3 Software and hardware requirements STEP 7....118
5.2.4 System requirement for WinCC Advanced....121
5.2.4.1 Software and hardware requirements....121
5.2.4.2 Parallel installation....125
5.2.4.3 Add-ons....126
5.2.4.4 Licenses and Powerpacks....128

5.3 Installation log....136

5.4 Starting installation....137

5.5 Checking availability of updates and support packages and installing them....139

5.6 Displaying the installed software....142

5.7 Modifying or updating installed products....143

5.8 Repairing installed products....144

5.9 Starting to uninstall....146

5.10 Installing and uninstalling the migration tool....148
5.10.1 System requirements....148
5.10.2 Installing the migration tool....148
5.10.3 Uninstalling the migration tool....149

6 Migrating projects and programs....151

6.1 Overview of Migration Options....151

6.2 Migrating projects to a TIA Portal project....152
6.2.1 Migration of projects with the TIA Portal....152
6.2.2 Check migration readiness of hardware components....154
6.2.3 Preparing projects with the migration tool....155
6.2.3.1 Migrating projects with the migration tool....155
6.2.3.2 Calling the migration tool....157
6.2.3.3 Creating a migration file....157
6.2.4 Migrating projects....158
6.2.5 Displaying the history of the migration....160
6.2.6 Display migration log....160
6.2.7 Migrating WinCC flexible projects....161
6.2.7.1 Principles (WinCC flexible)....161
6.2.7.2 Migrating engineering data (WinCC flexible)....167
6.2.7.3 Migrating runtime data (WinCC flexible)....188
6.2.7.4 Migrating integrated projects (WinCC flexible)....191
6.2.7.5 Reference (WinCC flexible)....195
6.2.8 Migrating integrated projects....209
6.2.8.1 Migrating an integrated project....209
6.2.8.2 Post-editing integrated projects....211
6.2.8.3 Converting unspecified CPUs into specified CPUs....213
6.2.8.4 Creating an integrated HMI connection....214
6.2.8.5 Re-linking HMI tags....216
6.2.8.6 Deleting an unspecified connection....217

6.3 Migrating S7-1200 to firmware as of V4....217
6.3.1 Basic information on upgrading to V4....217
6.3.2 Migrating to V4....220
6.3.3 Special considerations after migrating to V4....221

6.4 Programming recommendations....224
6.4.1 The new S7-1500 CPU functions at a glance....224
6.4.2 Flexibly using enable output ENO....227
6.4.3 Querying and fixing errors in the program code....228
6.4.4 Using MOVE instructions in STL....234
6.4.5 Using IEC timers and counters....236
6.4.6 Using ARRAY data blocks....239
6.4.7 Reliable addressing....249
6.4.7.1 Symbolic addressing....249
6.4.7.2 Addressing with Slice access....250
6.4.7.3 Indirect addressing of ARRAY elements....251
6.4.7.4 Addressing tags indirectly....254
6.4.8 Handling specific data types....260
6.4.8.1 Using the VARIANT data type....260
6.4.8.2 Using the DB_ANY data type....275
6.4.8.3 Using PLC data types (UDT)....286
6.4.8.4 Calculating with floating-point numbers (REAL and LREAL) in SCL....290
6.4.8.5 Calculating with constants in SCL....295

7 First steps....299

7.1 Getting Started Documentation....299

8 Introduction to the TIA Portal....301

8.1 User interface and operation....301

8.1.1 Starting, setting up, and exiting the TIA Portal....301
8.1.1.1 Starting and exiting the TIA Portal....301
8.1.1.2 Overview of the program settings....302
8.1.1.3 Overview of the script and text editor settings....304
8.1.1.4 Overview of the print settings....305
8.1.1.5 Overview of the online and diagnostic settings....305
8.1.1.6 Changing the settings....306
8.1.2 Layout of the user interface....307
8.1.2.1 Views....307
8.1.2.2 Portal view....307
8.1.2.3 Project view....309
8.1.2.4 Library view....311
8.1.2.5 Project tree....312
8.1.2.6 Work area....316
8.1.2.7 Inspector window....324
8.1.2.8 Task cards....326
8.1.2.9 Reference projects....328
8.1.2.10 Details view....330
8.1.2.11 Overview window....331
8.1.2.12 User interface layout....335
8.1.3 Keyboard operation in the TIA Portal....339
8.1.3.1 Operation of the TIA Portal with the keyboard....339
8.1.3.2 Displaying an overview of all keyboard shortcuts....339
8.1.3.3 Basic functions of the TIA Portal....339
8.1.3.4 Using project-related functions....341
8.1.3.5 Arranging windows....341
8.1.3.6 Navigating through the program interface....342
8.1.3.7 Customizing editors....343
8.1.3.8 Editing objects....345
8.1.3.9 Text editing....346
8.1.3.10 Editing tables....347
8.1.3.11 Using online functions....348
8.1.3.12 Using the on-screen keyboard....349
8.1.4 Special features specific to the operating system....349
8.1.4.1 Influence of user rights....349
8.1.4.2 Expanding user rights....350

8.2 Help on the information system....351
8.2.1 General remarks on the information system....351
8.2.2 Opening the Help system....355
8.2.3 Searching the Help system for keywords....355
8.2.4 Full-text searches....356
8.2.5 Using favorites....356
8.2.6 Printing help topics....357
8.2.7 Configuring the display of tooltips and tooltip cascades....358
8.2.8 Safety Guidelines....359
8.2.9 Assembling customized documentation....360

8.3 Providing user-defined documentation....361
8.3.1 Using user-defined documentation....361
8.3.2 Specifying settings in the TIA Portal....364
8.3.3 Specifying settings with an XML file....365
8.3.4 Creating a homepage....366

8.3.5 Conventions for the creation....367
8.3.6 Calling user-defined documentation....370
8.3.7 Displaying the call log....371
8.3.8 Creating user-defined documentation....371

9 Editing projects....375

9.1 The basics of projects....375
9.2 Using logs....376
9.3 Creating and managing projects....376
9.3.1 Creating a new project....376
9.3.2 Compatibility of projects....377
9.3.3 Opening projects....379
9.3.4 Upgrading projects....380
9.3.5 Displaying properties of the project....382
9.3.6 Saving projects....383
9.3.7 Closing projects....384
9.3.8 Removing projects....384
9.3.9 Deleting projects....385
9.3.10 Archiving and retrieving projects....386
9.3.10.1 Working with project archives....386
9.3.10.2 Creating compressed project archive....387
9.3.10.3 Minimizing project....388
9.3.10.4 Retrieving compressed project....389
9.4 Using reference projects....390
9.4.1 Basics of reference projects....390
9.4.2 Opening and closing a reference project....390
9.4.3 Comparing reference projects....391
9.5 Editing project data....392
9.5.1 Compiling and loading project data....392
9.5.1.1 Compiling project data....392
9.5.1.2 Loading project data....394
9.5.2 Comparing project data....402
9.5.2.1 Basics of project data comparison....402
9.5.2.2 Carrying out an online/offline comparison....403
9.5.2.3 Carrying out offline/offline comparisons....404
9.5.2.4 Using the compare editor 405
9.5.3 Protecting project data....417
9.5.3.1 Protection concept for project data....417
9.5.3.2 Revoking access rights for devices....418
9.5.4 Printing project contents....419
9.5.4.1 Printing project documentation....419
9.5.4.2 Printing module labels....438
9.6 Undoing and redoing actions....446
9.6.1 Basics of undoing and redoing actions....446
9.6.2 Undoing an action....447
9.6.3 Redoing an action....448
9.7 Find and replace....449
9.7.1 Basics of searching....449
9.7.2 Searching and replacing in the editor....450

9.7.2.1 Basics for search in open editors....450
9.7.2.2 Overview of the "Find and replace" palette....451
9.7.2.3 Searching and replacing in the editor....452
9.8 Working with multi-language projects....454
9.8.1 Project text basics....454
9.8.2 Select project languages....456
9.8.3 Setting the editing language....456
9.8.4 Translating all project texts in tabular form....457
9.8.5 Translating text associated with individual objects....458
9.8.6 Translating texts using reference texts....458
9.8.7 Exporting project texts....459
9.8.8 Importing project texts....461
9.8.9 Application examples for multilanguage projects....462
9.9 Working with text lists....463
9.9.1 Text lists....463
9.9.2 Creating user-defined text lists....465
9.9.3 Editing user-defined text lists....466
9.9.4 Editing system-defined text lists....466
9.10 Using memory cards....467
9.10.1 Basics about memory cards....467
9.10.2 Adding a user-defined card reader....468
9.10.3 Accessing memory cards....468
9.10.4 Displaying properties of memory cards....469
9.11 Using libraries....470
9.11.1 Library basics....470
9.11.2 Using the "Libraries" task card....472
9.11.2.1 Overview of the "Libraries" task card....472
9.11.2.2 Using the element view....474
9.11.3 Using the library view....475
9.11.3.1 Overview of the library view....475
9.11.3.2 Opening and closing the library view....477
9.11.4 Using library management....478
9.11.4.1 Overview of the library management....478
9.11.4.2 Opening library management....480
9.11.4.3 Filtering the display of types....481
9.11.4.4 Displaying instances in the project....481
9.11.4.5 Displaying cross references of an instance....482
9.11.4.6 Displaying relations to other library objects....483
9.11.5 Using global libraries....483
9.11.5.1 Creating a global library....483
9.11.5.2 Compatibility of global libraries....484
9.11.5.3 Opening a global library....486
9.11.5.4 Upgrading global libraries....487
9.11.5.5 Displaying properties of global libraries....488
9.11.5.6 Displaying logs of global libraries....489
9.11.5.7 Saving a global library....490
9.11.5.8 Closing a global library....491
9.11.5.9 Deleting a global library....491
9.11.5.10 Archiving and disabling global libraries....492
9.11.5.11 Using global corporate libraries....495

9.11.6 Creating folders in a library....497
9.11.7 Using master copies....497
9.11.7.1 Basics on master copies....497
9.11.7.2 Adding master copies....498
9.11.7.3 Filtering master copies....500
9.11.7.4 Using master copies....500
9.11.8 Using types and their versions....501
9.11.8.1 Basics on types....501
9.11.8.2 State of type versions....503
9.11.8.3 Displaying a released type version....504
9.11.8.4 Displaying properties of a type or version....505
9.11.8.5 Working with types in the project library....506
9.11.8.6 Working with types in global libraries....520
9.11.8.7 Assigning a version....524
9.11.9 Editing library elements....526
9.11.10 Updating a library with the contents of another library....528
9.11.11 Harmonizing names and path structure....530
9.11.12 Clean up library....531
9.11.13 Comparing library elements....532

9.12 Using cross-references....533
9.12.1 Using cross-references....533

9.13 Simulating devices....533
9.13.1 Simulation of devices....533
9.13.2 Starting the simulation....534

10 Editing devices and networks....535

10.1 Configuring devices and networks....535
10.1.1 Hardware and network editor....535
10.1.1.1 Overview of hardware and network editor....535
10.1.1.2 Network view....537
10.1.1.3 Device view....539
10.1.1.4 Topology view....542
10.1.1.5 Overview of settings for hardware configuration....543
10.1.1.6 Printing hardware and network configurations....544
10.1.1.7 Activating the page break preview for printout....546
10.1.1.8 Changing the print options....546
10.1.1.9 Inspector window....547
10.1.1.10 Hardware catalog....548
10.1.1.11 Enabling product support....550
10.1.1.12 Displaying product support for hardware components....551
10.1.1.13 Keyboard operation: Navigation in the editor....552
10.1.1.14 Keyboard operation: Editing objects....553
10.1.2 Configuring devices....554
10.1.2.1 Basics....554
10.1.2.2 Configuring individual devices....564
10.1.2.3 Comparing devices....579
10.1.3 Configure networks....581
10.1.3.1 Networking devices....581
10.1.3.2 Communication via connections....603
10.1.3.3 Displaying and configuring topology....661
10.1.3.4 Industrial Ethernet Security....677

10.1.4 Creating configurations....846
10.1.4.1 Information about the web server....846
10.1.4.2 Things you should know about PROFIBUS DP operating modes....847
10.1.4.3 Configuring automation systems....848
10.1.4.4 S7-1200 CM/CP....896
10.1.4.5 SCALANCE X, W and M....925
10.1.4.6 Configuring PROFIBUS DP....1085
10.1.4.7 Configurations for PROFINET IO....1109
10.1.4.8 Bus coupling with PN/PN coupler....1143
10.1.4.9 Integrating external tools....1144
10.1.4.10 Loading a configuration....1146
10.1.5 Displaying alarms....1157
10.1.5.1 Overview of the alarm display....1157
10.1.5.2 Archive view....1158
10.1.5.3 Layout of the alarms in the archive view....1158
10.1.5.4 Receiving alarms....1158
10.1.5.5 Export archive....1159
10.1.5.6 Clear archive....1159
10.1.5.7 "Active alarms" view....1160
10.1.5.8 Layout of the alarms in the "Active alarms" view....1160
10.1.5.9 Status of the alarms....1160
10.1.5.10 Acknowledging alarms....1161
10.1.5.11 Ignoring alarms....1161
10.1.5.12 Sort table in alarm display....1162
10.1.5.13 Keyboard commands in the alarm display....1162
10.1.6 Additional information on configurations....1163
10.1.6.1 Functional description of S7-1200 CPUs....1163
10.1.6.2 Identification systems....1237
10.1.6.3 Distributed I/O....1244
10.1.6.4 IPv6 configuration....1352

10.2 Device and network diagnostics....1354
10.2.1 Hardware diagnostics....1354
10.2.1.1 Overview of hardware diagnostics....1354
10.2.1.2 Showing non-editable and current values of configurable module properties....1365
10.2.1.3 Showing the current values of dynamic modules properties....1371
10.2.1.4 Checking a module for defects....1375
10.2.1.5 Changing the properties of a module or the programming device/PC....1382
10.2.1.6 Diagnostics in STOP mode....1398
10.2.1.7 Online accesses in the Online and Diagnostics view....1400
10.2.1.8 Checking PROFIBUS DP subnets for faults....1403
10.2.2 Connection diagnostics....1406
10.2.2.1 Overview of connection diagnostics....1406
10.2.2.2 Displaying the connection status using icons....1406
10.2.2.3 Detailed connection diagnostics....1407

11 Programming the PLC....1411

11.1 Creating the user program....1411
11.1.1 Programming basics....1411
11.1.1.1 Operating system and user program....1411
11.1.1.2 Blocks in the user program....1412
11.1.1.3 Block calls....1424
11.1.1.4 Using and addressing operands....1441

11.1.1.5 Program flow control....1473
11.1.2 Declaring PLC tags....1479
11.1.2.1 Overview of PLC tag tables....1479
11.1.2.2 Structure of the PLC tag tables....1480
11.1.2.3 Rules for PLC tags....1481
11.1.2.4 Creating and managing PLC tag tables....1485
11.1.2.5 Declaring PLC tags....1487
11.1.2.6 Grouping PLC tags for inputs and outputs in structures....1490
11.1.2.7 Declaring global constants....1493
11.1.2.8 Editing properties....1496
11.1.2.9 Monitoring of PLC tags....1498
11.1.2.10 Editing PLC tag tables....1499
11.1.3 Creating and managing blocks....1506
11.1.3.1 Creating blocks....1506
11.1.3.2 Specifying block properties....1517
11.1.3.3 Managing blocks....1524
11.1.4 Programming blocks....1529
11.1.4.1 Program editor....1529
11.1.4.2 Programming code blocks....1551
11.1.4.3 Programming data blocks....1703
11.1.4.4 Programming PLC data types....1734
11.1.4.5 Using external source files....1744
11.1.5 Comparing PLC programs....1750
11.1.5.1 Basic information on comparing PLC programs....1750
11.1.5.2 Comparing blocks....1754
11.1.5.3 Comparing PLC tags....1778
11.1.5.4 Comparing PLC data types....1780
11.1.6 Compiling and downloading blocks....1782
11.1.6.1 Compiling blocks....1782
11.1.6.2 Downloading blocks for S7-1200/1500....1786
11.1.7 Protecting blocks....1802
11.1.7.1 Protecting blocks....1802
11.1.7.2 Setting up and removing block copy protection....1804
11.1.7.3 Setting up block know-how protection....1805
11.1.7.4 Opening know-how protected blocks....1806
11.1.7.5 Printing know-how protected blocks....1807
11.1.7.6 Changing a password....1808
11.1.7.7 Removing block know-how protection....1808

11.2 Displaying program information....1809
11.2.1 Overview of available program information....1809
11.2.2 Displaying an assignment list....1810
11.2.2.1 Introduction to the assignment list....1810
11.2.2.2 Layout of the assignment list....1811
11.2.2.3 Symbols in the assignment list....1812
11.2.2.4 Displaying an assignment list....1813
11.2.2.5 Setting the view options for the assignment list....1814
11.2.2.6 Filter options in the assignment list....1815
11.2.2.7 Defining filters for assignment list....1815
11.2.2.8 Filtering an assignment list....1816
11.2.2.9 Defining retentive memory areas for bit memories....1817
11.2.2.10 Enabling the display of retentive bit memories....1818
11.2.3 Displaying the call structure....1818

11.2.3.1 Introduction to the call structure....1818
11.2.3.2 Symbols in the call structure....1820
11.2.3.3 Layout of the call structure....1821
11.2.3.4 Displaying the call structure....1822
11.2.3.5 Setting the view options for the call structure....1822
11.2.3.6 Introducing the consistency check in the call structure....1823
11.2.3.7 Checking block consistency in the call structure....1824
11.2.4 Displaying the dependency structure....1824
11.2.4.1 Introduction to the dependency structure....1824
11.2.4.2 Layout of the dependency structure....1826
11.2.4.3 Symbols in the dependency structure....1826
11.2.4.4 Displaying the dependency structure....1827
11.2.4.5 Setting the view options for the dependency structure....1828
11.2.4.6 Introducing the consistency check in the dependency structure....1828
11.2.4.7 Checking block consistency in the dependency structure....1829
11.2.5 Displaying CPU resources....1830
11.2.5.1 Introducing resources....1830
11.2.5.2 Layout of the "Resources" tab....1832
11.2.5.3 Displaying resources....1833
11.2.5.4 Selecting the maximum load memory available....1834

11.3 Displaying cross-references....1834
11.3.1 General information about cross references....1834
11.3.2 Structure of the cross-reference list....1835
11.3.3 Displaying the cross-reference list....1836
11.3.4 Displaying cross-references in the Inspector window....1838

11.4 Testing the user program....1839
11.4.1 Basics of testing the user program....1839
11.4.2 Testing with program status....1840
11.4.2.1 Introduction to testing with program status....1840
11.4.2.2 Setting the call environment....1841
11.4.2.3 Switching test with program status on/off....1843
11.4.2.4 Editing blocks during the program test....1845
11.4.2.5 Modifying tags in the program status....1845
11.4.2.6 Switching display formats in the program status....1846
11.4.2.7 Examples of program status display....1847
11.4.3 Testing with the watch table....1854
11.4.3.1 Introduction to testing with the watch table....1854
11.4.3.2 Layout of the watch table....1855
11.4.3.3 Basic mode and expanded mode in the watch table....1856
11.4.3.4 Icons in the watch table....1857
11.4.3.5 Creating and editing watch tables....1858
11.4.3.6 Entering tags in the watch table....1860
11.4.3.7 Monitoring tags in the watch table....1868
11.4.3.8 Modifying tags in the watch table....1873
11.4.4 Testing with the force table....1881
11.4.4.1 Introduction for testing with the force table....1881
11.4.4.2 Safety precautions when forcing tags....1883
11.4.4.3 Layout of the force table....1883
11.4.4.4 Basic mode and expanded mode in the force table....1884
11.4.4.5 Icons in the force table....1885
11.4.4.6 Open and edit force table....1886

11.4.4.7 Entering tags in the force table....1887
11.4.4.8 Monitoring tags in the force table....1894
11.4.4.9 Forcing tags in the force table....1898
11.4.4.10 Stop forcing tags....1906

11.5 Data types....1908
11.5.1 Overview of the valid data types....1908
11.5.2 Binary numbers....1912
11.5.2.1 BOOL (bit)....1912
11.5.2.2 Bit strings....1913
11.5.3 Integers....1917
11.5.3.1 SINT (8-bit integers)....1917
11.5.3.2 USINT (8-bit integers)....1918
11.5.3.3 INT (16-bit integers)....1918
11.5.3.4 UINT (16-bit integers)....1919
11.5.3.5 DINT (32-bit integers)....1920
11.5.3.6 UDINT (32-bit integers)....1921
11.5.3.7 LINT (64-bit integers)....1922
11.5.3.8 ULINT (64-bit integers)....1923
11.5.4 Floating-point numbers....1925
11.5.4.1 REAL....1925
11.5.4.2 LREAL....1926
11.5.4.3 Invalid floating-point numbers....1927
11.5.5 Timers....1929
11.5.5.1 S5TIME (duration)....1929
11.5.5.2 TIME (IEC time)....1930
11.5.5.3 LTIME (IEC time)....1931
11.5.6 Date and time....1931
11.5.6.1 DATE....1931
11.5.6.2 TIME_OF_DAY (TOD)....1932
11.5.6.3 LTOD (LTIME_OF_DAY)....1932
11.5.6.4 DATE_AND_TIME (date and time of day)....1933
11.5.6.5 LDT (DATE_AND_LTIME)....1934
11.5.6.6 DTL....1935
11.5.7 Character strings....1936
11.5.7.1 CHAR (character)....1936
11.5.7.2 WCHAR (character)....1937
11.5.7.3 STRING....1937
11.5.7.4 WSTRING....1939
11.5.8 Array....1941
11.5.8.1 Format of array (16-bit limits)....1941
11.5.8.2 Format of array (32-bit limits)....1942
11.5.8.3 Example of a one-dimensional array....1943
11.5.8.4 Example of a multi-dimensional array....1944
11.5.9 Structures....1945
11.5.9.1 STRUCT....1945
11.5.10 Pointer....1946
11.5.10.1 POINTER....1946
11.5.10.2 ANY....1948
11.5.10.3 VARIANT....1951
11.5.11 Parameter types....1953
11.5.11.1 Parameter types....1953
11.5.12 PLC data types....1954

11.5.12.1 PLC data types....1954
11.5.12.2 Example of a PLC data type....1954
11.5.13 System data types....1955
11.5.13.1 System data types....1955
11.5.14 Hardware data types....1957
11.5.14.1 Hardware data types....1957
11.5.15 Data type conversion for S7-1500:....1959
11.5.15.1 Overview of data type conversion....1959
11.5.15.2 Implicit conversion....1961
11.5.15.3 Explicit conversion....2010
11.5.16 Data type conversion for S7-1200:....2091
11.5.16.1 Overview of data type conversion....2091
11.5.16.2 Implicit conversion....2093
11.5.16.3 Explicit conversion....2113
11.5.17 Data type conversion for S7-300, S7-400....2157
11.5.17.1 Overview of data type conversion....2157
11.5.17.2 Implicit conversion....2159
11.5.17.3 Explicit conversion....2171

11.6 Instructions....2193
11.6.1 General parameters of the instructions....2193
11.6.1.1 Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions....2193
11.6.1.2 Evaluating errors with output parameter RET_VAL....2195
11.6.2 Basic instructions....2199
11.6.2.1 LAD....2199
11.6.2.2 FBD....2471
11.6.2.3 SCL....2752
11.6.3 Extended instructions....2971
11.6.3.1 Date and time-of-day....2971
11.6.3.2 String + Char....3001
11.6.3.3 Process image....3062
11.6.3.4 Distributed I/O....3069
11.6.3.5 PROFIenergy....3152
11.6.3.6 Module parameter assignment....3206
11.6.3.7 Interrupts....3216
11.6.3.8 Alarms....3246
11.6.3.9 Diagnostics....3262
11.6.3.10 Pulse....3310
11.6.3.11 Recipes and data logging....3312
11.6.3.12 Data block functions....3348
11.6.3.13 Addressing....3358
11.6.4 Technology....3374
11.6.4.1 S7-1200 Motion Control....3374
11.6.4.2 High-speed counters....3439
11.6.4.3 PID Control....3442
11.6.5 Communication....3608
11.6.5.1 S7 communication....3608
11.6.5.2 Open User Communication....3625
11.6.5.3 Web server....3721
11.6.5.4 Communications processor....3722
11.6.5.5 TeleService....3854

11.7 Programming examples....3861

11.7.1 LAD programming examples....3861
11.7.1.1 Example of controlling a conveyor belt....3861
11.7.1.2 Example of detecting the direction of a conveyor belt....3863
11.7.1.3 Example of detecting the fill level of a storage area....3864
11.7.1.4 Example of calculating an equation....3867
11.7.1.5 Example of controlling room temperature....3868
11.7.2 FBD programming examples....3871
11.7.2.1 Example of controlling a conveyor belt....3871
11.7.2.2 Example of detecting the direction of a conveyor belt....3872
11.7.2.3 Example of detecting the fill level of a storage area....3874
11.7.2.4 Example of calculating an equation....3877
11.7.2.5 Example of controlling room temperature....3879
11.7.3 SCL programming examples....3881
11.7.3.1 Example of controlling a conveyor belt....3881
11.7.3.2 Example of detecting the direction of a conveyor belt....3882
11.7.3.3 Example of detecting the fill level of a storage area....3884

12 Visualize processes....3889

12.1 Creating screens....3889
12.1.1 Basics....3889
12.1.1.1 Screen basics....3889
12.1.1.2 Device-dependent functional scope of screens....3891
12.1.1.3 Elements and basic settings....3892
12.1.1.4 Working with screens....3895
12.1.1.5 Working with templates....3900
12.1.1.6 Working with pop-up screens....3907
12.1.1.7 Working with slide-in screens....3910
12.1.1.8 Working with styles....3916
12.1.2 Working with objects....3932
12.1.2.1 Overview of objects....3932
12.1.2.2 Overview of objects....3935
12.1.2.3 Options for editing objects....3939
12.1.2.4 Inserting an object....3939
12.1.2.5 Deleting an object....3941
12.1.2.6 Positioning an object....3942
12.1.2.7 Resizing objects....3943
12.1.2.8 Aligning objects....3946
12.1.2.9 Moving an object forwards or backwards....3947
12.1.2.10 Showing objects outside the screen area....3948
12.1.2.11 Rotating objects....3949
12.1.2.12 Flipping objects....3951
12.1.2.13 Flashing....3952
12.1.2.14 Designing an object....3952
12.1.2.15 Designing the fill pattern....3953
12.1.2.16 Formatting text in an object....3955
12.1.2.17 Formatting graphics in an object....3956
12.1.2.18 Connecting tags and text lists in the text....3958
12.1.2.19 Designing a border....3959
12.1.2.20 Designing table-based objects....3961
12.1.2.21 Defining color gradients....3962
12.1.2.22 Using predefined styles....3963
12.1.2.23 Inserting multiple objects of the same type (stamping)....3964

12.1.2.24 Selecting multiple objects 3966
12.1.2.25 Repositioning and resizing multiple objects....3968
12.1.2.26 Managing own controls....3968
12.1.2.27 External graphics 3971
12.1.2.28 Managing external graphics....3973
12.1.2.29 Storing an external image in the graphics library....3974
12.1.2.30 Working with object groups....3977
12.1.2.31 Configuring the keyboard access....3984
12.1.2.32 Examples....3987
12.1.3 Working with text lists and graphics lists....3990
12.1.3.1 Working with text lists....3990
12.1.3.2 Working with graphics lists....4001
12.1.4 Dynamizing screens....4012
12.1.4.1 Basics on dynamizing screens....4012
12.1.4.2 Basics on dynamizing screens....4012
12.1.4.3 Dynamization in the inspector window....4013
12.1.4.4 Dynamization in the inspector window....4015
12.1.4.5 Dynamization with animations....4018
12.1.4.6 Dynamization in the property list....4030
12.1.4.7 Dynamizing with system functions....4034
12.1.5 Working with function keys....4036
12.1.5.1 Working with function keys 4036
12.1.5.2 Assigning function keys globally....4039
12.1.5.3 Local assignment of function keys....4040
12.1.5.4 Assigning a function to a function key....4041
12.1.5.5 Assigning operator authorization for a function key....4043
12.1.5.6 Assigning a graphic to a function key....4045
12.1.5.7 Configuring LED tags....4047
12.1.5.8 Example: Using function keys for screen navigation....4049
12.1.6 Working with layers....4051
12.1.6.1 Basics on working with layers....4051
12.1.6.2 Moving objects on layers....4052
12.1.6.3 Setting the active layer....4053
12.1.6.4 Showing and hiding layers....4054
12.1.6.5 Renaming layers....4055
12.1.7 Working with faceplates....4057
12.1.7.1 Basics on faceplates....4057
12.1.7.2 Device dependency of faceplates....4058
12.1.7.3 Faceplate editor....4059
12.1.7.4 Creating and managing faceplates....4062
12.1.7.5 Dynamizing faceplates....4079
12.1.7.6 Examples of faceplates....4081
12.1.8 Display and operating elements....4088
12.1.8.1 Device dependency of the objects....4088
12.1.8.2 Objects....4097
12.1.9 Configuring screen navigation....4194
12.1.9.1 Basics on screen navigation....4194
12.1.9.2 Assigning screen change to button....4195
12.1.9.3 Assigning screen change to function key....4196

12.2 Working with Tags....4197
12.2.1 Basics....4197
12.2.1.1 Basics of tags....4197

12.2.1.2 Overview of HMI tag tables....4198
12.2.1.3 External tags....4200
12.2.1.4 Addressing external tags....4202
12.2.1.5 Internal Tags....4205
12.2.1.6 User-defined PLC data types (UDT)....4205
12.2.2 Working with Tags....4207
12.2.2.1 Creating tags....4207
12.2.2.2 Editing tags....4211
12.2.2.3 Configuring Tags....4218
12.2.3 Working with arrays....4236
12.2.3.1 Basics on arrays....4236
12.2.3.2 Creating array tags....4238
12.2.3.3 Examples of arrays....4239
12.2.4 Working with user data types....4239
12.2.4.1 Basics on user data types....4239
12.2.4.2 Creating a user data type....4241
12.2.4.3 Creating user data type elements....4243
12.2.4.4 Managing versions of user data types....4244
12.2.4.5 Creating tags with a user data type data type....4245
12.2.5 Working with cycles....4247
12.2.5.1 Cycle basics....4247
12.2.5.2 Defining cycles....4248
12.2.6 Logging tags....4249
12.2.6.1 Basic principles for data logging....4249
12.2.6.2 Data logging in Runtime Advanced and Panels....4250
12.2.7 Displaying Tags....4268
12.2.7.1 Displaying tags with Basic Panels....4268
12.2.7.2 Displaying tags with Runtime Advanced and Panels....4270

12.3 Working with alarms....4278
12.3.1 Basics....4278
12.3.1.1 Alarm Logging in WinCC....4278
12.3.1.2 Alarm Logging in WinCC....4279
12.3.1.3 Alarm Procedures....4280
12.3.1.4 Alarm states....4285
12.3.1.5 Alarm classes....4286
12.3.1.6 Acknowledgement....4290
12.3.1.7 Alarm groups....4292
12.3.1.8 Alarm number....4293
12.3.2 Working with Alarms 4293
12.3.2.1 Alarm components and properties....4293
12.3.2.2 Configuring Alarms....4295
12.3.2.3 Configuring the Outputting of Alarms....4315
12.3.2.4 Acknowledging alarms....4333
12.3.2.5 Configuring alarm buffer overflow....4336
12.3.3 Logging Alarms....4337
12.3.3.1 Alarm Logging Basics....4337
12.3.3.2 Creating an alarm log....4339
12.3.3.3 Logging Alarms....4341
12.3.3.4 Configuring an alarm view for logged alarms....4342
12.3.3.5 Direct access to the ODBC log database....4343
12.3.3.6 Setting up the ODBC data source....4343
12.3.3.7 Configuring a checksum for a log....4344

12.3.3.8 Evaluating the checksum of log data....4346
12.3.3.9 Checksums for logs in regulated projects....4348
12.3.3.10 Log response to language switching in runtime....4348
12.3.3.11 Managing logging behavior when Runtime starts....4349
12.3.3.12 Controlling the Logging in relation to the Fill Level....4350
12.3.4 Using Alarms in Runtime....4351
12.3.4.1 Alarms in Runtime....4351
12.3.4.2 Alarms in Runtime....4353
12.3.4.3 Using the Alarm Window or Alarm View....4355
12.3.4.4 Using the Simple Alarm Window, Alarm View....4358
12.3.4.5 Using the Alarm Indicator....4361
12.3.4.6 Acknowledging alarms....4362
12.3.4.7 Filtering Alarms....4363
12.3.5 Reference....4364
12.3.5.1 System functions for alarms....4364
12.3.5.2 Alarm events....4365
12.3.5.3 System functions for logs....4366
12.3.5.4 Structure of *.csv files that contain alarms....4366
12.3.5.5 System events....4368
12.3.6 Configuring system diagnostics....4401
12.3.6.1 System diagnostics basics....4401
12.3.6.2 System diagnostics views....4402
12.3.6.3 Basic Panel basics....4406
12.3.6.4 Configuring system diagnostics objects....4409

12.4 Working with logs....4418
12.4.1 Working with logs for Runtime Advanced and Panels....4418
12.4.1.1 Log Basics....4418
12.4.1.2 Properties of Logs....4418
12.4.1.3 Storage locations of logs....4420

12.5 Working with recipes....4423
12.5.1 Basics....4423
12.5.1.1 Definition and applications....4423
12.5.1.2 Examples for using recipes....4425
12.5.1.3 Recipe structure....4425
12.5.1.4 Display of recipes....4427
12.5.1.5 Transferring recipe data records....4429
12.5.1.6 Configuration of recipes....4432
12.5.1.7 Special features of some devices....4433
12.5.1.8 Synchronization of recipe data records with the PLC....4435
12.5.1.9 "Recipes" editor....4436
12.5.2 Displaying and Editing Recipes in Runtime....4439
12.5.2.1 Recipe screen and recipe view....4439
12.5.2.2 Simple recipe view....4440
12.5.2.3 Configuration options of the simple recipe view....4441
12.5.2.4 Advanced recipe view....4444
12.5.2.5 Configuration options of the advanced recipe view....4445
12.5.2.6 Advanced recipe view (as of V13)....4447
12.5.2.7 Configuration options of the advanced recipe view (V13 or higher)....4448
12.5.2.8 Response of the recipe view in Runtime....4451
12.5.2.9 Basics on the recipe screen....4452
12.5.3 Configuring Recipes....4454

12.5.3.1 General configuration procedure....4454
12.5.3.2 Creating and Editing Recipes....4455
12.5.3.3 Configuring the display of recipes....4463
12.5.3.4 Importing recipes into the configuration and exporting them....4467
12.5.4 Using Recipes in Runtime....4470
12.5.4.1 Simple recipe view....4470
12.5.4.2 Advanced recipe view....4477
12.5.4.3 Exporting and importing recipe data records....4483
12.5.4.4 Reactions to modifications of the recipe structure....4485
12.5.5 Examples....4485
12.5.5.1 Example of creating a recipe....4485
12.5.5.2 Example of configuring a recipe screen....4488
12.5.5.3 Scenario for Entering Recipe Data Records in Runtime....4489
12.5.5.4 Scenario for a manual production sequence....4491
12.5.5.5 Scenario for an Automatic Production Sequence....4492

12.6 Working with reports....4493
12.6.1 Basics....4493
12.6.1.1 Reports....4493
12.6.1.2 Structure of reports....4494
12.6.2 Working with reports....4496
12.6.2.1 Creating reports....4496
12.6.2.2 Printing reports....4501
12.6.2.3 Working with objects....4502
12.6.3 Operation in Runtime....4516
12.6.3.1 Printing reports....4516
12.6.4 Objects in reports....4516
12.6.4.1 Audit report....4516
12.6.4.2 Date/time field....4517
12.6.4.3 I/O field....4518
12.6.4.4 Graphic view....4520
12.6.4.5 Graphic I/O field....4521
12.6.4.6 Alarm report....4521
12.6.4.7 Recipe report....4524
12.6.4.8 Page number....4525
12.6.4.9 Symbolic I/O field....4526
12.6.4.10 Text field....4526

12.7 Configuring user administration....4527
12.7.1 Field of application of the user administration....4527
12.7.2 Form of the user administration....4528
12.7.3 Basics....4529
12.7.3.1 Users....4529
12.7.3.2 Users work area....4531
12.7.3.3 User groups....4531
12.7.3.4 User groups work area....4532
12.7.3.5 Settings for the user administration....4533
12.7.3.6 Settings for the user administration....4534
12.7.4 Building up and structuring a user administration....4536
12.7.4.1 Basics of user administration....4536
12.7.4.2 Administering users for Runtime....4537
12.7.4.3 Managing users on the server....4543
12.7.4.4 Administering users in Runtime....4548

12.7.4.5 Configuring access protection....4557
12.7.5 Reference....4558
12.7.5.1 Objects with access protection....4558
12.7.5.2 Objects with access protection....4558
12.7.5.3 Default user groups and authorizations....4559
12.7.6 Examples....4559
12.7.6.1 Example: Configuring a button with logon dialog box....4559
12.7.6.2 Example: Logging the logon and logoff events....4560
12.7.6.3 Example of a user administration....4561

12.8 Working with system functions and Runtime scripting....4569
12.8.1 Basics....4569
12.8.1.1 Runtime scripting....4569
12.8.1.2 System functions....4570
12.8.1.3 User-defined functions....4572
12.8.2 Working with function lists....4574
12.8.2.1 Basics of the function list....4574
12.8.2.2 Properties of a function list....4575
12.8.2.3 Configuring a function list....4575
12.8.2.4 Editing a function list....4577
12.8.3 Working with user-defined VB functions....4578
12.8.3.1 "Scripts" editor....4578
12.8.3.2 Access to HMI tags....4580
12.8.3.3 Access to objects....4581
12.8.3.4 Calling system functions....4582
12.8.3.5 Calling user-defined VB functions....4583
12.8.3.6 Transfer and return of values in VBS....4584
12.8.3.7 Create customized VB functions....4585
12.8.3.8 Testing the syntax of customized functions....4587
12.8.3.9 Renaming customized VB functions....4588
12.8.3.10 Executing customized VB functions....4589
12.8.3.11 Protecting user-defined functions....4590
12.8.4 Debugging user-defined VB functions....4592
12.8.4.1 Debugging user-defined VB functions....4592
12.8.4.2 Integrating the Debugger....4594
12.8.4.3 Registering Microsoft Script Debugger....4598
12.8.4.4 Starting and stopping the debugger....4599
12.8.5 Runtime behavior in Runtime....4600
12.8.5.1 Executing a function list in Runtime....4600
12.8.5.2 Executing user-defined functions in Runtime....4601
12.8.5.3 Processing sequence for user-defined functions and system functions....4602
12.8.5.4 Making object properties dynamic in Runtime....4604
12.8.6 Examples....4604
12.8.6.1 Example: Converting Fahrenheit into degrees Celsius....4604
12.8.6.2 Example: Converting inches into meters....4606
12.8.6.3 Example: Changing the operating mode on the HMI device with the current display....4608
12.8.7 Reference....4612
12.8.7.1 Function list....4612
12.8.7.2 Events....4836
12.8.7.3 VB scripts....4858

12.9 Mobile Panels....5944
12.9.1 Field of application of the Mobile Panel Wireless....5944

12.9.2 How does the transponder system work?....5945
12.9.3 How does the RFID system work?....5948
12.9.4 Configuring the Mobile Panel V2 for fail-safe operation (up to V12)....5950
12.9.4.1 Configuration overview....5950
12.9.4.2 Creating PLC for fail-safe operation....5951
12.9.4.3 Installing a general station description (GSD)....5952
12.9.4.4 Creating a module from the GSD file....5952
12.9.4.5 Creating a connection between a module and the PLC....5955
12.9.4.6 Creating Mobile Panel....5956
12.9.5 Configuring KTP Mobile Panels for fail-safe operation....5957
12.9.5.1 Creating KTP Mobile Panel for fail-safe operation....5957
12.9.5.2 Configuring the termination of the PROFIsafe connection....5958
12.9.6 Basics....5959
12.9.6.1 Zones....5959
12.9.6.2 Zones work area at transponders....5960
12.9.6.3 Zones work area on connection boxes....5961
12.9.6.4 Effective ranges....5961
12.9.6.5 Effective ranges working area....5962
12.9.6.6 Effective ranges (RFID)....5964
12.9.6.7 Work area for effective ranges (RFID)....5964
12.9.7 Working with zones on transponders....5966
12.9.7.1 Configuring a zone in the transponder system....5966
12.9.7.2 Displaying the screen on entering a zone....5967
12.9.7.3 Displaying an object in relation to the zone....5968
12.9.8 Working with zones on connection boxes....5969
12.9.8.1 Configuring the zone of a connection box....5969
12.9.8.2 Configuring a screen for connection to connection box....5970
12.9.9 Working with effective ranges....5971
12.9.9.1 Overview....5971
12.9.9.2 Configure effective range....5972
12.9.9.3 Configuring the effective range name....5973
12.9.9.4 Configuring additional Mobile Wireless objects....5974
12.9.10 Working with effective ranges (RFID)....5974
12.9.10.1 Overview....5974
12.9.10.2 Configuring effective range (RFID)....5975
12.9.10.3 Configuring the effective range name (RFID)....5976
12.9.11 Reference....5977
12.9.11.1 PROFIsafe address....5977
12.9.11.2 Power Management....5978
12.9.11.3 Zone ID / connection point ID....5978

12.10 Planning tasks....5979
12.10.1 Field of application of the Scheduler....5979
12.10.2 Working with tasks and triggers....5980
12.10.3 Basics....5980
12.10.3.1 Work area of the "Scheduler" editor....5980
12.10.3.2 Function list....5982
12.10.3.3 Function list....5982
12.10.3.4 Triggers....5983
12.10.4 Planning jobs....5985
12.10.4.1 Planning tasks with acyclic triggers....5985
12.10.4.2 Planning tasks with cyclic triggers....5986
12.10.4.3 Planning tasks with event triggers....5987

12.10.4.4 Administer task....5988
12.10.5 Examples....5989
12.10.5.1 Example: Terminating Runtime every day....5989
12.10.5.2 Example: Update user following change of user....5990
12.10.5.3 Example: Changing the starting point of a job in Runtime....5991

12.11 Communicating with PLCs....5996
12.11.1 Basics of communication....5996
12.11.1.1 Communication between devices....5996
12.11.1.2 Devices and networks in the automation system....5997
12.11.1.3 Data exchange using tags....6000
12.11.1.4 Data exchange using area pointers....6001
12.11.1.5 Communication drivers....6001
12.11.2 Editors for communication....6002
12.11.2.1 "Devices & networks" editor....6002
12.11.2.2 Network view....6003
12.11.2.3 Network data....6006
12.11.2.4 Diagnostics of online connections....6008
12.11.2.5 Device view....6010
12.11.2.6 Topology view....6012
12.11.2.7 Inspector window....6014
12.11.2.8 Hardware catalog....6016
12.11.2.9 Information on hardware components....6018
12.11.3 Networks and connections....6019
12.11.3.1 SIMATIC communication networks....6019
12.11.3.2 Configuring networks and connections....6024
12.11.4 Data exchange....6034
12.11.4.1 Data exchange using tags....6034
12.11.4.2 Data exchange using area pointers....6041
12.11.5 Device dependency....6049
12.11.5.1 Basic Panel....6049
12.11.5.2 Panel....6055
12.11.5.3 Comfort Panel....6059
12.11.5.4 Multi Panel....6066
12.11.5.5 Mobile Panel....6070
12.11.5.6 PC systems....6075
12.11.5.7 Parallel communication....6079
12.11.6 Communicating with SIMATIC S7 1500....6081
12.11.6.1 Communication with SIMATIC S7 1500....6081
12.11.6.2 Communication via PROFINET....6082
12.11.6.3 Communication via PROFIBUS....6106
12.11.6.4 Data exchange....6121
12.11.6.5 Performance features of communication....6141
12.11.6.6 Configuring connections in the "Connections" editor....6148
12.11.6.7 Configuring connections in the "Connections" editor....6155
12.11.6.8 Configuring time synchronization....6160
12.11.7 Communicating with SIMATIC S7 1200....6164
12.11.7.1 Communication with SIMATIC S7 1200....6164
12.11.7.2 Communication via PROFINET....6164
12.11.7.3 Communication via PROFIBUS....6191
12.11.7.4 Data exchange....6208
12.11.7.5 Performance features of communication....6231
12.11.7.6 Creating connections in the "Connections" editor....6238

12.11.7.7 Time synchronization....6246
12.11.8 Communicating with SIMATIC S7 300/400....6251
12.11.8.1 Communication with SIMATIC S7 300/400....6251
12.11.8.2 Communication via PROFINET....6251
12.11.8.3 Communication via PROFIBUS....6272
12.11.8.4 Communication via MPI....6287
12.11.8.5 Data exchange....6301
12.11.8.6 Performance features of communication....6320
12.11.8.7 Creating connections in the "Connections" editor....6326
12.11.9 Communicating with the SIMATIC S7-1500 Software Controller....6338
12.11.9.1 Communication with SIMATIC S7-1500 Software Controller....6338
12.11.9.2 Communication via PROFINET....6339
12.11.9.3 PROFINET parameters....6350
12.11.9.4 Data exchange....6356
12.11.9.5 Performance features of communication....6377
12.11.9.6 Configuring connections in the "Connections" editor....6383
12.11.9.7 Configuring connections in the "Connections" editor....6385
12.11.9.8 Configuring time synchronization....6388
12.11.10 Communicating with SIMATIC ET 200 CPU....6391
12.11.10.1 Communication with SIMATIC ET 200 CPU....6391
12.11.10.2 Communication via PROFINET....6392
12.11.10.3 Communication via PROFIBUS....6409
12.11.10.4 Data exchange....6422
12.11.10.5 Performance features of communication....6445
12.11.10.6 Configuring connections in the "Connections" editor....6451
12.11.10.7 Configuring connections in the "Connections" editor....6460
12.11.10.8 Configuring time synchronization....6465
12.11.11 Communicating with SIMATIC S7 200....6469
12.11.11.1 Communication with SIMATIC S7 200....6469
12.11.11.2 Creating a connection to SIMATIC S7 200....6469
12.11.11.3 Parameters for the connection....6471
12.11.11.4 Data exchange....6478
12.11.11.5 Performance features of communication....6498
12.11.12 Communicating with SIMATIC LOGO!....6503
12.11.12.1 Communication with SIMATIC LOGO!....6503
12.11.12.2 Creating a connection to SIMATIC LOGO!....6504
12.11.12.3 Connection parameters....6505
12.11.12.4 Data exchange....6509
12.11.12.5 Performance features of communication....6513
12.11.13 Configuring direct keys....6518
12.11.13.1 Direct keys....6518
12.11.13.2 Changing the operating mode of the HMI device....6519
12.11.13.3 Configuring direct keys....6520
12.11.13.4 PROFINET IO direct keys....6521
12.11.13.5 PROFIBUS DP direct keys....6540
12.11.14 Communication via SIMATIC HMI HTTP....6557
12.11.14.1 Basic information on SIMATIC HMI HTTP....6557
12.11.14.2 Configuring a connection via SIMATIC HMI HTTP....6559
12.11.14.3 Performance features of communication....6572
12.11.15 Communication via OPC....6573
12.11.15.1 Communication via OPC....6573
12.11.15.2 Configuring a connection via OPC....6574

12.11.15.3 Performance features of communication....6576
12.11.16 Communication via routing....6578
12.11.16.1 Communication via routing....6578
12.11.16.2 Example for communication via routing....6580
12.11.16.3 Configuring communication via routing....6582
12.11.17 PROFINET IO and IRT....6584
12.11.17.1 PROFINET IO....6584
12.11.17.2 IRT communication....6585
12.11.17.3 Configuring the HMI device as IO device....6588
12.11.17.4 Parameters for PROFINET IO and IRT....6589
12.11.17.5 Performance characteristics of PROFINET IO and IRT....6592
12.11.18 Media redundancy....6593
12.11.18.1 Restrictions with media redundancy....6593
12.11.18.2 Media redundancy....6593
12.11.18.3 Media Redundancy Protocol (MRP)....6595
12.11.18.4 Configuring media redundancy for HMI devices....6596
12.11.18.5 Parameters for media redundancy....6598
12.11.18.6 Managing MRP domains....6599
12.11.19 Communication with other PLCs....6601
12.11.19.1 Communication with other PLCs....6601
12.11.19.2 Distinctive features when configuring....6602
12.11.19.3 Parallel communication....6602
12.11.19.4 Communication drivers....6604
12.11.19.5 Data exchange using area pointers....6730
12.11.20 Special features of WinAC MP....6745
12.11.20.1 WinAC MP basics....6745
12.11.20.2 Communication options with WinAC MP....6747
12.11.20.3 Standard procedure for communication with WinAC MP....6751
12.11.20.4 Configuring the WinAC MP communications driver....6752
12.11.20.5 Transferring WinAC MP to the HMI device....6763
12.11.20.6 Transferring authorization to the HMI device....6767
12.11.20.7 WinAC MP system library....6768

12.12 Using global functions....6768
12.12.1 HMI device wizard basics....6768
12.12.2 HMI device wizard basics....6770
12.12.3 Working with libraries....6771
12.12.3.1 Basics on libraries....6771
12.12.3.2 Overview of the library view....6774
12.12.3.3 Master copies and types....6775
12.12.3.4 Libraries in WinCC....6776
12.12.3.5 Managing libraries....6778
12.12.3.6 Managing objects in a library....6786
12.12.3.7 Using types and their versions....6791
12.12.4 Importing and exporting project data....6795
12.12.4.1 Importing and exporting project data....6795
12.12.4.2 Import and export of recipes....6796
12.12.4.3 Importing and exporting alarms....6800
12.12.4.4 Importing and exporting tags....6807
12.12.4.5 Importing and exporting text lists....6814
12.12.5 Exchanging controller data from other projects....6818
12.12.6 Using cross-references....6819
12.12.6.1 General information about cross references....6819

12.12.6.2 Displaying the cross-reference list....6820
12.12.6.3 Structure of the cross-reference list....6820
12.12.6.4 Displaying cross-references in the Inspector window....6821
12.12.6.5 Rewiring tags in the screens....6823
12.12.7 Managing languages....6825
12.12.7.1 Languages in WinCC....6825
12.12.7.2 Language settings in the operating system....6826
12.12.7.3 Operating system settings for Asian languages....6826
12.12.7.4 Setting project languages....6827
12.12.7.5 Creating one project in multiple languages....6831
12.12.7.6 Using language-specific graphics....6838
12.12.7.7 Languages in Runtime....6842
12.12.7.8 Example of multilingual configuration....6849
12.12.8 Replacing devices....6852
12.12.8.1 Basics....6852
12.12.8.2 Device-specific functions....6853
12.12.8.3 Adjusting screens to the new device....6858
12.12.8.4 Example: Replacing devices....6863
12.12.9 Copying between devices and editors....6867
12.12.9.1 Basics....6867
12.12.9.2 Copying and pasting....6870
12.12.9.3 Copying between different RT and ES versions....6874
12.12.10 Using WinCC version compatibility....6875
12.12.10.1 Basics on version compatibility....6875
12.12.10.2 Editing projects of a previous WinCC version....6878
12.12.10.3 Upgrading projects....6879
12.12.10.4 Upgrading a global library....6880
12.12.10.5 Changing between device versions....6881
12.12.10.6 Changing the device version....6882
12.12.11 Viewing memory card data....6883
12.12.11.1 Basics....6883
12.12.11.2 Working with backups....6883
12.12.11.3 Working with HMI device images....6886
12.12.12 Managing colors centrally....6890
12.12.12.1 Basic principles for color management....6890
12.12.12.2 Finding and replacing colors....6891

12.13 Compiling and loading....6892
12.13.1 Establishing a connection to the HMI device....6892
12.13.2 Basic Panels....6893
12.13.2.1 Runtime settings....6893
12.13.2.2 Overview of compiling and loading projects....6895
12.13.2.3 Compiling a project....6896
12.13.2.4 Simulating projects....6897
12.13.2.5 Loading projects....6902
12.13.2.6 Runtime start....6905
12.13.2.7 Error messages during loading of projects (Basic)....6906
12.13.2.8 Adapting the project for another HMI device....6908
12.13.2.9 Basics of operating in Runtime....6909
12.13.3 Runtime Advanced and Panels....6922
12.13.3.1 Runtime settings....6922
12.13.3.2 Overview of compiling and loading projects....6927
12.13.3.3 Compiling a project....6929

12.13.3.4 Simulating projects....6930
12.13.3.5 Loading projects....6937
12.13.3.6 Runtime start....6952
12.13.3.7 Error messages during the download of projects....6954
12.13.3.8 Adapting the project for another HMI device....6956
12.13.3.9 Viewing memory card data....6958
12.13.3.10 Basics of operating in Runtime....6965
12.13.4 Viewing memory card data....6983
12.13.4.1 Basics....6983
12.13.4.2 Working with backups....6983
12.13.4.3 Working with HMI device images....6986
12.13.5 Servicing the HMI device....6990
12.13.5.1 Overview of HMI device maintenance (Basic Panels)....6990
12.13.5.2 Overview of HMI device maintenance tasks (Basic Panels)....6991
12.13.5.3 ProSave....6992
12.13.5.4 Backup of HMI data....6993
12.13.5.5 Backing up and restoring data of the HMI device....6994
12.13.5.6 Updating the operating system....6996
12.13.5.7 Updating the operating system on the HMI device....6997
12.13.5.8 Transferring license keys....6998
12.13.5.9 Managing licenses....6999
12.13.5.10 Installing and uninstalling an option....7001
12.13.6 Printer driver....7002
12.13.6.1 Validity....7002
12.13.6.2 Supported HMI devices....7002
12.13.6.3 Installation....7003
12.13.6.4 Transferring the Options....7004
12.13.6.5 Setting up the printer driver....7005

12.14 Performance features....7007
12.14.1 Engineering system....7007
12.14.2 Basic Panel....7009
12.14.3 Basic Panel 2nd Generation....7013
12.14.4 Panel....7016
12.14.5 Mobile Panel....7020
12.14.6 Multi Panel....7026
12.14.7 Comfort Panel....7029
12.14.8 WinCC Runtime Advanced....7034
12.14.9 General technical specifications....7038
12.14.9.1 Permitted characters....7038
12.14.9.2 Recommended printers....7038
12.14.9.3 Printing via print server....7039
12.14.9.4 Memory requirement of recipes....7040
12.14.9.5 Memory requirements of recipes for Basic Panels, OP 77A, and TP 177A....7041

12.15 Options....7042
12.15.1 WinCC Audit....7042
12.15.1.1 Basics....7042
12.15.1.2 Enabling GMP compliant configuration....7046
12.15.1.3 Using the Audit trail....7048
12.15.1.4 Configuring audit functions....7068
12.15.1.5 Performance features of GMP relevant configuration....7082
12.15.2 WinCC Sm@rtServer....7083

12.15.2.1 Basics....7083
12.15.2.2 Remote control via Sm@rtServer....7115
12.15.2.3 E-mail notification from runtime....7127
12.15.2.4 Display integrated Service-Pages....7131
12.15.2.5 Access via SIMATIC HMI HTTP Protocol....7142
12.15.2.6 Connection to the Office-world....7151

12.16 Interfaces....7155
12.16.1 OPC....7155
12.16.1.1 Basics....7155
12.16.1.2 Configuring an OPC server....7158
12.16.1.3 Configuring an OPC client....7161
12.16.1.4 Security concept of OPC UA....7166
12.16.1.5 Reference....7168

12.17 Migrating to WinCC in the TIA Portal....7171
12.17.1 Overview of the migration to WinCC in the TIA Portal....7171
12.17.2 WinCC flexible....7172
12.17.2.1 Libraries....7172
12.17.2.2 Screens and templates....7174
12.17.2.3 Scripts in faceplates....7176
12.17.2.4 Synchronization of recipes....7177
12.17.2.5 Special considerations for converting....7178

13 Using technology functions....7179

13.1 PID control....7179
13.1.1 Principles for control....7179
13.1.1.1 Controlled system and actuators....7179
13.1.1.2 Controlled systems....7180
13.1.1.3 Characteristic values of the control section....7182
13.1.1.4 Pulse controller....7184
13.1.1.5 Response to setpoint changes and disturbances....7188
13.1.1.6 Control Response at Different Feedback Structures....7189
13.1.1.7 Selection of the controller structure for specified controlled systems....7197
13.1.1.8 PID parameter settings....7198
13.1.2 Configuring a software controller....7198
13.1.2.1 Overview of software controller....7198
13.1.2.2 Steps for the configuration of a software controller....7200
13.1.2.3 Add technology objects....7201
13.1.2.4 Configure technology objects....7202
13.1.2.5 Call instruction in the user program....7203
13.1.2.6 Downloading technology objects to device....7204
13.1.2.7 Commissioning software controller....7205
13.1.2.8 Save optimized PID parameter in the project....7205
13.1.2.9 Comparing values....7206
13.1.2.10 Parameter view....7209
13.1.2.11 Display instance DB of a technology object....7226
13.1.3 Using PID_Compact....7226
13.1.3.1 Technology object PID_Compact....7226
13.1.3.2 PID_Compact V2....7227
13.1.3.3 PID_Compact V1....7243
13.1.4 Using PID_3Step....7258
13.1.4.1 Technology object PID_3Step....7258

13.1.4.2 PID_3Step V2....7259
13.1.4.3 PID_3Step V1....7276
13.1.5 Using PID_Temp....7291
13.1.5.1 Technology object PID_Temp....7291
13.1.5.2 Configuring PID_Temp....7292
13.1.5.3 Commissioning PID_Temp....7313
13.1.5.4 Cascade control with PID_Temp....7321
13.1.5.5 Multi-zone controlling with PID_Temp....7327
13.2 Using S7-1200 Motion Control....7330
13.2.1 Introduction....7330
13.2.1.1 Motion functionality of the CPU S7-1200....7330
13.2.1.2 Hardware components for motion control....7331
13.2.2 Basics for working with S7-1200 Motion Control....7333
13.2.2.1 Drive connection via PTO....7333
13.2.2.2 PROFIdrive/analog drive connection....7340
13.2.2.3 Hardware and software limit switches....7346
13.2.2.4 Jerk limit....7347
13.2.2.5 Homing....7348
13.2.3 Guidelines on use of motion control....7349
13.2.4 Using versions....7349
13.2.4.1 Overview of versions....7349
13.2.4.2 Changing a technology version....7352
13.2.4.3 Compatibility list of tags....7352
13.2.4.4 Status of limit switch....7355
13.2.5 Positioning axis technology object....7357
13.2.5.1 Integration of the positioning axis technology object....7357
13.2.5.2 Tools of the positioning axis technology object....7360
13.2.5.3 Adding a positioning axis technology object....7362
13.2.5.4 Configuring the positioning axis technology object....7363
13.2.6 Technology object command table....7415
13.2.6.1 Use of the command table technology object....7415
13.2.6.2 Command table technology object tools....7416
13.2.6.3 Adding the technological object command table....7416
13.2.6.4 Configuring the command table technology object....7417
13.2.7 Download to CPU....7434
13.2.8 Commissioning....7435
13.2.8.1 Axis control panel....7435
13.2.8.2 Tuning....7438
13.2.9 Programming....7440
13.2.9.1 Overview of the Motion Control statements....7440
13.2.9.2 Creating a user program....7441
13.2.9.3 Programming notes....7444
13.2.9.4 Behavior of the Motion Control commands after POWER OFF and restart....7446
13.2.9.5 Monitoring active commands....7446
13.2.9.6 Error displays of the Motion Control statements....7457
13.2.9.7 Restart of technology objects....7458
13.2.10 Axis - Diagnostics....7459
13.2.10.1 Status and error bits (technology objects as of V4) 7459
13.2.10.2 Motion status....7462
13.2.10.3 Dynamics settings....7462
13.2.10.4 PROFIdrive frame....7463
13.2.11 Appendix....7464

13.2.11.1 Using multiple axes with the same PTO....7464
13.2.11.2 Using multiple drives with the same PTO....7467
13.2.11.3 Tracking jobs from higher priority classes (execution levels)....7468
13.2.11.4 Special cases when using software limit switches for drive connection via PTO....7470
13.2.11.5 Reducing velocity for a short positioning duration....7476
13.2.11.6 Dynamic adjustment of start/stop velocity....7477
13.2.11.7 List of ErrorIDs and ErrorInfos (technology objects as of V4)....7477
13.2.11.8 Tags of the positioning axis technology object as of V4....7498
13.2.11.9 Tags of the command table technology object as of V4....7532
13.2.11.10 Versions V1...4....7533

14 Using online and diagnostics functions....7587

14.1 Displaying accessible devices....7587

14.2 Changing the device configuration online....7588

14.3 Connecting devices online....7589
14.3.1 General information about online mode....7589
14.3.2 View in online mode....7591
14.3.3 Default setting online connection data....7592
14.3.4 Establishing or changing an online connection....7593
14.3.5 Canceling an online connection....7595
14.3.6 Connecting online with several devices....7595
14.3.7 Disconnecting online connections of multiple devices....7596

14.4 Creating a backup of an S7 CPU....7597
14.4.1 Backup options for S7 CPUs....7597
14.4.2 Backing up S7-300 and S7-400 CPUs....7598
14.4.2.1 Creating a backup of a device....7598
14.4.2.2 Restoring the software and hardware configuration of a device....7599
14.4.2.3 Backing up a device configuration....7600
14.4.3 Backing up S7-1200 and S7-1500 CPUs....7601
14.4.3.1 Creating a backup of a device....7601
14.4.3.2 Backing up a device configuration....7602
14.4.3.3 Restoring the configuration of a device....7603

14.5 Configuring the PG/PC interface....7604
14.5.1 Online access....7604
14.5.2 Basics of assigning parameters for the PG/PC interface....7606
14.5.3 Showing or hiding interfaces....7606
14.5.4 Displaying and modifying interface properties....7607
14.5.5 Adding interfaces....7607
14.5.6 Setting parameters for the Ethernet interface....7608
14.5.6.1 Setting parameters for the Industrial Ethernet interface....7608
14.5.6.2 Displaying operating system parameters....7609
14.5.6.3 Connecting the PG/PC interface to a subnet....7609
14.5.6.4 Setting parameters for the Ethernet interface....7610
14.5.6.5 Assigning a temporary IP address....7610
14.5.6.6 Managing temporary IP addresses....7611
14.5.6.7 Resetting the TCP/IP configuration....7611
14.5.7 Setting parameters for the MPI and PROFIBUS interfaces....7612
14.5.7.1 Setting parameters for the MPI and PROFIBUS interfaces....7612
14.5.7.2 Setting MPI or PROFIBUS interface parameters automatically....7613
14.5.7.3 Setting parameters for the MPI interface....7614

14.5.7.4 Setting parameters for the PROFIBUS interface....7616
14.5.7.5 Overview of the bus parameters for PROFIBUS....7618
14.5.7.6 Resetting the MPI or PROFIBUS configuration....7619

14.6 Using the trace and logic analyzer function....7620
Preface....7620
14.6.1 Security information....7621
14.6.1 Description....7621
14.6.1.1 Supported hardware....7621
14.6.1.2 Recording of measured values with the trace function....7621
14.6.1.3 Trace configuration, recording, installed trace and measurement....7623
14.6.1.4 Data storage....7624
14.6.2 Software user interface....7625
14.6.2.1 Project navigator....7626
14.6.2.2 Working area....7628
14.6.2.3 Inspector window....7633
14.6.3 Operation....7633
14.6.3.1 Quick start....7633
14.6.3.2 Using the trace function - overview....7638
14.6.3.3 Project tree....7638
14.6.3.4 Working area - general....7640
14.6.3.5 Working area - Configuration tab....7644
14.6.3.6 Working area - Diagram tab....7644
14.6.4 Devices....7648
14.6.4.1 S7-1200/1500 CPUs....7648
A Service & Support....7661

14.7 Establishing a remote connection with TeleService....7664
14.7.1 Basics of working with TeleService....7664
14.7.1.1 Introduction to TeleService....7664
14.7.1.2 TeleService functionality....7665
14.7.1.3 Telephone book at TeleService....7665
14.7.2 Working with the phone book....7666
14.7.2.1 Basics on working with the phone book....7666
14.7.2.2 Structure of the phone book....7667
14.7.2.3 Symbols in the phone book....7668
14.7.2.4 Manage phone book....7668
14.7.3 Remote connections as dial-up connections....7674
14.7.3.1 Basics for establishing a dial-up connection....7674
14.7.3.2 Telephone networks and modems....7675
14.7.3.3 Access protection for dial-up connections....7678
14.7.3.4 TS adapter MPI....7683
14.7.3.5 TS adapter IE....7690
14.7.3.6 Establishing a dial-up connection to a remote system....7696
14.7.4 Remote VPN connections....7698
14.7.4.1 Basics for establishing a VPN connection....7698
14.7.4.2 Basics of CA certificates....7699
14.7.4.3 Installing CA certificates for VPN connections....7701
14.7.4.4 Deleting CA certificates for VPN connections....7704
14.7.4.5 Establishing a VPN connection to a remote system....7704
14.7.4.6 TS Adapter IE Advanced....7706
14.7.5 CPU controlled TeleService remote connections 7711
14.7.5.1 Overview of CPU controlled remote connections....7711

14.7.5.2 Establishing a connection from and to remote systens (PG-AS-remote coupling)....7712
14.7.5.3 Data exchange between remote systems (AS-AS-remote coupling)....7713
14.7.5.4 Send SMS from a system....7715
14.7.5.5 Send an email from a system....7716
14.7.6 Notes on troubleshooting....7719
14.7.6.1 General information on troubleshooting for modem problems....7719
14.7.6.2 Recording a log file for the modem....7719
14.7.6.3 Dial-up connection to the TS Adapter is not established....7720
14.7.6.4 Dial-up connection from the TS Adapter is not established....7722
14.7.6.5 Modem connection is interrupted....7723
14.7.6.6 Checklist for troubleshooting the modem....7723
14.7.6.7 Modem alarms....7724
14.7.6.8 Possible error messages with VPN connections....7725

15 Using Team Engineering....7727

15.1 Shared commissioning of projects....7727
15.1.1 Basics for shared commissioning....7727
15.1.2 Requirements for shared commissioning....7730
15.1.3 Procedure for shared commissioning....7731
15.1.4 Rules for shared commissioning....7734

15.2 Exchanging data with Inter Project Engineering (IPE)....7740
15.2.1 Basics of Inter Project Engineering (IPE)....7740
15.2.2 Requirements for Inter Project Engineering (IPE)....7743
15.2.3 Overview for working with Inter Project Engineering (IPE)....7743
15.2.4 Creating "Device proxy data" in the source project....7746
15.2.5 Creating an IPE file by means of the "Device proxy data"....7747
15.2.6 Using controller data from other projects with IPE....7747
15.2.6.1 Using controller data from other projects in an HMI device....7747
15.2.6.2 Communication with device proxies....7759
15.2.6.3 Integrated configuring with WinCC and SIMATIC Manager....7766

16 Hardware documentation....7779

16.1 General information on the hardware documentation....7779

16.2 HMI....7779
16.2.1 Basic Panels....7779
16.2.1.1 Basic Panels....7779
16.2.2 Panels....7779
16.2.2.1 Panels of the 70 series....7779
16.2.2.2 Panels of the 170 series....7779
16.2.2.3 Panels of the 270 series....7780
16.2.3 Comfort Panels....7780
16.2.3.1 Comfort Panels....7780
16.2.4 Multi Panels....7780
16.2.4.1 170 series....7780
16.2.4.2 270 series....7780
16.2.4.3 370 series....7780
16.2.5 Mobile Panels....7780
16.2.5.1 170 series....7780
16.2.5.2 270 series....7781
16.2.6 Key Panels....7781
16.2.6.1 Key Panels....7781

16.2.6.2 Push Button Panels....7781
16.2.7 WinAC for Multi Panels....7781
16.2.7.1 WinAC for Multi Panel....7781
16.2.8 PC-based Automation....7781

16.3 Controller....7782
16.3.1 SIMATIC S7-1200....7782
16.3.1.1 CPU....7782
16.3.1.2 Signal boards....7788
16.3.1.3 Communication boards....7790
16.3.1.4 Battery boards....7790
16.3.1.5 Digital input modules....7790
16.3.1.6 Digital output modules....7792
16.3.1.7 Digital input and digital output modules....7794
16.3.1.8 Analog input modules....7797
16.3.1.9 Analog output modules....7800
16.3.1.10 Analog input and analog output modules....7801
16.3.1.11 Communications modules....7802
16.3.1.12 Technology modules....7806

16.4 Distributed I/O....7807
16.4.1 ET 200SP....7807
16.4.1.1 Interface modules....7807
16.4.1.2 Digital input modules....7808
16.4.1.3 Digital output modules....7810
16.4.1.4 Analog input modules....7811
16.4.1.5 Analog output modules....7813
16.4.1.6 Communication modules....7814
16.4.1.7 Power modules....7815
16.4.1.8 Special modules....7815
16.4.1.9 Technology modules....7816
16.4.1.10 BusAdapter....7817
16.4.2 ET 200MP....7817
16.4.2.1 Interface modules....7817
16.4.2.2 Digital input modules....7818
16.4.2.3 Digital output modules....7819
16.4.2.4 Digital input and digital output modules....7821
16.4.2.5 Analog input modules....7821
16.4.2.6 Analog output modules....7821
16.4.2.7 Analog input and analog output modules....7822
16.4.2.8 Communications modules....7822
16.4.2.9 Power supply module....7824
16.4.2.10 Technology modules....7825
16.4.3 ET 200AL....7825
16.4.3.1 Interface modules....7825
16.4.3.2 Digital input modules....7826
16.4.3.3 Digital input modules....7826
16.4.3.4 Analog input modules....7826
16.4.3.5 Communications modules....7827

Index....7829

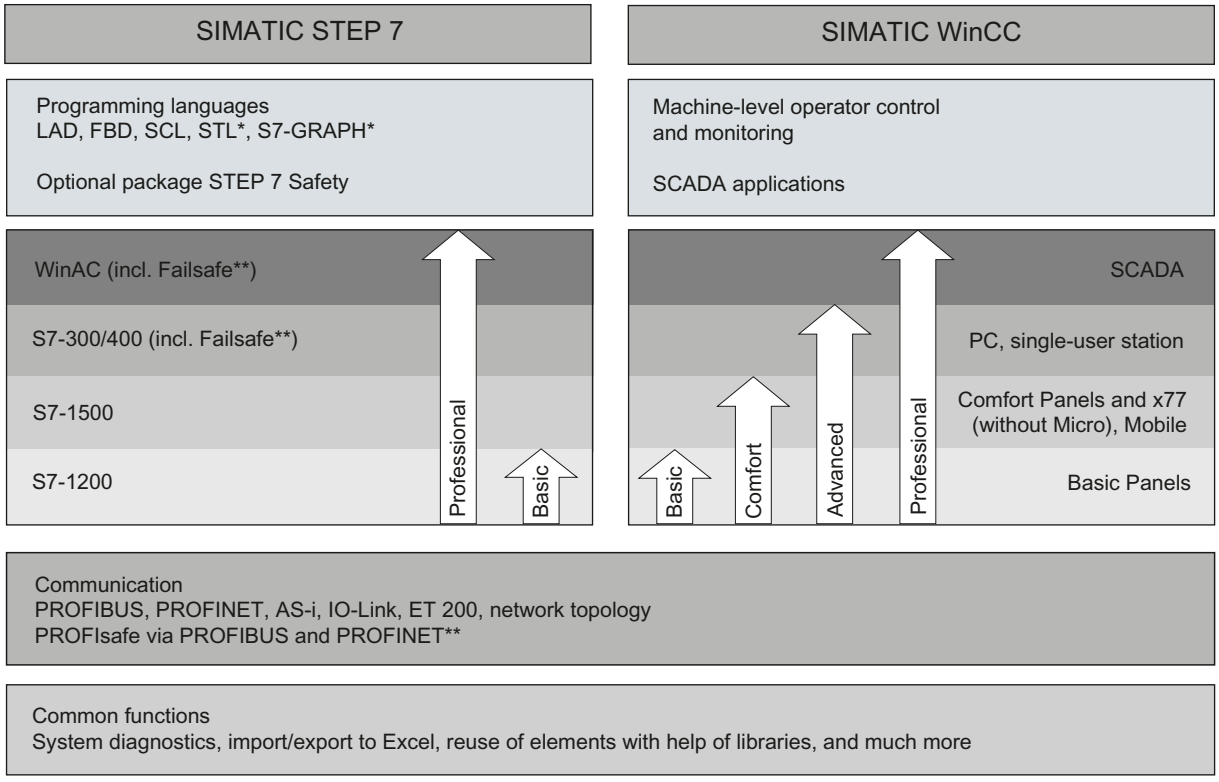
System overview of STEP 7 and WinCC

1

1.1 Scaling of STEP 7 and WinCC in the TIA Portal

Scope of performance of the products

The following graphic shows the scope of performance of the individual products of STEP 7 and WinCC:



* Only with STEP 7 Professional for S7-300/400/WinAC and S7-1500
** With installed optional package "STEP 7 Safety Advanced"

STEP 7

STEP 7 (TIA Portal) is the engineering software for configuring the SIMATIC S7-1200, S7-1500, S7-300/400 and WinAC controller families. STEP 7 (TIA Portal) is available in two editions, depending on the configurable controller families:

- STEP 7 Basic for configuring the S7-1200
- STEP 7 Professional for configuring S7-1200, S7-1500, S7-300/400 and WinAC

WinCC

WinCC (TIA Portal) is an engineering software for configuring SIMATIC Panels, SIMATIC Industrial PCs, and Standard PCs with the WinCC Runtime Advanced or the SCADA System WinCC Runtime Professional visualization software.

WinCC (TIA Portal) is available in four editions, depending on the configurable operator control systems:

- WinCC Basic for configuring Basic Panels
  WinCC Basic is included with every STEP 7 Basic and STEP 7 Professional product.
- WinCC Comfort for configuring all panels (including Comfort Panels, Mobile Panels)
- WinCC Advanced for configuring all panels and PCs with the WinCC Runtime Advanced visualization software
  WinCC Runtime Advanced is a visualization software for PC-based single-station systems. WinCC Runtime Advanced can be purchased with licenses for 128, 512, 2k, 4k as well as 8k PowerTags (tags with a process interface).
- WinCC Professional for configuring panels and PCs with WinCC Runtime Advanced or SCADA System WinCC Runtime Professional. WinCC Professional is available in the following editions: WinCC Professional for 512 and 4096 PowerTags as well as "WinCC Professional max. PowerTags".
  WinCC Runtime Professional is a SCADA system for structuring a configuration ranging from single-station systems to multi-station systems including standard clients or web clients. WinCC Runtime Professional can be purchased with licenses for 128, 512, 2k, 4k, 8k, and 64k PowerTags (tags with a process interface).

With WinCC (TIA Portal), it is also possible to configure a SINUMERIK PC with WinCC Runtime Advanced or WinCC Runtime Professional and HMI devices with SINUMERIK HMI Pro sl RT or SINUMERIK Operate WinCC RT Basic.

1.2 Options for STEP 7 Engineering System

Additional STEP 7 products

For applications with increased safety requirements, STEP 7 Professional can be supplemented with the STEP 7 Safety Advanced option.

When using the STEP 7 Safety Advanced option, you can configure failsafe I/O and program safety programs for F-CPUs in LAD and FBD.

1.3 Options for WinCC Engineering and Runtime systems

SIMATIC Panels as well as WinCC Runtime Advanced and WinCC Runtime Professional contain all essential functions for operator control and monitoring of machines or plants. Additional options allow you to extend the functionality in some cases to increase the range of available tasks.

Options for Comfort Panels, Mobile Panels, Multi Panels

The following possible extensions are available for Comfort Panels, Mobile Panels, and Multi Panels:

- WinCC SmartServer (remote operation)
- WinCC Audit (audit trail and electronic signature for regulated applications)

Note

In contrast to WinCC flexible 2008, functions from the WinCC flexible /Sm@rtService, WinCC flexible /Sm@rtAccess options as well as the WinCC flexible /OPC Server option are incorporated into the basic functionality.

Options for WinCC Runtime Advanced

The following possible extensions are available for WinCC Runtime Advanced:

- WinCC SmartServer (remote operation)
- WinCC Recipes (recipe system)
- WinCC Logging (logging of process values and alarms)
- WinCC Audit (audit trail for regulated applications)
- WinCC ControlDevelopment (extension by means of customer-specific controls)

Note

In contrast to WinCC flexible 2008, functions from the WinCC flexible /Sm@rtService, WinCC flexible /Sm@rtAccess options as well as the WinCC flexible /OPC Server option are incorporated into the basic functionality.

Options for WinCC Runtime Professional

The following possible extensions are available for WinCC Runtime Professional:

- WinCC Client (standard client for structuring multi-station systems)
- WinCC Server (supplements WinCC Runtime to include server functionality)
- WinCC Recipes (recipe system, formerly WinCC /UserArchives)
- WinCC WebNavigator (Web-based operator control and monitoring)
- WinCC DataMonitor (display and evaluation of process states and historical data)
- WinCC ControlDevelopment (extension by means of customer-specific controls)

Note

In contrast to WinCC V7, functions from the WinCC /OPC-Server and WinCC / ConnectivityPack options are incorporated into the basic functionality. Likewise, the basic functionality includes the Runtime API from WinCC /ODK.

What's new in WinCC Advanced? 2

2.1 What's new in WinCC V13 SP1?

New features in WinCC Advanced V13 SP1

You can find all important new features in WinCC Advanced V13 SP1 here:

Topic	New features in the TIA Portal
HMI devices	You can now configure the following new HMI devices: • Mobile Panels: KTP700 Mobile and KTP900 Mobile The following HMI devices have been extended in terms of their functionality: • Basic Panels 2nd Generation
Configuring screen	The following editors for creating HMI screens have been extended: • "Design" editor • "Styles" editor The following screen objects can now be configured: • "Slide-in screen" • "Pop-up screen" • "PDF view" • "Camera view" You can search for and replace colors within HMI screens.
Working with tags	Synchronization of HMI tags and PLC tags of data type "UDT". The range of data types for Panels and Runtime Advanced has been expanded.
Exchanging data with Inter Project Engineering (IPE)	The exchange of controller data using proxy devices has been extended.
Automating projects with scripts	You can automate projects with scripts using the "Openness" option. Install the "Openness" option using the DVD: "Support/Siemens_TIA_Openness_V13_SP1.exe"

What's new in STEP 7 Basic? 3

3.1 What's new in STEP 7 Basic?

New features in the TIA Portal

You can find all important new features in TIA Portal V13 SP1 here:

Topic	New features in the TIA Portal
Installation	The new virus scanner 360 Safety Guard is supported.
Migrating projects and programs	You can find many new examples of effective programming of S7-1200/1500 in the programming recommendations. (Page 224)
Editing projects	Information, warnings and errors can be hidden in the Inspector window. (Page 324) The creation of user-defined documentation is possible. (Page 361) Function extensions in libraries: • Types can be changed in the project view. (Page 501) • Drag-and-drop is also possible for multiple selection of objects.
Editing devices and networks	Devices can be compared at module level in the offline/offline comparison. (Page 579) HW system constants have unique names. The server module (6ES7 193-6PA00-0AA0) of the ET 200SP is automatically inserted after a compilation Installed GSD/GSDML files can be removed. The selection of partners for direct data exchange has been simplified. Device numbers are displayed in the network overview. PROFINET device names can be assigned easily. You can simply zoom in on the device and network views. Configuration control is available for S7-1200 as of FW4.1.

Topic	New features in the TIA Portal
Programming the PLC	New functions are available for controlling tags in data blocks. WCHAR and WSTRING data types are available. (Page 1936) Global constants cannot be used as ARRAY limits. New VARIANT instructions for creating generic user programs are available in all languages. (Page 260) The following instructions are also available for the S7-1200 CPU. As an example, the instructions in LAD are listed here: • EQ_Type (Page 2292) • NE_Type (Page 2294) • EQ_ElemType (Page 2295) • NE_ElemType (Page 2296) • IS_NULL (Page 2297) • NOT_NULL (Page 2298) • IS_ARRAY (Page 2299) • MOVE_BLK_VARIANT (Page 2346) • VariantGet (Page 2367) • VariantPut (Page 2368) • CountOfElements (Page 2369) Specifically in STL and SCL, the following VARIANT instructions are also available for the S7-1200 CPU: • VARIANT_TO_DB_ANY (Page 2890) • DB_ANY_TO_VARIANT (Page 2892) New extended instructions: • GetInstancePath (Page 3060) • GetSymbolPath (Page 3057) • ReconfigIOSystem (Page 3113) PLC data types (UDT) can be expanded in the PLC tag table A search is available in the "Instructions" task card. During a block call, the called block can be replaced with another block.
Using technology functions	A new PID controller "PID_Temp" for temperature processes is available. High-precision input/output with time-based IO is available. Position-controlled axis is available.
Using online and diagnostics functions	Scalable options for online backup of devices are available. (Page 7597) Preferred interfaces for the online connection can be saved as default in the settings. To establish an online connection to devices in an external subnet, you can assign an alternative IP address to the device.

Topic	New features in the TIA Portal
Using Team Engineering	Developing function extensions for HMI and PLC in parallel with Inter Project Engineering: • MPI is supported • H systems are supported
Support packages	The "Openness" option package with API functions and XML format for the import/export of project data is available for installation in the "Support" folder on the DVD. PLCSIM is also available for the S7-1200 CPU.

See also

Overview of versions (Page 7349)

Readme

4

4.1 Notes on the TIA Portal

4.1.1 General notes

The information in this readme file supersedes statements made in other documents.

Read the following notes carefully because they include important information for installation and use. Read these notes prior to installation.

Display of Asian characters in the TIA Portal

Due to a change in behavior in Microsoft Windows, it may occur that texts are not displayed correctly in the TIA Portal when a Chinese TIA Portal is installed on another Asian operating system (e.g. Korean). To view the texts in TIA Portal correctly, open the Windows Control Panel and select "English" under "Language for non-Unicode programs". Note, however, that this may cause display problems in other programs.

Installing new .Net versions or .Net service packs

- Close the TIA Portal before installing a new .Net version or a new .Net service pack on your programming device/PC.
- Restart the TIA Portal only after successful installation of the new .Net version or the new .Net service pack.

Notes on handling

- If a project in the list of projects last used is located on a network drive that is not connected, you may experience delays when opening the "Project" menu.
- When you insert a CPU, you may need to wait for some time if the project editor is open at the same time. This generally takes longer when you insert the first CPU in a newly created project. To be able to continue working more quickly, you should close the project editor before inserting a CPU.
- The alarm "Application is not responding" may appear in Windows 7 with functions that take a long time to run (loading the CPU for example). If this occurs, wait until the function has correctly finished.
- If you have installed a Microsoft mouse with IntelliPoint, you may find that it superimposes components over the buttons of the title bar. If this is the case, uninstall the IntelliPoint software from Microsoft.
- Enabling the "Virtual Desktop" options with NVIDIA graphics cards can cause problems. In this case, disable the "nView virtual desktop manager" of your NVIDIA graphics driver.

Using the TIA Portal via a remote desktop

In principle, it is possible to use the TIA Portal via a remote desktop connection. During configuration, you should, however, avoid disconnecting the connection to the desktop client. In rare cases, this can lead to the software user interface being blocked.

If you experience this blockage, follow these steps on the desktop client.

1. Open the Windows Task-Manager and close the "rdpclip.exe" process.
2. Type in "rdpclip.exe" in the command prompt to restart the process.

Note that the current content of the clipboard will be lost. You can, however, then continue configuration as usual. To be on the safe side, you should restart the TIA Portal at the next opportunity.

Migration of projects with the TIA Portal

After the migration of hardware configurations and program blocks from earlier automation solutions, first check the functionality of the migrated project before you use it in productive operation.

Working with automatically synchronized network drives

Automatic synchronization after a network interruption can result in current (local) project data being stored as a "backup" on the network drive through user interactions. This could cause outdated project data to be loaded from the network drive when opening the project. For this reason, we do not recommend that you store TIA Portal projects on synchronized network drives.

If, however, you do work on synchronized drives, you can continue working locally in the event of a network interruption. In this case, you must always ensure that the TIA Portal application is closed while data is synchronized. The synchronization itself must be implemented in such a way that the current (local) project data replaces the project data on the network drive.

Entry of decimal places

With certain Windows language settings, it may occur that the entry of values with a comma as decimal place is not recognized (entering "1,23" leads to an error). Instead, use the international format ("1.23").

Information on the TIA Portal in online support

Overview of the most important technical information and solutions for the TIA Portal in the Siemens Industry online support.

Internet link: TIA Portal in Siemens Industry online support (www.siemens.com/industry/onlinesupport/tiaportal)

All information on service and support in the Siemens Industry online support:

Internet link: Service and support in Siemens Industry online support (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=99&lang=en&referer=%2fWW%2f&func=cslib.csinfo2&siteid=csius&extranet=standard&viewreg=WW&groupid=4000002)

Here, you can also subscribe to the newsletter that provides you with latest information relating to your products.

Starting the TIA Portal

When you start the TIA Portal, Windows attempts to update the Certificate Revocation List (CRL) of "windowsupdate.com".

If no Internet access is available and there are multiple DNS servers, a timeout may occur and delay the start of the TIA Portal.

4.1.2 Notes on libraries

Contents

Information that could not be included in the online help and important information about product characteristics.

Comparing library elements

If master copies and types have the same name, the corresponding objects in the project are overwritten when the action "Copy" is used. Please note that this takes place without a prompt. The same behavior occurs when the name of the master copy is different to the name of the type, but an object within the master copy has the same name as the type.

4.1.3 Notes on memory cards

Contents

Information that could not be included in the online help and important information about product characteristics.

Notes on SIMATIC memory cards

The SIMATIC memory cards have been formatted and set up by Siemens for use with S7-1200 and S7-1500 modules. This format must not be overwritten; otherwise, the card will no longer be accepted by the modules. Formatting with Windows tools is therefore not permitted.

Behavior in case of open force job

Note that an active force job is retained even after you have loaded a new project to the SIMATIC memory card. This means you should first delete the active force job before you remove a SIMATIC memory card from the CPU and before you overwrite the card in the PC with a new project. If you use a SIMATIC memory card with unknown content, you should format the SIMATIC memory card before the next download.

Access protection for memory cards in USB card readers

By improving the security mechanisms for online access and engineering of S7-1500 CPUs, the data storage on memory cards has been changed. For this reason, this version of STEP 7 cannot evaluate the passwords of the configured protection level when reading project data from memory cards that is accessed via a USB card reader. The changed behavior affects the memory cards for CPUs of the S7-1200/1500 series. Therefore, use physical safeguards to protect critical project data on memory cards for these devices.

Note

This restriction is not related to online access to devices or the know-how protection of program blocks.

4.1.4 Notes on the hardware configuration

Content

Information that could not be included in the online help and important information about product characteristics.

Subnet addressing for CP 1613 and CP 1623

CP 1613 and CP 1623 are communication modules with microprocessor. To ensure secure management of communication links, these are processed on the module. The protocol stack in your PC is used for diagnostic purposes (SNMP, DCP). To allow both protocol stacks (i.e. CP 1613/23 Firmware and CP 1613/23 NDIS access) access to the same partners, is recommended to place both stacks of a module in the same subnet.

Editing a device IP address

Do not use the address range from 192.168.x.241 to 192.168.x.250 when editing a device IP address. If necessary, this address range is automatically assigned by the system to a programming device. Depending on the subnet mask, this applies also for all network classes.

Drivers for the CP 5512 communications processor

The drivers for the CP 5512 are no longer included in this software version. If you want to use the CP 5512, the following needs to be installed prior to the installation of this software:

http://support.automation.siemens.com/WW/view/78453460 (http://support.automation.siemens.com/WW/view/en/78453460)

Please note that the CP 5512 can be used with a maximum of Windows 7 32-bit. As of Windows 8 or with 64-bit operating systems, the CP 5512 is not compatible.

Note

Since 2009, the functional successor the USB 2.0 module CP 5711 has been available.

4.1.5 Notes on instructions

Contents

Information that could not be included in the online help and important information about product characteristics.

Instructions not valid for all CPU firmware versions

The firmware version of your CPU determines the following:

- Whether a specific instruction is available for this CPU:
  - If you select CPU 1211C DC/DC/DC with firmware version V1.0, for example, the extended instruction "T_COMBINE" will not be available to you. It is grayed out in the "Instructions" task card.
  - However, if you select CPU 1211C DC/DC/DC with firmware version V3.0, the extended instruction "T_COMBINE" will be available to you in versions V1.1 and V1.2.
- Which versions of the instruction are available. You can select the different versions in the "Version" column of the "Instructions" task card.

4.1.6 Notes on online and diagnostics

Contents

Information that could not be included in the online help and important information about product characteristics.

Display of interfaces via online access

If the Ethernet interfaces for online access are not displayed sporadically, install hotfix KB2588507 (for Microsoft Windows) from the Microsoft Support website.

Internet link: http://support.microsoft.com/kb/2588507 (http://support.microsoft.com/kb/2588507/en-us)

Online operation in hibernate mode

We recommend that you do not use the two options "Hibernate" and "Sleep" in online operation; if you do, communication problems could occur. If necessary, adapt the computer's energy options.

4.2 WinCC Comfort/Advanced

4.2.1 Security information

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Any third-party products that may be in use must also be taken into account. For more information about industrial security, visit

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit

http://support.automation.siemens.com (http://support.automation.siemens.com)

Passwords

Various passwords are set by default in WinCC. For security reasons, you should change these passwords.

- On HMI devices with version V11 or V12, the password "100" is preset for the Sm@rtServer and for the integrated Web server. A default password is not preset for HMI devices with version V13.
- For the user "Administrator", the default password is "administrator".

Integrated Web server

It is always possible on a PC to access HTML pages in Runtime, even though the option"HTML pages" is disabled. Setup always installs the standard pages of the Web Server on the PC. Assign an administrator password to prevent unauthorized access to the pages.

Communication via Ethernet

In Ethernet-based communication, end users themselves are responsible for the security of their data network. The proper functioning of the device cannot be guaranteed in all circumstances; targeted attacks, for example, can lead to overload of the device.

Ending Runtime automatically

If automatic transfer is enabled on the HMI device and a transfer is started on the configuration PC, the running project is automatically stopped on the HMI device. The HMI device then switches autonomously to "Transfer" mode. Transfer mode may cause undesired reactions in the system.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode. To block access to the transfer settings and thus avoid unauthorized changes, assign a password in the Control Panel.

Use of SSL 3.0

For security reasons, the use of the protocol SSL 3.0 is not recommended. The use of the protocol SSL 3.0 is disabled by default on Comfort Panels. If you nevertheless wish to activate the use of SSL 3.0, select the option "Use SSL 3.0" in Internet Explorer or in "Start Center > Settings" under "Internet options > Advanced".

For RT Advanced, the use of SSL 3.0 can be disabled in Internet Explorer or in the Control Panel under "Internet Options > Advanced" by deactivating the "Use SSL 3.0" option.

Network settings

The following tables show the network settings of each product which you need in order to analyze the network security and for the configuration of external firewalls:

WinCC Advanced (without simulation)					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
ALM	4410*	TCP	Inbound, Outbound	License service	This service provides the complete functionality for software licenses and is used by both the Automation License Manager as well as all license-related software products.
HMI Load	1033	TCP	Outbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.
HMI Load	2308	TCP	Outbound	HMI Load (RT Advanced)	This service is used to transmit images and configuration data to panels.
* Default port that can be changed by user configuration					

WinCC Simulation for Basic Panels					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
HMI Load	1033	TCP	Inbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.

WinCC Simulation for Basic Panels					
EtherNet/IP	44818	TCP	Outbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
	2222	UDP	Inbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
Modbus TCP	502	TCP	Outbound	Modbus TCP channel	The Modbus TCP protocol is used for connections to Schneider PLCs.
RFC 1006	102	TCP	Outbound	S7 channel	Communication with the S7 controller via Ethernet/PROFINET
Mitsubishi MC	5002	TCP	Outbound	Mitsubishi MC channel	The Mitsubishi protocol is used for connections to Mitsubishi PLCs.

WinCC Simulation for Panels and Runtime Advanced					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
DCP	---	Ethernet	Outbound	PROFINET	The DCP protocol (Discovery and basic Configuration Protocol) is used by PROFINET and provides the basic functionality for locating and configuring PROFINET devices.
LLDP	---	Ethernet	Inbound, Outbound	PROFINET	The LLDP protocol (Link Layer Discover Protocol) is used by PROFINET for topology detection.
SMTP	25	TCP	Outbound	SMTP Communication	This service is used by WinCC Runtime Advanced to send e-mails.
HTTP	80*	TCP	Inbound	Sm@rtServer	The Web server is only available when Sm@rtService is activated. The used port may differ depending on automatically selected settings.
RFC 1006	102	TCP	Outbound	S7 channel	Communication with the S7 controller via Ethernet/PROFINET
NTP	123	UDP	Outbound	Time synchronization	The NTP protocol (Network Time Protocol) is used for time synchronization in IP-based networks.
SNMP	161	UDP	Outbound	PROFINET	The SNMP client functionality is used by STEP 7 to read status information from PROFINET devices.
HMI Load	2308	TCP	Outbound	HMI Load (RT Advanced)	This service is used to transmit images and configuration data to panels.
HTTPS	443*	TCP	Inbound	Sm@rtServer	The Web server with HTTPS protocol is only available when Sm@rtService is activated. The used port may differ depending on automatically selected settings.
VNC server	5900*	TCP	Inbound	Sm@rtServer	This service is only available when Sm@rtService is activated.
	5800*	TCP	Inbound	Sm@rtServer	This service is only available when Sm@rtService is activated.
VNC client	5500	TCP	Outbound	Sm@rtServer	This service is only available when Sm@rtService is activated.
* Default port that can be changed by user configuration					

PROFINET protocols for Panels and Runtime Advanced					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
DCP	---	Ethernet	Outbound	Lifelist, PROFINET Discovery and configuration	The DCP protocol (Discovery and basic Configuration Protocol) is used by PROFINET and provides the basic functionality for locating and configuring PROFINET devices.
LLDP	---	Ethernet	Inbound, Outbound	PROFINET Link Layer Discovery protocol	The LLDP protocol (Link Layer Discover Protocol) is used by PROFINET for topology detection.
MRP	---	Ethernet	Outbound	PROFINET medium redundancy	The MRP protocol (Medium redundancy protocol) enables control of redundant transmission paths using a ring topology.
PROFINET IO Data	---	Ethernet	Inbound, Outbound	PROFINET Cyclic IO data transfer	Cyclic data exchange is used by panels for direct keys and LEDs.
NARE	---	Ethernet	Inbound, Outbound	Name Address Resolution	This protocol is used to resolve network names and assign IP addresses.
PROFINET Context Manager	34964	UDP	Inbound, Outbound	PROFINET connection less RPC	The PROFINET Context Manager provides an endpoint mapper in order to establish an application relation (PROFINET AR).

Communication connections for Panels and WinCC Runtime Advanced					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
Telnet	23	TCP	Inbound	Telnet	This service can be used for maintenance.
SMTP	**	TCP	Outbound	SendEMail	This service is used by Windows CE / PC Runtime to send e-mails.
HTTP	80*	TCP	Inbound	Hypertext Transfer Protocol	The HTTP protocol is used for communication with the internal Web server.
RFC 1006	**	TCP	Outbound	S7 channel	Communication with the S7 controller via Ethernet/PROFINET.
HMI Load	102	TCP	Inbound	Transfer	This service is used to transmit images, Runtime, and configuration data to the panel via PN/IE
NTP	**	UDP	Outbound	Time synchronization	The NTP protocol (Network Time Protocol) is used for time synchronization in IP-based networks.
DCOM***	135	TCP	Inbound	OPC server	This service is a component of the Microsoft Windows operating system. Communication via OPC (DA) is based on DCOM. Therefore, this service is required to initialize OPC (DA) connections.

Communication connections for Panels and WinCC Runtime Advanced					
DCOM***	**	TCP	Outbound	OPC server	The communication via OPC (DA) is based on DCOM and uses unspecified ports assigned by the system. This should be taken into consideration when using OPC (DA) and creating rules for the firewall.
NetBIOS over TCP/IP	**	UDP	Outbound	With the use of Remote File Share	Register / log on to a remote server.
NetBIOS over TCP/IP	**	UDP	Outbound	With the use of Remote File Share	Register / log on to a remote server.
SNMP	161	UDP	Inbound	Simple Network Management Protocol	The SNMP client functionality is used by STEP 7 to read status information from PROFINET devices.
HTTPS	443*	TCP	Inbound	Secure Hypertext Transfer Protocol	The HTTP protocol is used for communication with the panel-internal Web server via Secure Socket Layer (SSL).
Modbus TCP	**	TCP	Outbound	Modbus TCP channel	The Modbus TCP protocol is used for connections to Schneider PLCs.
Mitsubishi MC	**	TCP	Outbound	Mitsubishi MC channel	The Mitsubishi protocol is used for connections to Mitsubishi PLCs.
Printing	**	TCP	Outbound	Printing	Printing on the control panel (via Ethernet).
HMI Load	2308	TCP	Inbound	Transfer	This service is used to transmit images and configuration data to panels. On Comfort Panels, this service is replaced by DeviceManager and SCS in V13 and higher. This service is used to transmit configuration data to WinCC Runtime Advanced.
HMI Load	50523	TCP	Inbound	Transfer	This port is used if port 2308 is not available. This service is used to transmit images and configuration data to panels. On Comfort Panels, this service is replaced by DeviceManager and SCS in V13 and higher. This service is used to transmit configuration data to WinCC Runtime Advanced.
ALM	4410*	TCP	Inbound, Outbound	Application License Manager	This service of RT Advanced makes available the complete functionalities for software licenses and is used by the Automation License Manager.
OPC UA	4870*	TCP	Inbound	OPC UA server	This service is required for communication via OPC UA.
HMI Load	5001	TCP	Inbound	Device Manager	This service is used to transmit images and Runtime to panels.
HMI Load	5002	TCP	Inbound	SCS (System Configuration Server)	This service is used to transmit configuration data to panels.
VNC client	5500	TCP	Inbound	Sm@rtServer	Reverse VNC server connection. Receive mode is set for the VNC client.
VNC server	5800*	TCP	Inbound	Sm@rtServer	VNC server connection HTTP
	5900*	TCP	Inbound	Sm@rtServer	VNC server connection

Communication connections for Panels and WinCC Runtime Advanced					
SIMATIC Logon	**	TCP	Outbound	UMAC (User Management to the Access Control)	Register / log on to a remote server.
Allen Bradley Ethernet IP	**	TCP	Outbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
Reserved	49152 ... 65535	TCP/UDP	Outbound		Dynamic port range is used, for example, to connect to the remote file sharing.
* Default port that can be changed by user configuration ** Port is assigned automatically. *** Supported by WinCC Runtime Advanced only.					

4.2.2 News

News about the TIA Portal

You can learn about the latest news about various topic areas of the TIA Portal in Siemens Industry Online Support.

All news concerning WinCC can be found here: News (http://support.automation.siemens.com/WW/view/en/88360672)

4.2.3 Notes on use

Contents

Information that could no longer be included in the online help and important information about product features.

Working with standard user rights

If you are working with standard user rights in Windows 7, "User Account Control (UAC)" must not be disabled. The "User Account Control" is enabled in Windows 7 by default.

For more information on the "User Account Control", refer to the online help for Windows 7.

On-screen keyboard

Once you have opened the TIA Portal, you can no longer call the on-screen keyboard.

To call the on-screen keyboard in Windows, use the following command: “Start > All Programs > Accessories > Ease of Access > On-screen keyboard”.

Copying HMI devices with HMI connections

If you copy an HMI device with HMI connections to a PLC, the HMI connection in the new HMI device is not automatically connected to an existing PLC with the same name. This applies to copying within a project as well as copying across projects.

To access the PLC tag via HMI tag in the new HMI device, you have to complete the configuration of the HMI connection immediately after copying. Proceed as follows:

1. Open the "Devices & Networks" editor.
2. Connect the new HMI device to the desired network.
3. Open the connection table.
4. Select the HMI connection of the new HMI device.
5. Select the desired PLC under "Partner".

If you compile the new HMI device or connect additional PLC tags in between copying the HMI device and completing the connection, there may be some instances in which an additional HMI connection to the same PLC is created. This is especially true if you connect HMI tags with DB array elements.

Device replacement

After an HMI device has been replaced, you should check the appearance of the configured screens. Changing the size of the display may result in changes to the position and appearance of screen objects, e.g. recipe view and alarm view.

Device replacement - communication

If an HMI device is replaced, error messages of the type "... are not supported in the new configuration. and will therefore be removed" may be generated. These alarms refer to configured connections of the device and are triggered, for example, if the HMI devices have a different number of interfaces. These connections are marked red after a device replacement. If you would like to continue to use these connections, you have change the configuration of the connection. Proceed as follows:

1. Open the "Devices and Networks" editor.
2. Click "Network" in the toolbar of the network view.
3. Network the interface of the HMI device with the interface of the CPU.
4. Click in the table area of the network view on the "Connections" table.
5. Select the connection marked red.
6. Enter the new interface under "Properties > General > Interface" in the Inspector window.

Specifying the time of modification in the overview window

The times of modification displayed in the overview window only refer to changes to the object itself. Changes to subordinate objects, e.g. screen objects in a screen, do not cause the time of the last change to the screen to change in the overview window.

HMI device wizard

When you create a device with a color display using the HMI device wizard, the graphics of the navigation buttons may be displayed in black and white. This error only occurs, however, if the new device is created with the same name as a device with a monochrome display which has been deleted in the meantime.

You can avoid this error by always deleting the associated graphics in the Graphics collection whenever you delete a device from the project.

Objects with object references in the project library

Two copying methods can be used in WinCC flexible.

- With "simple copy" a WinCC flexible screen including an IO field, for example is copied. Only the object name of a tag configured on the IO field is copied, as this is a reference.
- With "copy", a screen, an IO field contained there and a tag configured on the IO field together with its properties are copied.

These two methods can also be used for storing an object in a library. Project libraries and the objects contained there are migrated during migration and can be used in WinCC.

In WinCC, however, only one copying method is available. With regard to tags, it functions like "simple copy" in WinCC flexible. With regard to graphics, graphics lists and text lists, it functions like "copy" in WinCC flexible.

If you stored objects with references to tags in a library in WinCC flexible, you must reconfigure the referenced objects when using them in WinCC.

Reports with Asian character sets on Windows CE devices

If Asian characters sets are displayed in an illegible manner in a report on a Windows CE device, you must change the character sets for the objects in the report. it is sufficient to change to a different character set and then reset the original character set.

Installation sequence for Startdrive

When you install Startdrive on a PC, adhere to the following installation sequence:

- Install STEP 7 V13.0.
- Install Startdrive.

Compatibility with V12

Empty projects are installed in the installation directory under ..\Portal V13\SampleProjects to allow the TIA Portal to be opened in compatibility mode:

- V12.0.1.4 project with the name "TIA_Portal_Project_V12.0.1.4.ap12", so that TIA Portal V13 SP1 can be opened in V12 compatibility mode
- V13.0.0.3 project with the name "TIA_Portal_Project_V13.0.0.3.ap13", so that TIA Portal V13 SP1 can be opened in V13 compatibility mode

This project must be copied to a local directory with full access before it can be used. For more information on this, refer to FAQ ID 66027369.

File browser on a Windows 8 PC with touch screen

You can only operate the file browser dialog with a mouse, keyboard or on-screen keyboard (without using the touch function) on a Windows 8 PC with touch screen. We recommend using the file browser dialog of the Windows operating system with the help of a script on a touch screen PC with Windows 8.

Making characters available for HMI tags of the type WString/WChar

In tags of the type WString or WChar, you specify strings that can be displayed with specific fonts.

WinCC Runtime Advanced supports all fonts that are installed on the configuration PC. If you want to use a different font, install it on the configuration PC.

The fonts Courier New and Tahoma are installed by default on the HMI device. Tahoma supports the display of characters in all languages except Chinese, Korean and Japanese.

You can download the following fonts to the HMI device with the additional ProSave options:

- SimSun for the input of strings in Chinese (PRC), Chinese (Singapore), English, German, Italian, French and Spanish.
- Gulim for the input of strings in Korean.
- MS PGothic for the input of strings in Japanese.

If you download any other font to the HMI device, only the characters used in tags are retained during compilation. Unused characters are removed when the project is compiled.

To use fonts that are not installed on an HMI device, provide the characters as configured texts in a screen:

1. Create a new screen.
2. Insert a text box.
3. In the properties of the text box, set a font that contains characters that you wish to use later.
4. Insert the characters of the font that you wish to use later into the text box.

After the project is compiled, you can use the characters configured in the text in HMI tags of the type WString/WChar.

4.2.4 Migration

Contents

Information that could no longer be included in the online help and important information about product features.

Project languages in WinCC

WinCC V13 does not support all project languages that were available in WinCC flexible, such as Arabic. If you receive an empty project as the result of your migration, you may want to check the set editing language. Do not set the project languages that are not supported as editing language in the source project. Proceed as follows:

1. Open the project with WinCC flexible.
2. Change the editing language to English, for example.
3. Save the project.
4. Restart the migration.

Migrating an integrated project with ProTool objects

The "PROTOOL option package(s) missing in STEP 7" error message output during migration of a WinCC flexible project that is integrated in STEP 7 indicates that WinCC flexible 2008 SP3 is installed on your system. Moreover, the project still contains objects that were configured using ProTool. Do not open the project with WinCC flexible 2008 SP3! Proceed as follows to migrate the project:

1. Copy the project to a computer on which WinCC flexible 2008 SP2 and STEP 7 are installed.
2. Open the project in the SIMATIC Manager.
3. Remove all ProTool objects from the project.
4. Execute the "Save as" command in the "File" menu.
5. Enable the option "With reorganization" in the "Save project as" dialog.
6. Click OK.
7. Copy the project back to the original computer.
8. Restart the migration.

Migrating a WinCC V7 project: Border line of rectangles

In a WinCC V7 project, you have configured a rectangle with the settings "Line weight = 1" and "Draw insider border = yes".

You then migrate the WinCC V7 project to WinCC V13. To have the rectangle displayed correctly, follow these steps.

1. Open the Inspector window of the rectangle.
2. Open the property list.
3. Disable "Widen border line inwards".

Progress bar

As long as the progress bar still shows a value of 100%, the software is still busy running remaining tasks such as the closing of references. The software will not respond to user input while this status is given.

Open a project created with WinCC V11

When you open a V11 project with a WinCC V13 version, it is no longer be possible to open this project with an older version afterwards.

Managing third-party ActiveX controls

The migration also supports third-party ActiveX controls. However, the controls must be registered in the operating system. If an ActiveX control is not registered, migration is canceled.

If you save a project with the migration tool and perform the migration yourself on another PC, the controls must also be registered on this PC.

Migrating integrated projects with alarm views

An alarm view is enabled with all alarm classes in an integrated project. The alarm classes are disabled during migration of the project.
Once the migration of the project is completed, check the settings in the alarm view.
Enable the require alarm classes in the Inspector window of the alarm view if needed under "Properties > General".

Migrating more extensive projects from WinCC V7

We recommend the use of a 64-bit operating system for the migration of more extensive projects from WinCC V7.

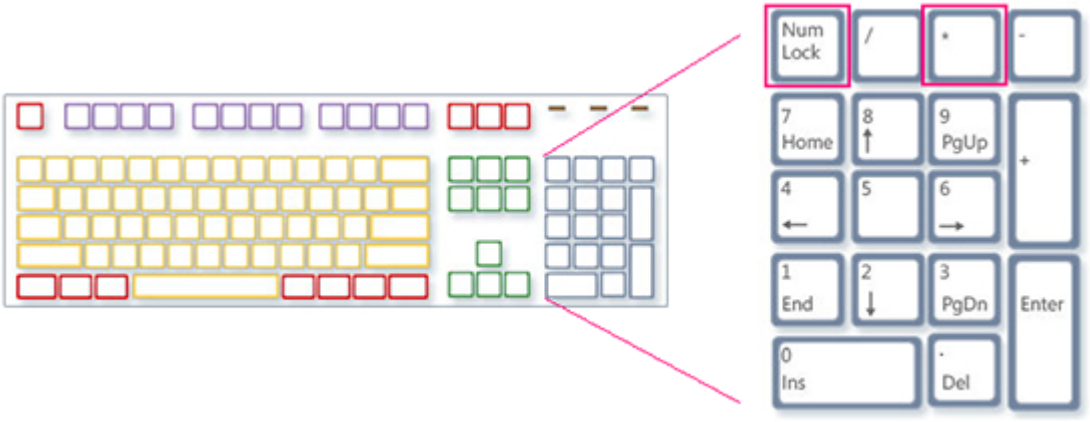
To use the migration tool on a computer with a 64-bit operating system for large projects, start the tool via the command line with the parameter 64-bit, as in the following example:
C:\Program Files (x86)\SIEMENS\Automation\MIGTOOL_V13\Bin
\Siemens.Automation.MigrationApplication.exe 64bit

Migrating projects from WinCC V7

In TIA Portal V13, you can continue to use projects from WinCC V7.2 after migration. Projects from earlier WinCC versions cannot be migrated directly to WinCC TIA Portal version V13. If you wish to continue using such projects in TIA Portal V13, you must first migrate them to the WinCC V7.2 Classic page. To do so, use WinCC V7.2 with the latest update.

Migration log

As of TIA Portal V13.0 alarms are sorted into a tree structure in the migration log. This means all alarms that are part of a specific subsystem are stored in one folder. The result is an increase in the number of folders in the tree structure. Press the shortcut <NUM+ X> to expand the currently selected folder and all subfolders in one single step.



Restrictions for user-specific project data

1. Folders and files you have created in the WinCC V7.2 project directory are not copied to the new project directory during the migration. You must adapt all scripts that access such directories and files after the migration.
2. C standard functions of WinCC V7 are not migrated. If you have made changes to the C standard functions in the WinCC V7 project, you must apply these changes manually to the TIA Portal after the migration.

Migration of texts in Spanish (international sorting) and Spanish (traditional sorting)

If the WinCC V7 project includes texts in Spanish (traditional sorting), these texts are migrated as Spanish (Spain) in WinCC V13.

If the WinCC V7 project includes texts in Spanish (international sorting), these texts are migrated as Spanish (Spain) in WinCC V13.

If the WinCC V7 project includes texts in Spanish (international sorting) as well as Spanish (traditional sorting), only the texts from Spanish (traditional sorting) are migrated as Spanish (Spain) in WinCC V13. The texts from Spanish (international sorting) are not taken into consideration.

See also

Object support during migration (Page 172)

Object support during migration

4.2.5 Engineering System

4.2.5.1 Screens and Screen Objects

Contents

Information that could no longer be included in the online help and important information about product features.

Screen objects after HMI device replacement

If you upgrade a device to the new HMI device versions, you should check the screens contained in the project. Because of the new appearance and improved operability, texts of symbolic I/O fields may not be completely legible and may be concealed by operator controls.

Copying display objects between two projects or two devices

In Project_1 configure an alarm window in the Global Screen, for example. You copy the alarm window and paste it in the Global Screen in Project_2.

The enabled alarm classes are partly not enabled in the alarm window after pasting.

This behavior applies to the following display objects:

- Alarm window
- Alarm indicator
- Alarm view

Display of the cross-references in the Inspector window

The Inspector window displays objects used by a screen object in the "About > Cross-reference" tab.

A screen is open and an object selected. You are using an HMI tag at the object as process tag.

The object and the linked HMI tag are displayed in the cross-references. All locations of use of the object and the HMI tags are listed.

If the HMI tag is interconnected with a PLC tag or a DB tag, the locations of use of the interconnected PLC tag or DB tag are displayed.

Event names in case of alarms in the "Info" tab of the Inspector window

In some alarms of the Inspector window the event names in the "Info" tab will deviate from the names in the "Properties" tab.

Name in the "Properties" tab of the Inspector window	Name in the "Info" tab of the Inspector window
Cleared	ClearScreen
Loaded	GenerateScreen
Enable	Activate
Change	Change
When a dialog is opened	ONMODALBEGIN
When a dialog is closed	ONMODALEND
User change	PASSWORD
Screen change	SCREEN
Disable	Deactivate
Press	Press
Outgoing	Going
Incoming	Coming
Limit "high limit error" violated	AboveUpperLimit
Limit "low limit error" violated	BelowLowerLimit
Click	Click
Loop-in alarm	LoopInAlarm
Release	Release
Alarm buffer overflow	OVERFLOW
Acknowledge	Acknowledgement
Runtime stop	Shutdown
Press key	KeyDown
Release key	KeyUp
Toggle ON	SwitchOn
Toggle OFF	SwitchOff
Value change	Change value

Faceplates

Faceplates cannot be rotated or mirrored.

Tab sequence in screens with faceplates

If you have configured a tab sequence in screens with faceplates in WinCC V12 or WinCC V12 SP1, you should check the tab sequence of these screens in WinCC V13. The tab sequence may have been changed in both the screen and the faceplate.

Tag prefix of a screen window in WinCC Runtime Professional

The objects of the "Controls" palette do not support the tag prefix that can be configured for a screen window.

I/O field with "decimal" display format and format pattern without "s" prefix

You have linked a process tag to an I/O field. The I/O field is assigned the "decimal" display format.

You may select a signed or an unsigned display format.

A "Format pattern" setting without "s", e.g., "999" has the following effects:

1. You cannot set negative values using the I/O field in Runtime.
2. If the tag assumes a negative value, the I/O field generates a two's complement and a corrupted positive value is output.

Trend view on Basic Panels

The trend view buttons are not displayed on Basic Panels. You can operate the trend view using the function keys of the HMI device that are assigned corresponding system functions.

Grouping of screen objects

When you group screen objects in WinCC, performance problems can arise in WinCC in the case of large nesting depths.

ActiveX and .NET controls

ActiveX and .NET controls are always positioned in the foreground in runtime.

The configuration of ActiveX and .NET controls on levels is not supported.

Use of bitmaps as icons

In Windows 8 and Windows 8.1, the use of bitmaps with a size of 48x48 pixels and a color depth of 32 bits as icons is not supported.

Border line of rectangles

In a WinCC V7 project, you have configured a rectangle with the settings "Line weight = 1" and "Draw insider border = yes".

You then migrate the WinCC V7 project to WinCC V12. To have the rectangle displayed correctly, follow these steps.

1. Open the Inspector window of the rectangle.
2. Open the property list.
3. Disable "Widen border line inwards".

Border line of “Graphic I/O field" object

In WinCC V13, the dynamization of the border line of the "Graphic I/O field" object in "Two states" mode has no effect during runtime.

Dynamization of instances of a faceplate type in a group

You are using the instance of a faceplate type in an object group. The properties of the instance are also displayed as properties of the group. Any dynamization with tags, scripts or animations of the group is not displayed in Runtime.

Preview in screen window

You use your own designs with shadows for screen objects. The screen objects can be displayed in a screen window.
The shadows of the screen objects are not displayed in the preview of the screen window. The response occurs only in the engineering system. It is displayed correctly in Runtime.

Assigning graphics to a softkey

A graphic can only be assigned to a softkey if the bottom edge of the permanent window does not conceal the area of the softkey graphic.

Renaming a PLC in Runtime

If a PLC is connected to the PLC code viewer in WinCC Professional and Runtime is running, changing the name of the PLC during runtime will trigger a error. Do not change the PLC name, the IP address, or other properties of the HMI connection during runtime.

Panels and RT Advanced with device version V13: many visually different screen objects

The use of screen objects with very many visually different properties (e.g., very many different styles) can reduce the performance of the user interface in Runtime and can increase the amount of available memory space used. Avoid using, for example, very many different corner radii: 0 pixel, 1 pixel, 2 pixels, 3 pixels, etc.

The use of many differently sized “Gauge” objects can have the same effect. Avoid “Gauge” of height 48 pixels, 49 pixels, 51 pixels, etc. Instead, use sizes such as: 50 pixels, 70 pixels, 100 pixels.

Dynamization of grouped objects

For groupings with multiple nestings (group in group, faceplate in group, group in faceplate, etc.), only the events of the outermost group and the innermost objects can be used for dynamization with system functions. System functions that are configured at events of the lower-level group or lower-level faceplate are not executed.

Number of characters in text fields, lists and alarm texts

The number of available characters in the text of a screen object is dynamic and depends on the HMI device and the memory format. Control instructions and formatting are taken into consideration when entering text data and the maximum number of characters is reduced accordingly.

Transparency in WinCC V13

Transparent graphics can be displayed without any problems in Runtime. This is true for all Comfort Panels and WinCC Runtime Advanced with version 13.0.0.0.
To use the transparency in a graphic view or in a graphic I/O field, the "Fill pattern" property must be set to "Transparent" and the property "Use transparent color" must be disabled.

If the property "Use transparent color" is enabled in a device with version 13.0.0.0, the transparency of the graphic is lost and the transparent pixels are displayed in black. We recommend that you check the "Use transparent color" property at the points of use of transparent graphics after upgrading existing projects to device version 13.0.0.0. If the graphics are not displayed correctly, disable this property.

Addressing tags in the "PLC code display" object

The "PLC code display" object only supports symbolic addressing of tags. If the operand is not addressed symbolically, the network with the operands cannot be displayed and an error message is generated.

Characters that can be used in the property name of the screen blocks with RT Professional

Note the following rules when entering property names of the faceplates:

- The name must start with a letter.
- The name can contain alphanumeric characters and the underscore.
- The name may not contain more than 255 characters.
- Do not use any UNICODE characters (for example, Chinese characters).

Playing videos on Comfort Panels

You can play video sequences on Comfort Panels in the "Media Player" screen object. You can find more detailed information on playing videos on the Internet under: http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/62101921) (entry ID 62101921)

Selecting the transfer protocol for the camera view

To display the screens of a network camera in the camera view, define the transfer protocol in the properties of the screen object. Select the TCP or UDP transfer protocol depending on your network and the type of network camera you are using. In most cases, a TCP connection has reliable synchronization with the camera view.

Displaying SIMOTION PLC websites in the HTML browser

If the SIMOTION PLC websites are not displayed correctly in the HTML browser, insert "/basic" after the website URL to display the websites in basic mode.

Using the optional "CamControls" from older TIA Portal versions

If you want to continue using the control "CamControl" from older versions of TIA Portal configurations in TIA Portal V13 SP1 or newer, we highly recommend an update of the TIA Portal project version and HMI device version to V13 SP1.

Basic Panels: Animation of object groups

Basic Panels as of device version V13 SP1 support animation of object groups that consist of individual screen objects. Animation of nested groups is not supported, however.

RT Professional: Copying screen objects with linked user data types

After copying a screen object from a screen that uses a user data type to a screen that does not use a user data type, the tag references in the target screen are shown with the prefix "@NOP::". These tag references do not function during runtime if the target screen is to be shown in a screen window. If you want the tag references to function, interconnect the target screen temporarily to a user data type. The interconnection of the target screen to the user data type can then be deleted afterwards.

RT Professional: User data types linked to screen objects in faceplates

If a faceplate uses an element of a user data type and a faceplate is created from this screen object, all tag references in the faceplate are shown as "@NOP::". The tag references function during runtime despite this, however. For the tag references to be shown correctly, no element of a user data type should be used when creating a faceplate from an existing screen object.

Display differences between the configuration and the display on the HMI device

The display of the text configured in a screen object may be different on the HMI device due to the display configuration. If you are using the options for automatic size adjustment in the configuration, check the display on your HMI device in every language.

If texts that were configured with the "Fit object to contents" option cannot be displayed in full, they are reduced slightly on the HMI device. If this reduction leads to a distorted display of texts, disable the "Fit object to contents" option and expand the text with additional blanks. Optionally, you can increase the width of the object or use a shorter formulation for the text.

RT Professional: S7-GRAPH overview

If the name of the S7-GRAPH data block that is interconnected at an S7-GRAPH overview contains a period, the control does not work during runtime.

Migration and configuration of ActiveX controls

Changes to properties of ActiveX controls are only saved if the control synchronizes the OCXState property correctly in the case of changes.

For controls such as "Microsoft Date and Time Picker Control" that do not synchronize the OCXState property correctly, the changes are discarded after the screen is closed, during compilation or after the migration.

Exporting and importing function keys

Function keys are synchronized during the import. If a function key is created in the global screen and the key is empty in the screen, the corresponding function key will use the global definition in all screens.

If you want to disable the global use of function keys after the import, define empty keys in the screens and import the screen types in the following order: Global screen, templates, screens.

If you want to ensure when exporting the screens that the global definition of a function key is not used by the template or by the global screen, create an empty function key in the screen. Select the required function key in the screen, then enable the "Use global assignment" property and disable it again.

4.2.5.2 Tags and connections

Contents

Information that could not be included in the online help and important information about product features.

Display of deleted array elements at location of use of HMI tags

The locations of use of HMI tags, such as the process value of IO fields, are usually indicated by the tag name. If the element of an array tag is used, then the tag name will be extended by the index of the array element indicated in brackets.

If a used tag is no longer included in the project, then the tag name will still be displayed at the location of use. The field will be displayed with a red background to indicate the missing tag. If a used array element or the array tag is no longer present, then only the index of the array element will be displayed in brackets. The tag name will not be displayed. The field is highlighted in red. You can no longer identify the name of the associated array tag based on the location of use in this instance.

If you do not know which array tag was linked to this location of use, then it may be necessary to link the array element once again.

If a tag or array tag was created based on a reference, then the selected reference will be closed automatically.

If an HMI tag is connected with an array element of a PLC tag and the PLC tag does no longer exist in the project, then the same behavior will take place in the "HMI tags" editor.

Array tags as list entry of multiplex tags

You can use the array tags of the Char data type just like the tags of the String data type.

The use of an array tag of the Char data type as list entry of a multiplex tag in the "HMI tags" editor is not supported.

Multiplexing tags on a Basic Panel

If you multiplex a tag with an external tag on a Basic Panel, the address is read from the PLC in runtime during the first read cycle. The value of the address read is not available until the second read cycle.

Runtime Advanced and Panels: Importing array elements and structure elements

Array tags and structure tags are always imported in full with all elements. The elements of the array tags and structure tags are not filled further during import.

A new tag is created if the name of a tag corresponds to the name of an array or structure element in the import file.

Example:

The import file contains an array tag called "Otto" with 10 array elements. The array elements are then called Otto[1], Otto[2], for example.

If the import file contains a tag named "Otto[1]", the first element of the array tag will not be filled. Instead, a new tag will be created in the engineering system.

Local ID of HMI connections

You cannot edit the "Local ID" value in the HMI connection properties. You need the local ID, for example, for communication by way of AR_SEND. To enable usage of the "Local ID" for communication, proceed as follows:

1. Open the network view in the "Devices & Networks" editor.
2. Click "Connections".
3. Select an S7 connection.
4. Select the "Add new connection" command in the shortcut menu of the PLC.
5. Click on the interface.
6. Specify the "Local ID (hex)".
7. Click "Add" and then "Close".
8. Select "Properties > General" from the partner area of the Inspector window and enter the IP address of the HMI device for the new connection.
9. Configure the necessary raw data tags for communication in the HMI device.

Tags with the DTL data type

Tags that use the "DTL" data type element by element, can only be used as read-only.

Tag names in faceplates

Use of the "." or "@" character in names of tags in faceplate types is not permitted. Do not use these special characters in the tag names in faceplates.

RT Professional: Tags with symbolic addressing and "Char Array" data type

Tags with symbolic addressing and the "Char Array" data type are not released for communication of RT Professional and SIMATIC S7-1200 V3.

Array elements in WinCC

If you have connected an HMI tag with an array from a STEP 7 data block which does not start with a low limit of 0, the array elements are mapped in WinCC to the low limit of 0.

To ensure that you do not have to rethink between the STEP 7 indices and the WinCC indices when accessing the individual array elements, the low limits of arrays should also start at 0 in STEP 7.

Duration of the initialization of historical data

Initialization of the archives on some storage media can take up to 5 minutes. The successful completion of initialization is confirmed by a system message once it has been completed. If there a lack of any storage medium for archiving when Runtime starts, the appearance of the system message can also take up to 5 minutes.

4.2.5.3 Alarm system and alarm displays

Contents

Information that could not be included in the online help and important information about product properties.

Displaying special characters in alarm texts

When configuring alarm texts, a fixed character set is used in the Engineering System. This character set allows you to use numerous special characters in alarm texts.

Language-dependent fonts are used in runtime to display the texts, for example MS PGothic, SimSun. The fonts used in runtime do not support all special characters. As a result, some special characters are not displayed in runtime.

Use of multiplex tags in output boxes with alarm texts

It is also possible to use multiplex tags in the output boxes of alarm texts in the engineering system. During runtime, this leads to an incorrect display of the alarm, because the use of multiplex tags is not supported by the basic panels.

Parameters in user alarms

Contrary to the information in the online help, it is not possible to configure parameters for user alarms.
The menu command "Properties > Properties > Alarm parameters" is not available in the Inspector window.

Tags in alarm texts of Runtime Advanced

Tags of data type WChar or WString cannot be displayed in the alarm view in Runtime Advanced.

Boolean tags in alarm logs

Bool type tags are recognized as 0 and -1 in the alarm log. If you use a text list that is controlled by a tag of the type Bool in an alarm log, then add the entry for the value -1 to the corresponding text list.

Duration of the initialization of historical data

Initialization of the archives on some storage media can take up to 5 minutes. The successful completion of initialization is confirmed by a system message once it has been completed. If there a lack of any storage medium for archiving when Runtime starts, the appearance of the system message can also take up to 5 minutes.

Printing alarms in Runtime

In WinCC, only ASCII characters are supported when printing alarms directly in Runtime.

4.2.5.4 System functions and scripts

Contents

Information that could no longer be included in the online help and important information about product features.

Runtime error on startup on TP 277 and OP 277

At the start of the runtime on a TP 277 or OP 277, the error "Global unknown error with VBScript: 'Expected statement' in script ..." can be reported.

This error is triggered on devices with the Windows CE 3.0 operating system by user functions of the following type:

```
Sub VBFunction_1()
With HmiRuntime.Screens("Screen_1").ScreenItems("Button_1")
    .backcolor = vbred
    .visible = not .visible
End With
End Sub
```

If the error occurs, you need to re-program the user function as follows:

```
Sub VBFunction_1()
    HmiRuntime.Screens("Screen_1").ScreenItems("Button_1").backcolor = vbred
    HmiRuntime.Screens("Screen_1").ScreenItems("Button_1").visible = not .visible
End Sub
```

Hiding warnings for scripts

Enable "Runtime settings > General > Settings > Also start with faulty scripts".

As a result, Runtime is started irrespective of faulty scripts. In addition, script warnings will not be displayed during a complete compilation

Graphics in faceplates

You have added a graphic view in a faceplate and defined the "Graphic" property as the interface of the faceplate. The "Graphic" property can be made dynamic using the interface of the faceplate instance.
Use the following notation to address the graphic property via a script: "..\\..\\Graphic name".

Scripts for service projects

Since no interactive user is usually logged in service projects, C scripts and VB scripts lead to problems in the following cases:

- When the scripts require interaction, for example, operator input.
- When the scripts show message boxes.

There is no common data area for C scripting in service mode. Thus, for example, no global C variables can be exchanged between the "Scheduled tasks" and "Screens".

Find and Replace in local scripts in WinCC Professional

Use the "Find and Replace" function to find and replace texts within editors. It is not possible to use Find and Replace in local scripts.

System functions in local scripts

System functions can be converted into local C scripts or VB scripts.

If a WString type tag is used in a SetTag system function that was converted into a C script, an error message is generated in the Global Script diagnostics window when the function is executed. The message does not affect the execution of the function.

This concerns the following SetTag functions:

- SetTagWithOperatorEvent
- SetTagIndirect
- SetTagByTagIndirect
- SetTagIndirectByTagIndirect
- SetTagByProperty
- SetTagIndirectByProperty

Properties no longer used in Comfort Panels and RT Advanced

In WinCC V13, many new visual properties have been introduced and many existing properties have been harmonized.

In the course of revision of the object properties, some properties that could be used in VB scripts in earlier versions of WinCC are no longer supported for use in WinCC V13. Scripts that use properties that are no longer supported remain valid, but the relevant calls within the scripts have no function. These limitations pertain exclusively to Comfort Panels and RT Advanced in device version V13.0.

The following table shows the properties of screen objects that are no longer being used:

Property	Screen object
BorderBrightColor3D	Slider
BorderColor	Text field, I/O field, symbolic I/O field, date/time field
BorderInnerWidth3D	Slider
BorderOuterWidth3D	Slider
BorderShadeColor3D	Slider
BorderStyle3D	Text field, I/O field, button, symbolic I/O field, graphic I/O field, date/time field, bar, switch
BorderWidth	Text field, symbolic I/O field
CenterColor	Gauge
DialColor	Gauge
EdgeStyle	Text field, I/O field

System function "ChangeConnection"

Contrary to the description in the online help, the system function "ChangeConnection" is released for the following devices:

- SIMATIC S7-300/400
- SIMATIC S7-NC
- SIMOTION

The system function "ChangeConnection" is not released for SIMATIC S7-1200.

System function "SetPropertyByProperty"

If you change the background color of a gauge in a VBScript or CScript file with the "BackgroundColor" property, these changes are applied in Runtime for the dial color of the gauge. To change the background color of a gauge in a script file, use the "FrameColor" property.

Note

The "FrameColor" property is not displayed in the Script Editor via IntelliSense, but is accepted during compilation of the project.

Script parameters in C-scripting

The following labels are already assigned to the "char*" data type in runtime and must not therefore be used in C-scripting as internal tags or script parameters:

- "Object"
- "Property"
- "Tag"
- "Picture".

System function "EstablishPROFIsafe"

Contrary to the description in the online help, the system function "EstablishPROFIsafe" is not available.

RT Advanced: System function "ShiftAndMask"

If you are using the "ShiftAndMask" system function and the device version of the target HMI device is changed after configuration (e.g.,"13.1.0" to "13.0.0" or vice versa), you must check and test the parameters of this system function.

You can use the "Char" and "Word" data types for the "Source tag" and "Target tag" parameters as of the device version 13.1.0.

In the device versions before 13.1.0, these parameters must be assigned other data types:

- Use "SInt" instead of "Char"
- Use "Int" instead of "Word"

Otherwise, undesirable effects may occur, for example, incorrect or unexpected behavior of the configured system functions.

4.2.5.5 Reports

Contents

Information that could not be included in the online help and important information about product features.

Displaying controls in protocols

A project with WinCC V11 SP2 with Update 4 or earlier is upgraded to WinCC V13.

In this process, it may happen that archive data is not displayed in controls for protocols.

The following controls are affected:

- f(t) trend view
- f(x) trend view
- Table view

Remedial measures for f(t) trend view and f(x) trend view

1. In the control, select "Properties > Properties > General > Display > Online".
2. Recompile your project.
3. Load the project onto your HMI device.

Remedial measures for Table view:

1. In the control, select "Properties > Properties > General > Upon opening screen > Start update".
2. Recompile your project.
3. Load the project onto your HMI device.

4.2.5.6 Recipes

Contents

Information that could not be included in the online help and important information about product features.

Arrays in recipe elements

If you have configured both an array as well as the elements of this array for recipe elements of a recipe, the loading of data records aborts with the following error message: "290055: Import of data records aborted with error"

Use either the array or just the array elements for recipe elements of a single recipe.

4.2.5.7 User administration

Contents

Information that could no longer be included in the online help and important information about product features.

Encrypted connection with SIMATIC Logon V1.5 SP3

Secure encrypted communication between SIMATIC Logon and HMI operator devices is supported as of SIMATIC Logon version V1.5 SP3 and the HMI device version V13 SP1 for Comfort Panels, KTP Mobile Panels and RT Advanced.

HMI devices with a version V13 or earlier can only make an non-secure / non-encrypted connection to a SIMATIC Logon server.

Authentication and certificate management in SIMATIC Logon

If you want to use a secure encrypted connection, the first time a connection is made the SIMATIC Logon certificate is compared with the local certificate of the HMI device. Only when the comparison has been successfully performed can a secure encrypted connection be successfully established.

When you make a connection the first time, the compared certificate is stored under "SimaticLogon\rejected". If you trust the certificate of the server, copy the stored certificate into the local certificate store directory.

The certificate store is located in the following path:

- On the PC under "C:\Program Files\Siemens\CoRtHmiRtm\SimaticLogon\certs"
- On HMI panels under "\flash\simatic\SimaticLogon\certs"

Dynamic logon

In Runtime Professional, an automatic log-off time does not affect users that log on via a logon tag.

4.2.5.8 Communication

Contents

Information that could no longer be included in the online help and important information about product features.

Connection interruptions with Mitsubishi PLCs

After multiple connection interruptions, a situation may arise where all the connection resources of the Mitsubishi PLC are in use and the connection can no longer be established. It is recommended to check these connection resources in the PLC program of the PLC and also to enable them again.

Limited number of possible HMI connections

An error message is displayed during compilation of a device indicating that the configuration of the HMI connection in the "Devices & Networks" editor is invalid. The reason may be that the maximum number of possible connections of the HMI device or PLC has been exceeded.

Check the maximum number of available connections. Consult the device manuals of the devices you are using.

Communication by means of routing with S7 300/400

Communication of connection partners in different subnets is possible via routing with the following connections: PROFINET, PROFIBUS, MPI.

Use of PROFINET IO with panel HMI devices

When using PROFINET IO to connect the direct keys and LEDs of HMI devices to the PLC, you can define an offset for the address area of the inputs and outputs during configuration in HW Config.

The following restriction applies when a PROFINET IO-capable S7-400 CPU is used with one of the HMI devices listed below:

The offset for the start of the address area of the inputs must not be greater than the offset for the start of the address area of the outputs.

The restriction applies to the following HMI devices:

- OP 177B
- OP 277
- Mobile Panel 177

For the configuration of the address parameters, open the PLC with the S7-400 CPU in HW Config. Select the HMI device connected via PROFINET IO in the station window of HW Config. A table with the properties of the HMI device is displayed at the bottom of the station window in the detail view. Select the line containing the addresses of the HMI device in the table and open the object properties using the shortcut menu.

Enable the "Addresses" tab in the "Object properties" dialog. Configure the offset for the inputs under "Inputs > Start". Configure the offset for the outputs under "Outputs > Start".

Exceeding value ranges with Mitsubishi MC and Mitsubishi FX

With some data types, the Mitsubishi MC and Mitsubishi FX communication drivers do not check whether the value of a recipe tag exceeds the value range of the PLC tags. The data types affected are:

- 4-bit block
- 12-bit block
- 20-bit block
- 24-bit block
- 28-bit block

Coordination area pointer in an OPC connection

In principle, the coordination area pointer can be used eight times in an OPC connection. If you have configured an OPC connection and you automatically create another OPC connection using "Add", the coordination area pointer is only displayed once in the newly created connection. In this case, you should change the communication driver of the connection. If you then set OPC again as the communication driver, the coordination area pointer can again be used eight times.

RT Advanced communication via Station Manager (SIMATIC NET) with an S7-1200

For communication of a SIMATIC S7-1200 with a PC with WinCC RT Advanced via a router, the following restrictions apply to the PC:

- Windows 7: Only with installed SIMATIC NET 8.1
- Windows XP: Communication via Station Manager (SIMATIC NET) is not supported

These restrictions also apply if you are using WinAC MP or Station Manager. Connections with the help of the Station Manager of Runtime Advanced are always treated as routed connections.

Communication between S7-1200 and a multi panel

Communication between an S7-1200 and a multi panel with the "WinAC MP" option installed via ProSave is not possible. The HMI devices affected are:

- SIMATIC MP 177 6" Touch
- SIMATIC MP 277 8" Touch
- SIMATIC MP 277 8" Key
- SIMATIC MP 277 10" Touch
- SIMATIC MP 277 10" Key
- SIMATIC MP 377 12" Touch

- SIMATIC MP 377 12" Key
- SIMATIC MP 377 15" Touch
- SIMATIC MP 377 19" Touch

HMI connections in WinCC V13

HMI connections to SIMATIC S7-1200 PLCs with firmware versions earlier than V2.0 are not possible in WinCC V13.

Connections via PROFIBUS DP

When a connection between a PLC and an HMI device via PROFIBUS DP is interrupted and then re-established, sporadically all other PROFIBUS DP connections in the communication network are interrupted and re-established.

De-energize the disconnected station before reconnecting it.

"Set the IP suite (address) of the PLC in the Control Panel" with SIMATIC S7-1200 V1

The function "Set the IP suite (address) of the PLC in the Control Panel" has not been approved for the following PLCs:

- SIMATIC S7-1200 V1

Switching a connection

A connection may be interrupted when it is switched from an HMI device to a SIMATIC S7-300/400, to a SIMATIC S7-1500 or to a SIMATIC S7-1200 PLC.

Note the following settings in the SIMATIC S7-1500 or SIMATIC S7-1200 controllers:

- Absolute addressing of tags
- The "Disable PUT-GET communication" option must be selected
- The "Complete protection" protection level may not be set

RT Advanced in the Station Manager

If RT Advanced and WinAC RTX use the same communication processor of a PC, HMI communication with SIMATIC S7-1200 and SIMATIC S7-1500 is not possible.

Raw data communication in redundant projects

Simatic.NET, Named Connections and various communication blocks, such as BSEND/BRCV, for example, can only be used to a limited extent in a redundantly configured PC station because the connection parameters for the redundant partner server cannot be configured.

Non-integrated connection to a SIMATIC S7-1500 software controller

A non-integrated connection between an HMI device and a SIMATIC S7-1500 software controller is not supported in WinCC.

Transfer areas of the operating mode IO device and DP slave of the HMI devices

If you have activated the operating mode "IO device" or "DP slave" for HMI devices, no transfer areas should be added or deleted in the properties of the HMI device. If you have inadvertently deleted or added a transfer area, disconnect and reconnect the controller.

4.2.6 System-wide functions

Contents

Information that could no longer be included in the online help and important information about product features.

Using system diagnostics in the device proxy

To use the system diagnostics function in an IPE device proxy, for example, a system diagnostics view, insert the PLC alarms as content of a device proxy.

Importing and exporting project texts

In WinCC, you only import the previously exported project texts into the same project. Importing into a different project is not supported.

Initialize device proxy with data from a V13 project

You cannot initialize a device proxy with data from a V13 project in a project with the version V13 SP1.
Upgrade the V13 source project to version V13 SP1 to initialize the device proxy in the target project with the data from the source project.

4.2.7 Compiling and loading

Contents

Information that could no longer be included in the online help and important information about product features.

Compiling and loading

If internal errors or warnings occur during compiling, compile the complete project using the command "Compile > Software (rebuild all)" in the shortcut menu of the HMI device.

Before you start productive operation with your project, compile the entire project using the "Compile > Software (rebuild all)" command from the shortcut menu of the HMI device.

If you are using HMI tags that are connected to the control tags in your project, compile all modified blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Settings for update of operating system

If you select the command "Online > HMI device maintenance > Update operating system" from WinCC, you cannot change the settings such as the type of PG/PC interface or baud rate. The settings used during the last download are always used.

To make changes to the settings, open the "Extended download to device" dialog using the "Online > Extended download to device" command and change the settings. When you click the "Load" button the changed settings are saved.

Alternatively, you can perform an update of the operating system with changed settings with ProSave. You start ProSave via the Windows Start menu "Siemens Automation > Options and Tools > HMI Tools > SIMATIC ProSave".

Incorrect installation of ProSave

If you receive an error message during installation of ProSave when loading data to a target device or maintenance of an HMI device, then you cannot remedy this error using the repair function of setup. Remove ProSave via the Control Panel. Then start setup and install the "ProSave" component again.

Checking the address parameters

During compilation of an HMI device in the project tree with the command "Compile > Software" in the shortcut menu, the address parameters of the HMI device, such as the IP address, will not be checked. If you want to ensure that the address parameters are checked as well, you will have to compile the HMI device using the "Compile" button in the "Devices & Networks" editor of the toolbar.

Error message when downloading data to the PLC

A panel and a PLC are connected and communicating with other.

If a tag is accessed while downloading data from the panel to the PLC, an error message is displayed on the panel.

Delayed reaction in the "Extended download to device" dialog

If the settings in the "Extended download to device" dialog for "Type of the PG/PC interface" and "PG/PC interface" do not match the settings on the HMI device, this can result in the application not responding for up to a minute.

Extended download with an S7-1200 and a Comfort Panel

An S7-1200 PLC and a Comfort Panel are located in the same physical network as the PG/PC. You open the "Extended download to device" dialog for the Comfort Panel.

If you enable the option "Show all accessible devices", it may occur that the application stops responding.

OP77A, OP73, TP177A: Loading projects

When loading a project to an HMI device, it can happen that Runtime is not automatically ended, even though "Remote Transfers" is activated in the Panel.

If this happens, stop Runtime and manually set the transfer mode on the HMI device.

Loading a SIMATIC HMI application to a PC station

The following circumstances can lead to an error message during the first load of a SIMATIC PC station:

- A SIMATIC HMI application is configured in a PC station in the project
  - WinCC Runtime Advanced
  - WinCC Runtime Professional
  - WinCC Standby
  - or WinCC Client
- The property "S7RTM is installed" is activated.

Before you load a SIMATIC PC station for the first time, select the configured device HMI_RT (WinCC...) in the project tree. Open the "Extended download to device" dialog and select the appropriate interface and parameter settings. Click "Load".

You then load the PC station as normal.

Project transfer via USB

If you have connected more than one HMI device via USB to your configuration PC, project transfer is only possible to the last connected device.

Opening project files

When you run "HmiIRTm.exe", a dialog opens asking if you want to open the project file (.fwc).

The following options are available to you:

- "Yes": A dialog opens allowing you to select a project file (.fwc).
- "No": The dialog closes.

Tag simulator in WinCC Professional does not start

If the tag simulator does not start in WinCC Professional, it may be because the fm20.dll file has not been installed in the system yet. The fm20.dll file cannot be distributed and must be installed as part of a Microsoft application:
http://support.microsoft.com (http://support.microsoft.com/default.aspx?scid=kb;EN-US;224305)

You have the following two options for installing the fm20.dll file:

- Install an application, such as Microsoft Office 97, on the destination system. The installation routine also installs the fm20.dll file.
- You can also download and install the Microsoft ActiveX Control Pad free of charge. The fm20.dll file is also installed in this case. Additional information on the ActiveX Control Pad is available on the Microsoft Developer Network website:
http://msdn.microsoft.com/en-us/library/ms968493.aspx (http://msdn.microsoft.com/en-us/library/ms968493.aspx)

Error message when PLC is loaded with a newer version of the TIA Portal

PLC code display shows the current program status of PLC programs.

An error message is output if the PLC code display from an older version of WinCC accesses a program on the PLC loaded with a new version. Make sure that the PLC and the HMI are loaded with the same version of the TIA Portal.

Comfort panels as of device version 13.0: Backing up data while loading projects

If the transfer is interrupted for Comfort Panels with a device version 13.0 or higher, WinCC automatically ensures that no data is lost and that existing data is only deleted on the HMI device after complete transmission.

Loading projects to a file

The "Load project to file" functionality is only available in Runtime Advanced devices with version V13 or older.

Displaying characters in transfer alarms

If characters are not displayed correctly in the transfer alarms during transfers to HMI devices with the device version V12 or older, please check the region and language settings in Windows. Set the corresponding language under "Language for non-Unicode programs".

4.2.8 Runtime

4.2.8.1 Notes on operation in Runtime

Contents

Information that could no longer be included in the online help and important information about product features.

Focus in Runtime

If you have configured a low-contrast combination of focus color and border color in a V12 project, the focus may no longer be identifiable after changing the HMI device version in Runtime. Change one of the two colors.

Language behavior - Layout of on-screen keyboard

The layout of the on-screen keyboard is not switched when the runtime language changes.

Tag values exceed the maximum length

You enter a character string in a string tag via an I/O field. If the character string exceeds the configured number of tags, the character string will be shortened to the configured length.

Empty alarm texts

Runtime is running with a project. The project is saved on a network drive.

In the event of interruptions to the network drive connection, Runtime may attempt to load alarm texts from the network drive.

In the event of disconnection, the alarm window or the alarm view remains empty.

To avoid this, copy the project to a local drive before the starting the project in Runtime.

Complete download in Service mode

If you need to perform a "complete download" to the OS in Service mode from the engineering station, Runtime automatically stops and then starts again.

The project is then no longer in Service mode.

In this state, the power supply is interrupted and WinCC Runtime no longer starts automatically on the OS.

Remedy:

1. Switch the project manually to Service mode after you have performed the "complete download".
2. Close the project manually.

3. Enable Service mode.
4. Start Runtime again using the surrogate icon in the taskbar.

Slow reaction of SmartServer

The following programs may start and respond very slowly under Windows 7 and Windows 2008 servers:

- HMI TouchInputPC
- SmartServer: <Ctrl+Alt+Del> shortcut in the logon dialog

The delay is caused by the callback for the Internet certificate validation.

Remedy:

You can find the following files
on the product DVD under:
Support\Windows7\CRL_Check or CD_RT\ Support\Windows7\CRL_Check\:

- DisableCRLCheck_LocalSystem.cmd
- DisableCRLCheck_CurrentUser.cmd

1. Run the "DisableCRLCheck_LocalSystem.cmd" file with administrator rights. Select the command "Run as administrator" from the shortcut menu of the file.
2. Reboot the PC.

If the problem persists, follow these steps:

1. Double-click the file and run the "DisableCRLCheck_CurrentUser.cmd" file with user rights.
2. Reboot the PC.

Note

The callback for the certificate validation is disabled for all users or PCs. To restore the original state, perform the following files:

- RestoreDefaults_LocalSystem.cmd
- RestoreDefaults_CurrentUser.cmd

You can find the files in the following directory of the product DVD:

- Support\Windows7\CRL_Check or CD_RT\Support\Windows7\CRL_Check\

Starting Runtime

Only WinCC Runtime V13 can be started in TIA Portal V13. WinCC Runtime V11.02, V12 or V13 can be simulated in TIA Portal V13.

Avoiding corrupt files during power failure

If a power failure occurs in Windows systems while the WinCC system is active, files may be corrupt or destroyed. Operation with the NTFS file system provides better security.

Secure, continuous operation is only ensured by using an uninterruptible power supply (UPS).

Blocking task switching on Windows 7 and Windows 8.x

To use the "Block task switching" option on a Runtime PC, disable the Aero theme in Windows 7 and Windows 8.x. To disable the Aero theme, right-click on the Desktop and select "Personalize". In the "Personalization" menu, select the designs "Windows Basic" or "Windows - Classic".

4.2.8.2 Notes on operation of panels in Runtime

Contents

Information that could no longer be included in the online help and important information about product features.

License transfer via S7USB

You always need to run WinCC to transfer a license to a panel via S7USB.

Transferring licenses to a panel on 64-bit operating systems

If you are running a 64-bit operating system and the "Edit > Connect target systems > Connect HMI device" menu command is not available in Automation License Manager, open command line input and run the following command with administrator rights:

"%WINDIR%\system32\RegSvr32.exe" "%CommonProgramFiles%\siemens\AlmPanelPlugin\ALMPanelParam.dll"

Using the mouse wheel in Runtime

The use of the mouse wheel in Runtime is not supported on all panels.

Basic Panels, OP73, OP77A and TP177A: Displaying texts in runtime

The default font selected in the "Runtime settings > Languages & font" editor has an effect on the display of texts in runtime.

Text entries may be truncated if you selected an unfavorable font size or style.

This setting possibly has an effect on the following text entries:

- Tooltips
- long alarm text
- text in the dialogs

Basic Panels: Connections to S7-1200 and S7-1500 with Backup/Restore

If you use the "Backup/Restore" function, a maximum of two connections from Basic Panels to the controllers are possible at any given time.

- SIMATIC S7-1200
- SIMATIC S7-1500

Basic Panels: Backup on the memory card of the PLC

Create the backup file "A.psb" on the memory card of the PLC. An error, for example a connection break, occurs when creating the backup.
This will create a corrupt file on the memory card of the PLC. Such a file has "~$" as prefix. Delete the file with the prefix "~$" if you want to save a backup again under the same name "A.psb".

Basic Panels: Panel Data Storage and S7-1500F

The "Panel Data Storage" PDS function cannot be used on Basic Panels in conjunction with S7-1500F when the password for the protection level "Full access incl. fail-safe" is used.

"Panel Data Storage" function on Basic Panels

The "Panel Data Storage" (PDS) function provided by Basic Panels is only supported by SIMATIC S7-1200 as of firmware V4.0 and SIMATIC S7-1500. For the PDS function, the panel must be connected directly with the CPU and must not be connected via the CP.

4.2.8.3 Notes on operation of Runtime Advanced

Contents

Information that could not be included in the online help and important information about product features.

Authorization for starting Runtime

On a computer running the 32-bit version of Windows 7, WinCC Runtime Professional or WinCC Runtime Advanced can only be started when a user is a member of the automatically created group, "Siemens TIA Engineer".

.Net-Controls in Runtime

If you have incorporated a .Net Control in your project as "Specific .Net-Control", you have to copy the files belonging to these controls to the installation directory of WinCC Runtime, e.g. "C:\ProgramFiles\Siemens\Automation\WinCC RT Advanced". Otherwise, the control cannot be loaded in Runtime.

Starting Runtime

Only WinCC Runtime V13 can be started in TIA Portal V13. WinCC Runtime V11.02, V12 or V13 can be simulated in TIA Portal V13.

Disabling automatic checking for software updates

If the Engineering System is installed together with Runtime on a PC, the operator gets notifications above software updates. For the system to run reliably on a multi-user system, the same software version must be installed on all PCs.

It is possible to disable the automatic checking for software updates and to thus improve performance.

To disable the automatic checking for software updates, go to "Settings > General > Software updates" and clear the "Check daily for updates" check box.

4.2.9 HMI devices

4.2.9.1 Notes on HMI devices

Contents

Information that could no longer be included in the online help and important information about product features.

If the PC goes into standby or hibernate mode while the transfer is in progress, the panel status after interruption of the transfer is not defined.

Multi-key operation

Unintentional actions can be triggered by multi-key operation:

- When you are using a key device, you cannot press more than two function keys at the same time.
- When you are using a touch device, a standard PC or a panel PC, you can only press one function key or button at the same time.

TS Adapter with Ethernet interface

If an HMI device is connected via Ethernet and a TS adapter, it can not be reset to factory settings.

Simulation of the Basic Panels

Use an output field in an alarm text to output an external tag. The content of the output field will always be displayed with "0" during simulation.

Simulation with real PLC connection

The access point used by the simulation is independent from the settings of the engineering system and can only be altered in the Control Panel with the "Setting PG/PC Interface" tool. If the PLC connection is terminated right after the start of the simulation with alarm 140001, you should check the access point used by the simulation with "Setting PG/PC Interface".

1. Double-click "Setting PG/PC Interface" in the Control Panel. A dialog opens.
2. Select" "S7ONLINE" in the "Access point of application" field as standard for HMI.
3. Select the interface in the "Interface Parameter Assignment Used" area.
4. Exit the dialog "Setting PG/PC Interface" with OK

Loading of projects without recipe data records

You are using recipes in a project. You transfer the project to a Basic Panel without recipe data records.

You may encounter inconsistencies if you have altered the structure of the recipe in the engineering system and the device already held recipe data records.

Check the consistency of the data records in this case. The device will not issue a note for all structural changes.

Floating point numbers on MP 277, MP 377, TP 177B 4" and CP4

Only floating point numbers in the range from 10^{-293} ... 10^{+307} are displayed correctly on the HMI devices MP 277, MP 377, TP 177B 4" and CP4. If the tag value is outside this range, it is displayed as 0.

Mobile Panels V2

If you use Mobile Panels V2 in a project, it is not possible to open the project with WinCC V11 SP1. This affects projects with the following devices:

- Mobile Panel 277F IWLAN (RFID Tag)
- Mobile Panel 277F IWLAN V2
- Mobile Panel 277 IWLAN V2

"Zone ID/Connection point ID" tag of a Mobile Panel 277 IWLAN V2

The tag used for the "Zone ID/Connection point ID" must be of data type INT for Mobile Panel 277 IWLAN V2 devices. Adapt this data type if necessary when migrating a project.

HMI devices with operating system Windows CE 5.0 or higher

Owing to a modified client-server communication security setting, the time difference between the HMI device (client) and PC (server) must not exceed 1 day. If you back up recipe data from the HMI device on a network drive, for example, make sure that the time is set correctly on the PC (server) and the HMI device (client).

HMI devices with high communication load

S7 Diagnostics should be enabled if a Panel is assigned many connections to PLCs or other HMI devices. Otherwise, you will risk overload on the Panel.

Device replacement in the engineering system

In the engineering system, you replace a device with configured LED keys with a device without LED keys. Runtime start fails after you have transferred the project data to the device.

For this reason, delete the LED key configuration before you replace the device.

Restrictions for the HMI device, MP 377 15" Touch daylight readable

The following functions are not supported in WinCC V12 for the MP 377 15" Touch daylight readable HMI device:

- Option: Sm@rtServer
- System function: SetAndGetBrightness
- Direct keys

Upgrading Basic Panels to WinCC V13

Before you upgrade Basic Panels from version V12 to version V13, transfer the image of the V12 SP2 Update 5 or higher to the devices.
In the "SIMATIC ProSave [OS Update]" dialog, select the setting "Reset to factory settings".

In this way, you always start a functional update of the image.

Affected devices:

- KP300 Basic mono PN
- KP400 Basic color PN
- KTP400 Basic color PN

Connection switch in the Control Panel with Basic Panels

If you use the "Override protected connection information" function, the following restriction applies:

You cannot perform a connection switch in the Control Panel of a Basic Panel from a PLC without a protection level to a PLC with a "Complete protection" level.

2nd generation Basic Panels: Choosing a USB port

If you are not using a USB hub, select the USB port USB_X60.1 as storage path.

KTP400F Mobile

KTP400F Mobile is available as Hardware Support Package (HSP) for version V13 SP1.

Mobile Panels 277F IWLAN (RFID Tag): F_DB_STATES

The block F_DB_STATES is no longer delivered with WinCC V13 SP1. The block F_DB_STATES serves only for data exchange. You can configure a block to replicate the function of the F_DB_STATES, however. Note the description of the F_DB_STATES in the operating instructions of your HMI device or in the information system.

F_FB_KTP_Mobile and F_FB_KTP_RNG

The fail-safe function blocks F_FB_KTP_Mobile and F_FB_KTP_RNG are not contained in the software WinCC V13 SP1. The function blocks will be part of the WinCC V13 SP1 Update for the Mobile Panels 2nd generation.

Mobile Panels 277F IWLAN

Mobile Panels 277F IWLAN are not released for use with CPUs of the type S7-1500F with WinCC V13 SP1.

Mobile Panels 277F IWLAN are not released for use with CPs in connection with CPUs of the type S7-400F with WinCC V13 SP1.

Affected MLFBs: 6AV6645-0EB01-0AX1, 6AV6645-0EC01-0AX1, 6AV6645-0EF01-0AX1, 6AV6645-0GB01-0AX1, 6AV6645-0GC01-0AX1, 6AV6645-0GF01-0AX1.

4.3 STEP 7 Basic

4.3.1 Security information

Upgrades and updates

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit

http://support.automation.siemens.com (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=99&lang=en&referer=%2fWW%2f&func=cslib.csinfo2&siteid=csius&extranet=standard&viewreg=WW&groupid=4000002)

Network settings

The following tables show the network settings of each product you need to analyze the network security and to configure external firewalls:

STEP 7 Basic					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
ALM	4410*	TCP	Inbound/outbound	License service	This service provides the complete functionality for software licenses and is used by both the Automation License Manager as well as all license-related software products.
RFC 1006	102	TCP	Outbound	S7 communication	Communication to the S7 controller via Ethernet/PROFINET for programming and diagnostic purposes.
DCP	---	Ethernet	Outbound	PROFINET	The DCP protocol (Discovery and Basic Configuration Protocol) is used by PROFINET and provides the basic functionality for locating and configuring PROFINET devices.
SNMP	161	UDP	Outbound	PROFINET	The SNMP client functionality is used by STEP 7 to read status information from PROFINET devices.
* Default port that can be changed by user configuration					

WinCC ES Basic (without simulation)					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
ALM	4410*	TCP	Inbound/outbound	License service	This service provides the complete functionality for software licenses and is used by both the Automation License Manager as well as all license-related software products.
HMI Load	1033	TCP	Outbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.
* Default port that can be changed by user configuration					

Simulation RT Basic					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
HMI Load	1033	TCP	Inbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.

Simulation RT Basic					
Ethernet/ IP	44818	TCP	Outbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
	2222	UDP	Inbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
Modbus TCP	502	TCP	Outbound	Modbus TCP channel	The Modbus TCP protocol is used for connections to Schneider PLCs.
RFC 1006	102	TCP	Outbound	S7 channel	Communication to the S7 controller via Ethernet/PROFINET
Mitsubishi MC	5002	TCP	Outbound	Mitsubishi MC channel	The Mitsubishi protocol is used for connections to Mitsubishi PLCs.

4.3.2 Notes on use

Content

Information that could not be included in the online help and important information about product characteristics.

Online operation

The simultaneous online operation of STEP 7 V5.5 or earlier and STEP 7 Basic V13 has not been approved.

Simultaneous online connections on an S7-1200 CPU

It is not possible to establish an online connection from multiple instances of the TIA Portal simultaneously to the same S7-1200 CPU.

Removing/inserting the memory card

After removing or inserting a memory card, always perform a memory reset on the CPU in order to restore the CPU to a functional condition.

Removing and inserting Ethernet modules

If Ethernet modules are removed and re-inserted during operation, you must boot the PC; otherwise, the "Accessible devices" functionality in STEP 7 or NCM PC will not display all devices. While the PC boots, Ethernet modules must be activated.

Loading project data with TIA Portal V12 and V13 (S7-1200)

If you load the project data of an S7-1200 CPU with the TIA Portal V13, you can no longer use TIA Portal V12 to access this data. To do this, first restore the factory settings of the CPU. Read the additional information on this in the online help under "How to reset a CPU to factory settings".

Using project data of distributed IO-Link master modules from TIA Portal V12 in V13

The following procedure applies if you are using distributed IO-Link master modules in TIA Portal V12 that are not GSD devices and were configured with PCT: To continue using your project data from TIA Portal V12.0 in TIA Portal V13.0, you have to export them to PCT before you upgrade the project. After the upgrade, you must import the project data once again using the PCT.

Compatibility

The device configuration and program of an S7-1200 CPU must always be configured with the same STEP 7 version. Usually, the TIA Portal makes sure that no version conflicts occur by outputting appropriate notifications during loading to the device.

This automatic verification is not possible with S7-1200 CPUs with firmware version V1.x. In this case, users themselves must ensure that no version conflicts occur.

See also

TIA-Portal_Link (http://support.automation.siemens.com/WW/view/en/28919804/133000)

4.3.3 Editing devices and networks

4.3.3.1 General information on devices and networks

Contents

Information that could not be included in the online help and important information about product characteristics.

S7 PCT IO-Link

The S7 Port Configuration Tool is available for free download at the following link.

http://support.automation.siemens.com/WW/view/37936752 (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid0=33102519&objaction=csopen)

4.3.3.2 Use of modules on the S7-1200

Content

Information that could not be included in the online help and important information about product characteristics.

Use of modules on the S7-1200

The modules listed below are not supported on the S7-1200.

Family	Module	Order number
S7-300 FMs	SM 338	6ES7 338-4BC01-0AB0
	FM 350-1	6ES7 350-1AH03-0AE0
	FM 350-2	6ES7 350-2AH00-0AE0, 6ES7 350-2AH01-0AE0
	FM 351	6ES7 351-1AH01-0AE0, 6ES7 351-1AH02-0AE0
	FM 352	6ES7 352-1AH02-0AE0
	FM 355 S	6ES7 355-1VH10-0AE0
	FM 355 C	6ES7 355-0VH10-0AE0
	FM 355-2 C	6ES7 355-2CH00-0AE0
	FM 355-2 S	6ES7 355-2SH00-0AE0
S7-300 PtP-CP	CP 340	6ES7 340-1AH02-0AE0, 6ES7 340-1BH02-0AE0, 6ES7 340-1CH02-0AE0
	CP 341	6ES7 341-1AH01-0AE0, 6ES7 341-1AH02-0AE0, 6ES7 341-1BH01-0AE0, 6ES7 341-1BH02-0AE0, 6ES7 341-1CH01-0AE0, 6ES7 341-1CH02-0AE0
Network component	Diagnostics repeater	6ES7 972-0AB01-0XA0
ET 200S	1 Count 24 V	6ES7 138-4DA04-0AB0
	1 Count 5 V	6ES7 138-4DE02-0AB0
	1 Step 5 V	6ES7 138-4DC00-0AB0, 6ES7 138-4DC01-0AB0
	2 pulses	6ES7 138-4DD00-0AB0, 6ES7 138-4DD01-0AB0
	1 SI	6ES7 138-4DF01-0AB0
	1 SI Modbus	6ES7 138-4DF11-0AB0
	1 SSI	6ES7 138-4DB02-0AB0, 6ES7 138-4DB03-0AB0
	1 Pos Universal	6ES7 138-4DL00-0AB0
	SIWAREX	7MH4910-0AA01, 7MH4912-0AA01, 7MH4920-0AA01
ET 200M	SIWAREX	7MH4 900-2AA01, 7MH4 900-3AA01, 7MH4 950-1AA01, 7MH4 950-2AA01

Loading S7-1200 module comments to the PG/PC

In central configurations with S7-1200, comments of modules, submodules and signal boards are not loaded. With CPs/CMs, only the comments of the IE interface or DP interface are loaded. In distributed configurations with ET 200SP or ET 200MP, only the comment of the channels is loaded from the I/O modules.

4.3.3.3 Replacing ET 200S positioning modules

Contents

Information that could not be included in the online help and important information about product characteristics.

Replacing ET 200S positioning modules

This information relates to the positioning module "1 Step 5V" (6ES7 138-4DC00-0AB0) from a project which was created with TIA Portal V11.0. When replacing these modules from the TIA Portal V11.0 with a new version of these modules, the parameter settings are reset to the default values.

This is the case with one of the following procedures:

- Replace the positioning module 6ES7 138-4DC00-0AB0 with its successor module 6ES7 138-4DC01-0AB0 by means of a device exchange.
- Updating the module version using the appropriate button in the device properties in the Inspector window.

4.3.3.4 CP 343-2 on SIMATIC S7 Embedded Controller EC31-RTX

Contents

Information that could not be included in the online help and important information about product characteristics.

CP 343-2 on SIMATIC S7 Embedded Controller EC31-RTX

The module AS-Interface CP 343-2 (article no.: 6GK7 343-2AH01) can be inserted in an expansion rack of the SIMATIC S7 Embedded Controller EC31-RTX (article no.: 6ES7 677-1DDxx-0BB0), but the CP 343-2 cannot be operated with the EC31-RTX.

4.3.3.5 F-CM AS-i Safety ST for ET 200SP

Content

Information that could not be included in the online help and important information about product characteristics.

F-CM AS-i Safety ST (3RK7 136-6SC00-0BC1) from HSP0070

When using an F-CM AS-i Safety ST (3RK7 136-6SC00-0BC1) using HSP0070 in TIA Portal V13.0, after updating to Service Pack 1, the HSP0070 (F-CM AS-i Safety ST) must be updated to version V2.0. This is necessary to allow use of the module in TIA Portal V13 Service Pack 1.

4.3.3.6 S7 routing via IE/PB Link

Content

Information that could not be included in the online help and important information about product characteristics.

S7 routing via IE/PB Link

In the following situations, no routing via the IE/PB Link PN IO is possible:

- S7 routing between two CPUs of the S7-1500
- S7 routing of PG connections to CPUs of the S7-1200/1500
- S7 routing of HMI connections to CPUs of the S7-1200/1500

This behavior relates to the IE/PB Link with firmware version V2.1 (6GK1 411-5AB00).

4.3.3.7 Notes on online and diagnostics

Contents

Information that could not be included in the online help and important information about product characteristics.

Hardware detection followed by online connection

When the "Online > Hardware detection" command is performed for an unspecified CPU, the online configuration is not loaded from the CPU. If you do not load the configuration resulting from the hardware detection to the CPU, the device and network views will always show a difference between the offline and online configurations. It will appear that there are different configurations in the online and diagnostic views, although the MLFBs are identical in the actual CPU and the offline CPU.

4.3.3.8 Network components

S7-1200 telecontrol CPs Load, TeleService, project number, station number

Changing the project number or station number for the entire STEP 7 project

If you change the project number or the station number in the "CP identification" parameter group for a telecontrol CP, this parameter is changed for all CPs in the STEP 7 project.

Copying Security modules

Copying stations with activated Security functions to another project without activated Security functions is forbidden and can lead to serious inconsistencies.

Mobile wireless CPs: Download / TeleService

The following behavior applies to all mobile wireless CPs:

- CP 1242-7 (6GK7 242-7KX30-0XE0)
- CP 1242-7 GPRS V2 (6GK7 242-7KX31-0XE0)

- CP 1243-7 LTE-EU (6GK7 243-7KX30-0XE0)
- CP 1243-7 LTE-USA (6GK7 243-7SX30-0XE0)

Connection resources for TeleService

The TeleService function occupies a connection resource on the engineering station.

The function download to device or upload from device during a TeleService session occupies a second connection resource on the engineering station.

Download to device

Only use the "Download to device" function with the mobile wireless CP via a TeleService connection as follows:

1. Select the CP in STEP 7.
2. Select the "Online" > "Download to device" menu.
3. In the "Extended download" dialog that appears, select the TeleService interface.
4. Download the project data from the "Extended download" dialog.

Upload from device

The "Upload from device" function via a TeleService connection is supported by the mobile wireless CPs with the following TeleService server applications:

- TeleControl Server Basic as of version V3
- TeleService Gateway as of version V3

Security modules

Migration of projects with Ethernet CPs and activated security functions

In STEP 7 V5.5 projects that contain Industrial Ethernet CPs with activated security functions, the security settings are disabled during migration to STEP 7 Professional.

If necessary, follow these steps after the migration:

1. Activate the security functionality.
2. Configure the required security settings.

Migration of IP access protection lists when activating the security functions

An active IP access protection list is converted to firewall rules when security functions are activated. These rules are visible in advanced firewall mode and can be adjusted there. The advanced firewall mode is activated automatically.

Copying Security modules

Copying stations with activated Security functions to another project without activated Security functions is forbidden and can lead to serious inconsistencies.

Security online diagnostics of S7 CPs

Security online diagnostics of a CP with security capability is only possible if the online connection is established directly via the CP. If the online connection was established by STEP 7 via the CPU to the station, with the "Connect online" button on the security diagnostics page "Security" > "Status" of the security CP you can establish a direct connection to the CP to run security online diagnostics. As an alternative, you can also terminate the online connection to the CPU and enter the IP address of the CP in the "Station address" input box under the entry "Online access" in online diagnostics.

IKE mode

When negotiating the key in phase 1 the IKE mode "Main" should be preferred. This mode normally provides a reliable procedure compared with the "Aggressive" mode. One reason for using the "Aggressive" setting is when you have VPN groups with different pre-shared keys since only one pre-shared key is supported with the "Main" setting.

The IKE "Aggressive" mode should not be used in conjunction with certificates. In the IKE "Aggressive" mode, only use pre-shared keys.

The SOFTNET Security Client only supports the IKE "Main" mode.

A security module must not be used in VPN groups that use different IKE modes.

Downloading configuration data to an S7-300/400 via a VPN tunnel

When you download configuration data via the gigabit interface of a CP x43-1 Advanced to an S7-300/S7-400 station, the path via which the download takes place is stored in the project. If the project is then downloaded via a VPN tunnel established between a SCALANCE S module and the CP x43-1 Advanced, the download fails due to the changed path.

To download via the VPN tunnel, follow the steps below:

1. Using the "Go online" button, connect the engineering station to the gigabit interface of the CP x43-1 Advanced.
2. Disconnect the online connection to the CP x43-1 Advanced.
3. Download the project to the station via the gigabit interface of the CP x43-1 Advanced.

VPN tunnel establishment with 1200/1500 CPs capable of VPN

The establishment of VPN tunnel connections by 1200/1500 CPs capable of VPN with the authentication method "pre-shared key" is only possible if the VPN connection partner is also a 1200/1500 CP capable of VPN. The establishment of VPN tunnel connections by a 1200/1500 CP capable of VPN to all other VPN connection partners is only possible with the "certificate" authentication method.

Media converters

When you migrate a project, you need to reconfigure the media converters.

4.3.4 Programming a PLC

4.3.4.1 General notes on PLC programming

Content

Information that could not be included in the online help and important information about product characteristics.

Information about network security

For communications access between the TIA Portal and CPU or between HMI (except for HMI access using "GET/PUT communication") and CPU, there are integrated security functions. These provide greater protection from manipulation and higher access protection. To protect against unauthorized network access to a CPU with standardized communications access such as "GET/PUT", "TSEND/TRCV", "Modbus", "FETCH/WRITE", you should also take suitable additional measures (e.g. cell protection concept).

Renumbering PLC data types

For better performance, the S7 CPUs process PLC data types with numbers. The user is not aware of these numbers, because the system processes the numbers independently. This means number conflicts are automatically resolved. When using know-how protected blocks that use PLC data types, it may be useful to set up your own numbering system (recommended > 5000) because the automatic resolution of numbering conflicts may result in a password prompt (compilation required).

To renumber the default number of a PLC data type, follow these steps:

1. Open the project library in the "Libraries" task card.
2. Drag the compilable PLC data type to the "Types" folder.
   The "Add type" dialog opens.
3. Enter the properties of the new type.
4. Click "OK" to confirm.
5. Right-click the PLC data type.
   The shortcut menu opens.
6. Select "Edit type" in the shortcut menu.
7. Open the "PLC programming" task card once again.
   The extension "in testing" is now added to the name of the PLC data type.
8. Right-click the PLC data type.
   The shortcut menu opens.
9. Select "Properties" in the shortcut menu.
10. Change the number of the PLC data type in the "General" section.
11. Select "Release version" from the shortcut menu of the library.

The PLC data type now has the new number and can be used. The assigned number is retained even if the type of the PLC data type is revoked.

Generating external sources from blocks

When generating external sources from blocks, the changes made directly in the block interface to the default values of PLC data types are not exported to the sources. This means these values are not available when the sources are imported once again. The default values are applied instead. To prevent this loss of data for the modified default values, the changes must be made directly in the PLC data type and not in the block interface. In this case, the changes are also exported when generating external sources.

Indirect indexing of ARRAY components of data type bit string in SCL

In TIA Portal V13, you can address the components of an ARRAY for a CPU of the S7-1200 series in SCL with a tag of the data type BYTE, WORD or DWORD as index in addition to a tag of the integer data type if the IEC check is not set.

Explicit data type conversion in SCL

As of TIA Portal V13, the string is displayed with a leading sign during explicit data type conversion of SINT/INT/DINT/LINT_TO_STRING or WSTRING in SCL and transferred aligned to the left.

The result is an incompatibility with TIA Portal V12 SP1, in which the string was transferred aligned to the right during conversion.

Start value behavior during "Upload from device"

The start values that you have changed with the instruction "WRIT_DBL: Write to data block in the load memory" will be lost during execution of the action "Upload from device".

Online/offline differences in the project tree (S7-1200 FW V2.0 and V2.1)

If you use the instruction "WRIT_DBL: Write to data block in the load memory" to change a data block, the resulting difference between online and offline block is initially not displayed correctly by the symbols in the project tree. The difference is only shown when you terminate an online connection and then go back online.

Loading inconsistent programs to a device

In TIA Portal, it is not possible to download inconsistent programs to a device without a consistency check. During the loading process, all blocks of the program are implicitly checked and are compiled again in the event of inconsistencies. If, however, there are programs on your CPU which were loaded with earlier versions of STEP 7, these programs could demonstrate inconsistencies.

In this case, note the following:

If you load an inconsistent program from a device, you will not be able to load the program unchanged to the device afterwards, because a consistency check always takes place during the loading process and existing inconsistencies are corrected.

Process image of PTO/PWM outputs

Do not use PTO/PWM outputs in the process image (for example, for access in the user program, for online functions or in HMI). The update rate of the process image is much slower than the rate of the signal changes. The display in the process image therefore does not reflect the signal flow.

Monitoring blocks in LAD and FBD

If the start of the current path is outside the visible range, it may not be possible to determine the input value. In this case, the current path is shown grayed out.

Avoid using PLC data types generated by the system in libraries

Some instructions generate their own PLC data types during instancing which are saved in the "PLC data types" project folder. However, you should not use these system-generated PLC data types in any library, because they may be recreated by the system at any time and may result in an unfavorable system behavior.

Using global data blocks in assignments

It is not possible to assign the contents of a global data block to a structurally identical data block, e.g. using a move box.

4.3.4.2 Compatibility of PLC programs from V12 SP1 or V13

Content

Information that could not be included in the online help and important information about product characteristics.

Compatibility

You can continue to use in V13 SP1 all programs that were created with TIA Portal V12 SP1 or V13. However, because improvements were made to the compiler and errors corrected there in V13 SP1, it can occur in rare cases that the program reacts differently after the upgrade or that you have to adjust the program code manually. These cases are described in detail below.

In addition, you have the option of editing the project in compatibility mode. Compatibility mode is available for projects that were created with TIA Portal V12 SP1 and V13.

You can find additional information on the compatibility under "Compatibility of projects".

Instruction "S_CONV: Convert character string"

Der EN/ENO mechanism behaves differently in TIA Portal V13 SP1 than in TIA Portal V13.

In version V13, the ENO enable output returns the signal state "0" in case of error, even if you have deactivated the ENO enable output. If you have switched an additional instruction to the EN enable output, this is then not executed.

In version V13 SP1, the ENO enable output returns the signal state "1" in case of error, if you have deactivated the ENO enable output. If you have switched an additional instruction to the EN enable output, this is then executed as expected.

Instruction "MUX: Multiplex" (SCL)

Up to and including TIA Portal V13, the value of the tag at the input parameter was output unchanged as a function value, even if the K parameter had a negative integer. This behavior has changed in TIA Portal V13 SP1.

In TIA Portal V13 SP1, if you use the BOOL, STRING, DT or DTL data types at the input parameters of the MUX instruction and the K parameter is a negative integer, the value of the tag is changed.

Instruction "DEMUX: Demultiplex" (SCL)

Up to and including TIA Portal V13, no value was output at the OUTELSE output parameter if the value of the K parameter was < 0. In contrast, the value of the IN input parameter was output at the OUTELSE output parameter if the value of the K parameter was > available outputs. This behavior has changed in TIA Portal V13 SP1.

In the TIA Portal V13 SP1, if you specify at the K parameter a value that is outside the available outputs (K < 0 or K > available outputs), then the value of the IN input parameter is output at the OUTELSE output parameter.

Function values (Return)

As of TIA Portal V13 SP1, more stringent syntax rules apply to function call:

It is checked that the function value (Return) is written in any case, even if multiple possible program paths can be run through in the function. Therefore, there is no longer any risk that the function values will accidentally not be written during runtime.

However, you may possibly receive syntax errors in functions during compiling, which it was still possible to compile without errors in V13. In such cases change the program code so that the function valve is written in all possible program paths.

Example:

SCL

```
IF #MyIn1 = #MyIn2 THEN
   #Block_3 := #MyIn1 + 1;
END_IF;
```

In this example, the function value of "Block_3" is not written if the condition of the IF instruction is not fulfilled. The function value then contains an undefined value.

SCL

```
#Block_3 := #MyIn1;
IF #MyIn1 = #MyIn2 THEN
   #Block_3 := #MyIn1 + 1;
END_IF;
```

In this example the function valve of "Block_3" is definitely written in the program, since "MyIn1" is set as default before the IF loop is completed.

Comparing the hardware data types HW_IO and HW_DEVICE

As of TIA Portal V13 SP1, there is a stricter syntax rule that is valid for the comparison of the data types HW_IO and HW_DEVICE:

Up until and including TIA Portal V13 it was possible to compare the data types HW_IO and HW_DEVICE directly with each other.

If you want to compare these data types in TIA Portal V13 SP1 you have to first create a tag of the data type HW_ANY in the section "Temp" for the block interface and then copy the LADDR (from data type HW_DEVICE) to the tag. It is then possible to compare HW_ANY and HW_IO.

EN/ENO mechanism for STRING conversion

Conversion	Description
Strg_TO_Chars: Convert character string to Array of CHAR	The ENO enable output returns the signal state "0" even if you have deactivated the ENO enable output: • For an invalid character at the CHARS parameter • For an invalid ARRAY index at the PCHARS parameter • If the sum of the PCHARS and STRG parameters exceeds the length of the target ARRAY.
Chars_TO_Strg: Convert Array of CHAR to character string	The ENO enable output returns the signal state "0" even if you have deactivated the ENO enable output: • If the sum of the PCHARS and CNT parameters exceeds the length of the source ARRAY.

4.3.4.3 Instructions

Content

Information that could not be included in the online help and important information about product characteristics.

Instruction "TRCV_C: Receive data via Ethernet"

Contrary to the information provided in the online help, the communication connection is terminated immediately, and not after sending data, when the CONT parameter is set to the value "0".

Instruction "T_CONFIG: Configure interface"

The CPU is restarted after you have executed the "Configure interface" instruction in order to change an IP parameter. The CPU goes to STOP mode, a warm restart is carried out and the CPU starts up again (RUN mode). Make sure that the control process is in a secure operating mode after the CPU has been restarted following execution of the "Configure interface" instruction. Uncontrolled operation can result in serious material damage or personal injury due to malfunctions or programming errors, for example. Non-retentive data could be lost.

Parameters ERROR and STATUS

ERROR	STATUS (DW#16#..)	ERR_LOC	Explanation
0	00000000	0	After the instruction has been executed successfully, the STATUS parameter "00000000" does not return any value.

Instruction "DataLogCreate: Create data log"

In addition to the values described in the online help, the RET_VAL parameter can also adopt the following values:

Error code (W#16#...)	Explanation
8B24	Invalid assignment at parameter HEADER (for example, bit memory used as memory area).
8C24	Invalid assignment at parameter DATA (for example, bit memory used as memory area).

Instruction "DPSYC_Fr: Synchronize DP slaves / Freeze inputs"

In addition to the values described in the online help, the RET_VAL parameter can also have the following value:

Error code (W#16#...)	Explanation
80A4	Communication on central PROFIBUS disrupted.

Instruction "Get_IM_Data: Reading identification and maintenance data"

The explanation for the values of the RET_VAL parameter described in the online help has been changed as follows:

Error code (W#16#...)	Explanation
80B1	No valid data available. (The I&M data is either not supported by the (sub)module or you have not configured I&M data for this device.)

Instruction "T_RESET: Resetting the connection"

In addition to the values described in the online help, the STATUS parameter can also have the following value:

Error code (W#16#...)	Explanation
7000	No job processing active.

Instruction "TCON: Establish communication connection"

In addition to the values described in the online help, the STATUS parameter can also have the following value:

Error code (W#16#...)	Explanation
80A1	The specified connection or the port is already being used.

Instruction "RD_ADDR: Determine the IO addresses from the hardware identifier"

Output of the IO addresses with packed addresses of an ET200

With packed addresses of an ET200. the first module of the pack group returns all addresses. For the other modules, the following is output with the PIADDR and PQADDR parameters:

- With PROFINET as address "0".
- With PROFIBUS as address "0". The error code 16#8090 (the hardware identifier of the module at the LADDR parameter is invalid.) is also output.

For the number of bytes of the inputs and outputs (parameters PICOUNT and PQCOUNT), "0" is output in each case.

Instruction "RD_LGADR: Determine the IO addresses from the hardware identifier"

Output of the IO addresses with packed addresses of an ET200

With packed addresses of an ET200. the first module of the pack group returns all addresses. For the other modules, the following is output with the PEADDR and PAADDR parameters:

- With PROFINET as address "0".
- With PROFIBUS as address "0". The error code 16#8090 (the hardware identifier of the module at the LADDR parameter is invalid.) is also output.

For the number of bytes of the inputs and outputs (parameters PECOUNT and PACOUNT), "0" is output in each case.

Notes on the use of "RecipeExport" and "RecipeImport"

- The instructions RecipeExport and RecipeImport are not suitable for applications in which special protection is necessary for the recipe data. For higher protection of the data use the recipe functions of HMI.
- The data block with the recipe data exported with RecipeExport can be located in work or load memory. If the data block is both in work memory as well as in load memory, the values from work memory are used.
- You can use different separators for the separators of the row values in the recipe. Note that if you open the CSV file with the exported recipe data, for example in Excel, commas can be replaced by periods and vice versa. The reason for this is that different languages use different decimal separators. This can have effect when importing CSV files. The following convention applies for RecipeImport; the first detected possible separator is assumed to be the separator for all values of a table row.
  - Example 1: If a semicolon is detected as the first separator, during import it is assumed that the semicolon is used as the separator for all further values. As the decimal separator with a value of the type REAL, a comma is then used.
  - Example 2: If a comma is detected as the first separator, during import it is assumed that the comma is used as the separator for all further values. As the decimal separator with a value of the type REAL, a period is then used.

  If the recipe data record contains a character string, make use that the character string does not contain the separator you are using. With character strings, also make sure that they do not contain any control characters that specify a line break (with ASCII: LF, CR LF, CR).
  Note: Spaces and tabs cannot be used as separators. The lead to error code 80B2 during import.

Notes on the use of data logs

Data logs are stored as a CSV file and can be opened using the Web server, for example in Excel. If you use character strings within a data log, make sure that you do not use any characters in the character strings that are also used as the separator for the row elements of the CSV file.

Instructions STRG_VAL and VAL_STRG

The instructions STRG_VAL and VAL_STRG are also available in the SCL programming language in V13 SP1 in contrast to the restriction in the online help.

Passing parameters to asynchronous executable blocks

Code blocks (FBs/FCs) and data blocks (DBs) can be created with different access types ("standard" and "optimized"). In code blocks, you can call any instructions. Many instructions (e.g. "WRIT_DBL" and "READ_DBL") are executed asynchronously. These blocks cannot be supplied with tags from TEMP because the data must not be changed during execution.

Ensure that you do not use these instructions in programs in which code blocks of different access types are called reciprocally. This could cause the following to occur:

- A structure from a standard data block is directly or indirectly passed to an optimized code block, which forwards this structure directly or indirectly to one of the blocks mentioned above.
- The reverse scenario, whereby a structure from an optimized code block is directly or indirectly passed to a standard data block, which forwards this structure directly or indirectly to one of the blocks mentioned above.

Otherwise a hidden copy of the passed data is created in TEMP, which leads to the asynchronous blocks returning a negative acknowledgment.

4.3.4.4 Testing the user program

Testing with the watch table

Content

Information that could not be included in the online help and important information about product characteristics.

Multiple access to the same CPU

Access to a CPU from a PG/PC is permitted only when a TIA Portal is open. Multiple access to the same CPU is not permitted and can lead to errors.

Loading data blocks during an active control job

Note

Loading changed data blocks during an active control job can result in unforeseen operating states. The control job continues to control the specified address, although the address assignment may have changed in the data block. Complete active control jobs before loading data blocks.

Testing programs converted from older STEP 7 versions

To monitor and test a program converted from an older STEP 7 version, you first need to compile and download the program with the current STEP 7 version.

4.3.5 Inter Project Engineering (IPE)

4.3.5.1 Notes on IPE

Contents

Information that could not be included in the online help and important information about product characteristics.

Comment box of the device proxy

Contrary to the description in the information system for the TIA Portal V13, with the "Inter Project Engineering" functionality, the comment field for a device proxy cannot be edited.

Restricted functionality

In the TIA-Portal V13, the "S7 GRAPH overview" and "PLC code display" functionality cannot be used in conjunction with the IPE device proxy.

Using system diagnostics in the device proxy

To use the "system diagnostics" function in an IPE device proxy, for example, a system diagnostics view, insert the PLC alarms as content of a device proxy.

4.3.6 Technological functions

4.3.6.1 Notes on technological functions

S7-1200 Motion Control - MC_Home

The motion control instruction "MC_Home" has been extended with the absolute value encoder adjustment:

- **Absolute encoder adjustment (relative)**
  MC_Home.Mode = 6
  The current position is shifted by the value of the parameter "MC_Home.Position".
  The calculated absolute value offset is stored retentively in the CPU.
  (<Axis name>.StatusSensor.AbsEncoderOffset)
- **Absolute encoder adjustment (absolute)**
  MC_Home.Mode = 7
  The current position is set to the value of the parameter "MC_Home.Position".
  The calculated absolute value offset is stored retentively in the CPU.
  (<Axis name>.StatusSensor.AbsEncoderOffset)

S7-1200 Motion Control - MC_WriteParam

Motion control instruction "MC_WriteParam" for drive connection via PROFIdrive/analog output:

Contrary to the statements in the online help, you cannot write parameters with "MC_WriteParam" which require a restart of the technology object.

S7-1200 Motion Control - MC_ChangeDynamic

You can only use the motion control instruction "MC_ChangeDynamic" for drive connection via PTO (Pulse Train Output).

S7-1200 Motion Control - Application cycle MC-Servo [OB91]

To avoid disruptions in the program execution on the CPU, set the application cycle depending on the number of used axes as follows:

Application cycle = number of axes x 2 ms

Number of axes	Application cycle
1	2 ms
2	4 ms
4	8 ms
8	16 ms

The SINAMICS G120 drive updates the drive process image every 4 ms. To improve control, set the application cycle of the MC-Servo [OB91] to 4 ms or to a multiple of 4 ms.

S7-1200 Motion Control - Overflow MC-Servo [OB91]

Contrary to the online help, the CPU does not go into STOP at overflow of the MC-Servo [OB91].

If necessary, you can set the CPU to STOP at overflow of MC-Servo [OB91] via a time-error OB (OB80).

S7-1200 Motion Control - Process image partition "OB Servo PIP"

For optimal control of all I/O modules (e.g. hardware limit switches) used by Motion Control, assign them to the process image partition "OB Servo PIP". The assignment causes the I/O modules to be processed simultaneously with the technology object.

A high-speed counter (HSC) used by Motion Control is automatically assigned to the process image partition "OB Servo PIP".

S7-1200 Motion Control - Renaming technology objects

In order to assure the consistency of your project, after you rename the technology objects, download the project to the CPU while it is in STOP. A name change includes when you delete a technology object and create a new technology object with a new name and data block number of the deleted technology object.

S7-1200 Motion Control - ErrorIDs and ErrorInfos

The ErrorInfo 16#003D is also displayed if a drive with an analog drive connection has turned off.

The ErrorInfo 16#002C has been supplemented for the ErrorID 16#820A:

ErrorID	ErrorInfo	Description	Solution
16#820A		**It is not possible to restart the axis**	
	16#0013	The axis is enabled in the user program	Disable the axis with the "MC_Power" instruction; restart again
	16#0027	The axis is currently being operated in "Manual control" (axis control panel)	Exit "Manual control"; restart again
	16#002C	The axis is not disabled.	Disable the axis; restart the command
	16#0047	The technology object is not ready for restart.	Download the project again.
	16#0048	Condition for restart of the technology object is not satisfied.	Disable the technology object.

PtP module with CM 1243-5 PROFIBUS Master

You have to make the following settings in the instance DB for the instructions when using the PtP-Module CM PtP RS232 BA, CM PtP RS422/485 BA, CM PtP RS232 HF, CM PtP RS422/485 HF and CM PtP with a CM 1243-5 PROFIBUS Master with firmware ≤ V1.3.4:

- **Send_P2P**
  max_record_len = 240
- **Modbus_Master**
  Send_P2P.max_record_len = 240
- **Modbus_Slave**
  Send_P2P.max_record_len = 240

Installation 5

5.1 Notes on the installation

Contents

Information that could not be included in the online help and important information about product characteristics.

Automated Installation

A description of automated installation is available on the product DVD in the directory "Documents\Readme\<Language directory>".

Use of the same versions of TIA Portal products during installation

When installing different TIA Portal products, make sure that you use the same versions of service packs and updates for the installation. For example, if you have installed SP1 for STEP 7 V13, you must also install SP1 for WinCC V13. The service packs and updates must be installed for all products at the same time. Do not start the TIA Portal until all products have been upgraded.

You can download the service packs from the Internet under Siemens Industry online support (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=99&lang=en&referer=%2fWW%2f&func=cslib.csinfo2&siteid=csius&extranet=standard&viewreg=WW&groupid=4000002).

Target directory of the installation

Do not use any UNICODE characters (for example, Chinese characters) in the installation path.

Use of antivirus programs

During the installation, read and write access to already installed files is necessary. Some antivirus programs block this access. We therefore recommend that you disable antivirus programs during installation of the TIA Portal and enable them again afterwards.

Compatibility with V12 SP1 and V13

Empty projects from previous versions are installed in the installation directory under .. \<INSTALLDIR>\Sample Projects to allow the current TIA Portal to be opened in compatibility mode. These projects must be copied to a local directory with full access before they can be used. For more information on this, refer to FAQ ID 82169157.

FAQs on the TIA Portal

FAQs on the TIA Portal are available under FAQs (http://support.automation.siemens.com/WW/view/en/28919804/133000).

5.2 System requirements for installation

5.2.1 Notes on licenses

Availability of licenses

The licenses for the products of the TIA Portal are usually supplied on an installation data medium and transferred automatically by the Automation Licence Manager during the installation process of the TIA Portal.

Before you uninstall the TIA Portal, you must transfer and back up the licenses still required. Use the Automation License Manager for this purpose.

Provision of the Automation License Manager

The Automation License Manager is supplied on the installation data medium and is transferred automatically during the installation process.

If you uninstall the TIA Portal, the Automation License Manager remains installed on your system.

Working with the Automation License Manager

The Automation License Manager is a product of Siemens AG, which is used for handling license keys (technical representatives of licenses).

Software products that require license keys for operation, such as the TIA Portal, register the need for license keys automatically with the Automation License Manager . If the Automation License Manager finds a valid license key for this software, the software can be used according to the license usage terms associated with this license key.

Note

For additional information on how to manage your licenses with the Automation License Manager , refer to the documentation supplied with the Automation License Manager .

See also

Notes on the system requirements (Page 113)

Starting installation (Page 137)

Displaying the installed software (Page 142)
Modifying or updating installed products (Page 143)
Repairing installed products (Page 144)
Starting to uninstall (Page 146)
Installation log (Page 136)

5.2.2 Notes on the system requirements

System requirements for individual products

The system requirements may differ depending on the products you want to install. You should therefore check the individual system requirements of your products.

If you want to install several products, make sure that your system meets the demands of the product with the highest requirements.

Displaying PDF files

To be able to read the supplied PDF files, you require a PDF reader that is compatible with PDF 1.7 e.g. Adobe (R) Reader version 9.

Displaying the Welcome Tour

You need the Adobe (R) Flash Player Version 9 or higher to start the Welcome Tour for the TIA Portal.

See also

Notes on licenses (Page 112)
Starting installation (Page 137)
Displaying the installed software (Page 142)
Modifying or updating installed products (Page 143)
Repairing installed products (Page 144)
Starting to uninstall (Page 146)

5.2.3 System requirements STEP 7 Basic

5.2.3.1 Licensing of STEP 7

Introduction

You require a License Key to license the following STEP 7 editions:

- STEP 7 Basic
- STEP 7 Professional

You can install the corresponding License Key when you are installing STEP 7 or transfer it using the Automation License Manager after the installation has been completed.

Licenses for STEP 7

The following licenses with the corresponding license keys are available:

- STEP 7 Basic
- STEP 7 Professional
- STEP 7 Professional Combo

Validity of license keys for older versions of STEP 7

With a valid License Key for V13.x of STEP 7 Professional and STEP 7 Professional Combo, you can also operate older versions of STEP 7 without restrictions. The following table provides more detailed information about this:

Edition	License	Valid for
STEP 7 Basic V13.x	STEP 7 Basic	• STEP 7 Basic V13.x • STEP 7 Basic V12.x • STEP 7 Basic V11.x • STEP 7 Basic V10.5
STEP 7 Professional V13.x	STEP 7 Professional	• STEP 7 Professional V13.x • STEP 7 Professional V12.x • STEP 7 Professional V11.x • STEP 7 Basic V13.x • STEP 7 Basic V12.x • STEP 7 Basic V11.x • STEP 7 Basic V10.5
STEP 7 Professional V13.x	STEP 7 Professional Combo	• STEP 7 Professional V13.x • STEP 7 Professional V12.x • STEP 7 Professional V11.x • STEP 7 Basic V13.x • STEP 7 Basic V12.x • STEP 7 Basic V11.x • STEP 7 Basic V10.5 • STEP 7 V5.5 • STEP 7 V5.4 • STEP 7 Professional 2010 • STEP 7 Professional 2006

Starting without a valid license key

If you start an edition of STEP 7 without a valid License Key , the system alerts you that you are working in non-licensed mode. You have the one-time option of activating a Trial License. However, this license is valid for a limited period only and expires after 21 days.

When the Trial License expires, the following scenarios can occur:

- STEP 7 has never been licensed on the PC in question:
  - Operations requiring a license can no longer be performed in STEP 7.
- STEP 7 was already licensed on the PC in question:
  - A window requiring acknowledgment presents an alert for non-licensed mode every 10 minutes and for every action requiring a license.

License requirements for simulation

You do not require additional licenses when you use the menu command "Online > Simulation" to start the simulation in STEP 7.

If the following conditions are met, the appropriate licenses for the edition of STEP 7 you have installed are also required for the simulation.

- The engineering station is connected to a PLC.
- The connection to the PLC is configured and active.

See also

Handling licenses and license keys (Page 116)

5.2.3.2 Handling licenses and license keys

Introduction

In each case, you require a valid License Key to use STEP 7 Basic and STEP 7 Professional.

Installing license keys

When you install STEP 7 Basic, the License Key is installed automatically during setup. When you install STEP 7 Professional you will be prompted at the end of the setup to transfer the license from the supplied storage medium to your PC.

If you want to install additional License Keys, you have to use the Automation License Manager to do this.

When you install a license, the associated license key is removed from the license key storage location.

Notice

Destruction of license keys by copying

It is not possible to copy a License Key. The copy protection prevents the license keys from being copied. If you attempt to copy a License Key this will be destroyed.

Uninstalling license keys

License keys are always uninstalled using the Automation License Manager. You uninstall a License Key in the following cases:

- When backing up data.
- If you no longer require the license.

You can then use a valid license on another PC or HMI device.

Data backup

When backing up data on the HMI device or creating a backup during device replacement, remove the License Keys on the HMI device. To do this, open the Automation License Manager and back up the uninstalled license key to another storage location.

Notice

Destruction of license keys on PCs

Start by removing all license keys in the following situations:

- Before you format the hard disk.
- Before you compress the hard disk.
- Before you restore the hard disk.
- Before you start an optimization program that moves fixed blocks.
- Before you install a new operating system.

Read the description of Automation License Manager ("Start > Siemens Automation > Documentation"). Observe all warnings and notices.

On PC-based HMI devices and on non-PC-based HMI devices where Automation License Manager is used, the license key storage location may contain multiple license keys. This capability means you can store multiple licenses of the same type at one location. Save all license keys of an HMI device to the same storage location.

Notice

Always keep the original storage location of the license keys.

Invalid license after time zone change

The installed license no longer functions in the following case.

- If you change the time zone on a PC as follows:
  From a time based on a complete hour to a time not based on a complete hour.
  Example: You change the time zone from GMT +3:00 to GMT +3:30.

To avoid this inconvenience, uninstall the license key using the Automation License Manager under the time zone setting that was set when the license key was installed.

This behavior does not apply to the Trial License.

Defective license

A license is defective in the following cases:

- If the license key is no longer accessible at the storage location.
- If the license key disappears during its transfer to the destination drive.

You can use the Automation License Manager to repair the defective license. To do this, use the "Restore" function or the "Restore wizard" of the Automation License Manager. To perform this restore operation you must contact Customer Support.

You can find more detailed information on the Internet: http://support.automation.siemens.com (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=99&lang=en&referer=%2fWW%2f&func=cslib.csinfo2&siteid=csius&extranet=standard&viewreg=WW&groupid=4000002)

See also

Licensing of STEP 7 (Page 114)

5.2.3.3 Software and hardware requirements STEP 7

System requirements for installation

The following table shows the minimum software and hardware requirements for installation of the "SIMATIC STEP 7 Basic" software package:

Hardware/software	Requirement
Processor	Intel® Celeron® Dual Core 2.2 GHz (Ivy/Sandy Bridge)
RAM	4 GB
Available hard disk space	8 GB
Operating systems *	**Windows 7 (32-bit)** • Windows 7 Home Premium SP1 • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 7 (64-bit)** • Windows 7 Home Premium SP1 • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 8.1 (64-bit)** • Windows 8.1 • Windows 8.1 Professional • Windows 8.1 Enterprise **Windows Server (64-bit)** • Windows Server 2012 R2 StdE (full installation)
Screen resolution	15.6" wide screen display (1024 x 768)

* For more detailed information on operating systems, refer to the help on Microsoft Windows or the Microsoft homepage.

Recommended software and hardware requirements

The following table shows the recommended software and hardware for the operation of STEP 7.

Hardware/software	Requirement
Computer	From SIMATIC FIELD PG M4 PREMIUM (or similar PC)
Processor	From Intel® Core™ i5-3320M 3.3 GHz
RAM	8 GB or more
Hard disk	300 GB SSD
Monitor	15.6" Wide Screen Display (1920 x 1080)
Operating systems *	**Windows 7 (64-bit)** • Windows 7 Home Premium SP1 • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 8.1 (64-bit)** • Windows 8.1 • Windows 8.1 Professional • Windows 8.1 Enterprise **Windows Server (64-bit)** • Windows Server 2012 R2 StdE (full installation)

* For more detailed information on operating systems, refer to the help on Microsoft Windows or the Microsoft homepage

Supported virtualization platforms

You can install the "SIMATIC STEP 7 Basic" software package on a virtual machine. For this purpose, use one of the following virtualization platforms in the specified version or a newer version:

- VMware vSphere Hypervisor (ESXi) 5.5 as of Update 2
- VMware Workstation 10
- VMware Player 6.0
- Microsoft Windows Server 2012 R2 Hyper-V

These virtualization platforms can use the following operating systems as host operating system:

- Windows 7 Professional/Ultimate/Enterprise (64-bit)
- Windows Server 2008 R2 (64-bit)
- Windows Server 2012 R2 (64-bit)
- Windows 8.1 Professional/Enterprise (64-bit)

You can use the following host operating systems to install "SIMATIC STEP 7 Basic" within the selected virtualization platform:

- Windows 7 Professional/Ultimate/Enterprise (64-bit)
- Windows 8.1 Professional/Enterprise (64-bit)

Note

- The same hardware requirements apply to the host operating system as for the respective TIA products.
- The plant operator must ensure that sufficient system resources are available for the host operating systems.
- The hardware certified by the manufacturers is recommended for the use of HyperV server and ESXi.
- When you use Microsoft Hyper-V, accessible stations cannot be displayed.

Supported security programs

The following security programs are compatible with "SIMATIC STEP 7 Basic":

- Antivirus programs:
  - Symantec Endpoint Protection 12.1
  - Trend Micro Office Scan Corporate Edition 10.6
  - McAfee VirusScan Enterprise 8.8
  - Kaspersky Anti-Virus 2014
  - Windows Defender (as of Windows 8.1)
  - Safety Guard 360
- Encryption software:
  - Microsoft Bitlocker
- Host-based Intrusion Detection System:
  - McAfee Application Control 6.0

5.2.4 System requirement for WinCC Advanced

5.2.4.1 Software and hardware requirements

Introduction

Specific requirements for the operating system and software configuration must be met for the installation.

Note

WinCC is generally authorized for use in a domain or workgroup.

However, be aware that domain group policies and restrictions of the domain may hinder the installation. In this happens, remove the computer from the domain prior to installing Microsoft Message Queuing, Microsoft SQL Server and WinCC. Log onto the computer in question locally with administrative rights. Then perform the installation. After successful installation, you can enter the WinCC computer back into the domain. If the domain group policies and restrictions of the domain do not impede the installation, the computer need not be removed from the domain during the installation.

Be aware that domain group policies and restrictions of the domain may also hinder operation. If you cannot avoid these restrictions, run the WinCC computer in a workgroup.

Consult with the domain administrator if needed.

Installation requirements

The following table shows the minimum software and hardware requirements that have to be met for the installation of the "SIMATIC WinCC Comfort/Advanced" software package:

Hardware/software	Requirement
Processor type	Intel® Celeron® Dual Core 2.2 GHz (Ivy/Sandy Bridge)
RAM	4 GB
Free hard disk space	8 GB

Hardware/software	Requirement
Operating systems *	**Windows 7 (32 bit)** • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 7 (64 bit)** • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 8 (64 bit)** • Windows 8.1 Professional • Windows 8.1 Enterprise **Windows Server (64 bit)** • Windows Server 2008 R2 Standard Edition SP1 • Windows Server 2012 R2 Standard Edition
Screen resolution	1024 x 768
Network	Ethernet 10 Mbps or faster
Optical drive	DVD-ROM
Software	Microsoft .Net Framework 4.5

* For additional information on the operating systems, refer to the Microsoft Windows help or the Microsoft homepage.

Simultaneously opening multiple instances of WinCC on a configuration PC can also increase the hardware capacity required.

Note

"Aero Glass Style" of Microsoft Windows 7

A powerful graphics card is required for "Aero Glass Style". It requires DirectX9 capabilities and 128 MB of dedicated graphics memory.

The performance of the architecture of the graphics system can significantly influence the performance of WinCC.

Recommended hardware

The following table shows the recommended hardware for the operation of SIMATIC WinCC.

Hardware	Requirement
Computer	SIMATIC FIELD PG M4 PREMIUM or higher (or comparable PC)
Processor	Intel® Core™ i5-3320M 3.3 GHz or higher
RAM	8 GB or more
Hard disk	300 GB SSD
Monitor	15.6" Wide Screen Display (1024 x 768)
Optical drive	DL MULTISTANDARD DVD RW

Supported virtualization platforms

You can install the "SIMATIC WinCC Comfort/Advanced" software package on a virtual machine. To do this, use one of the following virtualization platforms:

- VMware vSphere Hypervisor (ESXi) 5.5 (as from Update 2)
- VMware Workstation 10
- VMware Player 6
- Microsoft Windows Server 2012 R2 Hyper-V

The following operating systems can serve as a host operating system for these virtualization platforms:

- Windows 7 Professional / Ultimate / Enterprise (64-bit)
- Windows 8.1 Professional/Enterprise (64-bit)
- Windows Server 2008 R2 (64-bit)
- Windows Server 2012 R2 (64-bit)

You can use the following guest operating systems to install "SIMATIC WinCC Comfort/ Advanced" on the selected virtualization platform:

- Windows 7 Professional / Ultimate / Enterprise (64-bit)
- Windows 8.1 Professional/Enterprise (64-bit)

Note

- The same hardware requirements apply to the guest operating system as to the respective TIA products.
- The plant operator must provide sufficient system resources for the guest operating systems.
- The hardware certified by the manufacturers is recommended for the use of HyperV server and ESXi.

Supported security programs

The following security programs are compatible with "SIMATIC WinCC Comfort/Advanced".

- Virus scanners:
  - Symantec Endpoint Protection 12.1
  - Trend Micro Office Scan Corporate Edition 10.6
  - McAfee VirusScan Enterprise 8.8
  - Kaspersky Anti-Virus 2014
  - Windows Defender (Windows 8.1 and higher)
- Encryption software:
  - Microsoft Bitlocker
- Host-based Intrusion Detection System:
  - McAfee Application Control 6.0

Installing Microsoft .Net Framework

The software requires .Net Framework 4.5; it is automatically installed and activated.

You will be prompted to install the required .Net Framework version if this cannot be performed by the software package installation. A restart may be necessary after installation of .Net Framework.

Online help for Windows 7 / Windows Server 2008

Windows 7 and Windows Server 2008 no longer support all online help formats by default. These online help formats are used in WinCC in the following cases:

- Calling WinCC Direct Help
- Calling the WinCC Information System from the WinCC editors or via the Direct Help links

The following component is installed during the installation of WinCC to allow you to continue to call the WinCC Direct Help:

- Microsoft Help Engine

You can also call the WinCC Information System under Windows 7 and Windows Server 2008 from the Windows Start menu or from the installation folder.

To call the WinCC Information System from the WinCC editors or via the Direct Help links, some changes have to be made to the operating system. You can find more information on this in the section "More information for advanced users" of Microsoft Support article "917607": http://support.microsoft.com/kb/917607 (http://support.microsoft.com/kb/917607)

Security settings during installation

When you install WinCC V13, security settings are changed in your operating system.

The security settings in question are listed during the installation.

You must confirm the changes to the security settings.

If you make changes to your operating system after the installation, the changes to the security settings made by the installation of the TIA Portal could be changed.

You can restore the changes to the security settings made by the installation of the TIA Portal: "Start > All Programs > Siemens Automation > Security Controller > Restore settings".

SQL instance of WinCC V13

If you have installed a WinCC V11 product and you wish to install WinCC V13, you must uninstall the WinCC instance of SQL Server 2005 before the installation.

A new WinCC SQL 2008 instance is installed with the installation of WinCC V13.

See also

Options for WinCC Engineering and Runtime systems (Page 126)

Licensing of WinCC Engineering System (Page 128)

5.2.4.2 Parallel installation

Parallel installations in TIA Portal V13

You will be prevented from starting the TIA Portal if you perform a non-permitted parallel installation of STEP 7 and WinCC. The following parallel installations are permitted in the TIA Portal:

- STEP 7 V13 and WinCC V13

A dialog opens during installation to inform you of any inconsistencies in your parallel installation. The following parallel installations are permitted:

- WinCC V13 and RT Advanced V13
- WinCC V13 and RT Professional V13

The engineering system and Runtime must always be of the same version after an installation.

Parallel installation of WinCC V13 and other SIMATIC HMI products

Parallel installation of WinCC V13 and versions of WinCC flexible earlier than WinCC flexible 2008 SP1 is not allowed.

Parallel installation of WinCC V13 with versions of WinCC earlier than WinCC V7.0 SP2 is not allowed. Parallel installation of WinCC V13 with WinCC V7.0 SP2 or WinCC V7.0 SP3 is not allowed for:

- WinCC V 13 Professional
- WinCC V 13 Runtime Professional

Parallel use

If the term "Combo" appears in the name or license key of the software after installation, the use of the following products/versions is permitted in accordance with paragraph 1.6 of the General Terms and Conditions (see also setup text):

- With license for "WinCC V13 Comfort Combo": WinCC flexible 2008 Standard
- With license for "WinCC V13 Advanced Combo": WinCC flexible 2008 Advanced

5.2.4.3 Add-ons

Options for WinCC Engineering and Runtime systems

SIMATIC Panels as well as WinCC Runtime Advanced and WinCC Runtime Professional contain all essential functions for operator control and monitoring of machines or plants. Additional add-ons allow you to extend the functionality in some cases to increase the range of available tasks.

Add-ons for Comfort Panels, Mobile Panels, Multi Panels

The following add-ons are available for Comfort Panels, Mobile Panels, and Multi Panels:

- WinCC SmartServer (remote operation)
- WinCC Audit (audit trail and electronic signature for regulated applications)

Note

In contrast to WinCC flexible 2008, functions from the WinCC flexible /Sm@rtService, WinCC flexible /Sm@rtAccess add-ons as well as the WinCC flexible /OPC Server add-on are incorporated into the basic functionality.

Add-ons for WinCC Runtime Advanced

The following add-ons are available for WinCC Runtime Advanced:

- WinCC Recipes (recipe system)
- WinCC Logging (logging of process values and alarms)
- WinCC SmartServer (remote operation)
- WinCC Audit (audit trail for regulated applications)

Note

For use of the WinCC SmartServer and WinCC Audit add-ons, one license each is required.

Note

In contrast to WinCC flexible 2008, functions from the WinCC flexible /Sm@rtService, WinCC flexible /Sm@rtAccess add-ons as well as the WinCC flexible /OPC Server add-on are incorporated into the basic functionality.

Add-ons for WinCC Runtime Professional

WinCC Runtime Professional contains the following components, which you can activate with a corresponding license:

- WinCC Server (supplements WinCC Runtime to include server functionality)
- WinCC Recipes (recipe system, formerly WinCC /UserArchives)
- WinCC Logging (logging system, previously WinCC/TagLogging)

You can select and install the following add-ons in the installation dialog:

- WinCC Client (standard client for building multiple-station systems)
- WinCC WebNavigator (Web-based operator control and monitoring)
- WinCC DataMonitor (display and evaluation of process states and historical data)

Note

To use the add-ons, one license each is required.

Note

In contrast to WinCC V7, functions from the WinCC / OPC Server add-on have been incorporated into the basic functionality. Likewise, the basic functionality includes the Runtime API from WinCC /ODK.

In addition to the Runtime add-ons, you can also expand WinCC Runtime Professional with customized controls. The WinCC ControlDevelopment add-on is required to develop controls.

See also

Software and hardware requirements (Page 121)

Installing and uninstalling options (Page 127)

Installing and uninstalling options

Introduction

Some add-ons are included in standard WinCC installation and are installed automatically.

During the installation of WinCC Engineering System, the following add-ons are available for you to select for installation:

- Simulation for WinCC Runtime Advanced

During the installation of WinCC Runtime Advanced, the following add-ons are available for you to select for installation:

- OPC XML Gateway
- WinCC Audit Viewer

Add-ons for non-PC-based HMI devices

The required Runtime add-ons are installed on the HMI device as follows:

- During the transfer of the project to the HMI device
  Add-ons that are no longer required are removed automatically.
- With ProSave

Procedure - activating add-ons

To activate an add-on for use, install the appropriate license key using the Automation License Manager.

In order to backup the license key of an add-on or use it on a different configuration PC or HMI device, you must uninstall this license key using the Automation License Manager.

Note

When no valid license key is installed on a PC with WinCC Runtime, Runtime runs in Demo mode.

See also

Options for WinCC Engineering and Runtime systems (Page 126)

5.2.4.4 Licenses and Powerpacks

Licensing of WinCC Engineering System

You require a license key for the following:

- WinCC Engineering System, for example, WinCC Professional
- Add-ons for WinCC Engineering System

You can transfer the license key for WinCC simultaneously with the WinCC installation. You transfer licenses for the WinCC add-ons after installation with the Automation License Manager.

Starting without a valid license key

If you start WinCC without a valid license, the system alerts you that you are working in non-licensed mode. You have the one-time option of activating a trial license. The trial license for Engineering editions WinCC Basic, Comfort/Advanced, and Professional expires after 21 days.

When the trial license has expired, the following scenarios may occur:

- WinCC was never licensed on the PC in question.
  - Operations requiring a license can no longer be performed in WinCC.
- WinCC was already licensed on the PC in question.
  - Non-licensed mode is indicated every 10 minutes and at every action requiring a license by a window, which you must acknowledge.

License requirements for simulation

When you want to start simulation in WinCC using the "Online > Simulation> With tag simulator" menu command, you do not need licenses for WinCC Runtime or for add-ons that require a license.

If the following conditions are met, you will need the appropriate licenses for WinCC Runtime and add-ons that require a license even for simulation:

- The engineering station is connected to a PLC.
- The connection to the PLC is configured and active.

You start the simulation with the "Online > Simulation > Start" menu command.

See also

Software and hardware requirements (Page 121)

Licensing WinCC Runtime on PC-based HMI devices (Page 129)

Licensing of HMI devices (Page 132)

Working with license keys (Page 132)

Powerpack (Page 135)

Installing a Powerpack (Page 136)

Licensing WinCC Runtime on PC-based HMI devices

With PC-based HMI devices, you require a license key for the following:

- WinCC Runtime with 128 tags, for example.
- WinCC add-ons

You can install the license key for WinCC Runtime during the installation of WinCC. You transfer licenses for the WinCC add-ons after installation with the Automation License Manager.

Note

Only the Certificate of License for WinCC Runtime V13 authorizes operation of WinCC Runtime V13 SP1.

The appropriate new license is required to license WinCC Runtime.

You can upgrade the runtime licenses of WinCC flexible 2008 and WinCC V7 with an upgrade to WinCC Runtime V13.

Productive operation of the software is only allowed with a valid Certificate of License that matches the version listed on it.

Non-licensed mode

WinCC Runtime and the Runtime add-ons can be used freely without license. Non-licensed mode is indicated every 10 minutes by a window, which you must acknowledge.

See also

Licensing of WinCC Engineering System (Page 128)

Validity of licensing for older versions of WinCC

Validity of license keys for older versions of WinCC

With a valid license key for WinCC Advanced V13.x, you can also operate older versions of WinCC without restrictions.

The following table provides more detailed information about this:

WinCC Engineering System

Edition	License	Validity
WinCC Basic V13.x	WinCC Basic	• WinCC Basic V11.0 SP2 • WinCC Basic V12.0 • WinCC Basic V13.0
WinCC Comfort/Advanced V13.x	WinCC Comfort	• WinCC Basic V11.0 SP2 • WinCC Comfort V10.5 • WinCC Comfort V11.0 • WinCC Basic, Comfort V12.0 • WinCC Basic, Comfort V13.0
	WinCC Comfort Combo	• WinCC flexible Compact, Standard >= 2005 • WinCC Basic V11.0 SP2 • WinCC Comfort V10.5 • WinCC Comfort V11.0 • WinCC Basic, Comfort V12.0 • WinCC Basic, Comfort V13.0
	WinCC Advanced	• WinCC Basic V11.0 SP2 • WinCC Comfort, Advanced V10.5 • WinCC Comfort, Advanced V11.0 • WinCC Basic, Comfort, Advanced V12.0 • WinCC Basic, Comfort, Advanced V13.0
	WinCC Advanced Combo	• WinCC flexible Compact, Standard, Advanced >= 2005 • WinCC Basic V11.0 SP2 • WinCC Comfort, Advanced V10.5 • WinCC Comfort, Advanced V11.0 • WinCC Basic, Comfort, Advanced V12.0 • WinCC Basic, Comfort, Advanced V13.0

WinCC Runtime Advanced

Edition	License	Validity
WinCC RT Advanced V13.x	WinCC RT Advanced (128) WinCC RT Advanced (512) WinCC RT Advanced (2048) WinCC RT Advanced (4096) WinCC RT Advanced (8192)	• WinCC RT Advanced V11.0 • WinCC RT Advanced V12.0 • WinCC RT Advanced V13.0

Validity of licenses for WinCC options

Generally, WinCC options have version-independent license keys. You can use WinCC options V13.x with V11 and V12 license keys and vice versa.

Licensing of HMI devices

Non-PC-based HMI devices are always equipped with Runtime licenses. A license key is not required for runtime operation.

A license may be required for an add-on for non-PC-based HMI devices. The license key of the respective license always activates one add-on for use.

License key

In order to license non-PC-based HMI devices with license keys, the "SIMATIC HMI License Manager Panel Plugin" add-on is needed.

WinCC Setup installs this add-on by default. You open the License Manager Panel Plugin in the Automation License Manager with the "Edit > Connect Target System > Connect HMI Device" menu command.

If WinCC is not installed, an installation of ProSave 7.2 or higher is required.

Note

Further information on license handling can be found in the Automation License Manager help.

Note

Verify that the current version of the operating system (or higher) is installed on the HMI device before you start licensing. If necessary, update the operating system using ProSave.

Non-licensed mode

Runtime add-ons can be used for a short time without restrictions without a valid license. Non-licensed mode is indicated every 10 minutes by a window, which you must acknowledge.

See also

Licensing of WinCC Engineering System (Page 128)

Working with license keys

Introduction

You transfer a license key to the HMI device in the following cases:

- To use the WinCC engineering system
- To operate WinCC Runtime
- To use add-ons for WinCC Runtime on PC-based HMI devices
- To use add-ons on non-PC-based HMI devices

You transfer a license key from the HMI device in the following cases:

- When backing up data
- If you no longer require the license

You can then use this license on another PC or HMI device.

When you transfer a license to an HMI device, the associated license key is removed from the license key storage location.

Note

A license key cannot be copied. The copy protection employed prevents the license keys from being copied.

Data backup

You transfer the license keys from the HMI device for data backup on the HMI device or as a backup for device replacement.

You use the Automation License Manager to back up license keys from an HMI device to the storage area of the license keys.

Notice

Destruction of license keys on non-PC-based HMI devices

License keys transferred as a result of backup/restore operations are destroyed in the case of the following HMI devices:

- 270 series
- 370 series

Carry out the following before beginning restoring:

- Use the Automation License Manager and ProSave to check whether license keys are on the HMI device.
- Transfer the license keys present from the HMI device
  After restoring has been carried out, transfer the license keys back to the HMI device.

Note

Destruction of license keys on PCs

Transfer all license keys to a storage location in the following cases:

- Before you format the hard disk
- Before you compress the hard disk
- Before you restore the hard disk
- Starting an optimization program that moves fixed blocks
- Installing a new operating system

Read the description of Automation License Manager ("Start > Simatic Automation > Documentation"). Observe all warnings and notices.

The license key storage location on PC-based HMI devices and on non-PC-based HMI devices where Automation License Manager is used may contain multiple license keys. This capability means you can store multiple licenses of the same type at one location. For the backup, use a single storage location for all license keys present on an HMI device.

Note

Always keep the original storage location of the license keys.

Invalid license after time zone change

The transferred license no longer functions in the following case.

- If you change the time zone on a WinCC PC as follows:
  - From a time based on a complete hour to a time not based on a complete hour. Example: You change the time zone from GMT +3:00 to GMT +3:30.

To work around this, transfer the license key from the HMI device with a time zone setting that was set when the license key was transferred to the HMI device.

Example:

You transferred the license key with a time zone setting based on a full hour. Then, also transfer the license key to a storage location with a time zone setting based on a full hour.

This behavior does not apply to the trial license.

Defective license

A license is defective in the following cases:

- If the license key is no longer accessible at the storage area.
- If the license key disappears during its transfer to the destination drive.

Note

Resetting of the system date to an earlier time causes all licenses to be become defective.

You can use the Automation License Manager to repair the defective license. Use the "Restore" function or the "Restore Wizard" of the Automation License Manager for this purpose. You must contact Customer Support in order to restore the license. For additional information see: http://support.automation.siemens.com (http://support.automation.siemens.com)

Note

The Runtime software can also be operated without errors if the license is missing or defective. The system alerts you at brief intervals that you are working in non-licensed mode.

Note

If you start the WinCC Engineering System without a valid license key, the system alerts you that you are working in non-licensed mode. You have the one-time option of activating a trial license. The trial license expires after 21 days.

When the trial license has expired, the following scenarios may occur:

- WinCC was never licensed on the PC in question.
  WinCC can no longer be started.
- WinCC was already licensed on the PC in question.
  WinCC can be started. Non-licensed mode is indicated every 10 minutes by a window, which you must acknowledge.

See also

Licensing of WinCC Engineering System (Page 128)

Powerpack

Introduction

The following upgrades are possible with a Powerpack:

- From a smaller to a larger edition of WinCC Engineering System
- To a runtime system with a larger tag volume

A Powerpack also includes a special license that you can use at any time to activate a higher level of WinCC for use.

Engineering system

You own WinCC Comfort, for example. You can use the "SIMATIC WinCC Advanced Powerpack WinCC Comfort -> WinCC Advanced V13" Powerpack to activate WinCC Advanced for use.

Runtime

You own WinCC Runtime Advanced with 128 tags, for example. You use the "SIMATIC WinCC Runtime Advanced Powerpack 128 PowerTags -> 512 PowerTags V13" Powerpack to increase the tag volume from 128 to 512 tags.

You own WinCC Runtime Advanced with 128 tags, for example. You use the "SIMATIC WinCC Runtime Professional Powerpack Runtime Advanced 128 PowerTags -> Runtime Professional 128 PowerTags V13" Powerpack to activate WinCC Runtime Professional with 128 tags for use.

See also

Licensing of WinCC Engineering System (Page 128)

Installing a Powerpack

Introduction

Install a Powerpack by transferring the corresponding license key. The Powerpack license key replaces the license key of an existing installation.

Requirement

- The conditions described in the "System requirements" section must be fulfilled.
- The license key for the license for which you have purchased a Powerpack is available on the PC.

Installing a Powerpack

1. Open the Automation License Manager.
2. Select the license key.
3. Select "License Key > Upgrade" in the menu.

Result

The Powerpack license key replaces the former license key. You cannot reverse this process.

See also

Licensing of WinCC Engineering System (Page 128)

5.3 Installation log

Function of the installation log

The progress during the following installation processes is logged in a file:

- Installing products
- Modifying or updating already installed products
- Repairing an existing installation
- Uninstalling products

If errors occur during the installation process or warnings are issued, these can be evaluated with the help of the log file. You can do this yourself or contact product support.

Installation logs storage location

The log file is the most recent file with the file extension ".log" and the name of which that starts with "SIA".

The location of the log file is stored in the environment variable "%autinstlog%". You can enter this environment variable in the address bar of Windows Explorer to open the folder with the log files. Alternatively, you can navigate to the corresponding directory by entering "CD %autinstlog%" in the command line.

The storage location is dependent on the operating system, e.g. "C:\ProgramData\Siemens\Automation\Logfiles\Setup" in English-language Windows.

Setup_Report (CAB file)

To make it easier to provide Product Support with all necessary files, an archive file that contains the installation log and all other required files is saved in CAB format. This log can be found at "%autinstlog%\Reports\Setup_report.cab". Send this CAB file to Product Support if you need assistance with installation. With this information, Product Support can determine whether the installation was executed properly. CAB files that were generated during earlier installation processes are saved with a date ID in the "Reports" directory.

See also

Notes on licenses (Page 112)

Starting installation (Page 137)

Checking availability of updates and support packages and installing them (Page 139)

Displaying the installed software (Page 142)

Modifying or updating installed products (Page 143)

Repairing installed products (Page 144)

Starting to uninstall (Page 146)

5.4 Starting installation

Introduction

Software packages are installed automatically by the setup program. The setup program starts once the installation medium has been inserted in the drive.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To install the software packages, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the product languages opens.
6. Select the languages for the product user interface, and click the "Next" button.

   Note

   "English" is always installed as the basic product language.

   The dialog for selecting the product configuration opens.
7. Select the products you want to install:
   - If you wish to install the program in a minimal configuration, click on the "Minimal" button.
   - If you wish to install the program in a typical configuration, click on the "Typical" button.
   - If you wish to personally select the products to be installed, click on the "User-defined" button. Then select the check boxes for the products you wish to install.
8. If you want to create a shortcut on the desktop, select the "Create desktop shortcut" check box.
9. Click the "Browse" button if you want to change the target directory for the installation. Note that the length of the installation path must not exceed 89 characters.
10. Click the "Next" button.
    The dialog for the license terms opens.
11. To continue the installation, read and accept all license agreements and click "Next".
    If changes to the security and permission settings are required in order to install the TIA Portal, the security settings dialog opens.
12. To continue the installation, accept the changes to the security and permissions settings, and click the "Next" button.
    The next dialog displays an overview of the installation settings.
13. Check the selected installation settings. If you want to make any changes, click the "Back" button until you reach the point in the dialog where you want to make changes. Once you have completed the desired changes, return to the overview by clicking on "Next".

14. Click the "Install" button.
    Installation is started.

    Note

    If no license key is found during installation, you have the chance to transfer it to your PC. If you skip the license transfer, you can register it later with the Automation License Manager.

    Following installation, you will receive a message indicating whether the installation was successful.

15. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
16. If the computer does not reboot, click "Exit".

Result

The TIA Portal along with the products and licenses you have ordered and the Automation License Manager have been installed on your computer.

See also

Installation log (Page 136)

Notes on the system requirements (Page 113)

Notes on licenses (Page 112)

Displaying the installed software (Page 142)

Modifying or updating installed products (Page 143)

Repairing installed products (Page 144)

Starting to uninstall (Page 146)

5.5 Checking availability of updates and support packages and installing them

By default, the TIA Portal checks automatically if new software updates or support packages are available, for example, Hardware Support Packages (HSPs). The automatic search for updates takes place after each computer restart and then cyclically every 24 hours. You can also deactivate the automatic search at any time or reactivate it. You can also search for updates manually.

If updates are found, you can download and install them.

Note

Updates and support packages from TIA Portal V13 or higher are supported.

Deactivate or activate automatic search for software updates

To deactivate or reactivate the automatic search for software updates, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Software Updates" group in the area navigation.
3. Deselect the "Look for updates daily" check box if you want to deactivate the automatic search for software updates.
4. Select the "Look for updates daily" check box if you want to reactivate the automatic search for software updates.

Manually searching for software updates

If you want to search for software updates manually, follow these steps:

1. Click "Installed software" in the "Help" menu.
   The "Installed software" dialog opens.
2. Click "Look for updates".
   The "TIA Updater" opens and the available updates are displayed.

Or:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Software Updates" group in the area navigation.
3. Click "Search for updates now".
   The "TIA Updater" opens and the available updates are displayed.

Or:

1. Open the TIA Updater with "Start > All Programs > Siemens Automation > Automation Software Updater".
2. Click "Look for updates".
   The available updates are displayed.

Download and install software updates

To download and install available software updates and support packages, follow these steps:

1. If the TIA Updater is not displayed, open it in one of these ways:
   - If the automatic search has found updates, you receive a message in the information area of the Windows taskbar and the icon for the TIA Update is displayed in the information area. Click either the message or the icon to open the TIA Updater.
   - Click "Installed software" in the "Help" menu and then "Look for updates".
   - In the Windows Start menu, click "Start > All Programs > Siemens Automation > Automation Software Updater".
2. Click "Download" in the line of the update or support package that you want to install. The update or support package is downloaded. The associated "Install" button becomes active as soon as the download process has been completed.

   Note

   Please note the following:

   1. You can initiate multiple download processes simultaneously.
   2. You can log off or also shut down the computer while the download process is running. In such cases, the download process continues in the background as soon as you log on again.
   3. In some cases, a link to an external web page is displayed instead of the "Download" button. In this case, download the software update from this site and install it manually.

3. Close the TIA Portal if it is still open.
4. In the TIA Updater, click the "Install" button of the software package you want to install. The installation dialog appears.

   Note

   Please note the following:

   1. It is not possible to install multiple updates simultaneously.
   2. Do not log off during an installation and do not shut down the computer. This avoids inconsistent versions of the software on your computer.

5. Click "Next".
   The selected product is installed.

Alternative procedures for the installation of support packages

Another procedure is available for the installation of a support package. To do this, follow these steps:

1. Click "Support packages" in the "Options" menu of the TIA Portal.
   The "Detailed information" dialog opens. A table lists all support packages from the directory that you selected as the storage location for support packages in the settings.
2. If you want to install a support package that is not in the list, you have the following options:
   - If the support package is already on your computer, you can add it to the list by selecting "Add from file system".
   - If you add a support package from the "Service & Support" page on the Internet, first you download it by selecting "Download from the Internet". Then you can add it from the file system.
3. Select the support package that you want to install.
4. Click "Install".
5. Close and then restart the TIA Portal.

See also

Installation log (Page 136)

5.6 Displaying the installed software

You can find out which software is installed at any time. In addition, you can display more information on the installed software.

Procedure

To display an overview of the software installed, follow these steps:

1. Click "Installed software" in the "Help" menu.
   The "Installed software" dialog opens. You will see the installed software products in the dialog. Expand the entries to see which version is installed in each case.
2. If you would like to display additional information on the installed automation software, click the link on the "Detailed information about installed software" dialog.
   The "Detailed information" dialog opens.
3. Chose the topic you want more information about in the area navigation.

See also

Notes on the system requirements (Page 113)

Notes on licenses (Page 112)

Starting installation (Page 137)

Modifying or updating installed products (Page 143)
Repairing installed products (Page 144)
Starting to uninstall (Page 146)
Installation log (Page 136)

5.7 Modifying or updating installed products

You have the option to modify installed products using the setup program or to update to a new version.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To modify or update installed products, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the installation variant opens.
6. Select the "Modify/Upgrade" option button and click the "Next" button.
   The dialog for selecting the product languages opens.
7. Select the check boxes of the product languages that you want to install. You can remove previously installed product languages by clearing the corresponding check boxes.

   Note

   Note that the product language "English" cannot be removed.

8. Click the "Next" button.
   The dialog for selecting the product configuration opens.
9. Select the check boxes of the components that you want to install. You can remove previously installed components by clearing the corresponding check boxes.

10. Click the "Next" button.

Note

Note that you cannot change the target directory because the existing installation is being modified.

If changes to the security and permission settings are required in order to install the TIA Portal, the security settings dialog opens.

11. To continue the installation, accept the changes to the security and permissions settings, and click the "Next" button.
The next dialog displays an overview of the installation settings.

12. Click the "Modify" button.
This starts the installation of the additional components.

Note

Following installation, you will receive a message indicating whether the existing installation was successfully changed.

13. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".

14. If the computer does not reboot, click "Exit".

Result

The existing installation has been modified on your computer.

See also

Notes on the system requirements (Page 113)

Notes on licenses (Page 112)

Starting installation (Page 137)

Displaying the installed software (Page 142)

Repairing installed products (Page 144)

Starting to uninstall (Page 146)

Installation log (Page 136)

5.8 Repairing installed products

You have the option to repair installed products by completely reinstalling them using the setup program.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To repair installed products, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the installation variant opens.
6. Select the "Repair" option button, and click the "Next" button.
   The next dialog displays an overview of the installation settings.
7. Click the "Repair" button.
   This starts the repair of the existing installation.

   Note

   Following installation, you will receive a message indicating whether the existing installation was successfully repaired.

8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

Result

The installed products have been reinstalled.

See also

Notes on the system requirements (Page 113)

Notes on licenses (Page 112)

Starting installation (Page 137)

Displaying the installed software (Page 142)

Modifying or updating installed products (Page 143)
Starting to uninstall (Page 146)
Installation log (Page 136)

5.9 Starting to uninstall

Introduction

Software packages are removed automatically by the setup program. Once started, the setup program guides you step-by-step through the entire removal procedure.

You have two options for removing:

- Removing selected components via the Control Panel
- Removing a product using the installation medium

Note

The Automation License Manager will not be removed automatically when you remove the software packages, because it is used for the administration of several license keys for products supplied by Siemens AG.

Removing selected components via the Control Panel

To remove selected software packages, follow these steps:

1. Open the Control Panel with "Start > Control Panel".
2. Click "Programs and Features".
   The "Uninstall or change a program" dialog opens.
3. Select the software package to be removed and click the "Uninstall" button.
   The dialog for selecting the setup language opens.
4. Select the language in which you want the dialogs of the setup program to be displayed and click "Next".
   The dialog for selecting the products you want to remove opens.
5. Select the check boxes for the products that you want to remove and click "Next".
   The next dialog displays an overview of the installation settings.
6. Check the list with the products to be removed. If you want to make any changes, click the "Back" button.
7. Click the "Uninstall" button.
   Removal begins.
8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

Removing a product using the installation medium

To remove all software packages, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Select the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click "Read product information" or "Read installation notes".
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the installation variant opens.
6. Select the "Uninstall" option button and click the "Next" button.
   The next dialog displays an overview of the installation settings.
7. Click the "Uninstall" button.
   Removal begins.
8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

See also

Installation log (Page 136)

Notes on the system requirements (Page 113)

Notes on licenses (Page 112)

Starting installation (Page 137)

Displaying the installed software (Page 142)

Modifying or updating installed products (Page 143)

Repairing installed products (Page 144)

5.10 Installing and uninstalling the migration tool

5.10.1 System requirements

System requirements for the migration tool

The following system requirements apply to the use of the migration tool:

- All products used to create the source project must be installed. The following products are supported:
  - STEP 7 V5.5 SP3
  - WinCC V7.2 with the latest updates
  - WinCC flexible 2008 SP3
  - Integrated projects from STEP 7 V5.5 and the WinCC products listed above
  - STEP 7 Distributed Safety V5.4
  - SINUMERIK STARTER and Startdrives
  - SIMOTION SCOUT V4.4
    You need the SCOUT Migration Tool PlugIn V4.4 to migrate SIMOTION SCOUT V4.4 projects.
- All optional packages needed to process the STEP 7 project are installed. For example, all HSPs for the devices used in the source project are required.

5.10.2 Installing the migration tool

Distribution of the migration tool

You can find the migration tool in the "Support" directory on the installation DVD of the TIA Portal. Alternatively, it is available for download from the Siemens Industry Online Support. For some products, (such as SIMATIC Failsafe or SIMOTION) additional plug-ins are required for the migration tool. These plug-ins can also be downloaded from the Siemens Industry Online Support or installed from the installation DVD of the specific products.

Normally, the migration tool is installed without the TIA Portal. Because the TIA Portal has its own integrated migration function, a separate installation of the migration tool is not necessary.

Procedure

To install the migration tool, proceed as follows:

1. Download the installation file from the Siemens Industry Online Support, or use the installation file from the "Support" directory of the installation DVD of the TIA Portal to perform the installation.
2. Run the installation file.
   The setup program for the migration tool will open.
3. First, select the language in which the setup should be displayed and click the "Next" button.
   The page for selecting the software language is displayed.
4. Since the migration tool is provided exclusively in English, you cannot choose any other language for the installation. Therefore, click "Next" to proceed to the next step.
   The page for selecting the product is displayed.
5. The migration tool consists solely of a software component. Therefore, the migration tool is already selected.
   To create a Desktop icon for starting the migration tool, select the check box "Create Desktop icon". Then click the "Next" button.
   The page for confirming the licensing terms is shown.
6. Click on an entry in the list of license terms to read the selected license term. If you agree with all license terms, select the check box "I accept the terms of the displayed license agreement". Then click the "Next" button.
   An overview of the installation is displayed.
7. Click the "Install" button.
   The installation is performed with the displayed settings.

5.10.3 Uninstalling the migration tool

The migration tool can be removed using the Control Panel.

Procedure

To remove the migration tool, follow these steps:

1. Open the Control Panel.
2. Double click on "Add or Remove Programs" in the Control Panel.
   The "Add or Remove Programs" dialog opens.
3. Select the entry for the migration tool in the "Add or Remove Programs" dialog, and click the "Remove" button.
   A security prompt appears.
4. Click the "Uninstall" button to confirm this prompt.
   The migration tool will be removed.

Migrating projects and programs

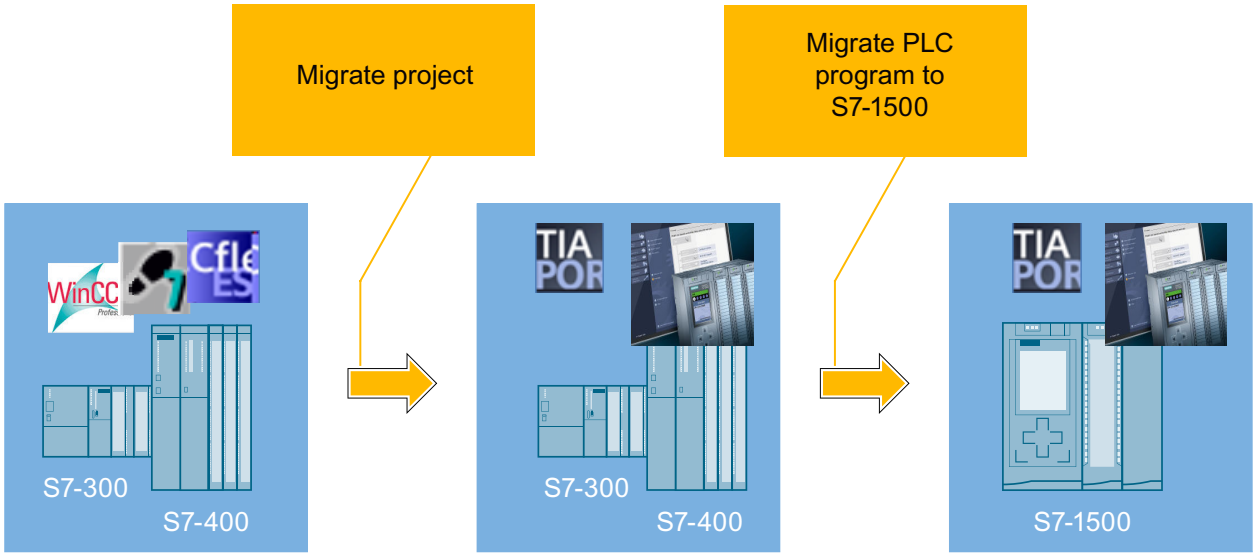
6

6.1 Overview of Migration Options

Migration paths

If you want to continue to use existing projects or programs with the latest version of the TIA Portal and the S7-1500, you have several options for migrating projects.

The following graphic provides an overview of the migration options:



Migrating projects to the TIA Portal

You use the "Migrate project" function to migrate projects that were created outside of the TIA Portal with STEP 7, WinCC, WinCC flexible, or SINUMERIK.

The result of the project migration is a TIA Portal project that you can use with your existing S7-300/400 series hardware and existing HMI devices.

Migrating PLC programs from S7-300/400 to S7-1500

To migrate a PLC program within the TIA Portal from an S7-300/400 series device to an S7-1500, you use the PLC migration.

The result of the PLC migration is an executable PLC program that is automatically adapted to the new system architecture of the S7-1500 as far as possible.

Optimizing PLC program for S7-1500

You also have the option to optimize your program for the S7-1500 by employing newly introduced programming methods. The optimization enables you to effectively use the higher performance, innovated memory technology and new features of the S7-1500 system.

Upgrading projects

You can also continue to use projects from previous versions of the TIA Portal. However, these projects do not need to be migrated. You upgrade projects from previous versions to the current version of the product or use projects from TIA Portal V12 SP1 in compatibility mode. You can find additional information on compatibility and upgrading projects in the section "Upgrading projects (Page 380)".

Note

Additional support for migration

You can find the latest information about migration in the Siemens Industry Online Support.

Migration of entire systems (http://support.automation.siemens.com/WW/view/en/83558085)

If you need further support, contact SIMATIC Customer Support.

See also

Compatibility of projects (Page 377)

Upgrading projects (Page 380)

Migration of controllers (http://support.automation.siemens.com/WW/view/en/83557459)

Migration of visualization (http://support.automation.siemens.com/WW/view/en/76878921)

Migration of communication (http://support.automation.siemens.com/WW/view/en/83558087)

Migration of I/O (http://support.automation.siemens.com/WW/view/en/83558994)

6.2 Migrating projects to a TIA Portal project

6.2.1 Migration of projects with the TIA Portal

Migration of existing projects

You can migrate projects from earlier automation solutions to the TIA Portal. Each time you migrate, a new project is created for the migrated data with which you can then work. Any TIA Portal projects already open are closed first.

The migration is then displayed in the table of the project history. From there, you have access to the migration log that is created automatically for the migration.

Supported products for migration

Refer to the chapter "System overview STEP 7 and WinCC" to find out which products are available for the TIA Portal. All products listed there are generally supported by the TIA Portal during migration.

Any additional requirements that must be met depend on the initial products that were used and the currently installed products. For more information on the migration options for your products, you can, for example, refer to the Siemens Industry Online Support and the documentation of your software products.

See also: Scaling of STEP 7 and WinCC in the TIA Portal (Page 33)

Procedure during migration

The migration process is divided into the following basic steps:

1. Preparing the initial project
   If the software for editing the initial project is not installed or not fully installed on the programming device/PC with the TIA Portal, or if the initial project is an integrated project, you must first convert the initial project to a migration file. To do this, install the migration tool on a programming device/PC on which the required software for editing the initial project is installed. Then, use the migration tool to convert the initial project, and copy the file to the programming device/PC on which the TIA Portal is installed. You can omit this step if the initial project and its associated software are on the same programming device/PC as the TIA Portal, and if the initial project is not an integrated project.
2. Performing migration
   Perform the migration within the TIA Portal. For the migration, specify as source either the migration file which you created with the migration tool or the initial project when all required software has been installed.
3. Checking the migration log
   A migration log is created for each migration. It contains information about modified project parts. You can call the log under "Common data > Logs" in the project tree or in the project history. After completion of the migration, the migration log will be displayed in the TIA Portal. Check the log following completion of the migration.
   If the migration failed, an XML file is created as a log under "\Logs" in the project directory. You can use any XML editor to open this log and view the reasons why the migration failed.
4. Correcting the migrated project
   Because the configurations of the initial project may not always be completely compatible with the TIA Portal, all configurations may not be reproduced identically in the migrated project. You should therefore work through the points in the migration log systematically. If you have not included the hardware configuration in the migration, you also have to convert the unspecified devices to the appropriate hardware.

Including the hardware configuration in the migration

By default, only the software parts of the project are included in the migration. An unspecified device is generated in the migrated project for the devices contained in the initial project. The hardware and network configurations and the connection are not migrated. You therefore convert the unspecified devices into suitable devices after the migration and recreate any network configurations and connections manually.

If you are certain that the hardware used in the initial project has a corresponding equivalent in the TIA Portal, you can include the hardware configuration in the migration. In this case, both the hardware configuration and the software are migrated. You can check with a tool (Page 154) to see which hardware components are supported.

See also

Display migration log (Page 160)

Scaling of STEP 7 and WinCC in the TIA Portal (Page 33)

6.2.2 Check migration readiness of hardware components

Introduction

Siemens offers a tool that can be used to check whether the hardware configuration used in an initial project is ready for migration to the TIA Portal.

Components integrated via GSD or GSDML files cannot be checked. For such modules, check manually in the TIA Portal whether the modules are available in the hardware catalog. If the modules are not available there, install the required GSD or GSDML files in the TIA Portal. You can obtain the required files from the manufacturer of the components.

In the result you can also see which software products and licenses have to be available on the programming device/PC with the installation of the TIA Portal to perform a migration. You can also see as of which firmware version the individual modules of the initial project are supported in TIA Portal. The result of the check can be output in a Microsoft Excel file or in PDF format.

Download

The tool for checking the migration readiness is available for download in the FAQs of the Siemens Industry Online Support under entry number 60162195 (http://support.automation.siemens.com/WW/view/en/60162195).

Source files for the check

To check readiness, you require one of the following source files which contains the article number of the hardware used in the initial project:

- .cfg file
  You can export the .cfg file from HW Config (STEP 7) with the menu command "Export as .cfg file" in the "Station" menu. The .cfg file contains all MLFBs of the devices used in the currently open station.
- Microsoft Excel file (in .xls file format)
  Regardless of the initial project used, you can create a Microsoft Excel list containing all MLFBs of the devices you want to migrate.
- File in .csv format
  The article numbers that are to be checked can also be saved to a .csv file instead of a Microsoft Excel list. To do this, use a standard text editor and enter the MLBs separated by comma and without space after the comma. Save the text file with the extension ".csv".

See also

Tool for checking migration readiness (http://support.automation.siemens.com/WW/view/en/60162195)

6.2.3 Preparing projects with the migration tool

6.2.3.1 Migrating projects with the migration tool

Preparation for migration

In many cases, a project that you wish to migrate will not be located on the same programming device/PC on which the latest version of the TIA Portal is installed. Therefore, the initial project must first be converted to a compatible format for the migration. The same applies to integrated projects.

After creation of the migration file, you copy the migration file to the programming device/PC on which the current version of the TIA Portal is installed. In the TIA Portal, you enter the migration file as the source for the migration and can create a project in the current file format of the TIA Portal.

Procedure for migration with the migration tool

The following steps are necessary to prepare a migration with the migration tool:

1. Install the migration tool on the programming device/PC where the source project is located. To do this, download the installation file from the Siemens Industry Online Support or install the migration tool from the setup DVD of the TIA Portal.
2. Start the migration tool, and use it to convert the source project to the migration file format with file extension ".am13".
   For this step, make sure that all software needed to process the source project is installed on the programming device/PC. This also includes all necessary service packs, hardware support packages and all expansion software that is needed to process the initial project. If individual products are not installed it may not be possible to perform the migration or the migration may be incomplete.
3. Copy the migration file to the target system on which a current version of the TIA Portal is installed.
   Note that the target system must have been installed with all software needed to configure the complete set of devices contained in the migration.
4. Perform the migration within the TIA Portal and specify the migration file with the extension ".am13" as the source.
5. Once migration is complete, check the migration log and systematically work through the information provided there for the newly created project. Read the information in the Inspector window with special care after the first compilation of the configuration.

Including the hardware configuration in the migration

By default, only the software parts of the project are included in the migration. An unspecified device is generated in the migrated project for the devices contained in the initial project. The hardware and network configurations and the connection are not migrated. You therefore convert the unspecified devices into suitable devices after the migration and recreate any network configurations and connections manually.

If you are certain that the hardware used in the initial project has a corresponding equivalent in the TIA Portal, you can include the hardware configuration in the migration. In this case, both the hardware configuration and the software are migrated. You can check with a tool to see which modules are supported.

See also

Migration of projects with the TIA Portal (Page 152)

Migrating projects (Page 158)

Calling the migration tool (Page 157)

Creating a migration file (Page 157)

6.2.3.2 Calling the migration tool

Starting the migration tool

During the installation, a "Migration to TIA Portal V13" shortcut is created as standard in the Start menu under "Siemens Automation > Migration Tool". Click this shortcut.

Alternatively, you can call the migration tool directly in Windows Explorer. During the installation, the migration tool is saved by default in one of the following folders:

- On a 64-bit operating system:
  C:\Program Files (x86)\Siemens\Automation\MIGTOOL_V13\Bin
- On a 32-bit operating system:
  C:\Program Files\Siemens\Automation\MIGTOOL_V13\Bin

To start the migration tool, click the "Siemens.Automation.MigrationApplication.exe" file in one of the directories.

See also

Creating a migration file (Page 157)

6.2.3.3 Creating a migration file

The section below describes how you can use the migration tool to convert the initial project to a migration file that can be read by the TIA Portal. Following conversion, this file is transferred to the target system and migrated there.

You can specify whether the migration file should contain the entire project, including the complete hardware configuration and the associated software, or whether you want to migrate the software only.

Requirement

- The suitable, original software with a valid license is installed for all configurations used in the initial project.
- The initial project is not provided with access protection.
- The initial project must be consistent, otherwise problem-free migration cannot be assured.

Procedure

To create the migration file, proceed as follows:

1. Select the path of the source file for the migration in the "Storage Location (Path)" field.
2. Specify the project parts that are to be migrated:
   - Select the "Include HW and Network data during the migration" check box to migrate not only the software but also the complete hardware parts and the network configuration of the project.
   - Select the "Copy SCADA runtime data" check box if you also want to migrate the runtime data, such as alarm archives, tag archives and user archives, in addition to the data of the engineering system.
3. Select the path and the file name for the migration file in the "Intermediate file" field.
4. Click the "Migrate" button.

Result:

A migration file is created. You now copy this file to the target system and migrate this file in the TIA Portal.

See also

Migrating projects (Page 158)

Calling the migration tool (Page 157)

Migrating projects with the migration tool (Page 155)

6.2.4 Migrating projects

Requirement

- A converted file in the format ".am13" is already available or the original software with a valid license is installed for all configurations used in the initial project.
- The initial project is not provided with access protection.
- The initial project must be consistent, otherwise problem-free migration cannot be assured.

Read the additional information on the requirements in the help for the respective products installed.

Note

System hibernation during the migration

While a migration is running, the system should not be changed to the standby or hibernate mode. Otherwise the migration will be aborted.

Procedure

To migrate a project, follow these steps:

1. Select the "Migrate project" command in the "Project" menu.
   The "Migrate project" dialog opens.
2. Specify the path and the file name for the project to be migrated in the "Source path" field.
   Select either a project in the ".am13" migration format or in the format of the initial project.
3. To include the hardware configuration in the migration, select the "Include hardware configuration" check box.
   If you have selected a migration file that was created with the migration tool, the check box cannot be selected. In this case, you must specify if you wish to include the hardware configuration in the migration before the conversion with the migration tool .
4. Select the "Copy WinCC Runtime Professional data" check box, if you also want to migrate the runtime data, such as alarm archives, tag archives and user archives, in addition to the data of the engineering system.
   If you have selected a migration file that was created with the migration tool, the check box cannot be selected. In this case, you must specify if you wish to include the SCADA runtime data in the migration before the conversion with the migration tool .
5. Select a name for the new project in the "Project name" box.
6. Select a path in the "Target path" box where the new project is to be created.
7. Enter your name or the name of another person responsible for the project in the "Author" field.
8. Enter a comment in the "Comment" box, if you require one.
9. Click "Migrate".

Result

The initial project is converted and a message appears after conversion is complete. The newly created project is then opened in the project view, and the migration log is opened in the TIA Portal.

Even if the migration failed, a project directory is created and a migration log in the form of an XML file is generated in this directory. The completion message that appears after the migration contains a link to this XML file. Click the link to open the XML file. Alternatively, you can find the XML file in the project directory under "\Logs".

See also

Post-editing integrated projects (Page 211)

Display migration log (Page 160)

Using logs (Page 376)

Migrating projects with the migration tool (Page 155)

Creating a migration file (Page 157)

6.2.5 Displaying the history of the migration

If a project was created by migration, the migration will be listed in the table of the project history. You can open the migration log in the table. The time of the migration is also shown.

Procedure

To display the migration in an overview table, follow these steps:

1. Select the open project in the project tree.
2. Select "Properties" in the shortcut menu of the project.
   The dialog with the properties of the project opens.
3. Select the "Project history" group in the area navigation.
   The overview table is displayed.

See also

Displaying properties of the project (Page 382)

6.2.6 Display migration log

A log is created for each successful migration. The log contains the following information:

- Migrated objects
- Modifications to objects made during migration
- Errors that occurred during migration
- In certain cases a link to more help with specific events.
  In this case, click the question mark to obtain more help.

Procedure

To display the log file of the migration, follow these steps:

1. Open the "Common data > Logs" folder in the project tree.
2. Double-click the desired log in the list.
   The contents of the log are displayed in the work area.

See also

Migration of projects with the TIA Portal (Page 152)

Using logs (Page 376)

6.2.7 Migrating WinCC flexible projects

6.2.7.1 Principles (WinCC flexible)

Migration (WinCC flexible)

Introduction

You can continue to use projects in WinCC from WinCC flexible. The following versions of WinCC flexible are supported:

- WinCC flexible 2008 SP2
- WinCC flexible 2008 SP3

The following sections describe the HMI devices that are supported and the required basic conditions for a successful migration.

Projects from ProTool Pro and earlier WinCC flexible versions cannot be migrated directly to WinCC. If you want to continue using such projects in WinCC, you have to migrate these to a supported version of WinCC flexible first and change the HMI device type.

If the project to be migrated contains components of a supported option package, the option package must be installed for successful migration to WinCC. If the option package is not installed, migration is canceled. This concerns the following option packages:

- SINUMERIK

The following option packages are not supported by the migration:

- ProAgent
- Open Platform Program - OPP

See also

Migrating projects from WinCC flexible (Page 164)

Compiling and loading a migrated project (WinCC flexible) (Page 166)

Supported HMI devices (Page 167)

Migration of connections (WinCC flexible) (Page 176)

Migration of tags (WinCC flexible) (Page 179)

Migration of runtime data (WinCC flexible) (Page 188)

Migration of integrated projects (WinCC flexible) (Page 191)

Migration principles (WinCC flexible)

Introduction

In the migration the project data are converted from a WinCC flexible project into the new data format of WinCC. The data will not be evaluated to see if they are consistent in the project you want to migrate. If errors or warnings are output in a source project during compilation, these will not be resolved as part of the migration. This means you should be able to compile the project without errors prior to migration. Note the scope of a project during migration. The features of WinCC apply for migration. For additional information refer to the online help in section "Process visualization > Performance features > Engineering System".

Unique object names

The objects are clearly identified by the folders in which they are contained in WinCC flexible. Screen objects in groups are clearly identified by the group name.

In WinCC, an object name must be unique within an HMI device. The name of screen objects must be unique within a screen.

The uniqueness of the name is verified during migration. If a name is not unique according to the new rule, the object in question will be renamed. A renamed object will receive the suffix "#Mign", where "n" stands for a sequential number.

Note

Changed OPC DA server name

If you have configured an OPC-DA server in the WinCC flexible project, the OPC-DA server is no longer available under the old name after migration.

In the migrated project, change the name of the OPC-DA server to the following value for the OPC clients concerned: "OPC.SimaticHMI.CoRtHmiRTm".

Points to note when renaming tags

If you have structured tags in folders in WinCC flexible, the name of the tag will be formed during migration from the folder name and the tag name. The names of the folders and tags are separated by the character \. The name of the tag after migration is then, for example: Plant1\Line3\Tag17.

If the name would otherwise be longer than 128 characters after migration, the name is formed by the character string #mig, a consecutive number, the character # and the tag name from WinCC flexible, for example #mig2#Tag17.

If you dynamically compose tag names in scripts, you must check the tags whose names were changed during migration with "#mig".

Affected objects

The following objects are renamed if necessary:

- Screens
- Faceplates
- Screen objects
- Graphics
- Recipes
- Structures
- Structural elements
- Alarm logs
- Tags
- Data logs
- Connections

Canceling migration

The migration is canceled in the following cases:

- If the project to be migrated is opened in the engineering system or in Runtime.
- If not enough memory space is available on the hard disk to create a copy for migration of the project.
- If the migration cannot address the project database due to problems with the installed SQL-Server.
- If the migration cannot address the project database due to missing user authorization.
- If you select the "*.hmi" file for the migration in an integrated project. You must select the "*.s7" file for the migration in an integrated project.
- If the project was created with a version not supported by the migration.

Saving the project in the migration format

You do not have to execute the migration of a WinCC flexible project completely on the PC on which the project is available. You can prepare the migration by saving the project in the migration format. The migration tool is available for saving a WinCC flexible project in the migration format. The migration tool exports the engineering data from the WinCC flexible project and saves the data in the migration format "*.AM13".

For the actual migration, copy the data in the migration format to a PC on which the TIA Portal is installed.

For more detailed information, please refer to the migration tool documentation.

Migrating projects from WinCC flexible

Introduction

When you migrate a project, data from a WinCC flexible project is loaded into a new project for WinCC. A new project is therefore created automatically for project migration. You cannot migrate to an existing project.

The migration can be started in both the Portal view and the Project view.

You should only migrate a project to a re-started TIA Portal.

Information on the migration of an integrated project can be found in the section Migration of integrated projects (WinCC flexible) (Page 191).

If you only want to save the project in migration format, you can use the migration tool. See Migration principles (WinCC flexible) (Page 162) for additional information.

Note

Further support for migration

You find current information on migration in the Siemens Industry Online Support:

Migration of visualization

If you need further assistance, please contact the SIMATIC Customer Support.

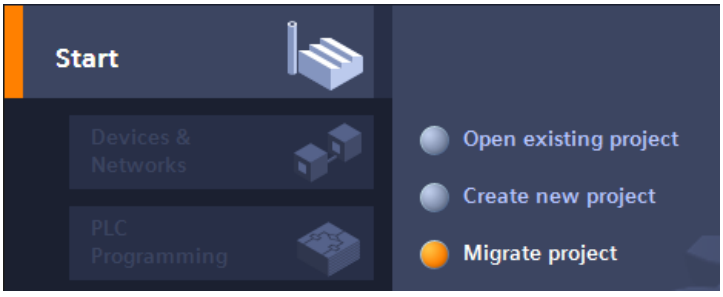
Requirement

- A project from WinCC flexible is available.
- The project is not open in WinCC flexible.

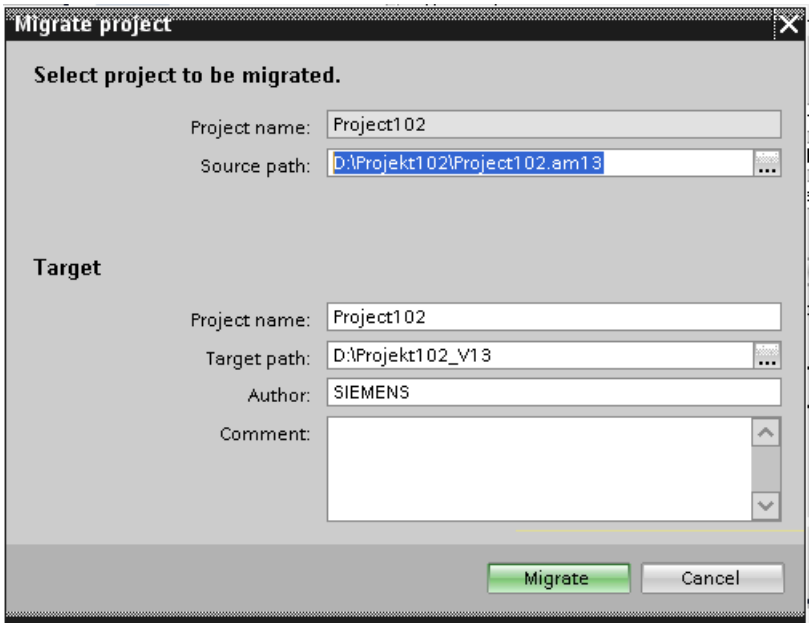
Procedure

Migrate a project in the Portal view as follows:

1. Select the action "Start > Migrate Project".



2. In the "Source path" box, navigate to the project you want to migrate.



3. Select the WinCC flexible project file "*.hmi" or "*.am13".
4. Change the information for the project to be created, if necessary. For example, change the project name or project path. The data to be migrated is created in the new project.
5. Click "Migrate".
   A new project is created and migration of the data is started:
   - The Project view opens.
   - The progress of the migration is shown in a migration window.
   - Warnings and errors about the migration process are displayed in the Inspector window under "Info > General".
   - All information about the migration is saved in a log file.
   - The project is saved and a message displayed upon completion of the migration. The message contains a link that you can use to open the log file.

When migration is complete, you will find a newly created device for each migrated HMI device in the project tree. These devices contain the migrated data, such as screens, alarms and tags.

Opening the migration log at a later point in time

The migration log is saved together with the migrated project. You can view the log at a later point in time. Open the log file as follows:

1. Open the "Common data > Logs" folder in the project tree. It contains the logs of all previously performed migrations.
2. Double-click the required migration log.
   The log is opened.

See also

Migration (WinCC flexible) (Page 161)

Migration of integrated projects (WinCC flexible) (Page 191)

Migration principles (WinCC flexible) (Page 162)

Compiling and loading a migrated project (WinCC flexible)

Compiling a migrated project

Once you have successfully migrated a WinCC flexible project, you need to recompile it before loading it to the HMI device. The project will only compile successfully if it was capable of error-free compiling prior to migration.

If errors occur during compilation of the migrated project, they have to be eliminated.

Once compiling is successfully completed, load the project to the HMI device.

Settings for download to the HMI device

The settings for loading the HMI device are not included in the migration. Once you have migrated the project, you must configure the settings for loading.

Select the HMI device in the project tree and select "Loading in device > Software (complete loading)" from the shortcut menu.The dialog "Advanced Loading" is opened. Configure the required settings for the interface. Click the "Load" button. The project is recompiled and the dialog "Load preview" is opened.

Expand the "Overwrite" entry and verify the settings for the following options:

- Would you like to overwrite the existing user administration data from this device
- Would you like to overwrite the existing recipe data on HMI system

Configure the options as you want to use them in the project in the future. Subsequently, load the project to the HMI device.

See also

Migration (WinCC flexible) (Page 161)

6.2.7.2 Migrating engineering data (WinCC flexible)

HMI devices (WinCC flexible)

Supported HMI devices

Introduction

When migrating projects from WinCC flexible you must bear in mind that WinCC does not support all HMI devices. You have to distinguish between the following cases:

- HMI device is supported by WinCC.
  The project is migrated 1:1 and gets the same HMI device after migration as before migration.
- The HMI device is replaced by a compatible successor model.
  The project is migrated. The migration replaces the HMI device with a compatible successor model. See HMI device change as a result of migration (WinCC flexible) (Page 169) for additional information.
- HMI device is not supported.
  If your WinCC flexible project contains an HMI device that is not supported by WinCC, the migration process is cancelled. To migrate the project, you must change the HMI device in WinCC flexible to a HMI device type supported by WinCC.

The following HMI device types are supported both by WinCC flexible and WinCC:

Basic Panels

- KTP400 Basic mono PN
- KTP400 Basic mono PN Portrait
- KTP600 Basic DP
- KTP600 Basic DP Portrait
- KTP600 Basic PN
- KTP600 Basic PN Portrait
- KTP600 Basic mono PN
- KTP600 Basic mono PN Portrait
- KTP1000 Basic DP
- KTP1000 Basic PN
- TP1500 Basic PN

Mobile Panels

- Mobile Panel 177 6" DP
- Mobile Panel 177 6" PN
- Mobile Panel 277 8"
- Mobile Panel 277 8" IWLAN V2
- Mobile Panel 277F 8" IWLAN V2
- Mobile Panel 277F 8" IWLAN (RFID Tag)
- Mobile Panel 277 10"

Panels

- OP 73
- OP 77A
- OP 77B
- OP 177B 6" mono
- OP 177B 6" color PN/DP
- TP 177B 4" color PN/DP
- TP 177A
- TP 177A Portrait
- TP 177B 6" mono DP
- TP 177B 6" color PN/DP
- OP 277 6"
- TP 277 6"

Multi Panels

- MP 177 6" Touch
- MP 277 8" Key
- MP 277 8" Touch
- MP 277 10" Key
- MP 277 10" Touch
- MP 377 12" Key
- MP 377 12" Touch
- MP 377 15" Touch
- MP 377 15" Touch daylight readable
- MP 377 19" Touch

Sinumerik PC

- OP 010 Key
- OP 012 Key
- OP 015 Key
- OP 015A Key
- TP 015A Touch+Key

HMI applications

- WinCC flexible Runtime

WinCC only supports the functions provided by these HMI device types.

Other functions which are not migrated because of the restricted device selection are documented in the following sections.

Adaptations before migration

If the HMI device was changed to an HMI device with a different screen size in the project to be migrated, you must recompile and save the project in WinCC flexible before the migration. The compilation process will adjust the size of the screens and screen elements.

See also

Migration (WinCC flexible) (Page 161)

HMI device change as a result of migration (WinCC flexible) (Page 169)

Configuration change after HMI device change (WinCC flexible) (Page 172)

Migration of connections (WinCC flexible) (Page 176)

HMI device change as a result of migration (WinCC flexible)

Introduction

WinCC flexible supports some HMI devices which are discontinued in the future. These HMI devices are no longer supported by WinCC. When migrating a WinCC flexible project, an HMI device that is not supported is replaced by a compatible successor device.

Only the HMI device type is changed during the migration. HMI device-specific data are not changed by the migration.

Inconsistencies in the project may occur due to a change in the HMI device. In a project which was compilable before the migration, errors may occur in the project compilation after changing the HMI device. E.g. because the changed HMI device supports different memory media to the previous HMI device.

HMI device change by the migration

The following table provides information about which successors replace the HMI devices that are not supported.

Unsupported HMI devices	Successors
C7-635 6" Key	MP 177 6“ Touch
C7-635 6" Touch	MP 177 6“ Touch
C7-636 6" Key	MP 177 6“ Touch
C7-636 10" Touch	MP 277 10“ Touch
HT 8	PCU50.3 -C (resolution: 640*480)
Mobile Panel 170 6"	Mobile Panel 177 6" DP
Mobile Panel 277 8" IWLAN	Mobile Panel 277 8" IWLAN V2
Mobile Panel 277F 8" IWLAN	Mobile Panel 277F 8" IWLAN V2
OP010 Key	PCU50.3B -C (resolution: 640*480)
OP012 Key	PCU50.3B -C (resolution: 800*600)
OP015 Key	PCU50.3B -C (resolution: 1024*768)
OP 08T	PCU50.3 -C (resolution: 640*480)
OP 73micro	OP 73
OP 170B 6" mono	OP 177B 6" color PN/DP
OP 270 6"	OP 277 6"
OP 270 10"	MP 277 10" Key
TP015A Touch Key	PCU50.3B (resolution: 1024*768)
TP 170A 6"	TP 177A 6"
TP 170B 6" mono	TP 177B 6" color PN/DP
TP 170B 6" color	TP 177B 6" color PN/DP
TP 170micro 6"	TP 177A
TP 177micro 6"	TP 177A
TP 270 6"	TP 277 6"
TP 270 10"	MP 277 10" Touch
MP 270 6" Touch	TP 277 6"
MP 270 10" Key	MP 277 10" Key
MP 270 10" Touch	MP 277 10" Touch
MP 370 12" Key	MP 377 12" Key
MP 370 12" Touch	MP 377 12" Touch
MP 370 15" Touch	MP 377 15" Touch
PC 477 12" Key	PC 477B 12" Key PB
PC 477 12" Touch	PC 477B 12" Touch PB
PC 477 15" Key	PC 477B 15" Key PB
PC 477 15" Touch	PC 477B 15" Touch PB
PC 477 19" Touch	PC 477B 19" Touch PB
PC 577 12" Key	IPC 577C 12" Key PB
PC 577 12" Touch	IPC 577C 12" Touch PB
PC 577 15" Key	IPC 577C 15" Key PB
PC 577 15" Touch	IPC 577C 15" Touch PB

Unsupported HMI devices	Successors
PC 577 19" Touch	IPC 577C 19" Touch PB
PC 670 10" Key	HMI IPC677C 12" Key PB
PC 670 12" Key	PC 677B 12" Key PB
PC 670 12" Touch	PC 677B 12" Touch PB
PC 670 15" Key	PC 677B 15" Key PB
PC 670 15" Touch	PC 677B 15" Touch PB
PC 677 12" Key	PC 677B 12" Key PB
PC 677 12" Touch	PC 677B 12" Touch PB
PC 677 15" Key	PC 677B 15" Key PB
PC 677 15" Touch	PC 677B 15" Touch PB
PC 677 17" Touch	PC 677B 17" Touch PB
PC 677 19" Touch	PC 677B 19" Touch PB
PC 870 12" Key	HMI IPC677C 12" Key PB
PC 870 15" Key	HMI IPC677C 15" Key PB
PC 870 15" Touch	HMI IPC677C 15" Touch PB
PC 877 12" Key	HMI IPC677C 12" Key PB
PC 877 15" Key	HMI IPC677C 15" Key PB
PC 877 15" Touch	HMI IPC677C 15" Touch PB
PC 877 19" Touch	HMI IPC677C 19" Touch PB
PC IL 70 12" Touch	HMI IPC577C 12" Touch PB
PC IL 70 15" Touch	HMI IPC577C 15" Touch PB
PC IL 77 12" Touch	HMI IPC577C 12" Touch PB
PC IL 77 15" Touch	HMI IPC577C 15" Touch PB
PC IL 77 19" Touch	HMI IPC577C 19" Touch PB
PC IL 77 12" Key	HMI IPC577C 12" Key PB
PC IL 77 15" Key	HMI IPC577C 15" Key PB
P012T Touch	IPC 677C 12" Touch PB (if a P350 is connected)
P012T Touch	IPC 477C 12" Touch PB (if a P320 is connected)

HMI device change by the user

If a project contains an HMI device that is not supported by WinCC and no compatible successor model exists, the migration is cancelled. If you want to migrate the project, you must change the HMI device to WinCC yourself before the migration. Use an HMI device which is supported both by WinCC flexible and WinCC for this.

See also

Supported HMI devices (Page 167)

Configuration change after HMI device change (WinCC flexible)

Note

If you migrate a project with embedded screens, this can result in the project file becoming larger than the old project file.

HMI device replacement by the migration

If the migration makes an HMI device replacement, it replaces the existing HMI device with a compatible successor model. The successor models are further developed and therefore more efficient than their predecessors. The new models therefore support all the properties of their predecessors. Therefore only a little rework is to be expected on the project due to the HMI device replacement.

HMI device replacement by the user

if you change an HMI device yourself before the migration, you must ensure that the HMI device used supports all properties which are used in the project. Compile the project after the HMI device replacement in WinCC flexible. The project must be perfectly compilable before the migration. Then migrate the project to WinCC.

See also

Supported HMI devices (Page 167)

Object support during migration

Introduction

When migrating projects from WinCC flexible, all configuration data involving an HMI device supported by WinCC will be migrated. Basically, all object types and functions that are available and can be mapped to the new project environment will be fully migrated.

Some global object types are not migrated, for example, dictionaries and global libraries.

Supported object types

The following object types are supported for migration:

- Animations
  Disabled animations are not migrated.
- Audit settings
- Audit reports
- Scheduler
- User administration

- Area pointers
- Faceplates
- Screens
- Screen template
- Data types
- Function lists
- Graphics lists
- Display and operating elements
  All display and operating elements that are available on the supported HMI devices are supported for migration.
- Alarms
- Alarm classes
- Alarm groups
- Project library
- Project languages
- Reports
- Recipes
- Runtime languages
- Runtime scripting
- Sm@rtAccess/Sm@rtService
- Structures
- System events
- System functions
- Texts
- Text lists
- Tags
- Connections
- Effective ranges
- Zones
- Cycles

Unsupported object types

The following object types are not supported by migration:

- Global libraries
- Dictionaries

- Project versions
- Change log

Mapping of the screen navigation

WinCC does not support the screen navigation from WinCC flexible. The data of the screen hierarchy from WinCC flexible are not migrated. To map the functionality of a screen navigation from WinCC flexible, the naivigation buttons are migrated to the button available in WinCC. The system functions migrated to the "ActivateScreen" system function are migrated to the "Click" event of the respective button. The following system functions used in WinCC flexible are not available for the screen navigation in WinCC:

- ActivateRootScreen
- ActivateLeftScreen
- ActivateRightScreen
- ActivateParentScreen
- ActivateFirstChildScreen

These system functions are migrated to the "ActivateScreen" system function. The "Screen name" parameter is taken from the data of the screen hierarchy. If one of the named system functions is called from a screen which is not contained in the screen hierarchy, the "ActivateScreen" function is created without parameters. You must configure the desired screen to this system function after the migration.

Tab sequence in pictures with faceplates

In pictures with faceplates, the tab sequence is changed in both the screen and the faceplate as a result of the migration.

Migration of the screen template

WinCC offers an extended concept for working with screen templates. WinCC offers a global screen and several templates for each device. During migration of a template from WinCC flexible, the objects contained there and the properties configured in the template are migrated to the extended concept of the screen templates of WinCC.

The following objects are migrated to the "global screen" of WinCC:

- Alarm window
- Alarm indicator
- Help indicator
- Function keys of HMI devices with function keys

All other objects and properties are migrated to a template of WinCC.

The connection of the objects and properties to the respective template is automatically adapted.

See also

Changes of values of object properties by the migration (WinCC flexible) (Page 175)

Changes of values of object properties by the migration (WinCC flexible)

Introduction

The standardization of object properties from WinCC V7 and WinCC flexible requires changes to the object properties during the migration process. The migration calculates the changes in such a way that the representation of the objects after migration is the same as prior to migration. Changes made during migration result in different units of measurements and values in the configuration for some object properties.

Migrating the font settings of an object

In WinCC V7 and WinCC flexible, the unit of measurement "point" is used to denote the size of the fonts used for an object. In WinCC, the unit of measurement "pixel" is used to denote the size of the fonts used for an object. During migration, the font size is converted accordingly to ensure that the representation of the font is the same size at zoom level 100%. The different units of measurement result in changes to the numerical values for the font sizes after migration.

Example:

Font style before migration	Font style after migration
Arial 10 points	Arial 13 pixels
Arial 16 points	Arial 21 pixels
Tahoma 10 points	Tahoma 13 pixels
Tahoma 16 points	Tahoma 21 pixels

Migration of object margins

In WinCC flexible, some objects permit the entry of values <0 and >127 for setup of the object margins for the configuration of the representation. In WinCC, the range of values for object margins is limited to values between 0 and 127. The migration changes values <0 to the value "0" and values >127 to the value "127".

See also

Object support during migration (Page 172)

Connections (WinCC flexible)

Migration of connections (WinCC flexible)

Introduction

If you migrate a project in which a supported communication driver is used, this driver is used further in WinCC. Objects which communicate via this driver are migrated 1:1. No rework is necessary.

Not all communication drivers which are available in WinCC flexible are supported in WinCC. If an unsupported driver is used in the project to be migrated, there are two possible scenarios:

1. A compatible spare driver is available for the used driver.
2. No compatible spare driver is available for the used driver.

Compatible spare driver is available

If a driver is available in WinCC which addresses the used PLC or a comptaible PLC, the driver is automatically replaced during the migration.

You get an appropriate warning if the used driver is replaced.

In this case, check whether all the external tags and area pointers are valid after the migration by means of the migration report.

If the used driver is replaced, all the connection parameters are reset to the standard values. The CPU type is adapted to the appropriate PLC. The properties of the connected tags and area pointers are not changed.

Compatible spare driver is not available

If no driver is available in WinCC which addresses the used PLC or a compatible PLC, the configured connection is not migrated. All external tags which were connected to the PLC by this driver are converted into internal tags. External tag properties are lost in the conversion, e.g. the address in the PLC. All changes to the tags are recorded in the migration report.

You must reconnect the tags with the PLC after the migration and configuring of a connection.

You get an appropriate warning if the driver is not migrated.

Converting the communication driver

The following tables show the mapping of communication drivers from WinCC flexible to WinCC.

Supported communication drivers

WinCC flexible	WinCC
Allen Bradley DF1	Allen Bradley DF1
Mitsubishi FX	Mitsubishi FX

WinCC flexible	WinCC
Modicon MODBUS	Modicon MODBUS RTU
Modicon MODBUS TCP/IP	Modicon MODBUS TCP/IP
Omron Hostlink/Multilink	Omron HostLink
OPC	OPC
SIMATIC HMI HTTP Protocol	SIMATIC HMI HTTP Protocol
SIMATIC S7 200	SIMATIC S7 200
SIMATIC S7 300/400	SIMATIC S7 300/400

Compatible communication drivers

WinCC flexible	WinCC
Allen Bradley DH485	Allen Bradley DF1
Allen Bradley E/IP C.Logix	Allen Bradley EtherNet/IP
Mitsubishi Protocol 4	Mitsubishi MC TCP/IP

Not supported communication drivers

WinCC flexible	WinCC
GE Fanuc SNP	Not supported
ISAC	Not supported
LG GLOFA-GM	Not supported
SIMATIC 500/505 DP	Not supported
SIMATIC 500/505 serial	Not supported
SIMATIC S5 AS511	Not supported
SIMATIC S5 DP	Not supported
Telemecanique Uni-Telway	Not supported

See also

Migration (WinCC flexible) (Page 161)
Supported HMI devices (Page 167)
Adapting configuration for non-migrated connection (WinCC flexible) (Page 178)
Migration of area pointers (WinCC flexible) (Page 179)
Migration of tags (WinCC flexible) (Page 179)

Adapting configuration for non-migrated connection (WinCC flexible)

Introduction

If a connection cannot be migrated you have the following options:

- Change the configuration in WinCC flexible before the migration
- Change the configuration after the migration

Change a connection before the migration

If the communication driver selected for the connection is not supported by the migration, you must select a driver which is supported by the migration. The HMI device must also support the selected communication driver.

If no suitable driver is available for the used HMI device, you must use a suitable HMI device. Then adapt the configuration in WinCC flexible and recompile the project. Migrate the project after successful adaptation and compiling. The connection is then also migrated.

Changing a connection before the migration

If you migrate a WinCC flexible project in which the connection is not migrated, all external tags of this connection are mapped to internal tags. You receive an appropriate entry in the migration protocol for every tag concerned.

You must configure a new connection after the migration. For this connection you select a communication driver which supports the used HMI device. If no suitable driver is available for the used HMI device, you must use a suitable HMI device.

If you have configured the new connection, you must reconnect all tags which were mapped to internal tags before the migration. Open the migration report to make reconfiguration easier. Reconfigure the tags according to the log entry.

If the project contained area pointers, you must also reconfigure these. You will also find corresponding entries for the area pointers in the migration log.

See the section Migrating projects from WinCC flexible (Page 164) for further information about the migration log.

See also

Migrating projects from WinCC flexible (Page 164)

Migration of connections (WinCC flexible) (Page 176)

Migration of area pointers (WinCC flexible)

Introduction

The migration of area pointers depends on the used communication driver.

- If the used communication driver is supported by the migration, area pointers from WinCC flexible are taken over unchanged in the migration.
- if the used communication driver is not supported by the migration, area pointers are not migrated.

Migration of the area pointers

If the connection used by the area pointers is fully migrated, the area pointers are also fully migrated. The parameters of the area pointers are taken over unchanged.

If you replace the communication driver of the connection before the migration, check whether all the parameters of the area pointers are still valid after the migration.

If the connection used by the area pointer is not migrated, the area pointers are not migrated either. See Adapting configuration for non-migrated connection (WinCC flexible) (Page 178) for additional information.

See also

Adapting configuration for non-migrated connection (WinCC flexible) (Page 178)

Migration of connections (WinCC flexible) (Page 176)

Migration of tags (WinCC flexible)

Introduction

You need to make some special considerations when migrating tags. The following aspects should be distinguished:

- Migrating data types of tags
- Migrating internal tags
- Migrating external tags
- Tag names
- Tag limits

Migrating data types

WinCC features some other data types and uses different data type names than WinCC flexible. When migrating a relevant tag, the data type from WinCC flexible is mapped to the corresponding data type in WinCC. See Migration of data types (WinCC flexible) (Page 195) for additional information.

Migrating internal tags

Internal tags are always migrated completely. Only the data type names and tag names may change due to migration.

Migrating external tags

If the connection used by the external tags is migrated, the external tags are also fully migrated. The migration of external tags depends on whether the used communication driver is supported by the migration. See Migration of connections (WinCC flexible) (Page 176) for additional information.

If the connection used by the external tags is not migrated, the external tags are not migrated either. The external tags are then mapped to internal tags. You must configure a new connection and reconnect the tags with the tags of the PLC after the migration.

Migrating names of tags

In WinCC flexible, tags located in different folders can have the same name. In WinCC, the tag name must be unique on the configured HMI device. This means that tags with the same name from different folders will be renamed during migration. See Migration principles (WinCC flexible) (Page 162) for additional information.

See also

Migration of data types (WinCC flexible) (Page 195)

Migration of connections (WinCC flexible) (Page 176)

Migration principles (WinCC flexible) (Page 162)

Migration (WinCC flexible) (Page 161)

Migration of alarm classes and alarm groups (WinCC flexible) (Page 181)

Migrating scripts (Page 184)

Migration of language-specific content (WinCC flexible) (Page 185)

Migration of libraries (WinCC flexible) (Page 187)

Migration of structures

The "StringChar" data type is not supported in HMI user data types in WinCC.

If you have used this data type in a structure in a WinCC flexible project, an invalid element is created by the migration in the HMI user data type.

After migration , you must re-work this user data type in WinCC and enable the user data type.

At the same time, check the offset of the following elements and any existing interconnections between a faceplate and the elements of the user data type. Adapt the elements if required.

Migration of logs

Storage location of logs

WinCC flexible allowed you to store logs in a database which was automatically set up upon WinCC flexible installation ("System-defined data source" setting). This option is not available in WinCC.

If you have used this setting in your WinCC flexible project, it will be changed to "User-defined name of data source" during migration. Before you can store logs in a database, you must configure an ODBC data source in the Windows control panel and configure the name of the user data source specified there as the "name of the data source" in the log in WinCC.

Migration of alarm classes and alarm groups (WinCC flexible)

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Error	Alarms
System	System
Warnings	Events

The names of the alarm classes can be changed as necessary after migration.

Migrating alarm groups

Migration will migrate only those alarm groups actually in use.

Alarm groups with an ID from 1-31 will be migrated 1:1.

A corresponding alarm group is created in WinCC for each alarm class in the system. These alarm groups created by the system are assigned IDs beginning with the number 32 and consecutively incremented. The 4 pre-defined message classes in every WinCC project are automatically given IDs 32-35 by their alarm groups. Additionally created alarm group and an additional ID is assigned to each user-defined alarm class. Therefore, the IDs for alarms groups with IDs > 31 may be changed after migration. This step also changes the assignment of the alarm group names to the IDs.

Example:

In the example, you can see the assignment of the IDs in WinCC for the migration.

Alarm groups	ID in WinCC flexible	ID in WinCC	
Alarm group 1-16	1-16	1-16	Default for alarm groups from system alarms
Alarm group 17-31	17-31	17-31	Custom alarm groups

Alarm groups	ID in WinCC flexible	ID in WinCC	
		32-35	Default in WinCC for alarm groups of pre-defined alarm classes.
Alarm group 32	32	36	Changed assignment of ID to alarm group in WinCC
Alarm group 33	33	37	Changed assignment of ID to alarm group in WinCC

Also note:

When migrating alarm groups that supposedly have the same group name, the migration adapts the name. This occurs, for example, when a group name contains a space at the end of the name. The migration deletes all existing spaces at the end of names. If two groups obtained the same group names due to this deletion, the migration adds the suffix "# Mign" to the group name of the following alarm groups, where "n" stands for a sequential number.

Example:

The following alarm groups exist in WinCC flexible:

"AlarmGroup_18"

"AlarmGroup_18 " - group name contains one space

"AlarmGroup_18 " - group name contains two spaces

"AlarmGroup_18" is the alarm group with the highest number.

Result after migration:

"AlarmGroup_18"

"AlarmGroup_18#Mig1"

"AlarmGroup_18#Mig1.1"

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Error	Errors
System	System
Warnings	Warnings

The names of the alarm classes can be changed as necessary after migration.

Display of ALARM_S messages and SIMATIC SFM messages

In WinCC flexible you can activate the display classes for ALARM_S messages in integrated projects. In WinCC flexible, you activate the display of SIMATIC SFM messages via a separate setting. The separate setting for activating the display of SIMATIC SFM messages is not required in WinCC. You control the display of SIMATIC SFM messages, and also the display of ALARM_S messages in WinCC only by activating the corresponding display class.

The changed concept may cause the display of messages to change following migration.

If all the display classes for ALARM_S messages are activated and the display of SIMATIC SFM messages is deactivated in the WinCC flexible project , ALARM_S messages and SIMATIC SFM messages are displayed following migration.

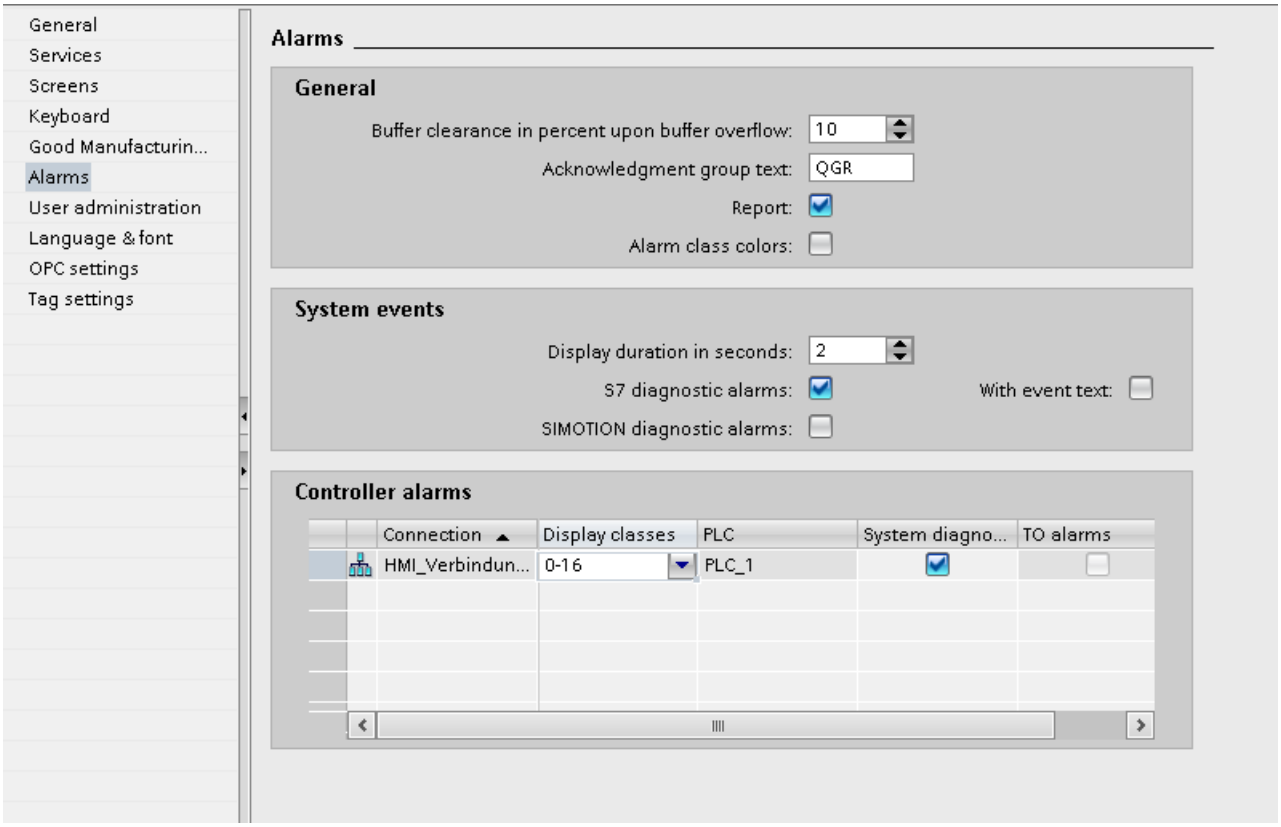
To ensure that only ALARM_S messages are displayed following migration, you have to assign the SIMATIC SFM messages to an unused display class after migration to STEP 7. You then have to deactivate this display class in WinCC.

If all the display classes for ALARM_S messages are deactivated and the display of SIMATIC SFM messages is activated in the WinCC flexible project , ALARM_S messages and SIMATIC SFM messages are not displayed following migration.

To ensure that only SIMATIC SFM messages are displayed following migration, you have to assign the SIMATIC SFM messages to an unused display class after migration to STEP 7. You then have to activate this display class in WinCC.

The display class is dependent on the settings in STEP 7. The default setting for SIMATIC SFM messages in Step 7 is the display class "0". To activate the display in WinCC, the display class "0" must be activated.

You activate the display class in WinCC in the Runtime settings of the respective HMI device in the "Messages" category.



See also

Migration of tags (WinCC flexible) (Page 179)

Migrating scripts

Introduction

The migration supports VB-scripts which were created in WinCC flexible. For a VB-script to be migrated successfully it must be functionable in WinCC flexible first.

Note

Script errors

The most efficient way to locate scripting errors in the course of the initial test run after migration is to use an installed Script Debugger and the diagnostics controls.

Migration of a VB script

A script is analyzed in the migration and adapted to the system behavior of WinCC if necessary. The following is adapted:

- System functions which have changed their names are renamed.
  This concerns the following system functions:

Function name in WinCC flexible	Function name in WinCC
IncreaseValue	IncreaseTag
DecreaseValue	DecreaseTag
SetBrightness	SetAndGetBrightness
SetValue	SetTag

- The access to tags as parameters of system functions is adapted to the system behavior of WinCC. In WinCC scripts are no longer transfromed as in WinCC flexible. The scripts are executed directly as a source code. Therefore the stricter rules for the VBS syntax apply. A tag call is always migrated with inverted commas. If several parameter types are allowed for a system function, these are migrated with the keyword "SmartTags".
  Example 1:
  The value of the "temperature" tag should be incremented by the value "1".
  Valid expressions in WinCC flexible:
  IncreaseValue temperature, 1
  IncreaseValue "temperature", 1
  IncreaseValue SmartTags("temperature"), 1
  Valid expression in WinCC:
  IncreaseTag "temperature", 1
  Example 2:
  You use a system function on which several parameters are allowed. The value of the "temperature" tag should be incremented by the value of the "heatcontrol" tag.
  Valid expression for WinCC flexible:
  IncreaseValue "temperature", "heatcontrol"
  Valid expression for WinCC:
  IncreaseTag "temperature", SmartTags("heatcontrol")
- Objects renamed by the migration are also renamed in the script when using in a script. See the section Migration principles (WinCC flexible) for further information about the renaming objects. You only have to observe the following rules when renaming objects: If you address objects whose names are dynamically generated by the script with the help of a script, the object names in the script can not be automatically changed by the migration. In such a case, you have to correct the generation of the object names in the script after migration.

See also

Migration of tags (WinCC flexible) (Page 179)

Migration of language-specific content (WinCC flexible)

Introduction

WinCC offers the same options for configuring projects in different languages as those available in WinCC flexible. All languages supported by WinCC are included in the migration of a project.

Migrating language-dependent content

The following language-dependent content is migrated:

- Project languages
- Project texts

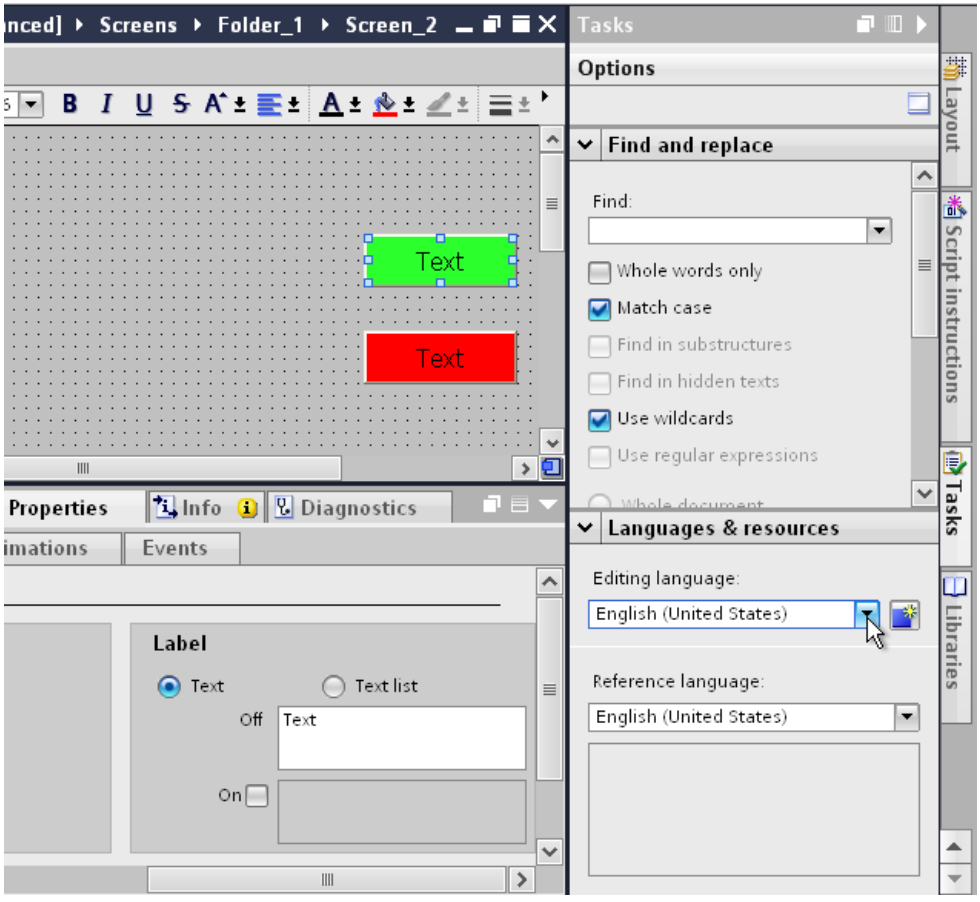
- Fonts for display in runtime
- Language-dependent graphics

You need to consider the following when migrating language-dependent content:

- The operating system on the PC performing the migration must support the project languages used in the project.
- The fonts used for runtime display must be installed on the PC performing the migration.
- Dictionaries are not supported by the migration.

Editiing language of integrated projects following migration

During migration of an integrated project, the project components to be migrated from STEP 7 and WinCC flexible also bring their respective settings for the editing language. In WinCC there is only one editing language for all project components. Migration activates for the mgrated project the editing language which was set in STEP 7 prior to migration. If this setting is not the same as the setting from WinCC flexible, the configured texts are no longer visible in WinCC. No text is displayed at the usage locations, or only the entry "Text" can be seen. To make the texts visible, you must change the editing language. Click the "Tasks" taskcard at the right-hand edge of the TIA portal and select the correct editing language in the "Language & Resources" area.



Unsupported languages

The migration of language-dependent content depends on whether or not WinCC supports the respective language.

If a project only contains project languages not supported by WinCC, the project will not be migrated.

If a project contains supported and unsupported project languages, only the supported languages will be migrated. The editing language and reference language are set to a supported language.

The following languages are not supported by WinCC:

- Arabic
- Hebrew
- Dhivehi
- Gujarati
- Kannada
- Tamil
- Telugu
- Urdu
- Punjabi
- Persian
- Syrian

See also

Migration of tags (WinCC flexible) (Page 179)

Migration of libraries (WinCC flexible)

Introduction

You need to consider two different cases when migrating from libraries:

1. Migrating a project library
2. Migrating a global library

Migrating a project library

A project library is stored together with the project data in the project file. For this reason, a project library is migrated with the same restrictions as the project data.

Migrating a global library

Global libraries are not supported by the migration. The library objects used in the project will be migrated, however. The library objects are copied when used in the project and then no longer have a connection to the library.

To migrate a global library, you must copy or move the objects contained in the library to the project library. The objects are then included in the migration. In WinCC, you move the migrated objects to a new global library that is created. You can copy or move both individual objects or entire library categories.

See also

Migration of tags (WinCC flexible) (Page 179)

6.2.7.3 Migrating runtime data (WinCC flexible)

Migration of runtime data (WinCC flexible)

Introduction

Only the configuration data are migrated by the migration when migrating a project. The runtime data are not affected. You need to update the runtime data following migration.

The runtime data consists of the following:

- Runtime project
  The runtime project contains the compiled project data.
- Recipe data and user administration
  The recipe data and user administration are data that can be changed in runtime.
- Log data
  The data of tag logs and alarm logs are acquired and logged in runtime.

Migrating recipe data and user administration

If the recipe data and user administration were changed in runtime, you need to back up this information from the HMI device before you load the migrated project.

Depending on the used HMI device, you have different options for saving the above data.

If the HMI device supports external memory media and the recipe data are saved there, the data remain on the memory medium. External memory media are for example StorageCard or a network drive. You use the data again after the migration. The user adminsitration is not saved on an external memory medium so you have to save these data, e.g. with ProSave.

If the recipe data are saved in the internal memory of the HMI device, save these data on an external memory medium. Use ProSave to save the recipe data. If you have already configured

appropriate system functions in your project, use the system functions. The following system functions are available for the recipe data:

- "ExportDataRecords" for the backup
- "ImportDataRecords" for the restore

You update the runtime project by compiling the project in WinCC again and loading it to the HMI device.

After you have loaded the migrated project on the HMI device, restore the recipe data and the user administration to the HMI device.

See also

Migration (WinCC flexible) (Page 161)

Backing up recipe data and user administration (WinCC flexible) (Page 189)

Restoring recipe data and user administration (WinCC flexible) (Page 190)

Backing up recipe data and user administration (WinCC flexible)

Introduction

To continue using the recipe data and user administration in a migrated project, you first need to back up this data from the HMI device. Then load the data into the migrated WinCC project. Use ProSave to back up the data.

Requirement

- The WinCC flexible project is running on the HMI device in Runtime.
- The HMI device is connected to a PC on which ProSave is installed.

Procedure

Proceed as follows to back up the recipe data and user administration:

1. Start ProSave.
2. Select the device type and the connection parameters in the "General" tab.
3. Open the "Backup" tab.
4. Select the "Recipes from the device memory" entry in the "Data type" box.
   Do not select "Complete backup" because otherwise you will not be able to select separately when restoring the recipe data.
5. Navigate to the desired location in the "Save as" box and click "Start Backup".
   The recipe data are saved.
6. Select "User administration" in the "Data type" box and click "Start Backup".
   The user administration is saved.

For additional information refer to the online help for ProSave.

Alternative procedure

ProSave is automatically installed with WinCC flexible. The entire functional range of ProSave is available on the configuration PC within WinCC flexible via the menu command "Project > Transfer".

Alternatively, you can back up the recipe data and user administration via the ProSave integrated in WinCC flexible. Start WinCC flexible and select the menu command "Project > Transfer > Backup". Back up the recipe data and user administration as described in steps 4-6.

See also

Migration of runtime data (WinCC flexible) (Page 188)

Restoring recipe data and user administration (WinCC flexible)

Introduction

To continue using saved recipe data and user administration after the migration, you first need to compile the migrated project and load it to the HMI device. You can then transfer the saved data to the HMI device. Use ProSave to restore the data.

Requirement

- The migrated project has been transferred to the HMI device and is running in runtime.
- The HMI device is connected to a PC on which ProSave is installed.

Procedure

Proceed as follows to load the saved recipe data and user administration to the HMI device:

1. Start ProSave.
2. Select the device type and the connection parameters in the "General" tab.
3. Open the "Restore" tab.
4. Navigate to the location of the saved recipe data in the "Opening..." box and select the file.
5. Click "Start Restore".
   The recipe data will be transferred to the HMI device..
6. Repeat steps 4-5 to restore the user administration.
   The user administration will be transferred to the HMI device.

For additional information refer to the online help for ProSave.

Alternative procedure

ProSave is automatically installed with WinCC. The entire functional range of ProSave is available on the configuration PC within WinCC flexible via the menu command "Project > Transfer".

You can also restore the recipe data and user administration via the ProSave integrated in WinCC. Start WinCC and select the menu command "Online > Device maintenance > Restore". Restore the recipe data and user administration as described in steps 4-6.

See also

Migration of runtime data (WinCC flexible) (Page 188)

Backing up log data (WinCC flexible)

Introduction

If an HMI device supports external memory media and the log data are saved there, the data remain on the memory medium. External memory media are for example StorageCard or a network drive. If a log is saved on an external memory medium, the migrated project accesses this storage location again after the migration. In this case the log data must not be backed up.

Backing up log data

If you want to back up the log data externally before the migration, you have the following options:

- Backup with the "ArchiveLogFile".
  If you have already configured the "ArchiveLogFile" function in the WinCC flexible project, use this function to back up the data.
- Copy the log files by Copy&Paste with the Windows Explorer to an external memory medium or network drive.

See also

Migration of runtime data (WinCC flexible) (Page 188)

6.2.7.4 Migrating integrated projects (WinCC flexible)

Migration of integrated projects (WinCC flexible)

Introduction

The controllers and HMI devices contained in a project integrated in STEP 7 are linked together by the configuration. The configuration data of WinCC flexible and STEP 7 are also connected. When an integrated project is migrated, the complete project will be migrated with components from WinCC flexible and STEP 7. The connections remain intact.

Note

It is advisable to compile and save an integrated project in WinCC flexible before you migrate it. You can be sure that the data in WinCC flexible and STEP 7 is synchronized if compilation was completed without errors.

Migrating an integrated project

When migrating an integrated project, the same requirements apply for the WinCC flexible component as those for the migration of a non-integrated WinCC flexible project. The objects and properties contained in the WinCC flexible component must be supported by WinCC, for example, the HMI device or the communication driver. The "Online" property must be activated on the configured connection. A connection with deactivated "Online" property is not migrated.

In addition to the requirements for the WinCC flexible component, there are also requirements for the STEP 7 component of the integrated project. The objects and properties contained in the STEP 7 V5.4 SP5 or V5.5 component must be supported in STEP 7. For detailed information, refer to the documentation for STEP 7.

To fully migrate an integrated project and then edit it, the following components must be installed on the PC performing the migration:

- STEP 7 V5.4 SP5 or STEP 7 V5.5
- WinCC flexible 2008 SP2 or WinCC flexible 2008 SP3
- STEP 7

If you only want to save the project in migration format, you can use the migration tool. See Migration principles (WinCC flexible) (Page 162) for additional information.

An integrated project is always fully migrated. If you only want to migrate the WinCC flexible project it contains, you need to separate it from the STEP 7 project before the migration. To separate the project from the integrated form, open the project in STEP 7 V5.4 SP5 or V5.5. Open the WinCC flexible project in the SIMATIC Manager. The project is opened with WinCC flexible. In WinCC flexible, select the menu command "Project > Copy project from STEP 7". WinCC flexible saves a non-integrated copy of the project.

See also

Migration (WinCC flexible) (Page 161)

Migrating an integrated project (Page 193)

Migration principles (WinCC flexible) (Page 162)

Migrating an integrated project

Introduction

When migrating an integrated project, the components from both the WinCC flexible project and the STEP 7 project will be migrated. This means you need to select the project file with the file extension "*.s7p" for migration. During migration, the data is copied from the existing project and migrated to a new project. You cannot migrate to an existing project.

The migration can be started in both the Portal view and the Project view.

You should only migrate a project to a re-started TIA Portal.

If you only want to save the project in migration format, you can use the migration tool. For more information, refer to basics of migration (WinCC flexible).

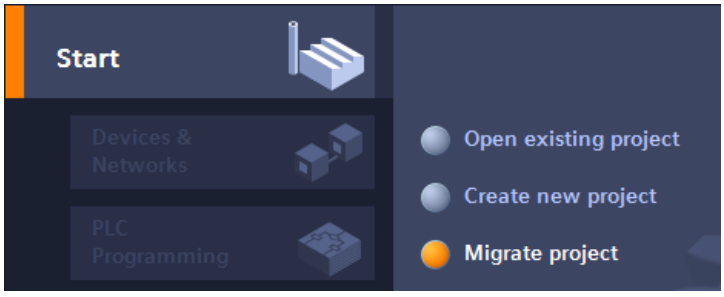
Requirement

- STEP 7 V5.4 SP5 or STEP 7 V5.5 and all option packages used are installed.
- STEP 7 and all option packages used are installed.
- The TIA Portal is restarted.
- No project is open in WinCC.
- An integrated project is available.
- The integrated project is not open.

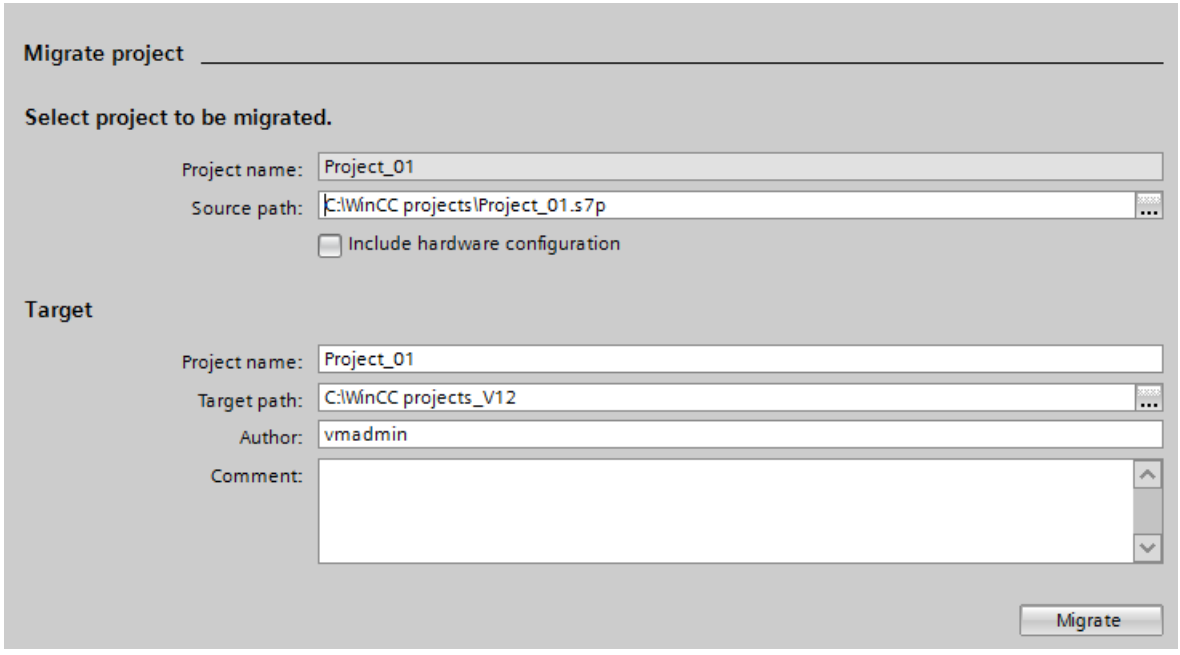
Procedure

To migrate an integrated project in the portal view, follow these steps:

1. Select the action "Start > Migrate Project".



2. In the "Source path" box, navigate to the project you want to migrate.



3. Select the "*.s7p" project file.
4. Change the information for the project to be created, if necessary. For example, change the project name or project path. The data to be migrated is created in the new project.
5. To migrate the project with hardware configuration, activate "Include hardware configuration".

6. Click "Migrate".
   A new project is created and migration of the data is started:
   - The Project view opens.
   - The progress of the migration is shown in a migration window.
   - Warnings and errors about the migration process are displayed in the Inspector window under "Info > General".
   - All information about the migration is saved in a log file.
   - A message is displayed upon completion of the migration. The message contains a link that you can use to open the log file.
7. Once migration is completed, save the project.

Once the migration is complete, you will find a newly created device for each migrated HMI device and controller in the project tree. These devices include the migrated data.

Opening the migration log at a later point in time

The migration log is saved together with the migrated project. You can view the log at a later point in time. Open the log file as follows:

1. Open "Common data > Logs" in the project navigation.
2. Double-click the log file. The migration log opens.

See also

Migration of integrated projects (WinCC flexible) (Page 191)

Migration principles (WinCC flexible) (Page 162)

6.2.7.5 Reference (WinCC flexible)

Migration of data types (WinCC flexible)

Introduction

To harmonize the data types used by PLCs and HMI systems, some data types of the internal HMI tags are renamed. The naming takes place in accordance with IEC conventions. Because only the names change, there are no changes to the internal tags for the configuration.

The following table describes the mapping of the internal data types from WinCC flexible to the data types in WinCC.

Migrating internal data types

The internal data types are mapped as follows during migration:

Internal data types WinCC flexible	Internal data types WinCC
Bool	Bool
Char	SInt
Byte	USInt
Int	Int
UInt	UInt
Long	DInt
ULong	UDInt
Float	Real
Double	LReal
String	WString
DateTime	DateTime

Migrating external data types

See the following pages for how to map the available communication drivers.

See also

Migrating data types of Allen-Bradley DF1 (WinCC flexible) (Page 197)
Migrating data types of Allen-Bradley DF485 (WinCC flexible) (Page 197)
Migrating data types of Allen-Bradley Ethernet IP (WinCC flexible) (Page 198)
Migrating data types of GE Fanuc SNP (WinCC flexible) (Page 198)
Migrating data types of LG GLOFA GM (WinCC flexible) (Page 199)
Migrating data types of Mitsubishi FX (WinCC flexible) (Page 199)
Migrating data types of Mitsubishi Protocol 4 (WinCC flexible) (Page 200)
Migrating data types of Modicon Modbus (WinCC flexible) (Page 201)
Migrating data types of Modicon Modbus TCP/IP (WinCC flexible) (Page 201)
Migrating data types of Omron Hostlink/Multilink (WinCC flexible) (Page 202)
Migrating data types of OPC (WinCC flexible) (Page 202)
Migrating data types of SIMATIC 500/505 DP (WinCC flexible) (Page 203)
Migrating data types of SIMATIC 500/505 serial (WinCC flexible) (Page 203)
Migrating data types of SIMATIC HMI HTTP Protocol (WinCC flexible) (Page 204)
Migrating data types of SIMATIC S5 AS511 (WinCC flexible) (Page 204)
Migrating data types of SIMATIC S5 DP (WinCC flexible) (Page 205)
Migrating data types of SIMATIC S7 200 (WinCC flexible) (Page 205)

Migrating data types of SIMATIC S7 300/400 (WinCC flexible) (Page 206)
Migrating data types of Telemecanique Uni-Telway (WinCC flexible) (Page 209)

Migrating data types of Allen-Bradley DF1 (WinCC flexible)

Migrating data types Allen-Bradley DF1

The data types of the Allen-Bradley DF1 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
ASCII	ASCII
BCD4	UInt
BCD8	UDInt
Bit	Bool
Int	Int
Long	DInt
Real	Real
UInt	UInt
ULong	UDInt

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of Allen-Bradley DF485 (WinCC flexible)

Migrating data types Allen-Bradley DH485

The Allen-Bradley DH485 communication driver is not supported by WinCC, it is replaced by the Allen-Bradley DF1 driver. The data types of the Allen-Bradley DH485 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
ASCII	ASCII
Bit	Bool
Int	Int
Long	DInt
Real	Real
UInt	UInt
ULong	UDInt

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of Allen-Bradley Ethernet IP (WinCC flexible)

Migrating data types Allen-Bradley Ethernet IP

The data types of the Allen-Bradley Ethernet IP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
DInt	DInt
Int	Int
Real	Real
SInt	SInt
String	String
UDInt	UDInt
UInt	UInt
USInt	USInt

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of GE Fanuc SNP (WinCC flexible)

Migrating data types GE Fanuc SNP

The GE Fanuc SNP communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the GE Fanuc SNP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
BCD4	UInt
BCD8	UDInt
Bit	Bool
Byte	USInt
DInt	DInt
Word	UInt
Int	Int
Real	Real
UInt	UInt
DWord	UDInt

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of LG GLOFA GM (WinCC flexible)

Migrating data types LG GLOFA GM

The LG GLOFA GM communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the LG GLOFA GM communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	USInt
DInt	DInt
DWord	UDInt
Int	Int
SInt	SInt
String	WString
Time	UDInt
UDInt	UDInt
UInt	UInt
USInt	USInt
Word	UInt

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of Mitsubishi FX (WinCC flexible)

Migrating data types Mitsubishi FX

The data types of the Mitsubishi FX communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
12 Bit Block	12-Bit Block
16 Bit Block	16-Bit Block
20 Bit Block	20-Bit Block
24 Bit Block	24-Bit Block
28 Bit Block	28-Bit Block
32 Bit Block	32-Bit Block
4 Bit Block	4-Bit Block

Data type in WinCC flexible	Data type in WinCC
8 Bit Block	8-Bit Block
Bit	Bool
Double	DWord
IEEE-Float	Real
String	String
Word	Word

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of Mitsubishi Protocol 4 (WinCC flexible)

Migrating data types Mitsubishi Protocol 4

The Mitsubishi Protocol 4 communication driver is not supported by WinCC, it is replaced by the Mitsubishi MC TCP/IP driver. The data types of the Mitsubishi Protocol 4 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
4 Bit Block	4-Bit Block
8 Bit Block	8-Bit Block
12 Bit Block	12-Bit Block
16 Bit Block	16-Bit Block
20 Bit Block	20-Bit Block
24 Bit Block	24-Bit Block
28 Bit Block	28-Bit Block
32 Bit Block	32-Bit Block
Bit	Bool
DInt	DInt
DWord	DWord
Int	Int
Real	Real
String	String
Word	Word

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of Modicon Modbus (WinCC flexible)

Migrating data types Modicon Modbus

The Modicon Modbus communication driver is not supported by WinCC, it is replaced by the Modicon Modbus RTU driver. The data types of the Modicon Modbus communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	+/- Double
+/-Int	+/- Int
16 Bit Group	16 Bit Group
ASCII	ASCII
Bit	Bit
Double	Double
Float	Float
Int	Int

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of Modicon Modbus TCP/IP (WinCC flexible)

Migrating data types Modicon Modbus TCP/IP

The data types of the Modicon Modbus TCP/IP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	+/- Double
+/-Int	+/- Int
16 Bit Group	16 Bit Group
ASCII	ASCII
Bit	Bit
Double	Double
Float	Float
Int	Int

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of Omron Hostlink/Multilink (WinCC flexible)

Migrating data types Omron Hostlink/Multilink

The Omron Hostlink/Multilink communication driver is not supported by WinCC, it is replaced by the Omron Host Link driver. The data types of the Omron Hostlink/Multilink communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-DEC	Int
+/-LDEC	DInt
ASCII	String
BIN	Bool
BYTE	Byte
DEC	UInt
IEEE	Real
LDEC	UDInt

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of OPC (WinCC flexible)

Migrating data types OPC

The data types of the OPC communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	VT_BOOL
Byte	VT_UI1
Char	VT_I1
Date	VT_DATE
Double	VT_R8
DWord	VT_UI4
Float	VT_R4
Long	VT_I4
Short	VT_I2
String	VT_BSTR
Word	VT_UI2

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of SIMATIC 500/505 DP (WinCC flexible)

Migrating data types SIMATIC 500/505 DP

The SIMATIC 500/505 DP communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the SIMATIC 500/505 DP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	DInt
+/-Int	Int
ASCII	WString
Bit	Bool
Double	UDInt
Int	UInt
Real	Real

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of SIMATIC 500/505 serial (WinCC flexible)

Migrating data types SIMATIC 500/505 seriell

The SIMATIC 500/505 seriell communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the SIMATIC 500/505 seriell communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	DInt
+/-Int	Int
ASCII	WString
Bit	Bool
Double	UDInt
Int	UInt
Real	Real

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of SIMATIC HMI HTTP Protocol (WinCC flexible)

Migrating data types SIMATIC HMI HTTP Protocol

The data types of the SIMATIC HMI HTTP Protocol communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	USInt
Char	SInt
DateTime	DateTime
Double	LReal
Float	Real
Int	Int
Long	DInt
String	WString
UInt	UInt
ULong	UDInt

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of SIMATIC S5 AS511 (WinCC flexible)

Migrating data types SIMATIC S5 AS511

The SIMATIC S5 AS511 communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the SIMATIC S5 AS511 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bit in D	Bool
Bit in W	Bool
DF	DInt
DH	UDInt
KC	WString
KF	Int
KG	Real
KH	UInt
KM	UInt
KT	UDInt
KY	UInt
KZ	UInt

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of SIMATIC S5 DP (WinCC flexible)

Migrating data types SIMATIC S5 DP

The SIMATIC S5 DP communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the SIMATIC S5 DP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bit in D	Bool
Bit in W	Bool
DF	DInt
DH	UDInt
KC	WString
KF	Int
KG	Real
KH	UInt
KM	UInt
KT	UDInt
KY	UInt
KZ	UInt

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of SIMATIC S7 200 (WinCC flexible)

Migrating data types SIMATIC S7 200

The data types of the SIMATIC S7 200 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	Byte
Char	Char
DInt	DInt
DWord	DWord
Int	Int
Real	Real

Data type in WinCC flexible	Data type in WinCC
StringChar	StringChar
Timer	Timer
Word	Word

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of SIMATIC S7 300/400 (WinCC flexible)

Migrating data types SIMATIC S7 300/400

The data types of the SIMATIC S7 300/400 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	Byte
Char	see below
Counter	see below
Date	Date
Date and Time	Date_And_Time
DInt	DInt
DWord	DWord
Int	Int
Real	Real
String	String
StringChar	see below
Time	Time
Time of Day	Time_Of_Day
Timer	see below
Word	Word

Special considerations for some data types

There are special considerations to be made when migrating external tags that contain data types of a SIMATIC S7-300/400 PLC.

Mapping of the S7 data type "Char"

The S7 data type "Char" is a data type for mapping characters according to the specification. However, since this data type is often used for reading and writing numerical values, it is mapped in WinCC to the S7 data type "Byte". If this should be the case during migration, an alarm will appear in the output window.

If the S7 data type "Char" is used for numerical values and negative numbers were configured at the point of use, the result is an error in mapping to the S7 data type "Byte". The S7 data type "Byte" cannot map any negative numbers. You have to adapt the configuration accordingly to correct the error. Use a signed data type, such as the data type "Int", for processing positive and negative numerical values.

If the S7 data type "Char" is used for mapping characters, you must change the configuration after migration. To represent characters, use the data type "String".

When an integrated project is migrated, the data type "Char" in WinCC is also migrated to the data type "Byte". With a connected PLC tag, the data type "Char" remains "Char". As a result of changing the data type of the HMI tag, symbolic addressing of the tags in question is not migrated. After migration. the tags are interconnected by absolute addresses and continue to work. If you want to restore symbolic addressing, you have to change the configuration accordingly after the migration.

Mapping an array of the S7 data type "Char"

An array of the S7 data type "Char" is mapped to an array of the data type "Byte" during migration.

If an array of the S7 data type "Char" is used for numerical values and negative numbers were configured at the point of use, the result is an error in mapping to an array of the S7 data type "Byte". The S7 data type "Byte" cannot map any negative numbers. You have to adapt the configuration accordingly to correct the error. Use a signed data type, such as the data type "Int", for processing positive and negative numerical values.

Mapping of the S7 data type "Counter"

An external tag with the S7 data type "Counter" with counter address is mapped to the S7 data type "Counter". The address will be retained.

If an external tag with the S7 data type "Counter" addresses a data block or a bit memory address, it is is mapped to the S7 data type "Word". The address will be retained. The migration sets the coding to "SimaticBCDCounter".

The S7 data type "Counter" has a value range of 0-999. When supplied by the S7 data type "Word" the value range may be exceeded on the PLC side. Ensure that you are observing the value range.

Example:

WinCC flexible

Tag	S7 data type	Address	Comment
Counter_Actual_Value	Counter	C10	BCD coded counter value
Counter_Setpoint_Value	Counter	DB10.DBW200	BCD coded counter value
Counter_Setpoint_Value#2	Counter	MW20	BCD coded counter value

WinCC

Tag	S7 data type	Address	Coding	Comment
Counter_Actual_Value	Counter	%C10	<Standard>	BCD coded counter value
Counter_Setpoint_Value	Word	%DB10.%DBW200	SimaticBCDCounter	BCD coded counter value
Counter_Setpoint_Value#2	Word	%MW20	SimaticBCDCounter	BCD coded counter value

Mapping of the data type "StringChar"

In WinCC there is no corresponding data type to which the "StringChar" data type can be mapped. Mapping in WinCC depends on the property "Length" of the S7 data type.

A tag of the "StringChar" data type with the "Length" property > 1 is migrated to an array of the S7 data type "Char". The length of the array corresponds to the length of the originally configured data type "StringChar".

If the property "Length" = 1, the data type in WinCC is migrated to an array of the S7 data type "Char" with length = 1. The expression for an array with an element is "Array[0 ..0] of Char".

Mapping of the S7 data type "Timer"

An external tag with the S7 data type "Timer" with timer address is mapped to the S7 data type "Timer". The address will be retained.

If an external tag with the S7 data type "Timer" addresses a data block or a bit memory address, it is is mapped to the S7 data type "S5 Time". The address will be retained.

Example:

WinCC flexible

Tag	S7 data type	Address	Comment
Timer_Actual_Value	Timer	T10	BCD coded timer value
Timer_Setpoint_Value	Timer	DB10.DBW200	BCD coded timer value
Timer_Setpoint_Value#2	Timer	MW20	BCD coded timer value

WinCC

Tag	S7 data type	Address	Comment
Timer_Actual_Value	Timer	%T10	BCD coded timer value
Timer_Setpoint_Value	S5Time	%DB10.%DBW200	BCD coded timer value
Timer_Setpoint_Value#2	S5Time	%MW20	BCD coded timer value

See also

Migration of data types (WinCC flexible) (Page 195)

Migrating data types of Telemecanique Uni-Telway (WinCC flexible)

Migrating data types Telemecanique Uni-Telway

The Telemecanique Uni-Telway communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the Telemecanique Uni-Telway communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Int	Int
+/-Long	DInt
ASCII	WString
Bool	Bool
Float	Real
Int	UInt
Long	UDInt

See also

Migration of data types (WinCC flexible) (Page 195)

6.2.8 Migrating integrated projects

6.2.8.1 Migrating an integrated project

Introduction

In integrated projects, you use SIMATIC controllers and WinCC components together in a project. When an integrated project is migrated, the complete project will be migrated with components from WinCC and STEP 7. Configured connections between control and visualization remain intact.

Migrating an integrated project

When migrating an integrated project, the same requirements apply for the STEP 7 component as those for migration of a non-integrated STEP 7 project. The objects and properties contained in the WinCC component must also be supported in WinCC (TIA Portal).

For an operating station (OS) to be migrated, it has to be located below a PC station and the WinCC application in the project tree of SIMATIC Manager. The following figure shows the assignment of the operating station within the initial project:

Additional migration requirements for integrated projects can be found in the documentation for WinCC.

Also note that the initial project must be compiled before the migration.

In order to fully migrate an integrated project, the following components must be installed on the programming device/PC for the migration:

- STEP 7 V5.4 SP5 or STEP 7 V5.5
- WinCC V7.2 with the latest update or WinCC flexible 2008 SP2 and SP3

To be able to fully post-edit an integrated project, the latest version of the following components must be installed on the PC for post-editing:

- STEP 7 Professional
- WinCC Basic, WinCC Comfort/Advanced or WinCC Professional, depending on the components used

Using the migration tool

It is necessary to use the migration tool under the following circumstances:

- The initial project is not located on the same programming device/PC as the installation of the TIA Portal.
- SCADA devices are included in the initial project. These can only be migrated with the migration tool.
- WinCC Professional V13 and STEP 7 with WinCC V7.2 cannot be installed on the same programming device/PC. Therefore, integrated projects with WinCC V7.2 parts must be prepared for migration using the migration tool.

Migration of the STEP 7 part of an integrated project

An integrated project is always fully migrated. Individual components cannot be migrated on their own. You can only migrate the included STEP 7 project alone, if you have previously deleted all HMI stations in the SIMATIC stations in the SIMATIC Manager and then recompiled the project in NetPro.

Alternatively, you can open the project in an installation of STEP 7 V5.4 SP5 or V5.5 without an installation of WinCC. Then, save the project again and select the "Reorganize" function during saving. The WinCC parts are then automatically removed when the copy is saved.

You can then migrate the STEP 7 project without the WinCC project.

Migration of an integrated project with the hardware configuration

In integrated projects, HMI devices are migrated even if the hardware configuration is not included in the migration. The STEP 7 part of the hardware configuration, including network configurations, and connections and interrupts, is migrated only if you include the hardware configuration in the migration. Otherwise, unspecified modules will be created for the STEP 7 devices and you will need to convert them into suitable modules after the migration.

HMI modules that are plugged into a PC station are converted to a separate Station during the migration. If you perform the migration without including the hardware configuration, the migrated project then contains a non-specified SIMATIC PC Station and a SIMATIC PC Station with the HMI devices. References to HMI devices are not imported during migration. When the hardware configuration is included, the migrated project contains two separate stations: the HMI Station and the PC Station.

Storage location of an integrated WinCC project

If you migrate an integrated project, the HMI part of it must be on the same PG/PC as the STEP 7 part of the project. If the HMI part is on a different PG, then only the STEP 7 part will be migrated.

Unsupported objects

The following components are not supported for migration:

- STEP 7 multiproject
  A STEP 7 multiproject cannot be migrated. Migration will be canceled.
- Central Archive Server - CAS
  If a CAS is part of an integrated project, then the migration will be carried out but the CAS data will not be migrated.

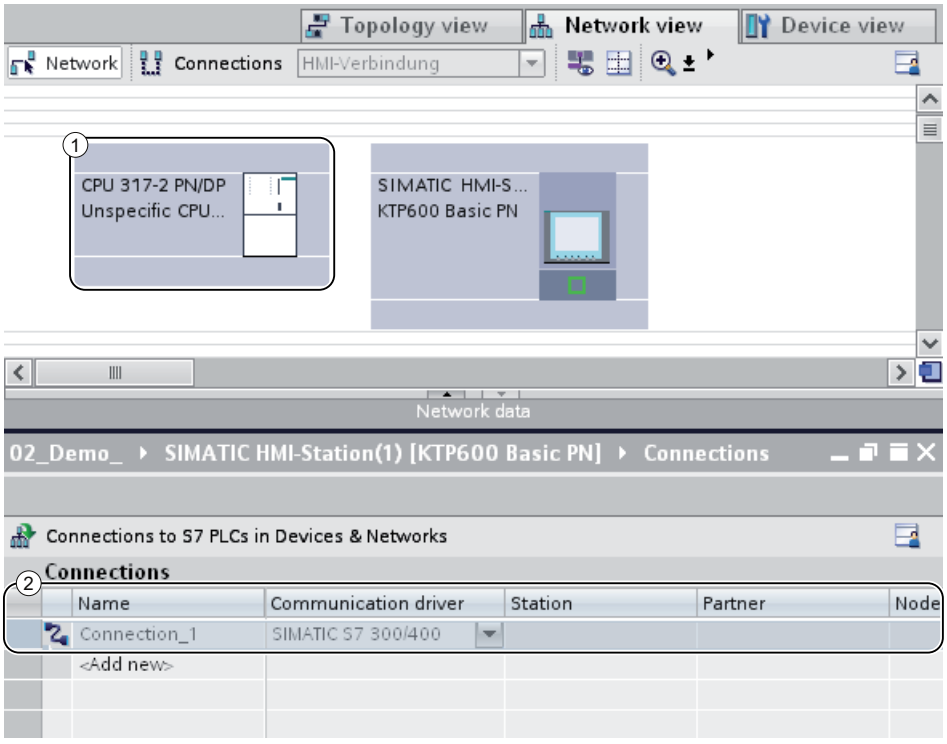
See also

Post-editing integrated projects (Page 211)

6.2.8.2 Post-editing integrated projects

If you have migrated an integrated project without hardware configuration, unspecified CPUs are used instead of the CPUs of the original project. Since no connection can exist between an unspecified CPU and an HMI device, connections from the source project are also imported only unspecified.

The following figure shows the state after a Migration without hardware configuration in an example project:



① The original CPU 317-2 PN/DP was replaced with an unspecified CPU during migration.

② The link between the CPU and HMI device is also unspecified and must be renewed.

Procedure

To continue to use an integrated project after the migration, follow these steps:

1. Convert the unspecified devices into suitable devices again.
2. Restore the integrated HMI connection between the HMI device and the PLC.
3. Connect all HMI tags to the newly created integrated connection.
4. Restore the connection between HMI tags and PLC tags.
5. Delete the non-integrated HMI connection.

In the following chapters a sample project is used to describe the individual steps in more detail.

See also

Converting unspecified CPUs into specified CPUs (Page 213)

Creating an integrated HMI connection (Page 214)

Re-linking HMI tags (Page 216)

Deleting an unspecified connection (Page 217)

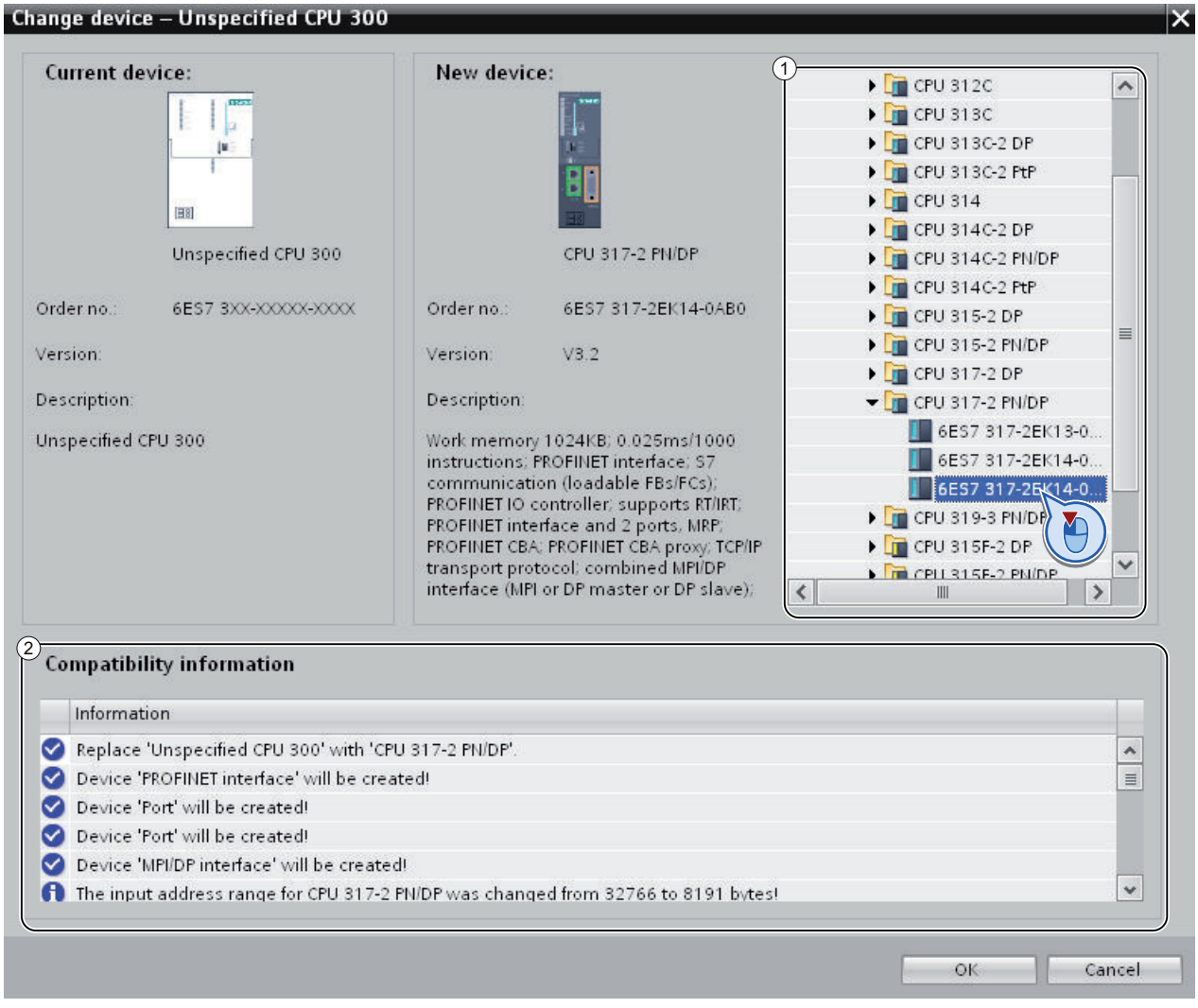
6.2.8.3 Converting unspecified CPUs into specified CPUs

The first step after the migration without hardware configuration is the conversion of the unspecified CPUs into specified CPUs. Unspecified CPUs are placeholders for certain CPUs from the hardware catalog that are not currently known. You can define general parameters and home the CPUs already in the user program. However, the project is not fully functional until the unspecified CPU has been specified.

Specifying a CPU using module replacement

To use module replacement to specify an unspecified CPU, follow these steps:

1. Select the unspecified CPU in the network or device view.
2. Select the "Replace device" command in the shortcut menu.
   The "Replace device" dialog opens.



3. Under "New device" in the tree structure, select the module with which you want to replace the unspecified CPU. (Area 1)
   "Compatibility information" provides you with information on the extent to which the selected CPU is compatible with the configuration in source project. (Area 2)
4. Click "OK".
5. Perform the above-described steps for all unspecified CPUs.

See also

Creating an integrated HMI connection (Page 214)

6.2.8.4 Creating an integrated HMI connection

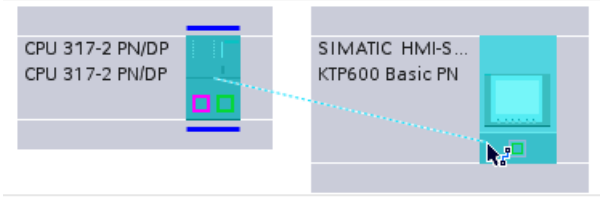
After you have specified the unspecified CPU, establish the connection to the HMI-device.

Procedure

To create a connection graphically, follow these steps:

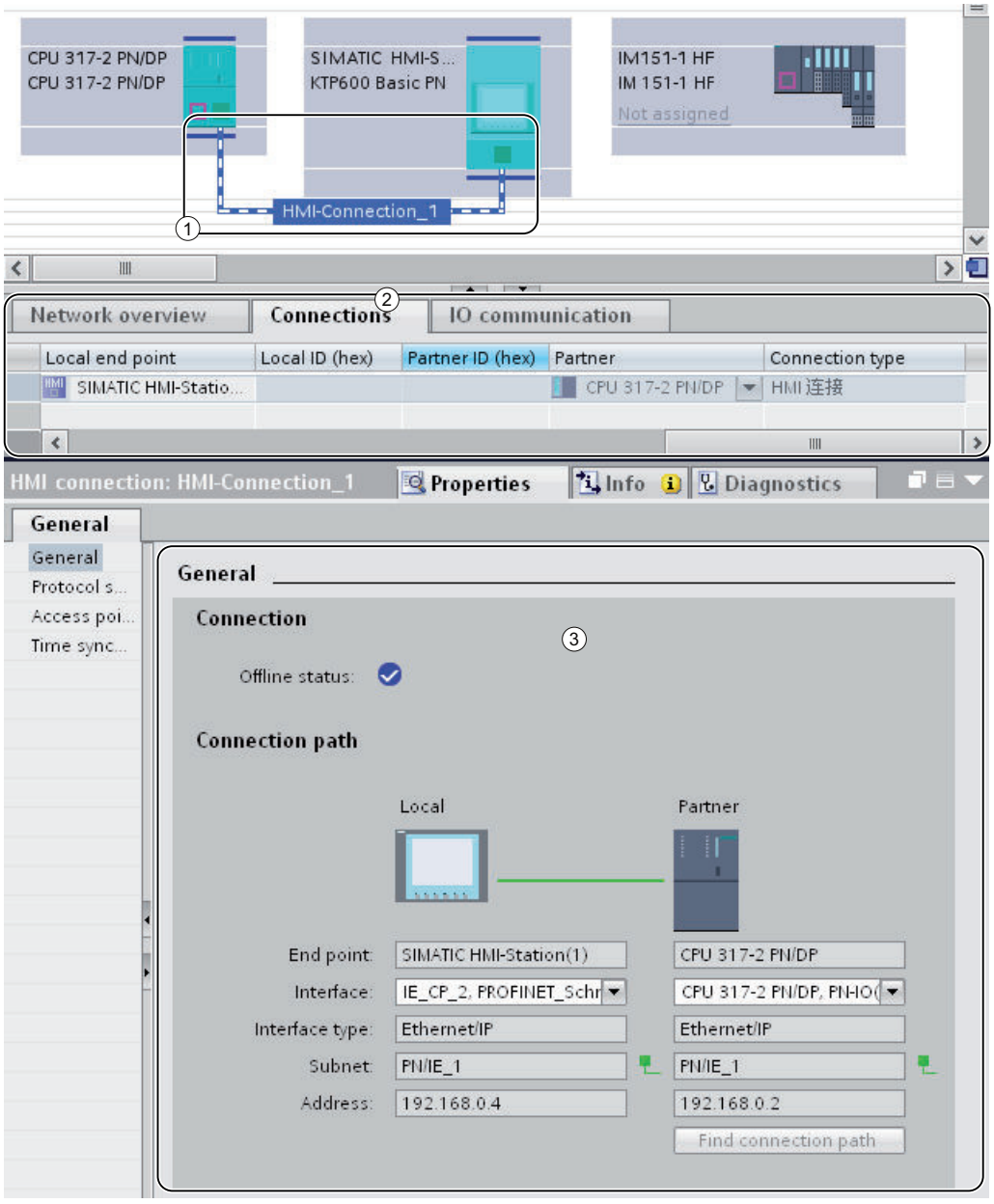
1. On the toolbar, click the "Connections" icon. This activates connection mode.

2. Select the connection type "HMI connection" in the adjacent drop-down list.
   The network view highlights in color all CPUs and HMI devices that can be used for an HMI connection.
3. You can now have the connection path automatically determined, or explicitly select a connection path via specific interfaces:
   - Allow connection path to be automatically determined
     Select the source CPU for a connection. Drag the mouse to the target components. Confirm the connection endpoint with another mouse click.
     Alternatively: While holding down the shift button, select the target components and with the right mouse button select the "Add new connection" command.
   - Selecting an explicit connection path from interface to interface
     Click on the subnet interface in the device for which you want to create a connection. Hold down the mouse button, drag the cursor to the relevant interface in the target device and then release the mouse button.



Result

The following figure shows the state after the integrated connection has been created:



① An integrated HMI connection is created and highlighted in the network view.
② The connection is shown in the connection table of the components.
③ The connection can be edited in the connection properties.

See also

Re-linking HMI tags (Page 216)

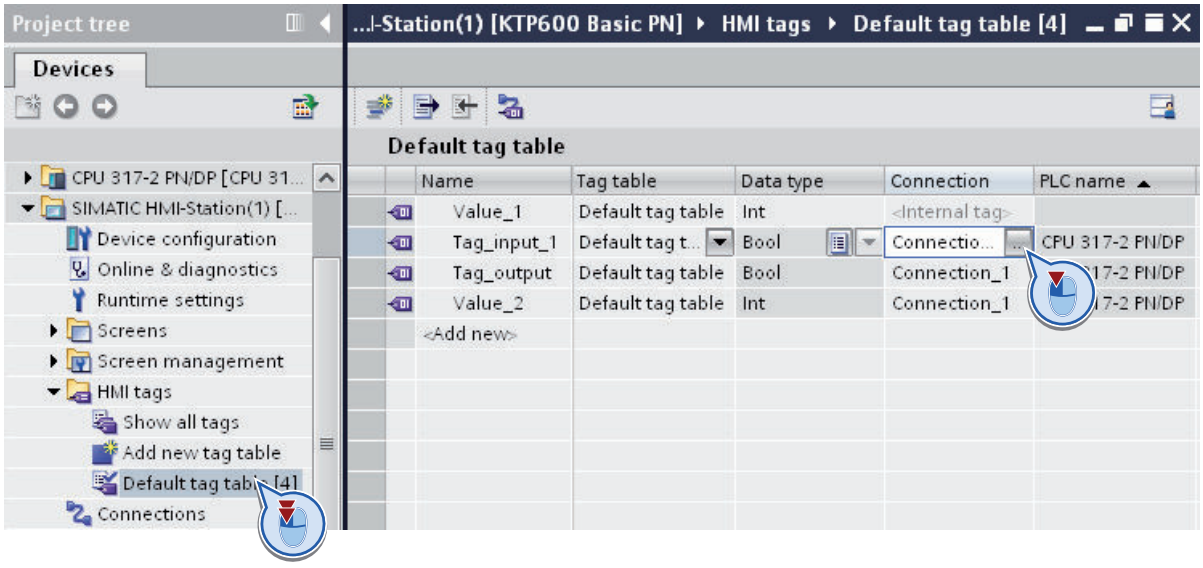
6.2.8.5 Re-linking HMI tags

When you have created a new HMI connection between the CPU and HMI device, you have to assign the existing HMI tags to the new connection. Perform the following steps for each line in the relevant tag table.

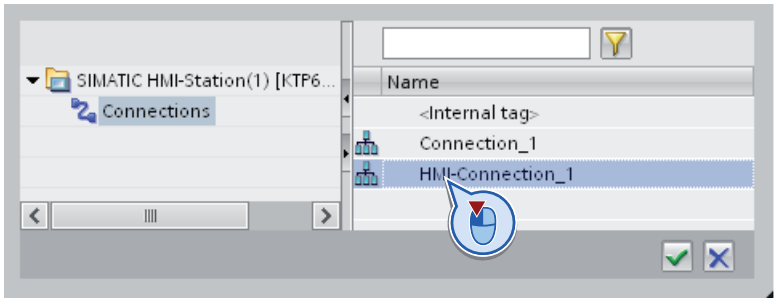
Procedure

To re-link HMI tags, follow these steps:

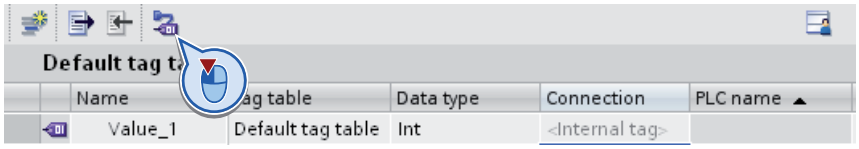
1. In the project tree, navigate to the HMI tags and double-click the relevant tag table to show this in the work area.
   The tag table opens.



2. Click the " ... " button in the "Connection" column.
   A dialog box for selecting the connection opens.
3. Select the newly established HMI connection.



4. Click the "✓" button to apply the selected connection.
5. On the toolbar, click the "Re-connect PLC tag" button.



See also

Deleting an unspecified connection (Page 217)

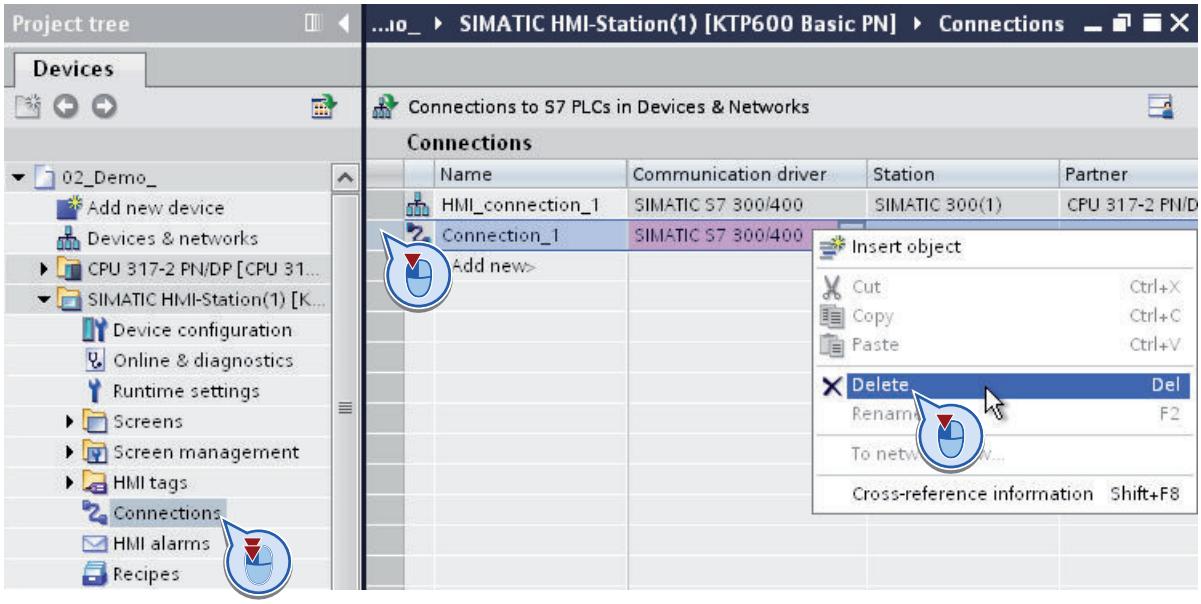
6.2.8.6 Deleting an unspecified connection

Finally, you can remove unspecified connections that still remain from the source project.

Procedure

To delete unspecified connections, follow these steps:

1. In the project tree, open the HMI device and double-click the "Connections" entry. The connection table opens.



2. Select the row with the old connection in the table.
3. Select the "Delete" command in the shortcut menu of the connection line.
4. Perform the above-described steps for all unspecified connections of the source project.

6.3 Migrating S7-1200 to firmware as of V4

6.3.1 Basic information on upgrading to V4

Introduction

If you have used a CPU with firmware version V3 in your project and want to upgrade to a CPU with firmware V4.0 or later, you can easily replace the device.

The TIA Portal offers the "Change device" function for this purpose. The project remains unchanged when the device is replaced. You can continue using the programs that you created with firmware version V3.

Rules

The following basic rules apply when replacing a device:

- Replacing a device is only possible if the project was created based on a CPU with firmware version V3.0. If your project was created based on firmware version V1.0 or V2.0, create a new CPU with firmware version V3.0 offline in the project, and copy your program to this CPU.
- It is not possible to replace a V4 CPU with a V3 CPU. If you want to continue using the existing V3 CPU, create a copy of this CPU before replacing the device.
- The program cannot be transferred to the new CPU via a memory card. Instead, use the "Change device" function, which is described in the following sections.

HMI panels

Configured HMI panels are treated differently during device replacement depending on the firmware version of the panel and the communication mode.

The following table shows the HMI connections that are supported with the migration:

Firmware version of the panel	PUT/GET communication	Migration to V4
V11 or later	No	S7-1200 does not support this configuration. Upgrade the firmware of the HMI panel to V12.0. Then compile and load the configuration.
V11 or later	Yes	S7-1200 supports this configuration. The connection is established automatically while you compile and load the project after replacing the device.
V12 or later	No	S7-1200 supports this configuration. The connection is established automatically while you compile and load the project after replacing the device.

During compilation of the program, you receive specific information on migrating the HMI panel.

Note

HMI TP 177B 4"

The HMI TP 177B 4" with firmware version V11.0.2 cannot be operated with S7-1200 V4. Replace the panel with a new device, if necessary.

S7-1200 expansion modules

If you are already using the following centrally plugged S7-1200 modules in your system, you must perform a firmware update for these modules in order to guarantee operation with S7-1200 V4.

- ASi - Master - CM 1243
- DP - Master - CM 1243-5
- WAN CP - CP1243-1

Newly shipped S7-1200 modules have the latest firmware installed ex works.

Protected blocks

Blocks equipped with know-how protection or copy protection cannot be converted to V4. If the project contains protected blocks, you must remove the protection prior to migration.

If these are supplied blocks and you do not know the password, ask your supplier either for the password or for a V4-compatible block.

Warning

Preventing personal injury and material damage

Changes are made to the program during device replacement in some cases. Therefore, thoroughly check the program in a test environment after replacing the device and before putting it in operation.

Note

Additional support

You can find the latest FAQs about migrating to S7-1200 V4 (http://support.automation.siemens.com/WW/view/en/82140966) in Siemens Industry Online Support.

If you need further assistance with migrating to S7-1200 V4, please contact SIMATIC Customer Support.

See also

Migrating to V4 (Page 220)

Special considerations after migrating to V4 (Page 221)

6.3.2 Migrating to V4

Requirement

- A CPU with firmware version V3 is available in the project.
- The project contains no protected blocks.

Procedure

Proceed as follows to replace a CPU:

1. Select the V3 CPU you want to replace.
2. Select the "Change device" command in the shortcut menu.
   The "Change device" dialog box appears.
3. Under "New device" in the tree structure, select the V4 CPU with which you want to replace your current V3 CPU.
4. Click "OK".
   The existing CPU is replaced by the new one.
5. Select the new CPU and select the "Compile > Hardware and software (only changes)" command in the shortcut menu.
   The device configuration and the user program are compiled again.
6. Optional: If necessary, apply know-how protection or copy protection to individual blocks in the program.
7. Select the new CPU and select the "Download to device > Hardware and software (only changes)" command in the shortcut menu.
   The device configuration and the user program are loaded into the new CPU.
   This completes the device replacement.

Warning

Preventing personal injury and material damage

Changes are made to the program during device replacement in some cases. Therefore, thoroughly check the program in a test environment after replacing the device and before putting it in operation.

Note

Additional support

You can find the latest FAQs about migrating to S7-1200 V4 (http://support.automation.siemens.com/WW/view/en/82140966) in Siemens Industry Online Support.

If you need further assistance with migrating to S7-1200 V4, please contact SIMATIC Customer Support.

See also

Basic information on upgrading to V4 (Page 217)

Special considerations after migrating to V4 (Page 221)

6.3.3 Special considerations after migrating to V4

Functional changes in V4

S7-1200 V4 offers significantly enhanced functionality. The most important functional changes that you need to consider after migrating from V3 are described briefly below.

You can find more information on S7-1200 in the "SIMATIC S7-1200 Programmable controller" system manual.

See also:

TIA Portal in Siemens Industry Online Support (http://support.automation.siemens.com/WW/view/en/65601780)

Organization blocks

With S7-1200 V4, you can specifically set the interruptibility of each organization block used. When a device is replaced, all the organization blocks are configured as non-interruptible, to ensure that the executability of your V3 program remains unchanged. The OB priorities from the V3 program also remain unchanged. After migration, you can change the settings for priority and interruptibility as needed.

The behavior of diagnostic interrupts in V4 has changed as follows:

In V3, the start information always contained information on the triggering module, including the channel number. In V4, this information is only generated for a pending diagnostics event. If no diagnostic event is pending, for example, because the fault has already been corrected, only the triggering module is indicated.

Access levels

S7-1200 V4 offers an extended access level concept. The following table shows how the protection levels of the V3 firmware are indicated in V4:

V3 protection level	V4 access level	Meaning
No protection	Full access (no protection)	Unrestricted access without password protection.
Read-only	Read access	HMI access and unimpeded communication between CPUs without password protection. A password is required for changes (write access) in the CPU and for changing the operating mode of the CPU (RUN/STOP).
Write/read protection	HMI access	HMI access and unimpeded communication between CPUs without password protection. A password is required to read the data in the CPU, for changes (writing) in the CPU, and for changing the operating mode of the CPU (RUN/STOP).
-	No access (complete protection)	No access without password input. A password is required for HMI access, for reading the data in the CPU, for changing (writing) data in the CPU, and for changing the operating mode of the CPU (RUN/STOP).

Instruction libraries

After migration to S7-1200 V4, instructions from the libraries of the firmware version V3 are still available. This ensures that you can continue to use your program unchanged. In addition, S7-1200 V4 offers many new instructions which are also compatible with the instructions of the S7-1500.

You can find more information on the instruction libraries of S7-1200 in the "SIMATIC S7-1200 Programmable controller" system manual.

See also:

TIA Portal in Siemens Industry Online Support (http://support.automation.siemens.com/WW/view/en/65601780)

Motion Control

When a device is replaced, the Motion Control objects from the libraries of the firmware versions V1 and V2 are replaced with the corresponding objects from the V3 libraries. The objects from V3 libraries are compatible so that you can continue to use the programs unchanged.

The libraries of the S7-1200 V4 offer many new Motion Control functions, which are compatible with the functions of the S7-1500. If you want to use V4 libraries, select them on the "Instructions" task card after replacing the device.

You can find more information on the new Motion Control functions in the "SIMATIC S7-1200 Programmable controller" system manual.

See also:

TIA Portal in Siemens Industry Online Support (http://support.automation.siemens.com/WW/view/en/65601780)

Web server

The following settings for operation via a web server are transferred from the V3 CPU to the V4 CPU during device replacement:

- Activate web server on this module
- Permit access only with HTTPS

If you want to operate the V4 CPU via a web server, you need to set up user accounts with assigned user rights in the user management. Only the standard web pages are available to standard users without any additional rights.

Note

Additional support

You can find the latest FAQs about migrating to S7-1200 V4 (http://support.automation.siemens.com/WW/view/en/82140966) in Siemens Industry Online Support.

If you need further assistance with migrating to S7-1200 V4, please contact SIMATIC Customer Support.

Communication via PUT/GET

Communication via PUT/GET is enabled after the device replacement. Note that the new integrated connection types offer a security standard higher than PUT/GET communication. If you do not use PUT/GET communication, you should disable it.

See also

Basic information on upgrading to V4 (Page 217)

Migrating to V4 (Page 220)

6.4 Programming recommendations

6.4.1 The new S7-1500 CPU functions at a glance

Higher performance

The S7-1500 represents a CPU series that provides much higher performance than the CPUs of the S7-300/400 series. When programming with STEP 7 V5.x, you were probably used to working with programming methods such as absolute addressing to achieve higher performance from the CPU and leaner program code.

Due to the high performance that the S7-1500 provides, these programming methods are made obsolete.

In the paragraphs below we would like to introduce some new programming options of the S7-1500.

Universal symbolism

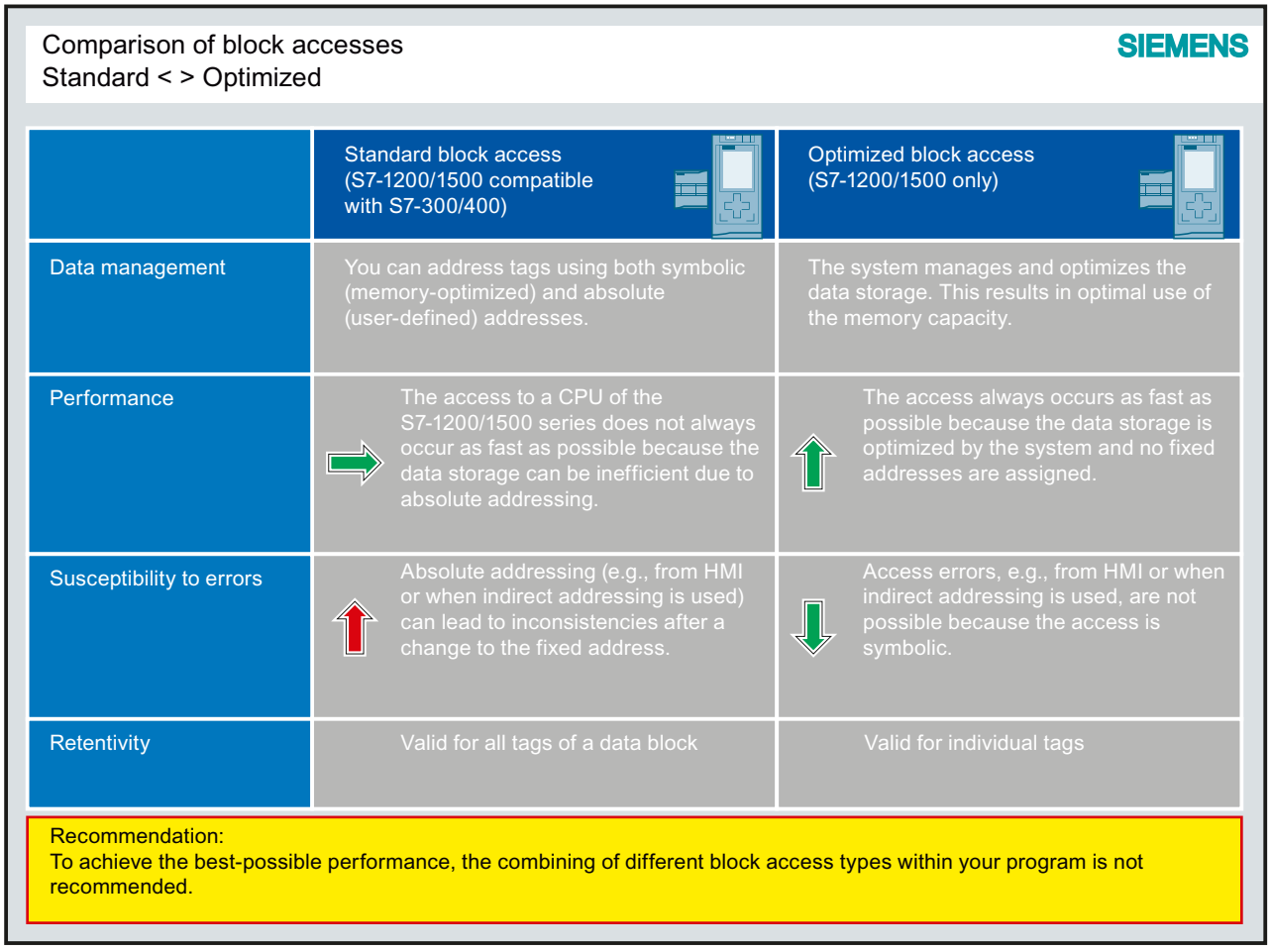
The S7-1500 allows you to use the symbolism throughout the entire project. Using the auto-complete function, you are given context-dependent support for programming with symbols within the programming editors. The data elements, for example those in a data block, are assigned only a symbolic name in the declaration but no fixed address within the data block. This allows you to fully exploit the high performance of the S7-1500 when accessing these data elements. The absolute addresses of operands need not be known and access errors are avoided.

Your program code will be clearer due to the symbolism and you have to comment less. All points of use are automatically updated when a correction is made to the symbolism.

An example of the use of universal symbols is available at: Symbolic addressing (Page 249)

Optimized block access

With optimized block access, the declared data elements are arranged automatically in the available memory area of the block in a way that makes optimal use of its capacity. The data is structured and saved in way that is optimal for the CPU used. The storage is left to the system. The data elements are assigned only a symbolic name in the declaration by which the tag within the block can be addressed. This allows you to increase the performance of the CPU. Access errors, from the HMI, for example, are not possible.



You can find additional information on blocks with optimized access under "See also".

New data types

The new data types LWORD, LINT, ULINT, LTIME, LTOD, LDT and the array (32-bit limit) offer much higher calculation accuracy when using mathematical functions. In the area of implicit and explicit data type conversion, you have more options in comparison to the CPUs of the S7-300/400 series.

You can find additional information on the new data types under "See also".

PLC data types

PLC data types (UDT) are data structures that you define and that can be used multiple times within the program. The structure of a PLC data type is made up of several components, each of which can contain different data types. You define the type of components in the declaration of the PLC data type.

You can use the PLC data type as a base data type for defining tags and as a template for creating global data blocks. If you later make changes to the PLC data type, the changes are automatically updated at all points of use.

You can also symbolically access individual elements of an array within a PLC data type.

You can find additional information on the new data types under "See also".

You can find an example for using PLC data types under: Using PLC data types (UDT) (Page 286)

Uniform instructions in all programming languages

You are provided with a uniform set of instructions in all programming languages (LAD, FBD, STL, SCL, and GRAPH).

Slice access

Slice access gives you the option of specifically addressing areas within declared tags. You can implement symbolic access to a single bit up to the level of the tag. The single bit is then addressed absolutely.

You can find additional information on slice access under "See also".

Indirect addressing

Indirect addressing offers you the option of addressing operands whose address is not calculated until runtime. All programming languages provide general methods for indirect addressing, for example, via POINTER. In the SCL programming language, you can also use PEEK and POKE instructions.

You can find additional information on indirect addressing under "See also".

Additional programming recommendations

You can find more information about programming recommendations and a programming guide in the Siemens Industry online support under:

- FAQs: Programming recommendations (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&Datakey=47071380&extranet=standard&groupid=4000002&viewreg=WW&nodeid0=29156492&objaction=csopen)
- Background and system descriptions in the Programming Guideline (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=100&lang=en&referer=%2fWW%2f&func=cslib.cssearch&nodeid0=4000024&viewreg=WW&siteid=csius&extranet=standard&groupid=4000002&objaction=cssearch&content=adsearch%2Fadsearch%2Easpx)

See also

PLC data types (Page 1954)

6.4.2 Flexibly using enable output ENO

Advantages

You can to detect runtime errors and avoid program termination using the EN/ENO mechanism for each instruction and block call. Overflow, for example, is reported via the ENO enable output for mathematical functions.

In STEP 7 TIA Portal, the ENO enable output is disabled by default in the programming languages LAD and FBD. If needed, you can activate the enable output and thereby deliberately target the instructions for which you want to have error evaluation.

This gives you the following advantages:

- The performance increases with ENO disabled.
- Runtime errors do not cause the CPU to go to STOP when ENO is enabled.

Procedure in STEP 7 TIA Portal

Proceed as follows to activate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to activate the EN/ENO mechanism.
2. Select the "Generate ENO" command from the shortcut menu.
   The ENO value is generated for the instruction. Other instructions are inserted with the enable output.

The tables below list the instructions for which ENO enable output can be disabled:

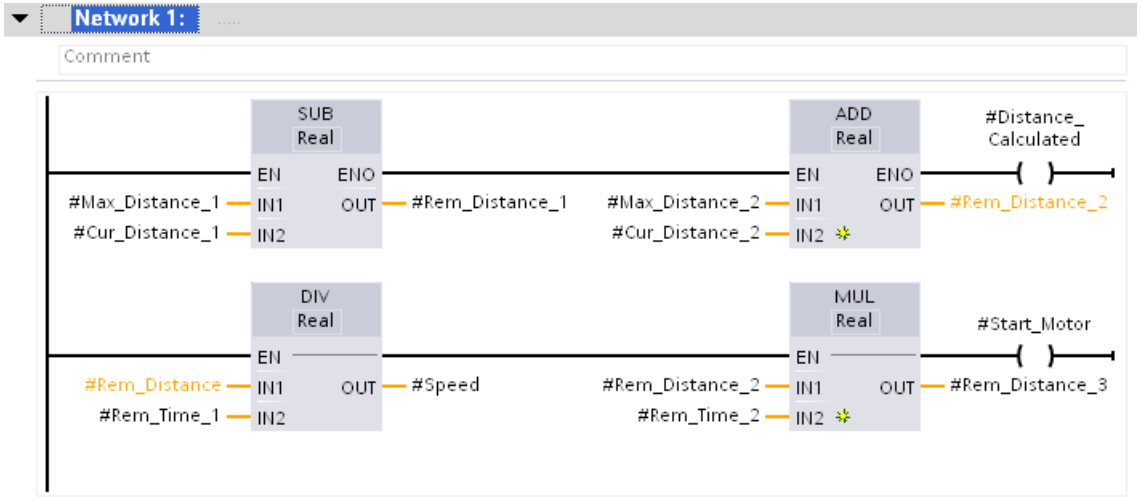
Basic instructions	
Math functions	ADD, SUB, MUL, DIV, MOD, INC, DEC, ABS, NEG, SQR, SQRT, LN, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN, FRAC, EXPT, MIN, MAX, LIMIT, CALCULATE
Move operations	MOVE, SWAP, MOVE_BLK, UMOVE_BLK, FILL_BLK, UFILL_BLK, MOVE_BLK_VARIANT
Conversion operations	CONVERT, ROUND, CEIL, TRUNC, FLOOR, NORM_X, SCALE_X
Word logic operations	AND, OR, XOR, INV, DECO, ENCO
Shift and rotate	SHR, SHL, ROR, ROL

Extended instructions	
String + Char	CONCAT, LEFT, RIGHT, MID, DELETE, INSERT, REPLACE, FIND, LEN, S_CONV
Date and time	T_CONV

You can find additional information on the EN/ENO mechanism in each programming language under "See also".

Programming example

The following example shows you how to use instructions with the ENO enable output enabled and disabled:



If you have activated the ENO enable output, for example with the SUB instruction, all subsequent instructions are also applied with an activated enable output. If an arithmetic error occurs during the execution of the SUB instruction, the ADD instruction is not executed.

The ENO enable output is disabled for the DIV instruction in the second branch. If a runtime error occurs during execution, the MUL instruction is executed anyway.

6.4.3 Querying and fixing errors in the program code

Error handling in the TIA Portal

Various error scenarios can occur within the program code during runtime. These can involve, for example, an access error or an overflow in mathematical operations. If you do not catch such a runtime error using a program code sequence within your program, the CPU reacts as follows in the event of an error, depending on the model:

- CPU S7-1200:
  - The CPU remains in RUN mode and writes an entry to the diagnostics buffer.
- CPU S7-1500:
  - When a programming error occurs, the CPU goes to STOP mode and writes an entry to the diagnostics buffer.
  - With an I/O access error, the CPU remains in RUN mode and writes an entry to the diagnostics buffer.

To be able to respond appropriately to possible errors, in addition to global error handling (for example, the use of the organization blocks "Programming error OB" or "I/O access error OB"), we recommend that you implement local error handling in your program (for example, use an EN/ENO mechanism, the parameters RET_VAL, STATUS and ERROR, or the instructions "GET_ERROR"/"GET_ERR_ID").

Note

STL programming language

In STL, you use the BR bit of the status word instead of the EN/ENO mechanism.

Global error handling intervenes at the end of each program cycle, while local error handling is capable of intervening immediately after an error occurs.

Note

BR bit and EN/ENO mechanism as first indicator

A first indicator for an error can be either the BR bit of the status word or the ENO enable output. If these return signal state "0", there is an error in the execution of the instruction. Signal state "1" indicates that there is no error and no further error analysis is necessary.

Options for local error handling

The following options for local error handling of programming and access errors are available:

Error handling method	Validity	Explanation
EN/ENO mechanism	S7-1200/1500	You can use the ENO enable output to detect and handle runtime errors. Execution of subsequent instructions depends on the signal state of this parameter. You can avoid program crashes by using the EN/ENO mechanism. The block status is passed on in the form of a Boolean tag.
Output parameter RET_VAL	S7-1200/1500	Using the RET_VAL parameter as the return value of system functions (SFCs). you can display general error codes that may relate to any instruction or specific error codes that apply only to the instruction at hand. A maximum of one tag of the data type INT or WORD can be output. This error handling method is suitable for querying a communication error or incorrect data access, for example. You can find more information on the error codes in the Siemens Industry Online Support under the following FAQ ID: 770453 (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid0=10805384&objaction=csopen)
Output parameters STATUS and ERROR	S7-1200/1500	Using the STATUS and ERROR parameters as return values of system function blocks (SFBs), you can query device-specific error information. The error information is output in a pre-defined structure.

Error handling method	Validity	Explanation
Instructions "GET_ERROR" and "GET_ERR_ID"	S7-1200/1500	The instructions give you the opportunity to obtain an error ID or detailed error information. You can use the error information for an access error, for example, to determine the parameter that caused the access error. In order for the instructions to output the required error information, they must be programmed in the user program for each individual block from which potential errors are to be evaluated. This option is ideal especially for custom libraries with error handling. If you work with the "GET_ERROR" or "GET_ERR_ID" instructions, no error OB is called and there is no entry in the diagnostics buffer. With this method of error handling, you actively intervene in the program sequence when an error occurs.
BR bit of the status word	S7-1200/1500	Using the BR bit of the status word, you can determine whether an error occurred when the instruction was executed. (BR bit = "0" => An error has occurred; BR bit = "1" => No error has occurred)

Warning

Output parameter RET_VAL

When errors occur in the supply of the input parameters during the execution of the instruction that contains the RET_VAL parameter, an invalid error code is output at the RET_VAL parameter and the output parameters of the instruction cannot be evaluated.

Example

The above-mentioned local error handling options can be programmed individually or in combination with one another. To ensure that each error scenario that can occur within your program is recognized, we recommend a combination of local error handling options, as shown in the example below.

For a more precise error analysis, in addition to the RET_VAL output parameter you can also use the instructions "GET_ERROR" or "GET_ERR_ID". These options provide you with error codes, the detailed explanations of which are available in the descriptions of the respective instructions.

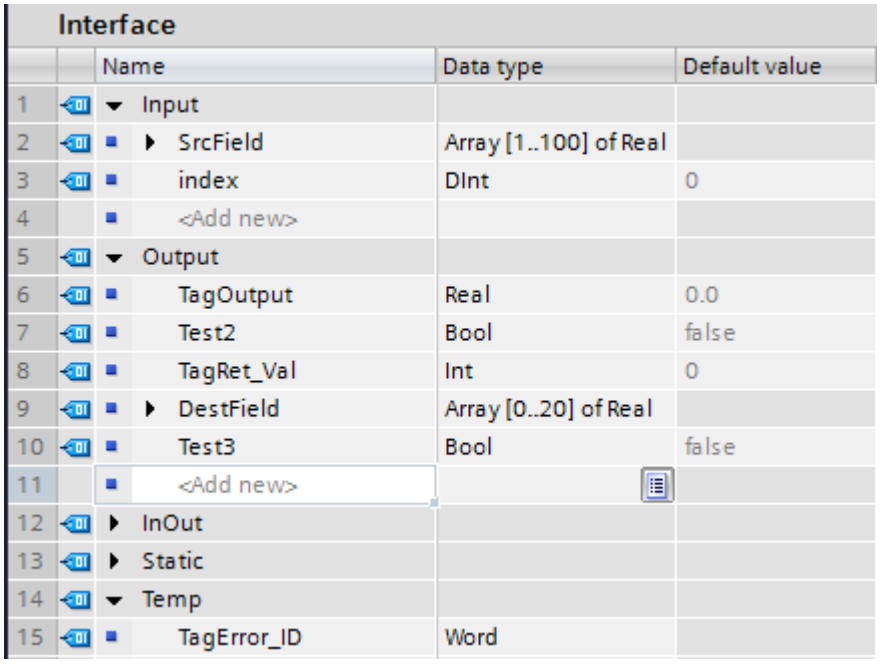
There are also error scenarios in which the RET_VAL output parameter does not output a valid error code. If an access error occurs while reading an input parameter, for example, the outputs of the instruction are no longer written, because the execution of the instruction is interrupted. In this case, we recommend that you integrate the two instructions "GET_ERROR" and "GET_ERR_ID" in your program because they provide reliable error information even when this type of error occurs.

Warning

Access error when reading an input parameter

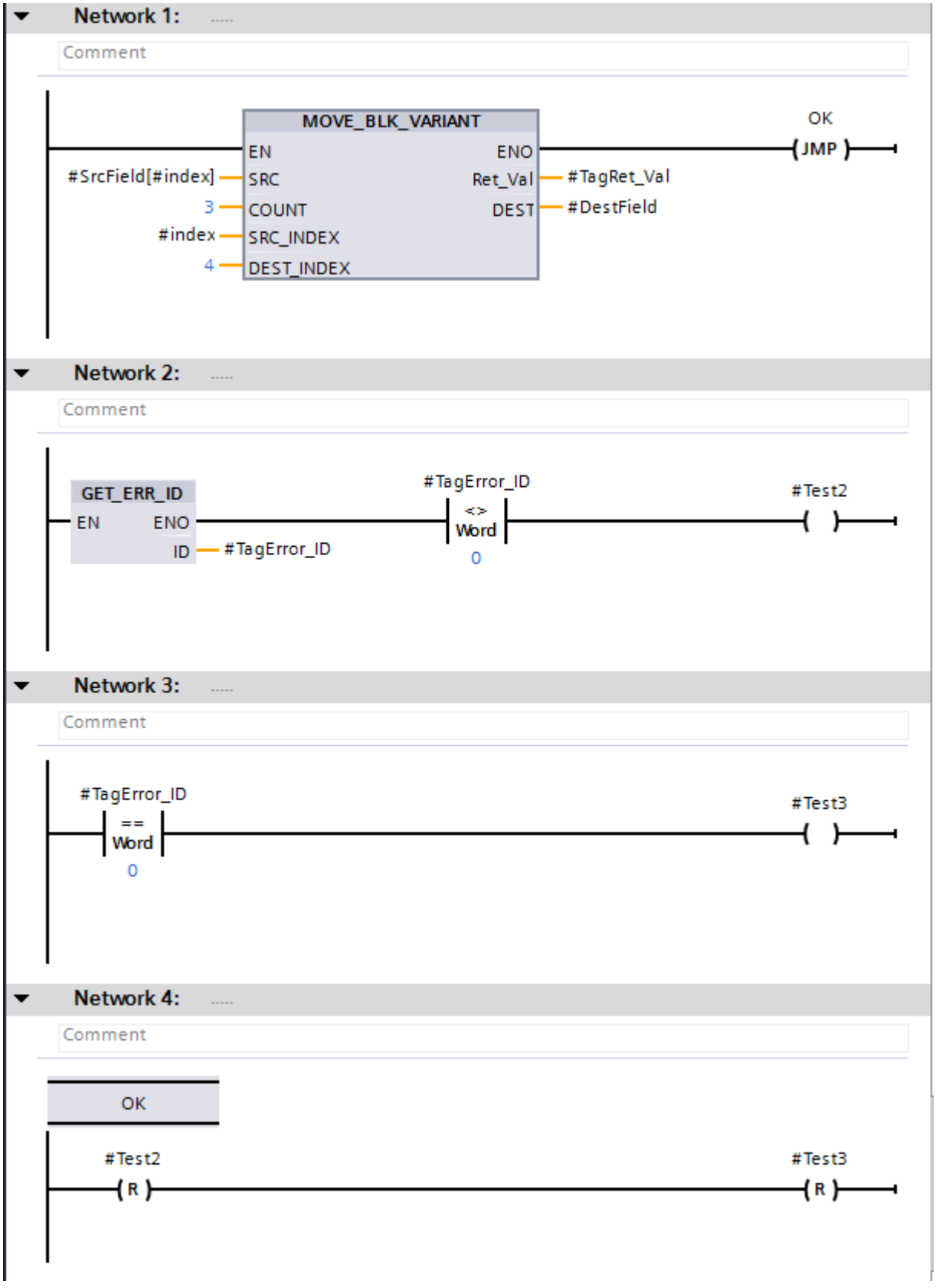
The RET_VAL parameter does not return a valid error code and no detailed error information is output to the diagnostics buffer.

The following example shows you how you can recognize an access error when reading an input parameter:



Interface

		Name	Data type	Default value
1		Input		
2		SrcField	Array [1..100] of Real	
3		index	DInt	0
4		<Add new>		
5		Output		
6		TagOutput	Real	0.0
7		Test2	Bool	false
8		TagRet_Val	Int	0
9		DestField	Array [0..20] of Real	
10		Test3	Bool	false
11		<Add new>		
12		InOut		
13		Static		
14		Temp		
15		TagError_ID	Word	


Network 1:
Comment
MOVE_BLK_VARIANT
EN
ENO
#SrcField[#index]
SRC
Ret_Val
#TagRet_Val
3
COUNT
DEST
#DestField
#index
SRC_INDEX
4
DEST_INDEX
OK
JMP
Network 2:
Comment
GET_ERR_ID
EN
ENO
ID
#TagError_ID
#TagError_ID
<>
Word
0
#Test2
Network 3:
Comment
#TagError_ID
==
Word
0
#Test3
Network 4:
Comment
OK
#Test2
R
#Test3
R

In network 1, the instruction "MOVE_BLK_VARIANT": Move block" is called. The "SrcField" source area is accessed at the SRC parameter using a variable index. If no errors occur during the execution of the instruction, the ENO enable output returns signal state "1", and program execution jumps to network 4 and continues there.

If an access error occurs during the execution of the instruction, for example, due to the variable index, the instruction "GET_ERR_ID: Get error ID locally" in network 2 returns an error ID. The comparison of the error ID for "UNEQUALS" with the value "0" in network 2 returns the result #Test2 = TRUE. The comparison of the error ID for "EQUALS" with the value "0" in network 3 returns the result #Test3 = TRUE.

The #TagRet_Val operand at the RET_VAL output parameter returns no valid error code in this case.

Exceptions

There are some instructions, however, for which you cannot program the error handling as described in the example above. This includes the following instructions:

- Instructions which generally do not have an EN/ENO mechanism
- Instructions in which the ENO was disabled
- S_COMP and T_COMP
- PEEK, PEEK_BOOL, POKE, POKE_BOOL and POKE_BLK

The BR bit or the ENO enable output is set to TRUE in these instructions even if it may result in an access error.

The following example shows you how to program reliable error handling in the STL programming language:

		Name	Data type
1		Input	
2		StringArray	Array[0..10] of String
3		index	DInt
4		Output	
5		TagResult	Bool
6		InOut	
7		Static	
8		Temp	
9		TagError_ID	Word
10		Constant	

STL	**Explanation**
SET L 0 T #Tag_ErrorID	// The operand #Tag_ErrorID is initialized with "0".
CALL S_COMP	// The instruction is called.

STL	Explanation
String EQ	
IN1 := #StringArray.[#index]	// Variable access to the ARRAY component.
IN2 := 'STRING'	// The two values are compared with each other.
OUT := #TagResult	// If both values are identical, the operand #TagResult receives the signal state "1".
A BR	// The BR bit is queried.
CALL GET_ERR_ID	// The instruction is called.
RET_VAL := #Tag_ErrorID	// The instruction outputs an error code in the case of an access error.

Even if the RLO bit has signal state "1", the access error is detected. You can query the error code with the evaluation of the #Tag_ErrorID operand of the "GER_ERR_ID instruction: Get error ID locally".

See also

GET_ERROR: Get error locally (Page 2414)

GET_ERR_ID: Get error ID locally (Page 2417)

Evaluating errors with output parameter RET_VAL (Page 2195)

Parameter STATUS (Page 3078)

6.4.4 Using MOVE instructions in STL

Possible applications

You can now program with MOVE instructions in STL on an S7-1500 CPU.

This gives you the following advantages:

- The program structure is easier to create.
- The performance of the CPU increases.

Programming in STEP 7 V5.x

In STEP 7 V5.x, you used the "BLKMOV: Move block" and "UBLKMOV: Move block uninterruptible" system functions to implement the MOVE functionality.

Procedure in STEP 7 TIA Portal

The following new MOVE instructions are available in STEP 7 TIA Portal:

- MOVE: Move value
- MOVE_BLK: Move block
- MOVE_BLK_VARIANT: Move block
- UMOVE_BLK: Move block uninterruptible

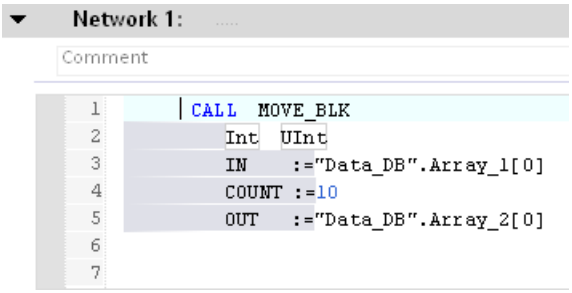
You can find additional information on the new MOVE instructions under "See also".

Programming example

The following example shows you how the "MOVE_BLK instruction works: Move block". An ARRAY block is copied into another ARRAY block:

Data_DB

	Name	Data type
1	Static	
2	Array_1	Array [0..10] of Int
3	Array_1[0]	Int
4	Array_1[1]	Int
5	Array_1[2]	Int
6	Array_1[3]	Int
7	Array_1[4]	Int
8	Array_1[5]	Int
9	Array_1[6]	Int
10	Array_1[7]	Int
11	Array_1[8]	Int
12	Array_1[9]	Int
13	Array_1[10]	Int
14	Array_2	Array [0..10] of Int
15	Array_2[0]	Int
16	Array_2[1]	Int
17	Array_2[2]	Int
18	Array_2[3]	Int
19	Array_2[4]	Int
20	Array_2[5]	Int
21	Array_2[6]	Int
22	Array_2[7]	Int
23	Array_2[8]	Int
24	Array_2[9]	Int
25	Array_2[10]	Int



Using the MOVE_BLK instruction, ten elements from "Array_1" of the "Data_DB" data block are copied into "Array_2" of the same data block.

6.4.5 Using IEC timers and counters

Advantages of IEC timers and counters

The universal use of IEC timers and counters makes your program code more efficient.

This gives you the following advantages:

- The blocks can be called multiple times with newly generated instance data blocks.
- The IEC counters have a large counting range.
- Compared to S5 timers, IEC timers have better performance and greater time accuracy.

Programming in STEP 7 V5.x

The S5 timers and counters in STEP 7 V5.x were addressed absolutely via a number. Due to this dependency on numbers, program blocks were not reusable with S5 timers and counters.

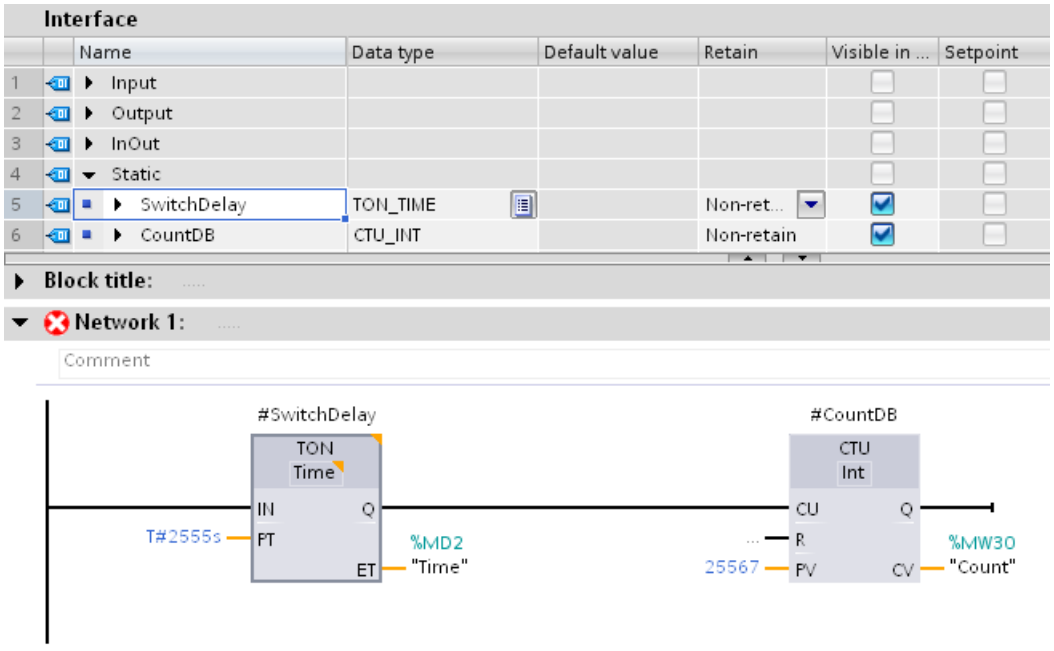
The value range of a timer was limited to 9990 seconds and that of a counter limited to a maximum of 999.

Procedure in STEP 7 TIA Portal

You declare IEC timers and counters in the program block where they are called or needed. The IEC timer is a structure of data type IEC_TIMER, IEC_LTIMER, or TON_TIME and TON_LTIME, for example, which you can also declare as a local tag in a block. The IEC counter is a structure of data type IEC_SCOUNTER, IEC_USCOUNTER, etc.

Programming examples in the TIA Portal

The following example shows you how to declare an IEC timer and IEC counter as a local tag:



The data of the TON IEC timer and the CTU IEC counter is stored as a local tag (multi-instance) in the block interface.

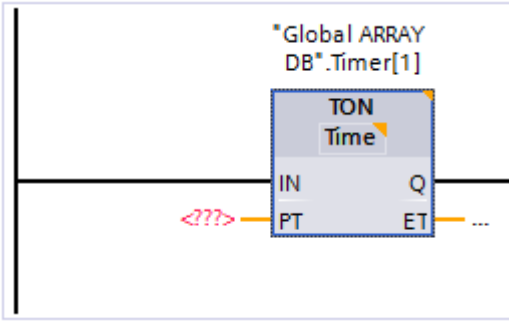
In the LAD and FBD programming languages, you also have the option of creating timers in a global data block and of using these as instances in your program code.

1. To do this, for example, create a global data block with an ARRAY of TON. The data type TON does not appear in the drop-down list, but can be entered manually:

Global ARRAY DB

		Name	Data type
1		Static	
2		Timer	Array[0..10] of TON
3		Timer[0]	TON
4		Timer[1]	TON
5		Timer[2]	TON
6		Timer[3]	TON
7		Timer[4]	TON
8		Timer[5]	TON
9		Timer[6]	TON
10		Timer[7]	TON
11		Timer[8]	TON
12		Timer[9]	TON
13		Timer[10]	TON

2. Create a LAD or FBD function block and drag the instruction "TON: Generate on-delay" to a network. Call the instance of timer TON as follows:



Call of a timer as multi-instance

If you want to use the IN parameter to start a timer as multi-instance, you must not initialize it beforehand in your program code. In this case, the timer called at the IN parameter does not recognize a positive signal edge and the timer is not started:

STL	Explanation
`Q "Tag_Output"`	`// When the "Tag_Output" output receives the signal state 1,`
`= #Timer_1.IN`	`// the IN parameter of the multi-instance timer #Timer_1 is initialized with a positive signal edge.`

STL	Explanation

CALL #Timer_1
???
IN := "Tag_Output"
PT := T#30s
Q := "Tag_4"
ET := "Tag_ElapsedTime"
	// When the multi-instance timer is now called and the IN parameter queried again, the timer is not started because there is no new positive signal edge.
	
	// Enter TIME as data type of the instruction.

This is why you must program the initialization of the multi-instance timer within the call.

STL	Explanation

CALL #Timer_1
???
IN := "Tag_Output"
PT := T#30s
Q := "Tag_4"
ET := "Tag_ElapsedTime"
	// The timer is called and started.
	
	// Enter TIME as data type of the instruction.

See also

Timers (Page 1929)

6.4.6 Using ARRAY data blocks

Using an ARRAY data block (S7-1500)

ARRAY data blocks are global data blocks that consist exclusively of an ARRAY. A data block with a tag of the ARRAY data type is sufficient for most applications because access can be programmed intuitively with a tag of the ARRAY data type (for example, #myArray[#index]) and offer a better runtime performance than ARRAY data blocks. In certain scenarios, however, it is necessary to process ARRAYs with different lengths. The ARRAY data block is particularly suitable for these scenarios. ARRAY data blocks with different length can be processed by means of the following instructions:

- ReadFromArrayDB: Read from Array data block
- WriteToArrayDB: Write to array data block
- ReadFromArrayDBL: Read from array data block in load memory
- WriteToArrayDBL: Write to array data block in load memory

For more information about the instructions, refer to "Basic instructions > LAD/FBD/STL/SCL/GRAPH > Move operations > ARRAY DB".

When the program code is being written, it is not necessary to know which ARRAY data block is read or written or what size it is. As a result, you can program the block flexibly so that it can also be used for buffers of various length. It is also not necessary to know the data type of the ARRAY elements, as the function in the program code using the DB_ANY data type does not yet require this information. The data type VARIANT is used so that you can also be flexible

in specifying the value to be read or written. The ARRAY data block is transferred only during runtime in order to then access the values in the program block. The data type of the ARRAY elements and the data type of the value that is to be read or written are determined. You can determine the number of objects and the fill level of the ARRAY data block using the ARRAY elements.

Note

When you create an ARRAY data block, specify the data type and the number of ARRAY elements. The data type of the value which is to be written to the ARRAY data block, for example, must match the data type of the ARRAY elements of the data block. In each case, connect a source area (PLC data type) with a destination area (ARRAY data block).

The advantage of this procedure is that the program code can already be created before you know which ARRAY data block and which value will be processed.

Programming example

The following example shows you how to use an ARRAY data block:

Individual pieces of material are transported on a conveyor belt. These pieces of material pass a scanner which can read out the information that a piece of material carries with it. The information is read out and transmitted to a panel. Because the scanner and the panel have different clock cycles/speeds, the material information must be cached in each case.

In the following programming example, we show you how to program the program code for passing on material information. For this purpose, you use a PLC data type (UDT) and an ARRAY data block.

Procedure

Create the PLC data type "UDT_Queue". You use this PLC data type as an instance which can be accessed by both functions ("FC_Enqueue" and "FC_Dequeue"). This is important, for example, for accessing the tag #Queue.Used, as the function "FC_Enqueue" increments the tag by one and the function "FC_Dequeue" decrements the tag by one.

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare the following lines within the PLC data type:
   DB > Data type: DB_ANY
   Size > Data type: DINT
   Used > Data type: DINT
   ReadPos > Data type: DINT
   WritePos > Data type: DINT

UDT_Queue

	Name	Data type	Default value	Comment
1	DB	DB_ANY	0	
2	Size	DInt	0	
3	Used	DInt	0	
4	ReadPos	DInt	0	Pointer Reading Position
5	WritePos	DInt	0	Pointer Writing Position

Program the instruction "FC_Enqueue", which writes the values from a PLC data type into an ARRAY data block. The PLC data type and the ARRAY data block must be still unknown at this time, as the interfaces with the data types VARIANT and DB_ANY are programmed:

1. Create an SCL function and name it "FC_Enqueue".
2. Declare the block interface as follows:

FC_Enqueue

	Name	Data type	Comment
1	Input		
2	Value	Variant	
3	Output		
4	InOut		
5	Queue	"UDT_Queue"	
6	DB	DB_ANY	
7	Size	DInt	
8	Used	DInt	
9	ReadPos	DInt	Pointer Reading Position
10	WritePos	DInt	Pointer Writing Position
11	Temp		
12	Error	Int	
13	<Add new>		
14	Constant		
15	Return		
16	FC_Enqueue	Int	

3. Write the following program code:

```
IF #Queue.Used < #Queue.Size THEN // As long as there is enough space in the DB
                                  // the function keeps on writing.

    #Error := WriteToArrayDB(db := #Queue.DB, index := #Queue.WritePos, value := #Value);

    IF #Error = 0 THEN

        #Queue.Used := #Queue.Used + 1;
        #Queue.WritePos := #Queue.WritePos + 1;

        IF #Queue.WritePos >= #Queue.Size THEN // If this condition is fullfilled
                                               // the end of the DB is reached.
                                               // End of writing pointer.

            #Queue.WritePos := 0;

        END_IF;

    END_IF;

ELSE

    #Error := 4711; // The queue is full.

END_IF;
```

With this function, you check whether there is still space free in the data block. If so, write the value that is specified at the parameter value into the data block at the parameter db. With each new item of material information that is written, the tag #Queue.Used and the pointer tag #Queue.WritePos are incremented by one. As soon as the cursor reaches the end of the data block, it is reset to 0. If the data block is full, the error code #4711 is returned.

Program the instruction "FC_Dequeue", which reads the material information from an ARRAY data block and writes it into a PLC data type. The PLC data type and the ARRAY data block must be still unknown at this time, as the interfaces are programmed with the data types

VARIANT and DB_ANY. The material information can then be displayed on a panel, for example:

1. Create an SCL function and name it "FC_Dequeue".
2. Declare the block interface as follows:

FC_Dequeue

	Name	Data type	Comment
1	Input		
2	Output		
3	InOut		
4	Value	Variant	
5	Queue	"UDT_Queue"	
6	DB	DB_ANY	
7	Size	DInt	
8	Used	DInt	
9	ReadPos	DInt	Pointer Reading Position
10	WritePos	DInt	Pointer Writing Position
11	Temp		
12	Error	Int	
13	Constant		
14	Return		
15	FC_Dequeue	Int	

3. Write the following program code:

```
IF #Queue.Used > 0 THEN // As long as there is some information in the DB
                        // the function keeps on reading.

    #Error := ReadFromArrayDB(db := #Queue.DB, index := #Queue.ReadPos, value => #Value);

    IF #Error = 0 THEN

        #Queue.Used := #Queue.Used - 1;
        #Queue.ReadPos := #Queue.ReadPos + 1;

        IF #Queue.ReadPos >= #Queue.Size THEN // If this condition is fullfilled
                                              // the end of the DB is reached.
                                              // End of reading pointer.

            #Queue.ReadPos := 0;

        END_IF;

    END_IF;

ELSE

    #Error := 4712; // The queue is empty.

END_IF;
```

With this function, you check whether material information is available in the data block. If so, read the value to which the pointer #Queue.ReadPos points, and write it to the tag #Value. With each item of material information that is read, the tag #Queue.Used is decremented by one and the pointer tag #Queue.ReadPos is incremented by one. As soon as the cursor reaches the end of the data block, it is reset to 0. If the data block is empty, the error code #4712 is returned.

In order to be able to write or read data, you require a specific PLC data type and an ARRAY data block, each of which must have the same data type.

To do this, create the PLC data type "UDT_Material". The scanner writes the read-out material information into this PLC data type.

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare the following lines within the PLC data type:
   ArticleNumber > Data type: DINT
   ArticleName > Data type: STRING
   Amount > Data type: REAL
   Unit > Data type: STRING

UDT_Material

	Name	Data type
1	ArticleNumber	DInt
2	ArticleName	String
3	Amount	Real
4	Unit	String

Create the ARRAY data block "DB_MaterialBuffer". The material information from the PLC data type "UDT_Material" is sent to this ARRAY data block with the help of the function "FC_Enqueue".

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_MaterialBuffer".
4. Select the type of the data block "ARRAY DB".
5. Select the PLC data type "UDT_Material" as ARRAY data type.
6. Specify "1000" as high ARRAY limit.
7. Click "OK".

DB_MaterialBuffer

	Name	Data type
1	DB_MaterialBuffer	Array[0..1000] of "UDT_Material"
2	DB_MaterialBuffer[0]	"UDT_Material"
3	ArticleNumber	DInt
4	ArticleName	String
5	Amount	Real
6	Unit	String
7	DB_MaterialBuffer[1]	"UDT_Material"
8	DB_MaterialBuffer[2]	"UDT_Material"
9	DB_MaterialBuffer[3]	"UDT_Material"
10	DB_MaterialBuffer[4]	"UDT_Material"
11	DB_MaterialBuffer[5]	"UDT_Material"

Create the start of the organization block (OB) "OB_MaterialQueue". In this organization block, you initialize the tags DB and Size.

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Organization block (OB)" button.
3. Specify the name "OB_MaterialQueue".
4. Select the type "Startup".
5. Select SCL as the language of the organization block.
6. Click "OK".
7. Write the following program code:

```
1  "DB_MaterialQueue".DB := "DB_MaterialBuffer";
2  "DB_MaterialQueue".Size := 1000;
3  "DB_MaterialQueue".Used := 0;
4  "DB_MaterialQueue".ReadPos := 0;
5  "DB_MaterialQueue".WritePos := 0;
```

   By assigning the data block, you connect the ARRAY data block "DB_MaterialBuffer" with the SCL functions "FC_Enqueue" and "FC_Dequeue". Enter the size of the ARRAY data block at the Size parameter. The start value of the Used parameter is "0". The first item of material information is thus written to the ARRAY element "0".

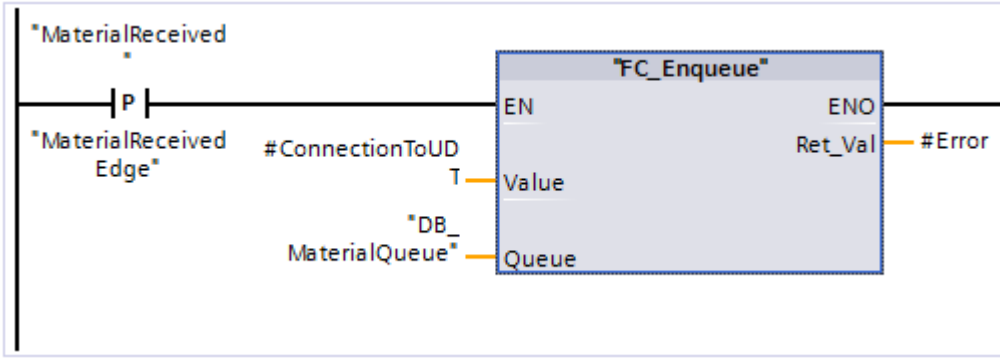
1. Select the folder "Program blocks" in the project tree and select the command "Compile > Software (only changes)" from the shortcut menu.
2. Declare the following tags in the "Default tag table":

Name	Data type	Address
MaterialReceived	Bool	%I0.0
MaterialReceivedEdge	Bool	%M0.0
MaterialDone	Bool	%I1.0
MaterialDoneEdge	Bool	%M0.1

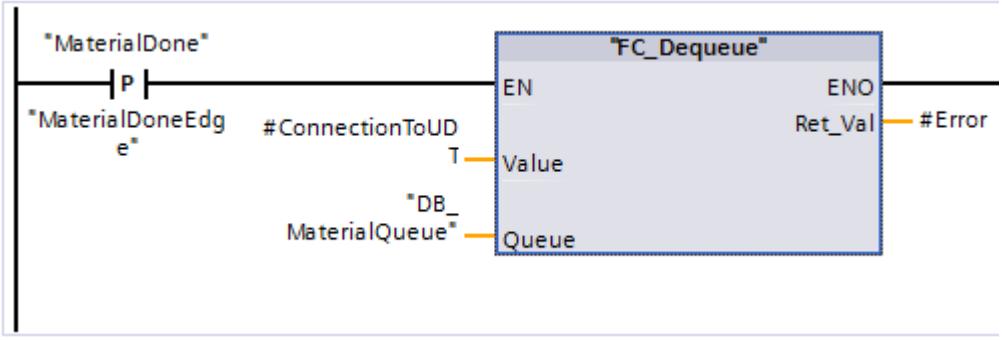
3. Call the SCL function "FC_Enqueue" within the function block in which the scanner reads out the material information.
4. Declare the tag "ConnectionToUDT" in the section "Temp" in the block interface and link this tag with the PLC data type "UDT_Material":

Temp	
ConnectionToUDT	"UDT_Material"
ArticleNumber	DInt
ArticleName	String
Amount	Real
Unit	String

5. Link the function call with the following tags and, at the enable input EN, create the signal edge "P: Scan operand for positive signal edge". Link the signal edge with the global tags from the default tag table:



6. Call the SCL function "FC_Dequeue"
7. Link the function call with the following tags and, at the enable input EN, create the signal edge "P: Scan operand for positive signal edge". Link the signal edge with the global tags from the default tag table:



Result

As soon as a positive signal edge occurs, material information is written to an ARRAY data block with the help of the instruction "WriteToArrayDB" and sent to the panel with the help of the instruction "ReadFromArrayDB".

See also

ReadFromArrayDB: Read from array data block (Page 2357)

WriteToArrayDB: Write to array data block (Page 2359)

ReadFromArrayDBL: Read from array data block in load memory (Page 2361)

WriteToArrayDBL: Write to array data block in load memory (Page 2364)

ReadFromArrayDB: Read from array data block (Page 2635)

WriteToArrayDB: Write to array data block (Page 2637)

ReadFromArrayDBL: Read from array data block in load memory (Page 2639)

WriteToArrayDBL: Write to array data block in load memory (Page 2642)

ReadFromArrayDB: Read from array data block (Page 2838)

WriteToArrayDB: Write to array data block (Page 2840)

ReadFromArrayDBL: Read from array data block in load memory (Page 2842)

WriteToArrayDBL: Write to array data block in load memory (Page 2845)

6.4.7 Reliable addressing

6.4.7.1 Symbolic addressing

Advantages of symbolic addressing

The use of universally applied and meaningful symbols in the overall project makes the program code easier to read and understand.

This gives you the following advantages:

- You do not have to write detailed comments.
- Data access is faster.
- No errors occur when accessing data.
- You no longer have to work with absolute addresses.
- The assignment of the symbol to the memory address is monitored by STEP 7, which means all points of use are automatically updated when the name or the address of a tag changes.

Programming in STEP 7 V5.x

STEP 7 V5.x already gave you the option to write a program that is easier to read by using descriptive names for operands and blocks. You did this by assigning the symbolic operands to memory addresses and blocks in the symbol table. In order for a change in the symbolism to also have an affect on the program code in the program editor, you had to use the "Operand priority" property to specify whether the symbol or the absolute value had priority.

The use of symbolic addressing allowed you to create a program with greater clarity. However, in some cases, such as when programming with user-defined data types (UDT), this could impair performance.

You could increase the performance by ignoring the symbolism in the UDT and using absolute addressing. This made it necessary, however, to know the data storage. Changes to the UDT were not automatically updated. Using absolute addressing, you could also access parts of a tag and edit these. The disadvantage of exclusively absolute addressing, however, was that the program code became cluttered after a certain level and you had to insert additional comments for better orientation.

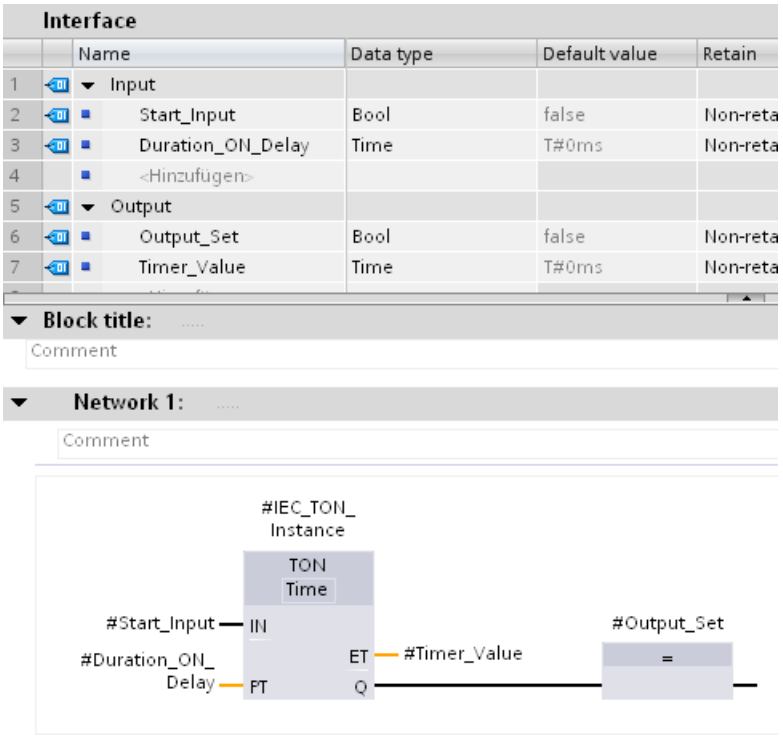
Procedure in STEP 7 TIA Portal

The S7-1500 CPU offers much higher performance than the S7-300/400 CPUs. To make full use of this high performance, we recommend that you enable optimized block access for all blocks and use symbolic addressing in the program code.

The program editor helps you work with symbols through context-sensitive input help, for example, auto-completion. Using it, you can easily access existing tags or instructions during programming.

Programming example

The following example shows you how to symbolically access individual elements:



You can use the tag names that you have defined in the block interface directly at the parameters of the TON instruction without knowing the absolute address of the tags.

6.4.7.2 Addressing with Slice access

Symbolically accessing tags of Bit string data type, bit by bit, byte by byte, word by word and double word by double word

You have the option to specifically address areas within declared tags. You can access areas of the 1-bit, 8-bit, 16-bit, or 32-bit width. The allocation of a memory area (e.g., BYTE or WORD) to a small memory area (e.g., BOOL) is also referred to as "Slice".

You can find additional information on the syntax of slice access under "See also".

Programming example

You can find a detailed example in the Siemens Industry Online Support under the following FAQ ID: 57374718 (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&Datakey=47071380&extranet=standard&groupid=4000002&viewreg=WW&nodeid0=29156492&objaction=csopen)

6.4.7.3 Indirect addressing of ARRAY elements

Implementing ARRAY access in TIA Portal with a variable index

It is advisable to use an ARRAY when you want to process assembled data of the same data type. For addressing ARRAY elements, you can specify either constants or tags of the integer data type as index. Integers with lengths of up to 32 bits are allowed here.

With indirect addressing using a tag, the index is only calculated during program runtime. You can, for example, use a different index for each cycle in program loops. You can also access an ARRAY within a PLC data type (UDT).

This gives you the following advantages:

- No addressing is necessary with address registers or with self-assembled pointers, for example, an ANY pointer.
- More flexibility within your program.
- The variable index is available in all STEP 7 programming languages.
- It uses the existing names of data blocks and ARRAY tags (symbolic addressing). This improves readability of the program code.
- The start address of the ARRAY does not have to be known.
- The program code is easier to create and the compiler generates optimized program code.

Procedure in STEP 7 V5.x

In STEP 7 V5.x, you had to use an address register with the help of a self-configured POINTER for indirect addressing of ARRAY elements. The following aspects had to be taken into consideration for this:

- The name of the ARRAY was not used. This reduced the readability of the program code and make additional comments necessary.
- The start address of the ARRAY had to be known to perform the addressing.

The SCL programming language already supported indirect addressing with variable index.

Programming example in STEP 7 V5.x

The "Data_classic" data block is required for the following STL example. To address an element of the "Quantities" ARRAY, the following commands must be used:

STL	Explanation
`OPN "Data_classic"`	`// The "Data_classic" data block is called.`
`L #index`	`// The value of the local tag #index is loaded into accumulator 1.`
`SLD 3`	`// Move bits 0 to 31 of accumulator 1 by 3 positions to the left.`
	`// Fill the now empty bit places with zeros.`
`LAR1`	`// Load address register 1 with contents of accumulator 1.`
`L DBW [AR1, P#10.0]`	`// Load the ARRAY element addressed with #index into accumulator 1.`
	`// P#10.0 = Start address of the array`

Programming example in STEP 7 TIA Portal

In the example below, you see the indirect addressing of an ARRAY element in STL in the TIA Portal.

Create a global data block for this purpose:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_Quantities".
4. Select the type of the data block "ARRAY DB".
5. Select the "DINT" data type as ARRAY data type.

6. Specify "10" as high ARRAY limit.
7. Click "OK".

DB_Quantities

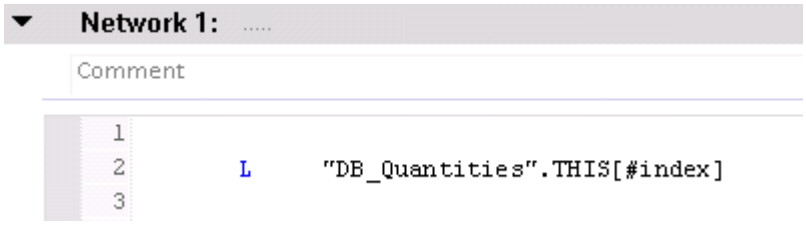
	Name	Data type
1	DB_Quantities	Array[0..10] of DInt
2	DB_Quantities[0]	DInt
3	DB_Quantities[1]	DInt
4	DB_Quantities[2]	DInt
5	DB_Quantities[3]	DInt
6	DB_Quantities[4]	DInt
7	DB_Quantities[5]	DInt
8	DB_Quantities[6]	DInt
9	DB_Quantities[7]	DInt
10	DB_Quantities[8]	DInt
11	DB_Quantities[9]	DInt
12	DB_Quantities[10]	DInt

1. Create a function block and name it "FB_Quantities".
2. Declare the block interface as follows:

FB_Quantities

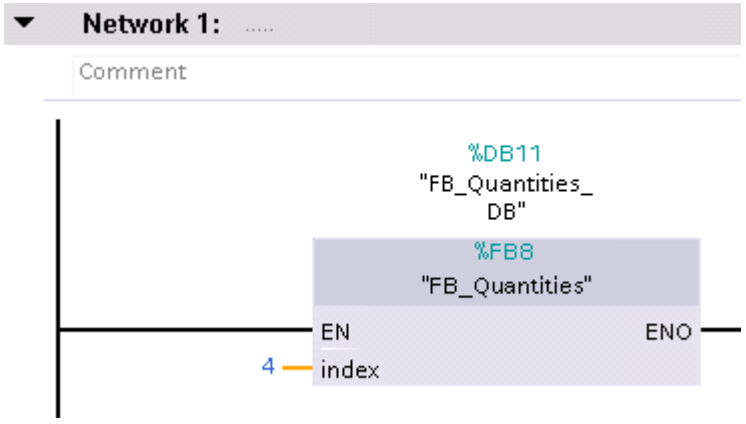
	Name	Data type
1	Input	
2	Output	
3	InOut	
4	Static	
5	Temp	
6	Index	Int
7	Constant	

3. Write the following program code:



You need just one more program line for addressing an ARRAY element in the TIA Portal. The value of the ARRAY element #index is loaded directly from the data block into accumulator 1.

4. Call the "FB_Quantities" function block in OB1 and assign it an index between 0 - 10:



Note the following to get the best performance:

- Declare tags that are used as an ARRAY index as an integer of less than or equal to 32 bits.
- Store intermediate results and ARRAY indexes in the temporary local data area.

See also

Array (Page 1941)

6.4.7.4 Addressing tags indirectly

Implementing tag access with variable index

Using a variable index, you can access tags that have different data types and are located in different areas. For addressing, you can specify either constants or tags of the integer data type as index. Integers with lengths of up to 32 bits are allowed here.

With indirect addressing using a tag, the index is only calculated during program runtime. You can, for example, use a different index for each cycle in program loops.

This gives you the following advantages:

- No addressing is necessary with address registers or with self-assembled pointers, for example, an ANY pointer.
- More flexibility within your program.
- The variable index is available in all STEP 7 programming languages.
- The existing names of data blocks and tags are used (symbolic addressing). This improves readability of the program code.
- The start address does not need to be known.
- The program code is easier to create and the compiler generates optimized program code.

1. Programming example

In the example below, you use the index to access three tags from different memory areas.

Access list of the three tags, each of which is assigned to an index:

Index	Accessing tag	Memory area
1	Input_WORD_0	IW 0
2	"Processdata".Temperature	DB 1
3	Output_WORD_4	QW 4

Declare the following two tags in the "Default tag table":

Default tag table

		Name	Data type	Address
1		Input_WORD_0	Word	%IW0
2		Output_WORD_4	Word	%QW4

Create a global data block:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_Processdata".
4. Select "Global DB" as the type of the data block.
5. Click "OK".
6. Declare the data block element "Temperature":

DB_Processdata

		Name	Data type	Start value
1		Static		
2		Temperature	DInt	50

Declare indirect access using an index within a function.

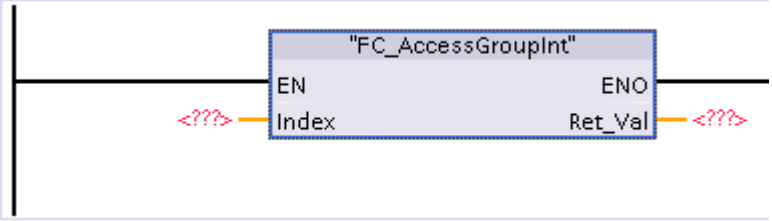
1. Create an SCL function and name it "FB_AccessGroupInt".
2. Declare the block interface as follows:

FC_AccessGroupInt		
	Name	Data type
1	Input	
2	Index	Int
3	Output	
4	InOut	
5	Temp	
6	Constant	
7	Return	
8	FC_AccessGroupInt	DInt

3. Write the following program code:

```
1
2  CASE #Index OF
3
4      1: // First case: "Input_WORD_0"
5          #FC_AccessGroupInt := "Input_WORD_0";
6      2: // Second case: "DB_Processdata".Temperature
7          #FC_AccessGroupInt := "DB_Processdata".Temperature;
8      3: // Third case: "Output_WORD_4"
9          #FC_AccessGroupInt := "Output_WORD_4";
10
11 END_CASE;
```

4. Call the "FC_AccessGroupInt" function in OB1:



Depending on which number (1, 2 or 3) you specify at the Index parameter, the first, second or third case of the "FC_AccessGroupInt" instruction is executed.

2. Programming example

In the example below, you use the index to access three different optimized data blocks.

Since all data blocks should contain the same tags, you can use a PLC data type (UDT) in this case.

1. To create a PLC data type, double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Rename the PLC data type to "UDT_SiloContents".
3. Declare the following lines within the PLC data type:
   MyBool > Data type: BOOL
   MyInt > Data type: INT
   MyWord > Data type: WORD

UDT_SiloContents

		Name	Data type
1	...	MyBool	Bool
2	...	MyInt	Int
3	...	MyWord	Word

Create three global data blocks.

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the names "DB_SiloWater", "DB_SiloSugar" and "DB_SiloMilk".

4. Select the data blocks "UDT_SiloContents" as the type of the data blocks.
5. Click "OK".

DB_SiloWater

		Name	Data type
1		Static	
2		MyBool	Bool
3		MyInt	Int
4		MyWord	Word

DB_SiloSugar

		Name	Data type
1		Static	
2		MyBool	Bool
3		MyInt	Int
4		MyWord	Word

DB_SiloMilk

		Name	Data type
1		Static	
2		MyBool	Bool
3		MyInt	Int
4		MyWord	Word

Create a function to read the values of the data block tags and to write these to a PLC data type.

1. Create an SCL function and name it "FC_AccessGroupSiloRead".
2. Declare the block interface as follows:

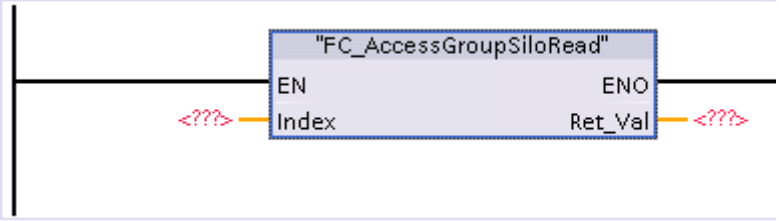
FC_AccessGroupSiloRead

		Name	Data type
1		Input	
2		Index	DInt
3		Output	
4		InOut	
5		Temp	
6		Constant	
7		Return	
8		FC_AccessGroupSiloRead	"UDT_SiloContents"

3. Write the following program code:

```
1
2  CASE #Index OF
3
4      1: // First case: "DB_SiloWater"
5          #FC_AccessGroupSiloRead := "DB_SiloWater";
6      2: // Second case: "DB_SiloSugar"
7          #FC_AccessGroupSiloRead := "DB_SiloSugar";
8      3: // Third case: "DB_SiloMilk"
9          #FC_AccessGroupSiloRead := "DB_SiloMilk";
10
11     ELSE // Optional case
12         #FC_AccessGroupSiloRead := "DB_SiloWater";
13
14 END_CASE;
```

4. Call the "FC_AccessGroupSiloRead" function in OB1:



Depending on which number (1, 2 or 3) you specify at the Index parameter, the first, second or third case of the "FC_AccessGroupSiloRead" instruction is executed.

Result

With this procedure, the programming is

- comprehensible, as you can use cross-reference lists
- reliable, as you use only the previously defined memory areas
- applicable to both standard and optimized data areas

6.4.8 Handling specific data types

6.4.8.1 Using the VARIANT data type

Overview of the VARIANT data type

Introduction

The VARIANT data type is a pointer or a reference to another data object. The data type VARIANT is typified, i.e., you can read out the data type of a referenced tag during program runtime.

Using the data type VARIANT, you can above all create generic, standardized function blocks (FB) or functions (FC) for various data types. Various instructions in all programming languages are available to you for this purpose. During the creation of the program, you can specify which data types the block should be able to process. The VARIANT data type supports you here by allowing the interconnection of any tags. You can then react accordingly to their data types in the block.

Each of the figures below shows you a section from the "Moving data" programming example. You can find the detailed program code under "See also".

Application cases for pointers with an S7-1200/1500 CPU as compared to S7-300/400

The following table provides you with an overview of the various applications for pointers on a CPU of the S7-300/400 series (ANY pointer) and their solution with a CPU of the S7-1200/1500 series.

In most applications, the use of a pointer is no longer required on a CPU of the S7-1200/1500 series. Instead, much simpler language resources are available.

It is only advisable to use the VARIANT data type for indirect addressing, when the data types are only determined during the program runtime.

What is the purpose of the ANY pointer?	Recommendations in the TIA Portal (S7-1200/ S7-1500)
Moving data of any source and destination data type in the program using the instruction "BLKMOV: Move block".	Definition of tags within a PLC data type. You can use the instructions "Serialize" and "Deserialize" to move the tags.
Initializing an ARRAY structure	Using the instruction "FILL_BLK: Fill block", you initialize or fill an ARRAY structure.
Moving ARRAY elements	Using the instruction "MOVE_BLK: Move block". you move the content of multiple elements of an ARRAY structure to another ARRAY structure.

What is the purpose of the ANY pointer?	Recommendations in the TIA Portal (S7-1200/ S7-1500)
Memory and performance optimization using structured data	Use the InOut section in the block interface to optimize memory and performance. You can find additional information in the "Programming Guideline for S7-1200/1500" under the following link Programming Guideline for S7-1200/1500 (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=4&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid4=20229695&objaction=csopen)
Access to individual bits/bytes of a WORD	Use the "slice access" Additional information is available here: Example of a slice access (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid0=29156492&objaction=csopen)
Determination of the length of structures or data blocks	Use an ARRAY and read its length using the instruction "CountofElements: Get number of ARRAY elements. The instruction only works in conjunction with the data type VARIANT.
Indirect addressing	You can use the VARIANT pointer for the indirect addressing of data types that will only be known during runtime. You can use the data type DB_ANY for indirect access to a data block.

Initializing the VARIANT data type

Initialize the VARIANT data type by assigning a specific tag to the VARIANT block parameter at the block call. This forms a reference to the address of the passed tag. To do this, create a block parameter of the VARIANT data type in the block interface. In the following example, these are the two block parameters SourceArray and DestinationArray in the section InOut.

FC_PartialArrayCopy

	Name	Data type
1	Input	
2	Count	UDInt
3	SourceIndex	DInt
4	DestinationIndex	DInt
5	Output	
6	InOut	
7	SourceArray	Variant
8	DestinationArray	Variant
9	Temp	
10	Error	Int
11	Constant	
12	<Add new>	
13	Return	
14	FC_PartialArrayCopy	Int

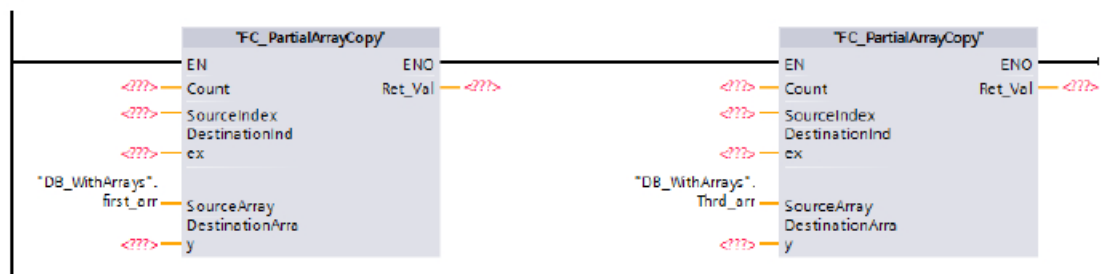
Note

Direct passing of a tag to a VARIANT tag is not possible, such as myVARIANT := #Variable

Passing various data types

In the following example, you see how the VARIANT block parameter can be initialized with different tags when a generic, standardized function is called multiple times:

The "FC_PartialArrayCopy" function is called twice. The VARIANT parameter SourceArray is interconnected with an ARRAY of "my_struct" with the left call. The VARIANT parameter SourceArray is interconnected with an ARRAY of REAL with the right call.



Reading out and checking data types

Various comparison instructions are available to you to read out the data type of a tag or an element and compare it with data types of other tags or elements.

In the figure below, you see the usage of multiple comparison instructions to check whether the elements of the ARRAYs have the same data type:

```
6  IF IS_ARRAY(#SourceArray) AND IS_ARRAY(#DestinationArray)
7      AND TypeOfElements(#SourceArray) = TypeOfElements(#DestinationArray) THEN
8      #Error := MOVE_BLK_VARIANT(COUNT := #Count, SRC := #SourceArray, SRC_INDEX := #SourceIndex,
9                                 DEST => #DestinationArray, DEST_INDEX := #DestinationIndex);
10 END_IF;
```

The MOVE_BLK_VARIANT instruction is only executed if the data types of the ARRAY elements are the same.

See also

ANY (Page 1948)

VARIANT (Page 1951)

TypeOf: Check data type of a VARIANT tag (Page 2798)

TypeOfElements: Check data type of an ARRAY element of a VARIANT tag (Page 2799)

IS_ARRAY: Check for ARRAY (Page 2800)

Programming example: Moving data (Page 267)

VARIANT instructions

VARIANT instructions

The following instructions for working with VARIANT are available to you in the TIA Portal:

Basic instructions		
Category	**Instruction**	**Description**
Comparator operations	EQ_Type	Compare data type for EQUAL with the data type of a tag
	NE_Type	Compare data type for UNEQUAL with the data type of a tag
	EQ_ElemType	Compare data type of an ARRAY element for EQUAL with the data type of a tag
	NE_ElemType	Compare data type of an ARRAY element for UNEQUAL with the data type of a tag
	IS_NULL	Query for EQUALS ZERO pointer
	NOT_NULL	Query for UNEQUALS ZERO pointer
	IS_ARRAY	Check for ARRAY
	TypeOf	Check data type of a VARIANT tag
	TypeOfElements	Check element data type of a VARIANT tag
	IS_ARRAY	Check for ARRAY

Basic instructions		
Category	**Instruction**	**Description**
Move operations	MOVE_BLK_VARIANT	Move block
	VariantGet	Read out VARIANT tag value
	VariantPut	Write VARIANT tag value
	CountOfElements	Get number of ARRAY elements
Conversion operations	VAR-IANT_TO_DB_ANY	Convert VARIANT to DB_ANY
	DB_ANY_TO_VARIANT	Convert DB_ANY to VARIANT

Note

Differences between MOVE, MOVE_BLK and MOVE_BLK_VARIANT

- You can use the "MOVE" instruction to copy complete structures.
- You can use the "MOVE_BLK" instruction to move parts of ARRAYs with known data type.
- The MOVE_BLK_VARIANT instruction is only required if you want to move parts of ARRAYs with a data type that is only known during program runtime.

You can find additional information on the individual instructions in the information system under "Basic instructions > Respective programming language".

You can also find additional instructions which also work with the VARIANT data type under the "Extended instructions".

See also

VARIANT (Page 1951)

Indirect addressing with the VARIANT data type (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid0=29156492&objaction=csopen)

Using VARIANT instructions

Introduction

In the following chapter, you will learn which application options you have with VARIANT instructions.

Evaluating data types of tags to which a VARIANT points

In the table below, you see which instructions are available to you to evaluate data types of tags to which a VARIANT points:

Function	Instruction	Description
To determine the data type	TypeOf(): Check data type of a VARIANT tag (This instruction is available only in SCL and only in conjunction with an IF instruction.)	You use this instruction to compare the data type to which a VARIANT tag points with the data type of any other tag. You can also make the comparison with a PLC data type.
	TypeOfElements(): Check element data type of a VARIANT tag (This instruction is available only in SCL and only in conjunction with an IF instruction.)	You use this instruction to compare the data type to which a VARIANT tag points with the data type of any other tag. You can also make the comparison with a PLC data type. If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.
	EQ_Type: Compare data type for EQUAL with the data type of a tag NE_Type: Compare data type for UNEQUAL with the data type of a tag	You use this instruction to compare the data type to which a VARIANT tag points with the data type of any other tag. You can also make the comparison with a PLC data type.
	EQ_ElemType: Compare element data type for EQUAL with the data type of a tag NE_ElemType: Compare element data type for UNEQUAL with the data type of a tag	You use this instruction to compare the data type to which a VARIANT tag points with the data type of any other tag. You can also make the comparison with a PLC data type. If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.
To evaluate ARRAY elements	IS_ARRAY: Check for ARRAY	You use this instruction to check whether the data type to which a VARIANT tag points is an ARRAY.
	CountOfElements: Get number of ARRAY elements	You use this instruction to read out how many ARRAY elements the tag has to which the VARIANT tag points.

You can find additional information on the individual instructions in the information system under "Basic instructions > Respective programming language".

Reading data to which a VARIANT points

To be able to use the data you must move this data to a tag as an intermediate step, as it cannot be processed directly.

Instruction	Description	Example		Result
		VARIANT points to	Destination data type	
VariantGet: Read out VARIANT tag value	You use this instruction to move the value of a single tag to another tag. The data types of both tags must match.	UDT_1	UDT_1	The instruction is executed
		REAL	REAL	
		DINT	DWORD	The instruction is not executed.

Assigning data to a VARIANT tag

You cannot use the instruction to initialize VARIANT tags. The VARIANT tags must therefore already be initialized when the data is returned to the tag. Do not use uninitialized temporary VARIANT tags.

Instruction	Description	Example		Result
		Source data type	VARIANT points to:	
VariantPut: Write VARIANT tag value	You use this instruction to move the value of a single tag to another tag. The data types of both tags must match.	UDT_1	UDT_1	The instruction is executed
		REAL	REAL	
		DINT	DWORD	The instruction will not be executed, as the data types are different.

Processing dynamic ARRAY structures

To evaluate ARRAY elements	TypeOfElements(): Check data type of an ARRAY element of a VARIANT tag (This instruction is available only in SCL and only in conjunction with an IF instruction.)	You use this instruction to compare the data type to which a VARIANT tag points with the data type of any other tag. You can also make the comparison with a PLC data type. If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.
	IS_ARRAY: Check for ARRAY	You use this instruction to check whether the data type to which a VARIANT tag points is an ARRAY.
	CountOfElements: Get number of ARRAY elements	You use this instruction to read out how many ARRAY elements the tag has to which the VARIANT tag points.
	MOVE_BLK_VARIANT: Move block	This instruction is used to move dynamic and type-safe (integrated type test) ARRAYs. You can freely select the limit values for the source and destination ARRAYs. The data types of ARRAY elements must match.

Note

Differences between MOVE, MOVE_BLK and MOVE_BLK_VARIANT

- You can use the "MOVE" instruction to copy complete structures.
- You can use the "MOVE_BLK" instruction to move parts of ARRAYs with known data type.
- The MOVE_BLK_VARIANT instruction is only required if you want to move parts of ARRAYs with a data type that is only known during program runtime.

Additional information on the use of the MOVE_BLK_VARIANT instruction can be found in the "Moving data" programming example.

See also

VARIANT (Page 1951)

Programming example: Moving data (Page 267)

Programming examples with VARIANT

Programming example: Moving data

Programming example

In this programming example, data values, which are collected for example during a production shift, are moved for further processing. The data is collected in an ARRAY. Using the "MOVE_BLK_VARIANT instruction: Move block", you can move either the entire ARRAY or only individual ARRAY elements dynamically and type-safe. You can freely select the ARRAY limits for the respective source and destination ARRAYs and these do not have to match. However, the data type of the data values that are to be moved must match. This instruction is available in all programming languages.

By using the VARIANT data type, you can use the created program code to move data for other production shifts as well, by specifying a different source and destination area at the block call.

Procedure

1. Create a function using the SCL programming language and give it the name "FC_PartialArrayCopy".
2. Declare the block interface as follows:

FC_PartialArrayCopy

	Name	Data type
1	Input	
2	Count	UDInt
3	SourceIndex	DInt
4	DestinationIndex	DInt
5	Output	
6	InOut	
7	SourceArray	Variant
8	DestinationArray	Variant
9	Temp	
10	Error	Int
11	Constant	
12	<Add new>	
13	Return	
14	FC_PartialArrayCopy	Int

3. Create the SCL program code as follows:
   You can find the program code below as template.

```
IF IS_ARRAY(#SourceArray) AND IS_ARRAY(#DestinationArray)
    AND TypeOfElements(#SourceArray) = TypeOfElements(#DestinationArray) THEN
    #Error := MOVE_BLK_VARIANT(COUNT := #Count, SRC := #SourceArray, SRC_INDEX := #SourceIndex,
                               DEST => #DestinationArray, DEST_INDEX := #DestinationIndex);
END_IF;
#FC_PartialArrayCopy := #Error;
```

4. Create the PLC data type "UDT_MyStruct":

UDT_MyStruct

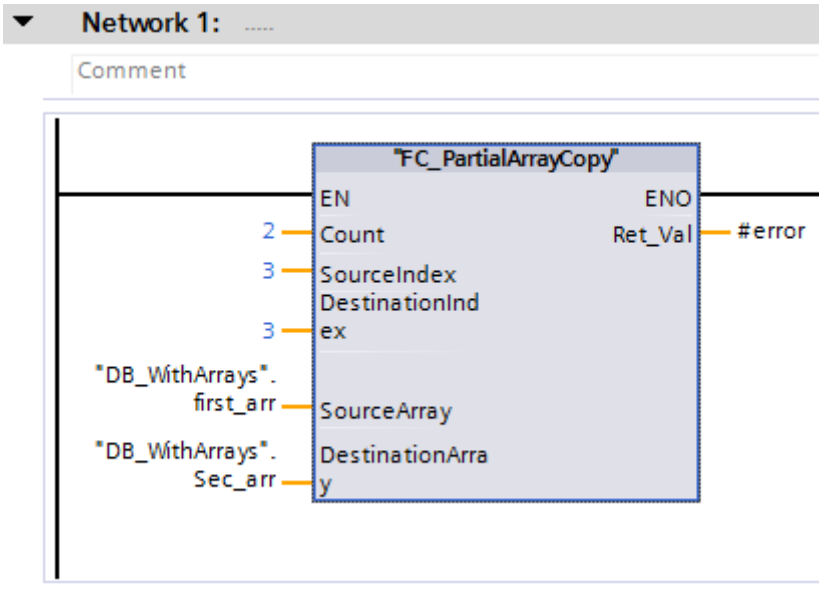
	Name	Data type	Default value
1	a	Bool	1
2	b	Bool	false
3	c	Bool	1
4	d	Bool	false
5	e	Bool	1
6	arr_int	Array[1..100] of Int	
7	f	Int	0
8	g	Real	0.0
9	h	LReal	0.0

5. Create the global data block "DB_WithArrays":

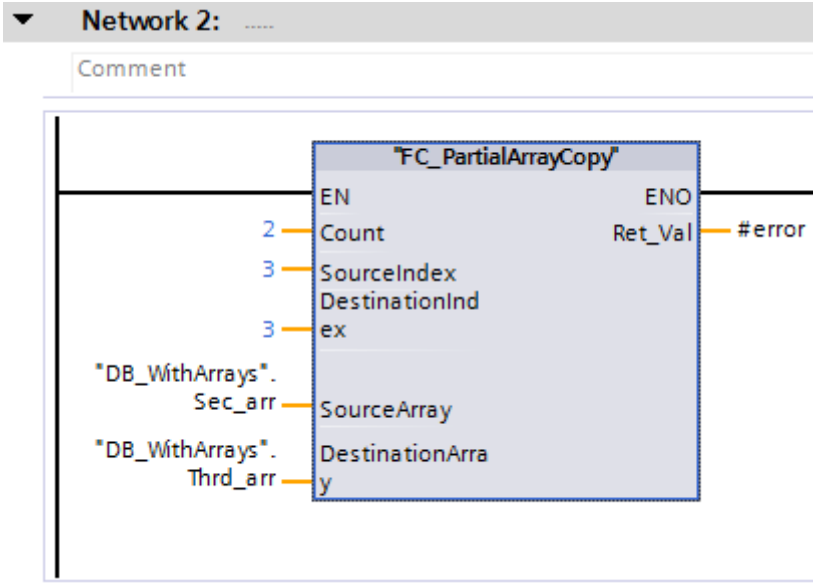
DB_WithArrays

	Name	Data type
1	Static	
2	first_arr	Array[0..100] of "UDT_MyStruct"
3	Sec_arr	Array[1..200] of "UDT_MyStruct"
4	Thrd_arr	Array[1..200] of Real
5	For_arr	Array[1..500] of Real

6. Call the "FC_PartialArrayCopy" function in an organization block, for example OB1, and initialize the parameters with the DB_WithArrays data block. Enter the named constants:



7. Instead of using the first two ARRAYs, which have the data type UDT_MyStruct, you can also use the third and fourth ARRAY, which have the data type REAL:



Result

As soon as the "FC_PartialArrayCopy" block is called in the program cycle, two data values, starting with the fourth element, are copied from the first ARRAY of the "DB_WithArrays" global data block to the second ARRAY of the data block. The copied data values are inserted in the second ARRAY, starting from the fourth element.

SCL program code for copying:

SCL

```
IF IS_ARRAY(#SourceArray) AND TypeOfElements(#SourceArray) =
TypeOfElements(#DestinationArray) THEN
#Error := MOVE_BLK_VARIANT(COUNT := #Count, SRC := #SourceArray, SRC_INDEX := #SourceIndex,
                           DEST => #DestinationArray, DEST_INDEX := #DestinationIndex);
END_IF;
#FC_PartialArrayCopy := #Error;
```

See also

VARIANT (Page 1951)

Programming example: Programming queues (FIFO)

Programming example

In the following example, you program a ring buffer which consists of an ARRAY and is written and read according to the FIFO principle. The program code has a read-VARIANT pointer and a write-VARIANT pointer. Using the VARIANT instructions, you can program the program code robustly and ensure reliable copying or deleting.

Using the VARIANT data type, program sections can be influenced during runtime. The VARIANT pointer is a type-safe pointer, i.e. a type test is performed during runtime. For blocks that have been created with the block property "optimized", sub-functions that were previously programmed with an ANY pointer can now be resolved with a VARIANT pointer. The VARIANT data type is used to transfer structures to system function blocks.

Procedure

1. Create an SCL function block and name it "FIFOQueue".
2. Declare the block interface as follows:

Declaration	Parameters	Data type	Comment
Input	request	BOOL	The instruction is executed when a positive signal edge is detected at the "request" parameter.
	mode	BOOL	0 = The first entry in the ring buffer is returned. 1 = An entry is written at the last position in the ring buffer.
	initialValue	VARIANT	Value with which the ARRAY of the ring buffer is initialized.
Output	error	INT	Error information
InOut	item	VARIANT	The entry that is either returned from the ring buffer or written to the ring buffer.
	buffer	VARIANT	An ARRAY that is used as a ring buffer.
Static	edgeupm	BOOL	Edge memory bit in which the RLO of the previous query is saved.
	firstItemIndex	INT	Index of the oldest entry in the ring buffer
	nextEmptyItemIndex	INT	Index of the next free entry in the ring buffer
Temp	edgeup	BOOL	Result of edge evaluation
	internalError	INT	Error information
	newFirstItemIndex	INT	Variable index
	newNextEmptyItemIndex	INT	Variable index
	bufferSize	UDINT	Number of ARRAY elements in the ring buffer

3. Create the following program code in the "FIFOQueue" function block:

```
(* This program code section is only executed once after a positive signal
edge. If there is no change in the signal state of the result of logic
operation, the program processing of the "FIFOQueue" FB is terminated. *)
#edgeup := #request & NOT #edgeupm;
#edgeupm := #request;
IF NOT (#edgeup) THEN
   RETURN;
END_IF;
```

```
// ------Validation of whether all parameter inputs are valid.----
(* This program code section checks whether the ring buffer is an ARRAY. If
so, the number of the ARRAY elements is read out. If it is not an ARRAY,
the program execution is terminated at this point and the error code "-10"
is output. *)
IF NOT (IS_ARRAY(#buffer)) THEN
   #error := -10;
   RETURN;
ELSE
   #bufferSize := CountofElements(#buffer);
END_IF;

(* This program code section checks whether the data type of the ARRAY
elements matches the data type of the entry (tag #item). If the data types
do not match, the program execution is terminated at this point and the
error code "-11" is output. *)
IF NOT (TypeOf(#item) = TypeOfElements(#buffer)) THEN
   #error := -11;
   RETURN;
END_IF;

(* This program code section checks whether the initial value of the ring
buffer matches the entry (tag #item). If the data types do not match, the
program execution is terminated at this point and the error code "-12" is
output. *)
IF NOT (TypeOf(#item) = TypeOf(#initialValue)) THEN
   #error := -12;
   RETURN;
END_IF;

(* This program code section checks whether the variable indices are within
the ARRAY limits. If they are not, the program execution is terminated at
this point and either error code "-20" or "-21" is output depending on the
index. *)
IF (#nextEmptyItemIndex >= #bufferSize) THEN
   #error := -20;
   RETURN;
END_IF;
IF (#firstItemIndex >= #bufferSize) THEN
   #error := -21;
   RETURN;
END_IF;
```

```
//-----------Program code execution, depending on the Mode
parameter-------------
// The execution of the instructions depends on the signal state of the Mode
parameter.
IF #mode = 0 THEN

// If the Mode parameter has the signal state "0", the first entry from the
passed ring buffer is returned.
(* This program code section checks whether the ring buffer is empty. If
this is the case, program execution is terminated at this point and the
error code "-40" is output. *)
   IF (#firstItemIndex = -1) THEN
      #error := -40;
      RETURN;
   END_IF;

// This program code section returns the first entry of the ring buffer.
   #internalError := MOVE_BLK_VARIANT(SRC := #buffer,
                                      COUNT := 1,
                                      SRC_INDEX := #firstItemIndex,
                                      DEST_INDEX := 0,
                                      DEST => #item);

   IF (#internalError = 0) THEN
(* This program code section checks whether the ring buffer contains ARRAY
elements. If it does, the first entry is passed further on and the index is
incremented by 1. *)
      #internalError := MOVE_BLK_VARIANT(SRC := #initialValue,
                                         COUNT := 1,
                                         SRC_INDEX := 0,
                                         DEST_INDEX := #firstItemIndex,
                                         DEST => #buffer);

// This program code section calculates the new index of the first entry.
      #newFirstItemIndex := #firstItemIndex +1;
      #newFirstItemIndex := #newFirstItemIndex MOD #bufferSize;

// This program section checks whether the ring buffer is empty.
      IF (#nextEmptyItemIndex = #newFirstItemIndex) THEN
// If the ring buffer is empty, the index is set to 0.
         #firstItemIndex := -1;
         #nextEmptyItemIndex := 0;
      ELSE
// The index of the first entry is changed.
         #firstItemIndex := #newFirstItemIndex;
      END_IF;
```

```
   END_IF;
ELSE


// If the Mode parameter has the signal state "1", the entry is written to
the passed ring buffer.
(* This program code section checks whether the ring buffer is full. If this
is the case, program execution is terminated at this point and the error
code "-50" is output. *)
   IF (#nextEmptyItemIndex = #firstItemIndex) THEN
      #error := -50;
      RETURN;
   END_IF;

// This program code section writes the entry to the ring buffer.
   #internalError := MOVE_BLK_VARIANT(SRC := #item,
                                      COUNT := 1,
                                      SRC_INDEX := 0,
                                      DEST_INDEX := #nextEmptyItemIndex,
                                      DEST => #buffer);

   IF (#internalError = 0) THEN
// This program code section increments the index by 1 and calculates the
new empty entry index.
      #newNextEmptyItemIndex := #nextEmptyItemIndex +1;
      #newNextEmptyItemIndex := #newNextEmptyItemIndex MOD #bufferSize;
      #nextEmptyItemIndex := #newNextEmptyItemIndex;

(* This program code section checks which index the "#firstItemIndex" tag
has. If the number = -1, the ring buffer is initialized and the entry is
written to the ring buffer. Therefore, "0" must be assigned to the tag. *)
      IF (#firstItemIndex = -1) THEN
         #firstItemIndex := 0;
      END_IF;
   END_IF;
END_IF;

//-------------------------Local error handling-----------------------------
(* This program code section checks whether a local error has occurred. If
this is the case, program execution is terminated at this point and the
error code "-100" is output. *)
IF (#internalError > 0) THEN
   #error := -100;
   RETURN;
END_IF;
```

```
// If no error occurred during program execution, the error code "0" is
output.
#error := 0;
```

Result

Call the SCL function block at the position in your program at which the FIFO queue is to run.

6.4.8.2 Using the DB_ANY data type

Using the DB_ANY data type (S7-1200/1500)

The DB_ANY data type is used to identify any data block. For CPUs of the S7-1200/1500 series, you have the option of accessing a data block that is not yet available during programming. For this purpose, create a block parameter of the DB_ANY data type in the block interface of the accessing block. The data block name or a tag of data type DB_ANY which was previously assigned to the data block name is transferred to this parameter during runtime. You can symbolically process the content of a data block by means of the following instructions:

- VARIANT_TO_DB_ANY: Convert VARIANT to DB_ANY
- DB_ANY_TO_VARIANT: Convert DB_ANY to VARIANT

For more information about the instructions, refer to "Basic instructions > LAD/FBD/STL/SCL > Conversion operations > VARIANT".

The advantage of this procedure: You can create the program code before you know which data block will be processed.

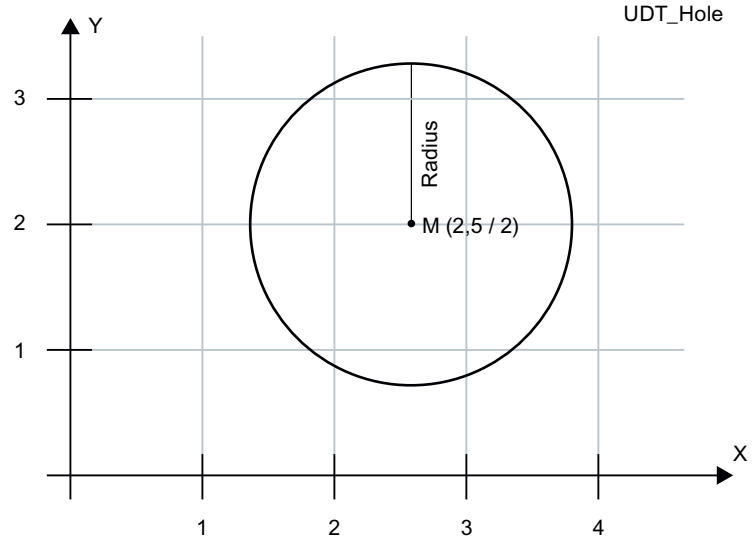
Programming example

The following example shows you how to use the DB_ANY data type:

A punching machine can punch out a variety of geometric shapes. The punching jobs are transferred to the machine and there are specific job data for each individual job. The job data differ in their job type as well as in their values.

Procedure - Creating the PLC data types

In the first job, a round hole is to be punched out of a piece of sheet metal. For the punching machine to be able to execute this job, it needs the center point coordinates and the radius of the hole. You can transfer these job data to the punching machine collectively in a PLC data type (UDT).

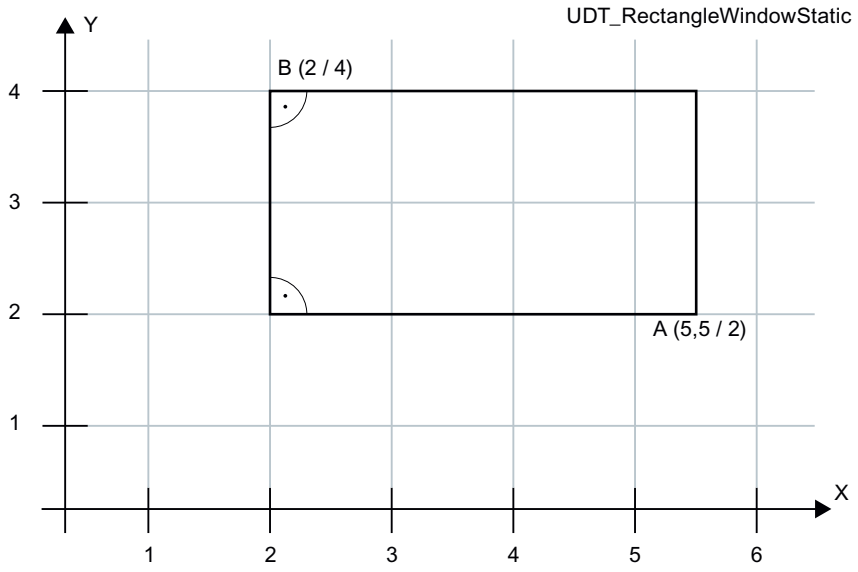


Create the PLC data type "UDT_Hole" to transfer the job data:

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare the following lines within the PLC data type:
   X-coordinate > REAL
   Y-coordinate > REAL
   Radius > REAL

UDT_Hole			
		Name	Data type
1		X-coordinate	Real
2		Y-coordinate	Real
3		Radius	Real

A rectangle is to be punched out of a piece of sheet metal in the second job. For this job, you need two coordinates which define the upper left-hand point and the bottom right-hand point of the rectangle. You can transfer these job data to the punching machine collectively in the PLC data type "UDT_RectangleWindowStatic".



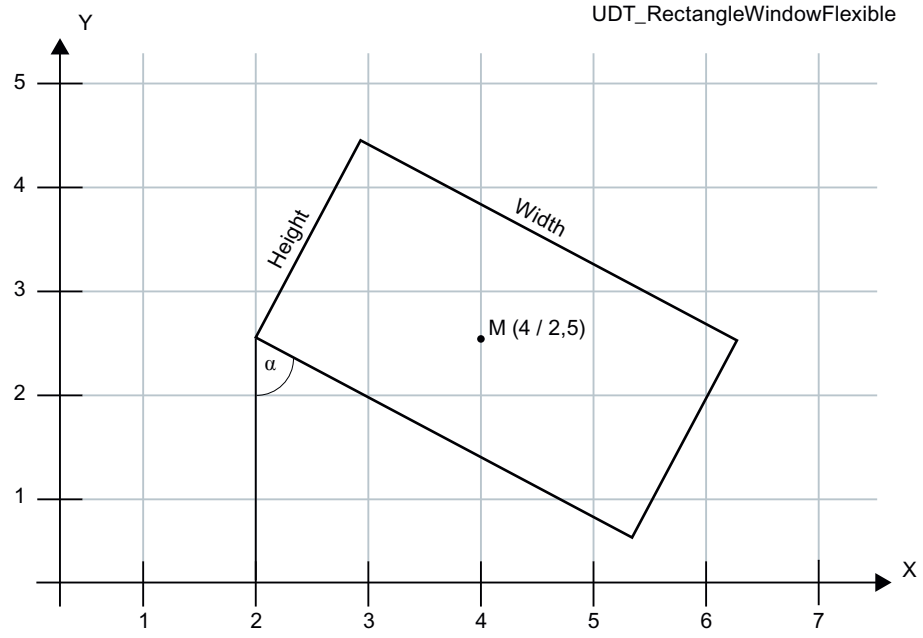
Create the PLC data type "UDT_RectangleWindowStatic":

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare the following lines within the PLC data type:
   X1-coordinate > REAL
   Y1-coordinate > REAL
   X2-coordinate > REAL
   Y2-coordinate > REAL

UDT_RectangleWindowStatic			
		Name	Data type
1		X1-coordinate	Real
2		Y1-coordinate	Real
3		X2-coordinate	Real
4		Y2-coordinate	Real

The job data of the "UDT_RectangleWindowStatic" can only be used to punch out rectangles whose edges are aligned parallel to the x and y axis.

If you want to punch out a rectangle with different alignment, i.e., not parallel to the x and y axis, you need an additional PLC data type. In it, you can specify the height and width, for example, as well as the alignment of the rectangle to the x axis by means of an angle.



Create the PLC data type "UDT_RectangleWindowFlexible":

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare the following lines within the PLC data type:
   X-coordinate > REAL
   Y-coordinate > REAL
   Height > REAL
   Width > REAL
   Angle > REAL

UDT_RectangleWindowFlexible			
		Name	Data type
1		X-coordinate	Real
2		Y-coordinate	Real
3		Height	Real
4		Width	Real
5		Angle	Real

The x and y coordinates specify the center of the rectangle.

Procedure - Creating the data blocks

In the next section, you will learn how to transfer simple geometric shapes whose job data you have defined in the PLC data types to the punching machine. The punch jobs are broken down into individual punches in the program code; these punches are then executed consecutively by the punching machine. The punching machine has a cross table on which a sheet metal has been firmly clamped. You can move a cross table along the x and/or y axis just as in a coordinate system. The cross table is moved by two motors. The tool has different dies to punch out shapes from the sheet metal, such as circles and rectangles of different sizes. The tool can also be rotated by up to 90 degrees to cut out rectangles at different alignments.

You now use the PLC data types to create a data block. The data block then includes the specific values, for example, for a hole.

Create the data block "DB_OrderHole":

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_OrderHole".
4. Select "Global-DB" as the type of the data block.
5. Click "OK".

Create the following tag in the data block and enter the corresponding start values:

DB_OrderHole

	Name	Data type	Start value
1	Static		
2	Hole	"UDT_Hole"	
3	X-coordinate	Real	10.0
4	Y-coordinate	Real	900.0
5	Diameter	Real	4.0

To manufacture a specific sheet metal part such as the side panel of a control cabinet, for example, the necessary geometric shapes are loaded to the punching machine. For this purpose, you need to create another data block which includes a list of data blocks.

Create the data block "DB_OrderList":

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_OrderList".
4. Select "Global-DB" as the type of the data block.
5. Click "OK".

Create the following job list in the data block:

DB_OrderList			
		Name	Data type
1		Static	
2		Order	Array[0..100] of DB_ANY
3		Order[0]	DB_ANY
4		Order[1]	DB_ANY
5		Order[2]	DB_ANY
6		Order[3]	DB_ANY
7		Order[4]	DB_ANY
8		Order[5]	DB_ANY
9		Order[6]	DB_ANY
10		Order[7]	DB_ANY
11		Order[8]	DB_ANY

Procedure - Creating the program code

The punching machine should now start processing the jobs. If it is already processing the jobs, it should get the next job from the job list and prepare it.

1. Create an SCL function block.
2. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
3. Click the "Function block (FB)" button.
4. Specify the name "FB_PickNextOrder".

5. Declare the block interface as follows:

FB_PickNextOrder

	Name	Data type
1	Input	
2	Output	
3	InOut	
4	Static	
5	OrderReady	Bool
6	OrderNo	DInt
7	Order	DB_ANY
8	Temp	
9	Error	Int

6. Write the following program code:

```
1
2 IF #OrderReady THEN
3     #Order := "DB_OrderList".Order[#OrderNo];
4     #OrderNo := #OrderNo + 1;
5     #OrderReady := false;
6 END_IF;
```

The next job on the list is prepared by breaking down the current punch job into individual punches. The punching machine must be able to recognize which punch job it is processing.

1. Create an SCL function.
2. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
3. Click the "Function (FC)" button.
4. Specify the name "FC_PrepareOrder".

5. Declare the block interface as follows:

FC_PrepareOrder

	Name	Data type
1	Input	
2	Order	DB_ANY
3	Output	
4	Error	Int
5	InOut	
6	Temp	
7	WindowFlexible	"UDT_RectangleWindowFlexible"
8	WindowStatic	"UDT_RectangleWindowStatic"
9	Hole	"UDT_Hole"
10	a	Variant
11	Constant	
12	Return	
13	FC_PrepareOrder	Void

6. Write the following program code. The calls of the three functions are not yet given a red underline because they still have to be created.

```
1
2  #a := DB_ANY_TO_VARIANT(in := #Order, err => #Error);
3
4 CASE TypeOf(#a) OF
5     UDT_Hole:
6         VariantGet(SRC := #a,
7                    DST => #Hole);
8         "FC_PrepareHole"(#Hole);
9         ;
10
11    UDT_RectangleWindowStatic:
12        VariantGet(SRC := #a,
13                   DST => #WindowStatic);
14        "FC_PrepareWindowStatic"(#WindowStatic);
15        ;
16
17    UDT_RectangleWindowFlexible:
18        VariantGet(SRC := #a,
19                   DST => #WindowFlexible);
20        "FC_PrepareRectangleWindow"(#WindowFlexible);
21        ;
22    ELSE
23        ;
24 END_CASE;
```

The "VariantGet" instruction copies the information from the #a tag to the #Hole tag.

In the following section, you create the functions "FC_PrepareHole" and "FC_PrepareWindowStatic".

A separate function is created for each job type. The punch jobs are broken down into individual punches here and collected in an ARRAY.

1. Create a PLC data type.
2. Double-click the "Add new data type" command under "PLC data types".
   A new PLC data type with the name "UserDataType_x" is created.
3. Rename the PLC data type to "UDT_Punch".
4. Declare the following lines within the PLC data type:
   Tool > DINT
   x > REAL
   y > REAL
   w > BOOL

UDT_Punch

	Name	Data type
1	Tool	DInt
2	x	Real
3	y	Real
4	w	Bool

5. Create the global data block "DB_PunchList":

DB_PunchList

	Name	Data type
1	Static	
2	Punch	Array[0..100] of "UDT_Punch"
3	Punch[0]	"UDT_Punch"
4	Punch[1]	"UDT_Punch"
5	Punch[2]	"UDT_Punch"
6	Punch[3]	"UDT_Punch"
7	Punch[4]	"UDT_Punch"
8	Punch[5]	"UDT_Punch"
9	Punch[6]	"UDT_Punch"
10	Punch[7]	"UDT_Punch"
11	Punch[8]	"UDT_Punch"

To prepare the punch job for a hole, create an SCL function and name it "FC_PrepareHole".

1. Declare the block interface as follows:

FC_PrepareHole

		Name	Data type
1		Input	
2		a	"UDT_Hole"
3		Output	
4		InOut	
5		Temp	
6		Constant	
7		Return	
8		FC_PrepareHole	Void

2. Write the following program code:

```
1
2  IF #a.Radius = 4.0 THEN
3      "DB_PunchList".Punch[0].Tool := 1;
4
5  ELSIF #a.Radius = 6.0 THEN
6      "DB_PunchList".Punch[0].Tool := 2;
7
8  ELSE
9      ;
10     "DB_PunchList".Punch[0].x := #a."X-coordinate";
11     "DB_PunchList".Punch[0].y := #a."Y-coordinate";
12
13 END_IF;
```

To prepare the punch job for a window, you need a function which combines four punch sequences into the punch job. Create an SCL function and name it "FC_PrepareWindowStatic".

1. Declare the block interface as follows:

FC_PrepareWindowStatic

	Name	Data type
1	Input	
2	a	"UDT_RectangleWindowStatic"
3	Output	
4	InOut	
5	Temp	
6	delta	Real
7	x	Real
8	x2	Real
9	y	Real
10	i	DInt
11	Constant	
12	Return	
13	FC_PrepareWindowStatic	Void

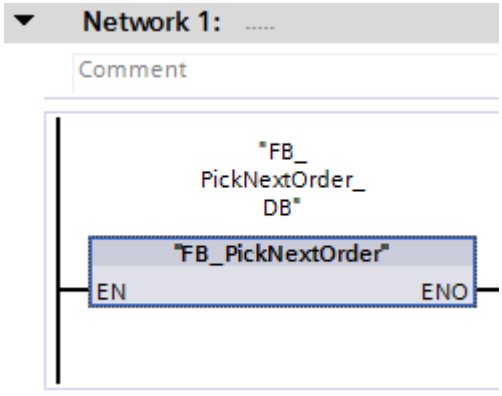
2. Write the following program code:

```
1
2  #delta := 6.0;  // Defined by tool 7
3  #i := 0;
4  #x := #a."X1-coordinate" + #delta;
5 WHILE #x < #a."X2-coordinate" - #delta  DO
6      "DB_PunchList".Punch[#i].Tool := 7;
7      "DB_PunchList".Punch[#i].x := #x;
8      "DB_PunchList".Punch[#i].y := #y + #delta;
9      "DB_PunchList".Punch[#i].w := 0; // Don't rotate
10     #x := #x + 2 * #delta;
11 END_WHILE;
12
13 #y := #a."Y1-coordinate" + #delta;
14 WHILE #y < #a."Y2-coordinate" - #delta DO
15     "DB_PunchList".Punch[#i].Tool := 7;
16     "DB_PunchList".Punch[#i].x := #x2 + #delta;
17     "DB_PunchList".Punch[#i].y := #y;
18     "DB_PunchList".Punch[#i].w := 0; // Don't rotate
19     #y := #y + 2 * #delta;
20 END_WHILE;
```

Call the SCL function block "FC_PrepareOrder" in the SCL function block "FB_PickNextOrder":

```
1
2  IF #OrderReady THEN
3      #Order := "DB_OrderList".Order[#OrderNo];
4      #OrderNo := #OrderNo + 1;
5      #OrderReady := false;
6      "FC_PrepareOrder"(Order:=#Order,
7                        Error=>#Error);
8
9  END_IF;
```

Then, call the "FB_PickNextOrder" in OB1:



Result

This example shows you how to use the instruction "DB_ANY_TO_VARIANT: Convert DB_ANY to VARIANT" to determine the PLC data type of a data block and how to select and execute a suitable function based on this.

6.4.8.3 Using PLC data types (UDT)

Using a PLC data type

PLC data types (UDT) are data structures that you define yourself and that can be used multiple times in the program. The structure can be composed of multiple elements of different data types. You define the data types of the individual elements during the declaration of a PLC data type.

PLC data types are frequently used if an assembled data record with various data types is required and these are to be processed from different points in the program. For example, these could be:

- Data records for material tracking
- Parameter sets for a motor setting
- Recipes

The use of PLC data types provides you with the following benefits:

- The elements of a PLC data type can also be addressed indirectly, which means the address is variable and is not calculated until during runtime.
- Tags based on a PLC data type inherit all properties of the PLC data type. When there is a change to the PLC data type, all tags based on this PLC data type are therefore adapted automatically.
- Using universal symbolism makes the program easier to read, because the names of the individual elements of a PLC data type are displayed in the program.
- Optimum utilization of the high performance of an S7-1500 CPU.
- The PLC data type can be passed as a complete structure for a block call.
- A simplified call interface due to a lower number of parameters to be supplied.

Procedure in STEP 7 V5.x

STEP 7 V5.x already gave you the option to create a data record as a structured tag using the STRUCT data type or a PLC data type (UDT). However, the performance was adversely affected by the use of symbolic addressing.

The declaration in the data blocks was mostly implemented as an anonymous structure. The blocks themselves were then programmed to pass the values of the structure as actual parameters and the calculated values were copied back into the structure. This enabled you to also pass the data block number and use absolute addressing in the block. The number of parameters that you had to supply was often very large. The actual data was stored in the data blocks and the calculated values were passed to other blocks. However, no symbolism was available when passing the data block tags.

Programming example in STEP 7 TIA Portal

You can assign both a formula parameter and an actual parameter to a PLC data type. This means that you no longer have to declare individual parameters. If a block has an input parameter that is based on a PLC data type, you must transfer a tag of the same PLC data type as actual parameter.

The following example shows the call and the parameter assignment of a function block (FB) with two formal parameters:

1. To create a PLC data type, double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Rename the PLC data type to "UDT_Material".
3. Declare the following lines within the PLC data type:
   ArticleNumber > Data type: DINT
   ArticleName > Data type: STRING
   Amount > Data type: REAL
   Unit > Data type: STRING

UDT_Material

		Name	Data type
1		ArticleNumber	DInt
2		ArticleName	String
3		Amount	Real
4		Unit	String

Use the PLC data type within a global data block. You can specify the PLC data type either directly as data type of the data block or within the data block as data type of a tag.

Create a global data block for this purpose:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_MaterialBuffer".
4. Select the type of the data block "ARRAY DB".
5. Select the PLC data type "UDT_Material" as ARRAY data type.

6. Specify "1000" as high ARRAY limit.
7. Click "OK".

DB_MaterialBuffer

	Name	Data type
1	DB_MaterialBuffer	Array[0..1000] of "UDT_Material"
2	DB_MaterialBuffer[0]	"UDT_Material"
3	ArticleNumber	DInt
4	ArticleName	String
5	Amount	Real
6	Unit	String
7	DB_MaterialBuffer[1]	"UDT_Material"
8	DB_MaterialBuffer[2]	"UDT_Material"
9	DB_MaterialBuffer[3]	"UDT_Material"
10	DB_MaterialBuffer[4]	"UDT_Material"
11	DB_MaterialBuffer[5]	"UDT_Material"

At the function block call, interconnect the formal parameters with tags from the global data block "DB_MaterialBuffer".

1. Create an SCL function block and name it "FB_Material".
2. Declare the block interface as follows:

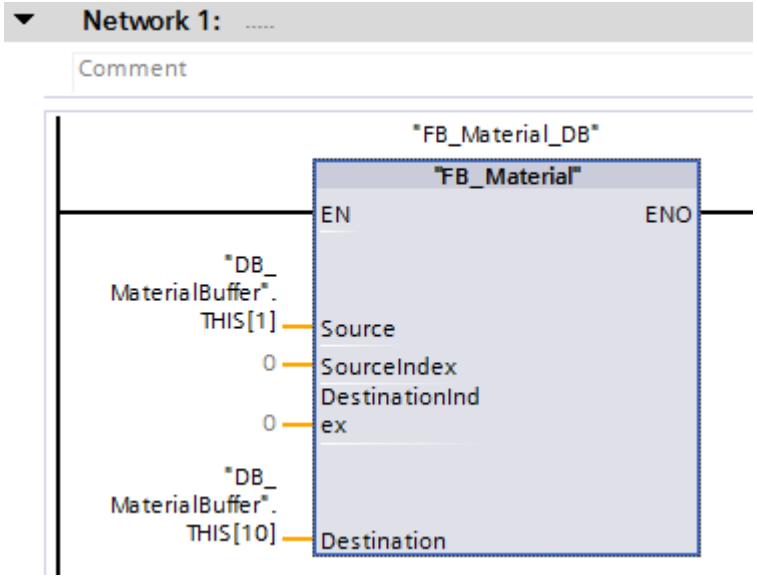
FB_Material

	Name	Data type
1	Input	
2	Source	Variant
3	SourceIndex	DInt
4	DestinationIndex	DInt
5	Output	
6	InOut	
7	Destination	Variant
8	Static	
9	Temp	
10	Result	Int
11	Constant	

3. Write the following program code:

```
1
2 #Result := MOVE_BLK_VARIANT(SRC:=#Source,
3                              COUNT:=3,
4                              SRC_INDEX:=#SourceIndex,
5                              DEST_INDEX:=#DestinationIndex,
6                              DEST=>#Destination);
```

4. Call the "FB_Material" function block in OB1 and interconnect the formal parameters with tags of the global data block "DB_MaterialBuffer":



The material data are moved within the global data block "DB_MaterialBuffer".

See also

PLC data types (Page 1954)

6.4.8.4 Calculating with floating-point numbers (REAL and LREAL) in SCL

Representation of the accuracy of floating-point numbers

The data type REAL, for example, is specified and calculated in the program with an accuracy of 6 decimal places. For the calculation of floating-point numbers (REAL and LREAL), it should be noted that this accuracy applies in general to each individual step of the calculation.

The exponents are adjusted when floating-point numbers are added and subtracted. The base and the exponents are thus equal during the addition and subtraction and only the mantissas are added. For additional information on the structure of floating-point numbers, refer to "See also".

Programming example

In the following programming example, you are to perform a calculation in which two operands of the data type REAL are to be added and one is to be subtracted. In the next step of the calculation, the constant 1 is divided by the previous result. To do this, create a global data block in which you declare your operands and a function in which you program the calculation operations.

Calculation formulas

```
y = a + b - c
Z = 1/y
```

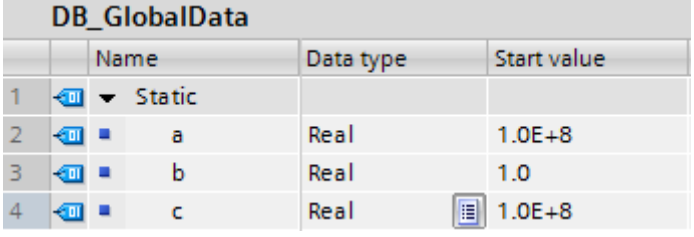
The operands are stored with the following values:

Operand	Value	REAL value
a	100 000 000	$1.000000*10^8$
b	1	$1.000000*10^0$
c	100 000 000	$1.000000*10^8$

Procedure

Create the data block "DB_GlobalData":

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_GlobalData".
4. Select "Global DB" as the type of the data block.
5. Click "OK".
6. Create the following tags in the data block and enter the corresponding start values:



The start value of both tags is 100000000.0 and is converted into 1.0E+8 according to the data type REAL.

Create an SCL function and name it "FC_Calculate".

1. Declare the block interface as follows:

FC_Calculate		
	Name	Data type
1	Input	
2	Output	
3	InOut	
4	Temp	
5	z	Real
6	y	Real
7	Constant	
8	Return	
9	FC_Calculate	Void

2. Write the following formulas in your program code and establish an online connection to see the result:

SCL

```
#y := "DB_GlobalData".a + "DB_GlobalData".b - "DB_GlobalData".c;
#z := 1/#y;
```

#y	0.000000E+000
"Global data".a	1.000000E+008
"Global data".b	1.000000E+000
"Global data".c	1.000000E+008

As you can see, the result at the operand is #y = 0, even though the number 1 is actually expected as result.

The incorrect result comes about as follows:

1. In the first step of the calculation, the operands a + b are added. The REAL values of the two operands (a = $1.000000*10^8$ and b = $1.000000*10^0$) look as follows after the exponents have been adjusted:
   a = $1.000000*10^8$ and b = $0.00000001*10^8$. The last two places of the second number (operand b) are truncated, as they can no longer be represented due to the accuracy of 6 decimal places. Therefore, a 0 instead of a 1 is added to the operand.
2. In the second calculation step, the operand C is subtracted from the result of the preceding calculation step (intermediate result = $1.000000*10^8$ - c = $1.000000*10^8$ is $0.000000e^0$).
3. If you now calculate the operand z in the next calculation step, you try to divide by zero.

#z	16#7F800000
#y	0.000000E+000

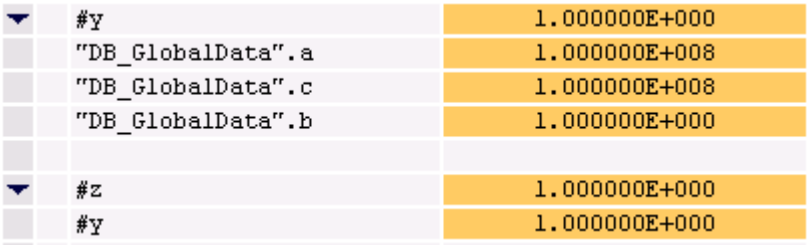
1. Possible solution

To work around such cases, you can simply adjust your calculation formula. Write the formula as follows instead:

Calculation formulas

```
y = a - c + b
Z = 1/y
```

Since the result 0.000000e^{0} is available in this case after the first calculation step (operand a - c), the addition of the REAL number in the second calculation step (intermediate result + b) leads to the correct result (y = 0.000000*10^{0}+ 1.000000*10^{0} = 1.000000*10^{0}).



▼	#y	1.000000E+000
	"DB_GlobalData".a	1.000000E+008
	"DB_GlobalData".c	1.000000E+008
	"DB_GlobalData".b	1.000000E+000
▼	#z	1.000000E+000
	#y	1.000000E+000

We recommend that you check how the calculation can be made most effectively before you program a calculation.

2. Possible solution

To calculate the above named formulas, use the LREAL data type instead of the REAL data type. Since the data type is processed with an accuracy of 15 decimal places, this problem does not even arise.

1. In the global data block "DB_GlobalData", create three new tags with the same values, each with the data type LREAL.

DB_GlobalData

	Name	Data type	Start value
1	Static		
2	a	Real	1.0E+8
3	a_LREAL	LReal	100000000.0
4	b	Real	1.0
5	b_LREAL	LReal	1.0
6	c	Real	1.0E+8
7	c_LREAL	LReal	100000000.0

2. In the block interface of the FC "FC_Calculate", also declare two new tags with the data type LREAL.

FC_Calculate

	Name	Data type
1	Input	
2	Output	
3	InOut	
4	Temp	
5	z_LREAL	LReal
6	z	Real
7	y_LREAL	LReal
8	y	Real
9	Constant	
10	Return	
11	FC_Calculate	Void

3. Use the new LREAL tags for the formulas in your program code and establish an online connection to see the result.

SCL

```
#y_LREAL := "DB_GlobalData".a_LREAL + "DB_GlobalData".b_LREAL -
"DB_GlobalData".c_LREAL;
#z_LREAL := 1/#y_LREAL;
```

▼	#y_LREAL	1.00000000000000E+000
	"DB_GlobalData".a_LREAL	1.00000000000000E+008
	"DB_GlobalData".b_LREAL	1.00000000000000E+000
	"DB_GlobalData".c_LREAL	1.00000000000000E+008
▼	#z_LREAL	1.00000000000000E+000
	#y_LREAL	1.00000000000000E+000

See also

REAL (Page 1925)

LREAL (Page 1926)

Invalid floating-point numbers (Page 1927)

6.4.8.5 Calculating with constants in SCL

Interpretation of typed and non-typed constants

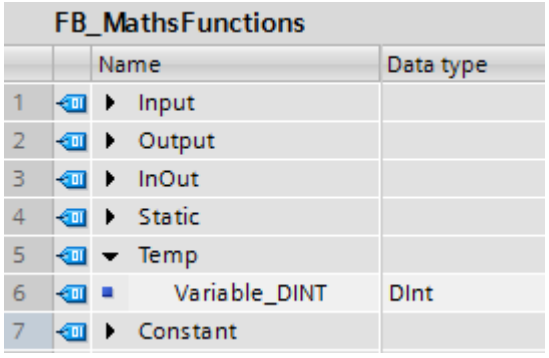
Constants are data with a fixed value that cannot change during program runtime. Constants can be read by various program elements during the execution of the program but cannot be overwritten. There are defined notations for the value of a constant, depending on the data type and data format. A distinction is made between the typed and non-typed notation.

We recommend that you do not mix typed and non-typed constants within mathematical functions as you may otherwise be faced with unwanted implicit conversions and thus end up with incorrect values.

Programming example

In the following programming example, you see a calculation operation with a typed and a non-typed constant.

1. Create an SCL function block and name it "FB_MathsFunctions".
2. Declare the "Variable_DINT" tag in the "Temp" section of the block interface.

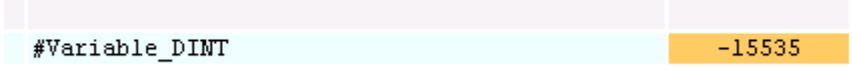


3. Write the following program code:
   Variable_DINT := INT#1 +50000;
   In this math operation, the typed constant INT#1 and the non-typed constant 50000 are to be added. The non-typed constant 50000 is underlined in yellow in the software to indicate that the constant value is outside the permitted range of the data type INT.

```
1
2  #Variable_DINT := INT#1 + 50000;
```

To see the result, go online.

1. Compile the SCL function block "FB_MathsFunctions" by right-clicking the command "Compile > Software (only changes)" to execute it.
2. Download the block with the command "Download to device > Software (only changes)".
3. Go online and monitor your block.



The data type of the typed constant defines the data type of the addition. This means that the addition is performed in the data type area INT. In the first step, the non-typed constant 50000 is implicitly converted into the data type INT. However, the conversion leads to a negative value (-15536). This value is then added to the typed constant (INT#1). The result is -15535. Since the tag to which the result of the addition is to be written is declared with the data type DINT, the number -15535 is implicitly converted into the data type DINT and written to the tag "Variable_DINT". However, the result remains negative.

1. Possible solution

One option for avoiding this undesired result is to type both constants. If you type both constants, the longer data type determines the calculation operation.

1. Write the following program code in the "FB_MathsFunctions" SCL function block:

```
3
4  #Variable_DINT := INT#1 + DINT#50000;
```

   In this calculation operation, the typed constant INT#1 and the typed constant DINT#50000 are to be added.

To see the result, go online.

1. Compile the SCL function block "FB_MathsFunctions" by right-clicking the command "Compile > Software (only changes)" to execute it.
2. Download the block with the command "Download to device > Software (only changes)".
3. Go online and monitor your block.

	#Variable_DINT	50001

   The constant INT#1 is converted into the DINT data type and the addition of the two constants is executed in the DINT data type area.

2. Possible solution

Another option for avoiding this undesired result is not to type both constants. If you do not type both constants, these are then interpreted as the widest possible data type on the current CPU. This means that on an S7-1500 series CPU, the two constants are interpreted as LINT data type.

1. Write the following program code in the "FB_MathsFunctions" SCL function block:

```
1
2  #Variable_DINT := 1 + 50000;
```

   In this calculation operation, the non-typed constant 1 and the non-typed constant 50000 are to be added.

To see the result, go online.

1. Compile the SCL function block "FB_MathsFunctions" by right-clicking the command "Compile > Software (only changes)" to execute it.
2. Download the block with the command "Download to device > Software (only changes)".
3. Go online and monitor your block.

The constants 1 and 50000 are interpreted as LINT data type and the result of the addition is again converted into the DINT data type.

See also

Overview of the valid data types (Page 1908)

First steps

7

7.1 Getting Started Documentation

Getting Started with the TIA Portal

The Getting Started documentation is available to help you begin using the TIA portal.

The Getting Started documentation contains instructions which show you, step-by-step how to create a project in the TIA portal and give you the chance to get a quick overview of all the possibilities the TIA portal offers you.

Contents

The Getting Started documents describe the creation of a single, continuous project for STEP 7 and WinCC that is extended with each chapter. You start with simple basic functions, and use more complex ones as you continue with the creation of the project.

In addition to the step-by-step instructions, the Getting Started documents also give you background information that explains the functions used and illustrate how they relate to each other.

Target audience

The Getting Started documents are intended for beginners, but are also useful for users migrating from a previous version of SIMATIC STEP 7 and WinCC.

Getting Started documentation

The documentation is available, free of charge at the Service&Support (https://support.automation.siemens.com) portal in PDF form.

You can download the documents here:

- STEP 7 Basic and WinCC Basic (as of TIA Portal V10.5) (http://support.automation.siemens.com/WW/view/en/40263542/0/en)
- STEP 7 Professional and WinCC Advanced (as of TIA Portal V11) (http://support.automation.siemens.com/WW/view/en/28919804/133300)

Multimedia Getting Started

A multimedia Getting Started is also available for the TIA Portal as of version 13.0.

The following link will take you to the home page of the Getting Started:

http://www.automation.siemens.com/salesmaterial-as/interactive-manuals/getting-started_simatic-s7-1500/_content/EN/content_en.html (http://www.automation.siemens.com/

salesmaterial-as/interactive-manuals/getting-started_simatic-s7-1500/_content/EN/content_en.html)

Introduction to the TIA Portal 8

8.1 User interface and operation

8.1.1 Starting, setting up, and exiting the TIA Portal

8.1.1.1 Starting and exiting the TIA Portal

Starting the TIA Portal

To start the TIA Portal, follow these steps:

1. In Windows, select "Start > All Programs > Siemens Automation > TIA Portal V13".
   The TIA Portal opens with the last settings used.

Exiting the TIA Portal

To exit the TIA Portal, follow these steps:

1. In the "Project" menu, select the "Exit" command.
   If the project contains any changes that have not been saved, you will be asked if you wish to save them.
   - Select "Yes" to save the changes in the current project and close the TIA Portal.
   - Select "No" to close the TIA Portal without saving the most recent changes in the project.
   - Select "Cancel" to cancel the closing procedure. The TIA Portal will remain open if you select this option.

8.1.1.2 Overview of the program settings

Overview

The following table shows the application settings that you can make:

Group	Setting	Description
General settings	User name	The user name of the user. The user name is stored in the project properties when a new project is created.
	User interface language	Language for the program interface
	Mnemonic	Specifies the mnemonics for programming: "German" uses the German mnemonics, for example, "E1.0". "International" uses international mnemonics, for example, "I1.0". For information on the differences in the mnemonics of the individual commands, refer to the description of the relevant programming language.
	Show list of recently used projects	Number of entries in the list of recently used projects in the "Project" menu
	Load most recent project during startup	The last opened project is opened automatically after the TIA Portal starts.
	Displaying truncated text in full	Texts which are truncated due to their length are displayed in a tooltip.
	Show tooltips (context-sensitive help is available)	Tooltips are displayed and you get context-sensitive help. If this function is disabled, you can open the tooltip with <F1>.
	Open cascade automatically in tooltips	After a brief time, the tooltips automatically expand to display a cascade containing additional help. If this option is cleared, the tooltips must be manually expanded.
Reset to default	All application settings	All changes that you made in the TIA Portal after installation are undone.
	Layout of the editors	Resets the complete layout of the application to the factory state.
	Show all alarm windows	All alarm windows whose appearance was manually suppressed are displayed again.
Start view	Most recent view	Starts the program in the last view that was used. This can be either the portal view or the project view.
	Portal view	Starts the TIA Portal in the portal view every time, irrespective of the last view you worked in.
	Project view	Starts the TIA Portal in the project view every time, irrespective of the last view you worked in.
View for objects in overview	Details	When several views are available, the detail view opens by default, for example, in the overview window.
	List	When several views are available, the list view opens by default, for example, in the overview window.
	Thumbnails	When several views are available, the symbol view opens by default, for example, in the overview window.

Group	Setting	Description
Storage settings	Recently used storage location	When a project is saved the first time, the most recently used file path is set by default.
	Default storage location	Enables the specification of file paths for: • Projects • Libraries • Configuration file for corporate libraries
Data exchange	Storage location for data import	Imported files are searched for at this storage path by default.
	Storage location for data export	This is the default storage path for the data export.
	Storage location for support packages	When support packages are downloaded, they are stored in the specified storage path and can be installed from there.
	Storage location for log files	Log files are stored at the location specified here.

See also

Starting and exiting the TIA Portal (Page 301)

Resetting the user interface layout (Page 338)

Changing the settings (Page 306)

Configuring the display of tooltips and tooltip cascades (Page 358)

8.1.1.3 Overview of the script and text editor settings

Overview

The following table shows the available settings for script and text editors:

Group	Setting	Description
Font	Font type and size	Sets the font type and size for the text in text editors.
Font colors	Color settings	You can choose the colors for individual text elements from the respective drop-down lists in the text editors. Optional settings are available for the following text elements: • Text • Keywords • Comments • Operands • Scripts • Standard functions • Instructions/system functions • Constant strings • Numeric constants • Constant tags • Tags • Object models • Formal parameters
	Reset to default	Resets all font colors in editors to their factory settings.
Tabs	Tab width	Sets the width of tabs.
	Use tabs	Enables the use of tabs.
	Use spaces	Specifies use of space characters instead of tabs.
Indent	None	Text entries are not automatically indented.
	Block	The lines or the selected paragraph are automatically indented.
	Smart	The lines or the selected paragraph are automatically indented. All unnecessary spaces are also removed.
View	Show white spaces	Shows control characters within a text.
STL (Statement List)	Font type and size	Sets the font type and size for STL program code.
SCL (Structured Control Language)	Show line numbers	Shows the row numbers in SCL programs.

See also

Changing the settings (Page 306)

8.1.1.4 Overview of the print settings

Overview

The following table shows the available settings for printing:

Group	Setting	Description
General	Always print table data as pairs of values	Tables are printed as a list and not in tabular form. The corresponding values are listed for each column. Enable this option, for example, if you want to print a table that is too large for the print area.
Hardware configuration	Active graphic view	The graphics of network and device view are included in the printout.
	Active table	A table associated with an editor is included when printing with the editor.
PLC Programming	Zoom factor	Specifies the size in which blocks are to be printed out.
	With interface	The interfaces of blocks are included in the printout.
	With comments	Comments on blocks are included in the printout.
	With line numbers	The line numbers of the program code are printed for text-based programming languages.
Motion & Technology	Dialog display/graphic	The content of the editor is printed out as a graphic if the editor supports this.
	Table	The parameters of the technology objects are printed out in the form of a table.
HMI screens	Show tab order	In the printout you can specify the order in which the runtime objects can be selected with the TAB key.

See also

Changing the settings (Page 306)

8.1.1.5 Overview of the online and diagnostic settings

Overview

The following table shows the settings that you can make for the online and diagnostic functions:

Group	Setting	Description
Preset connection path for online access	Type of the PG/PC interface	Specifies the type of the PG/PC interface that is used as the presetting in the dialogs for online access, e.g., in the "Go online" dialog.
	PG/PC interface	Specifies a particular PG/PC interface that is used as the presetting in the dialogs for online access, e.g., in the "Go online" dialog.
	Use preset connection path for online connection	Activates or deactivates the presettings for the PG/PC interface. If the check box is selected, the connection path specified in the settings is used as the presetting in the dialogs for online access.

Group	Setting	Description
Alarm display	Multiline	Displays the alarms in the Inspector window using multiple lines.
	Autoscroll	When a new alarm occurs, the display scrolls automatically to this alarm.
	Archive size	Maximum number of alarms that can be displayed in the Inspector window. If the set number of alarms is exceeded, older alarms are automatically deleted.

8.1.1.6 Changing the settings

Procedure

To change the settings, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation to change the settings described in the previous sections. Or click on one of the other entries in area navigation to make settings for your installed products.
3. Change the settings.

Result

The change will be adopted immediately, there is no need to save it explicitly.

See also

Overview of the program settings (Page 302)

Overview of the script and text editor settings (Page 304)

Overview of the print settings (Page 305)

8.1.2 Layout of the user interface

8.1.2.1 Views

Views

Three different views are available for your automation project:

- The portal view is a task-oriented view of the project tasks.
- The project view is a view of the components of the project, as well as the relevant work areas and editors.
- The library view (Page 311) shows the elements of the project library and the open global libraries.

You can change over between the two views using a link.

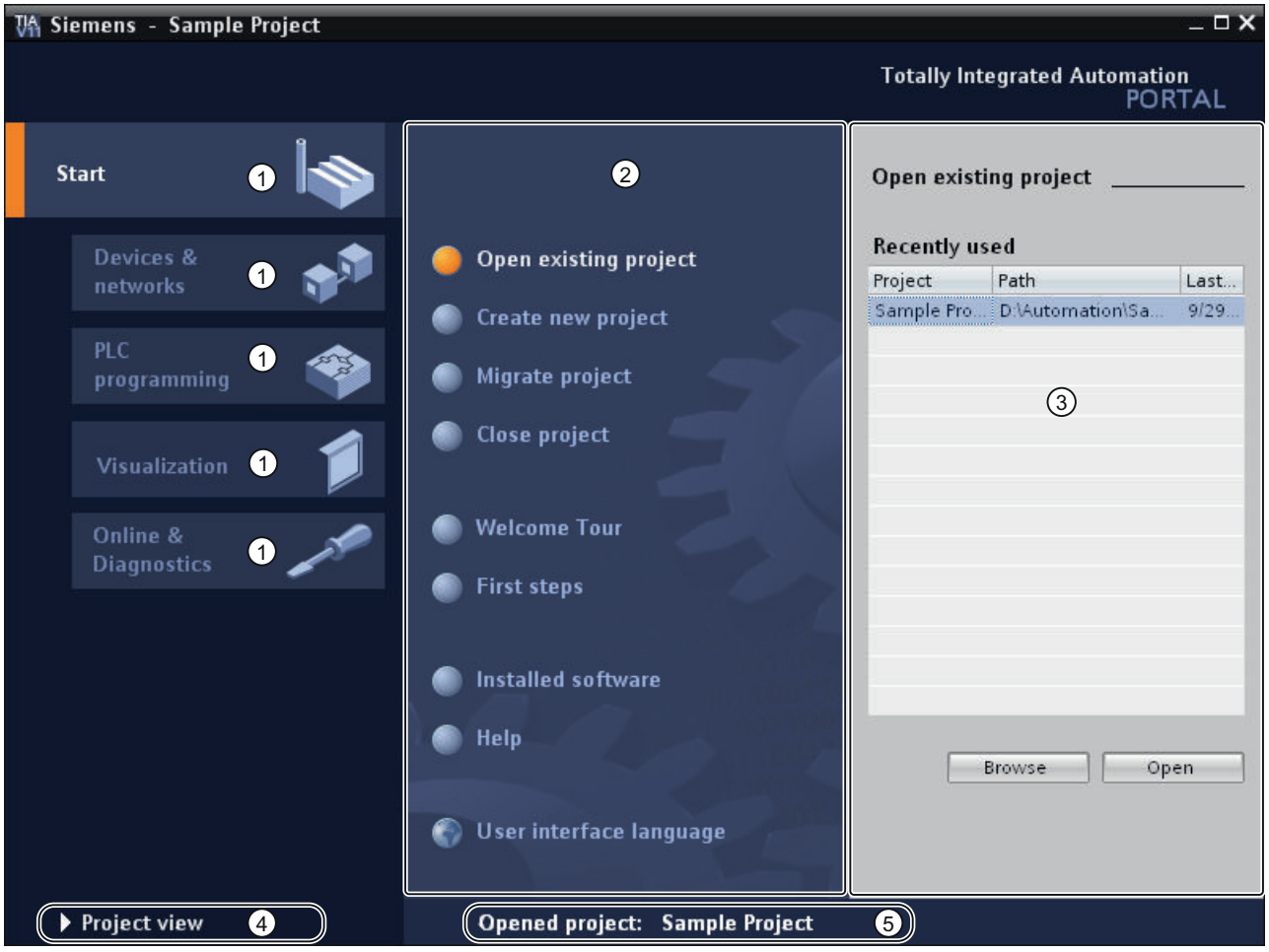
8.1.2.2 Portal view

Purpose of the portal view

The portal view provides you with a task-oriented view of the tools. Here, you can quickly decide what you want to do and call up the tool for the task in hand. If necessary, the view changes automatically to the project view (Page 309) for the selected task.

Layout of the portal view

The following figure shows an example of the components in the portal view:



① Portals for different tasks
② Actions for the selected portal
③ Selection panel for the selected action
④ Change to the project view
⑤ Display of the project that is currently open

Portals

The portals provide the basic functions for the individual task areas. The portals that are provided in the portal view depends on the products that have been installed.

Actions for the selected portal

Here, you will find the actions available to you in the portal you have selected. You can call up the help function in every portal on a context-sensitive basis.

Selection panel for the selected action

The selection panel is available in all portals. The content of the panel adapts to your current selection.

Change to the project view

You can use the "Project view" link to change to the project view.

Display of the project that is currently open

Here, you can obtain information about which project is currently open.

See also

Project tree (Page 312)

Basics of the work area (Page 316)

Inspector window (Page 324)

Basics on task cards (Page 326)

Details view (Page 330)

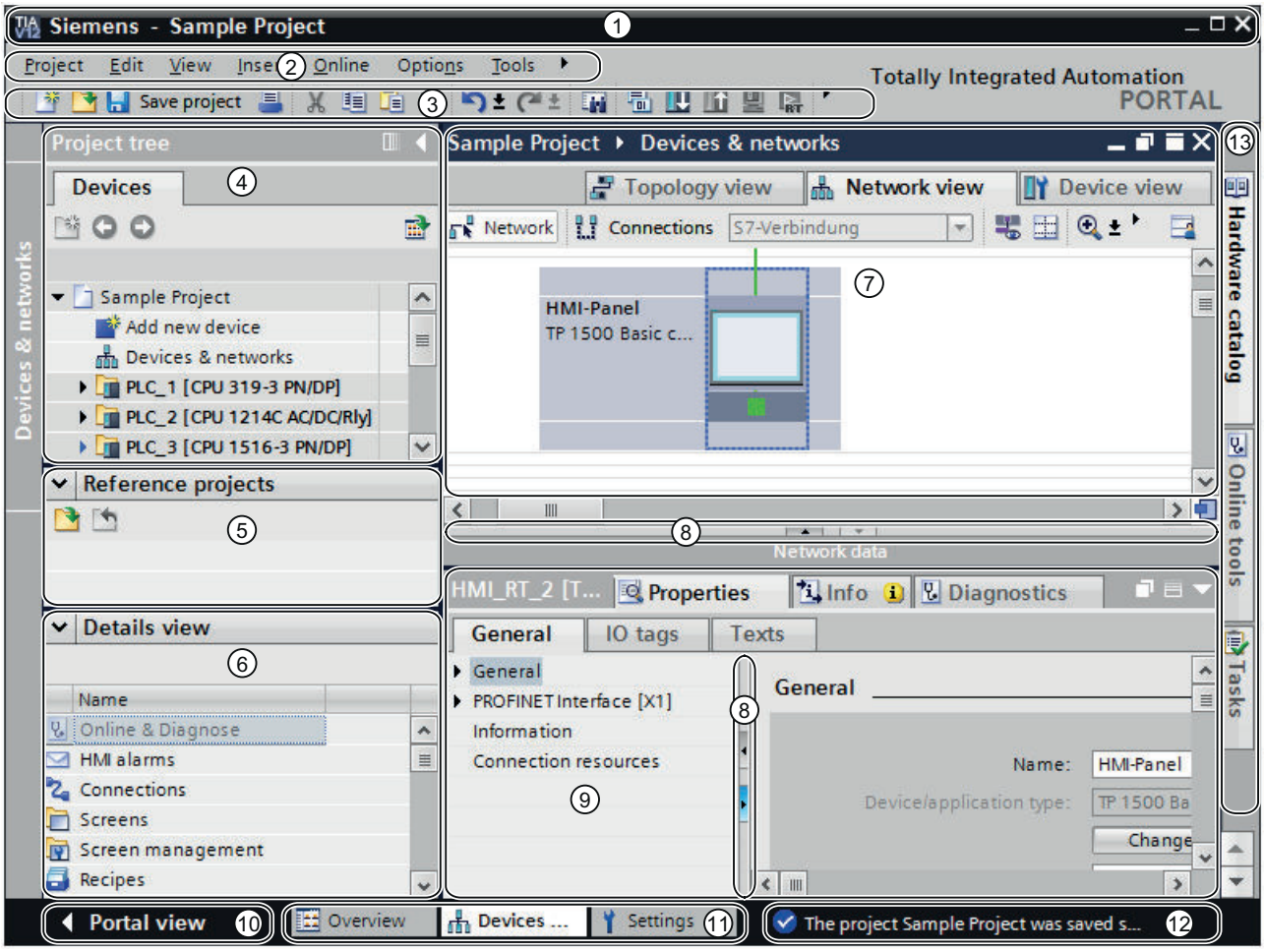
8.1.2.3 Project view

Purpose of the project view

The project view is a structured view of all components of the project.

Layout of the project view

The following figure shows an example of the components of the project view:



① Title bar
② Menu bar
③ Toolbar
④ Project tree (Page 312)
⑤ Reference projects (Page 328)
⑥ Details view (Page 330)
⑦ Work area (Page 326)
⑧ Dividers
⑨ Inspector window (Page 324)
⑩ Changing to the Portal view (Page 307)
⑪ Editor bar
⑫ Status bar with progress display
⑬ Task cards (Page 326)

Title bar

The name of the project is displayed in the title bar.

Menu bar

The menu bar contains all the commands that you require for your work.

Toolbar

The toolbar provides you with buttons for commands you will use frequently. This gives you faster access to these commands.

Dividers

Dividers separate individual components of the program interface. The arrows on the dividers allow you to display and hide the adjacent sections of the user interface.

Changing to the portal view

You can use the "Portal view" link to change to the portal view.

Editor bar

The Editor bar displays the open editors. If you have opened a lot of editors, they are shown grouped together. You can use the Editor bar to change quickly between the open elements.

Status bar with progress display

In the status bar, you will find the progress display for processes that are currently running in the background. This also includes a progress bar that shows the progress graphically. Hover the mouse pointer over the progress bar to display a tooltip providing additional information on the active background process. You can cancel the background processes by clicking the button next to the progress bar.

If no background processes are currently running, the status bar displays the last generated alarm.

See also

Basics of the work area (Page 316)

8.1.2.4 Library view

Function of the library view

The library view provides an overview of the elements in the project library and the open global libraries. You can switch to the library view using the "Libraries" task card.

See also: Overview of the library view (Page 475)

8.1.2.5 Project tree

Project tree

Function of the project tree

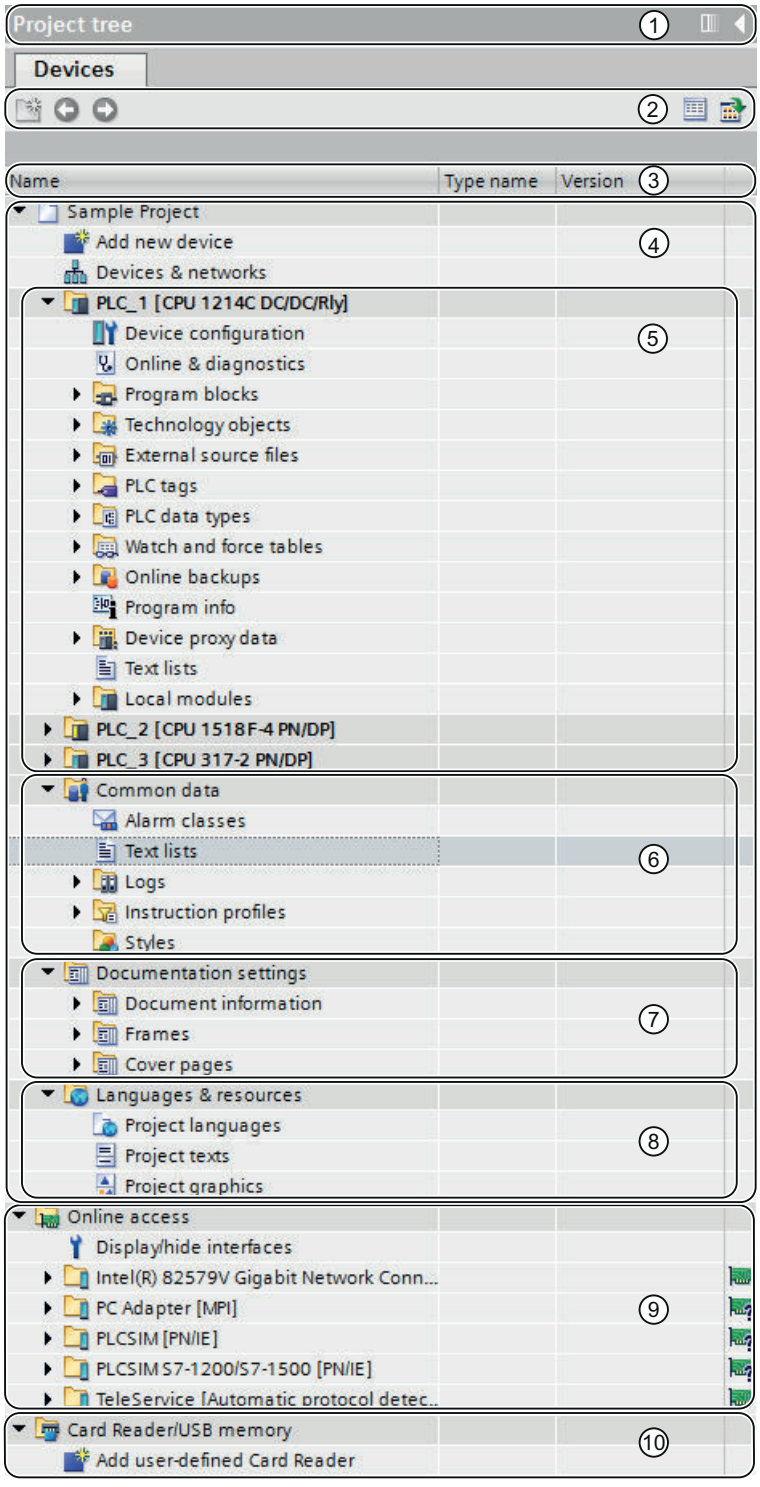
Using the project tree features gives you access to all components and project data. You can perform the following tasks in the project tree:

- Add new components
- Edit existing components
- Scan and modify the properties of existing components

You can select the objects of the project tree either with the mouse or via the keyboard by typing the first letter of the desired object. If more than one object begins with the same letter, the next lower object is selected. The project tree must be the focused user interface element in order for you to select an object with its initial letter.

Layout of project tree

The following figure shows an example of the project tree components:



① Title bar
② Toolbar
③ Table header
④ Project
⑤ Devices
⑥ Common data
⑦ Documentation settings
⑧ Languages & resources
⑨ Online access
⑩ Card Reader / USB memory

Title bar

The title bar of the project tree has a button for automatically and manually collapsing the project tree. Once it is collapsed manually, the button is "Reduced" to the left-hand margin. It changes from an arrow pointing left to one that is pointing right, and can now be used to reopen the project tree. You can use the "Collapse automatically" button collapse to project tree automatically when you do not need it.

See also: Maximizing and minimizing the work area (Page 318)

Toolbar

You can do the following tasks in the toolbar of the project tree:

- Create a new user folder; for example, in order to group blocks in the "Program blocks" folder.
- Navigate forward to the source of a link and back to the link itself.
  There are two buttons for links in the project tree. You can use these to navigate from the link to the source and back.
- Show an overview of the selected object in the work area.
  When the overview is displayed, the lower-level objects and actions of the elements in the project tree are hidden.

Table header

The "Name" column is shown by default. You can also show the columns "Type name" and "Version". If you are showing the additional columns, you see the name of the respective type as well as the version used for instances of types from the library.

Project

You will find all the objects and actions related to the project in the "Project" folder, e.g.:

- Devices
- Languages & resources
- Online access

Device

There is a separate folder for each device in the project, which has an internal project name. Objects and actions belonging to the device are arranged inside this folder.

Common data

This folder contains data that you can use across more than one device, such as common message classes, logs, scripts and text lists.

Documentation settings

In this folder, you can specify the layout for project documentation to be printed at a later point.

Languages & resources

You can determine the project languages and texts in this folder.

Online access

This folder contains all the interfaces of the programming device / PC, even if they are not used for communication with a module.

Card Reader / USB memory

This folder is used to manage all card readers an other USB storage media connected to the programming device / PC.

See also

Portal view (Page 307)

Project view (Page 309)

Basics of the work area (Page 316)

Inspector window (Page 324)

Basics on task cards (Page 326)

Details view (Page 330)

Showing and hiding columns (Page 316)

Showing and hiding columns

You can show additional columns in the project tree, if necessary. These additional columns show the name of the type associated with an instance as well as its version number.

Procedure

To show or hide additional table columns for the associated type and its version number, follow these steps:

1. Right-click on the table header of the project tree.
2. Select the "Show/Hide" command in the shortcut menu, and select the columns you want to display.
   The selected columns are displayed or hidden.

See also

Project tree (Page 312)

8.1.2.6 Work area

Basics of the work area

Function of the work area

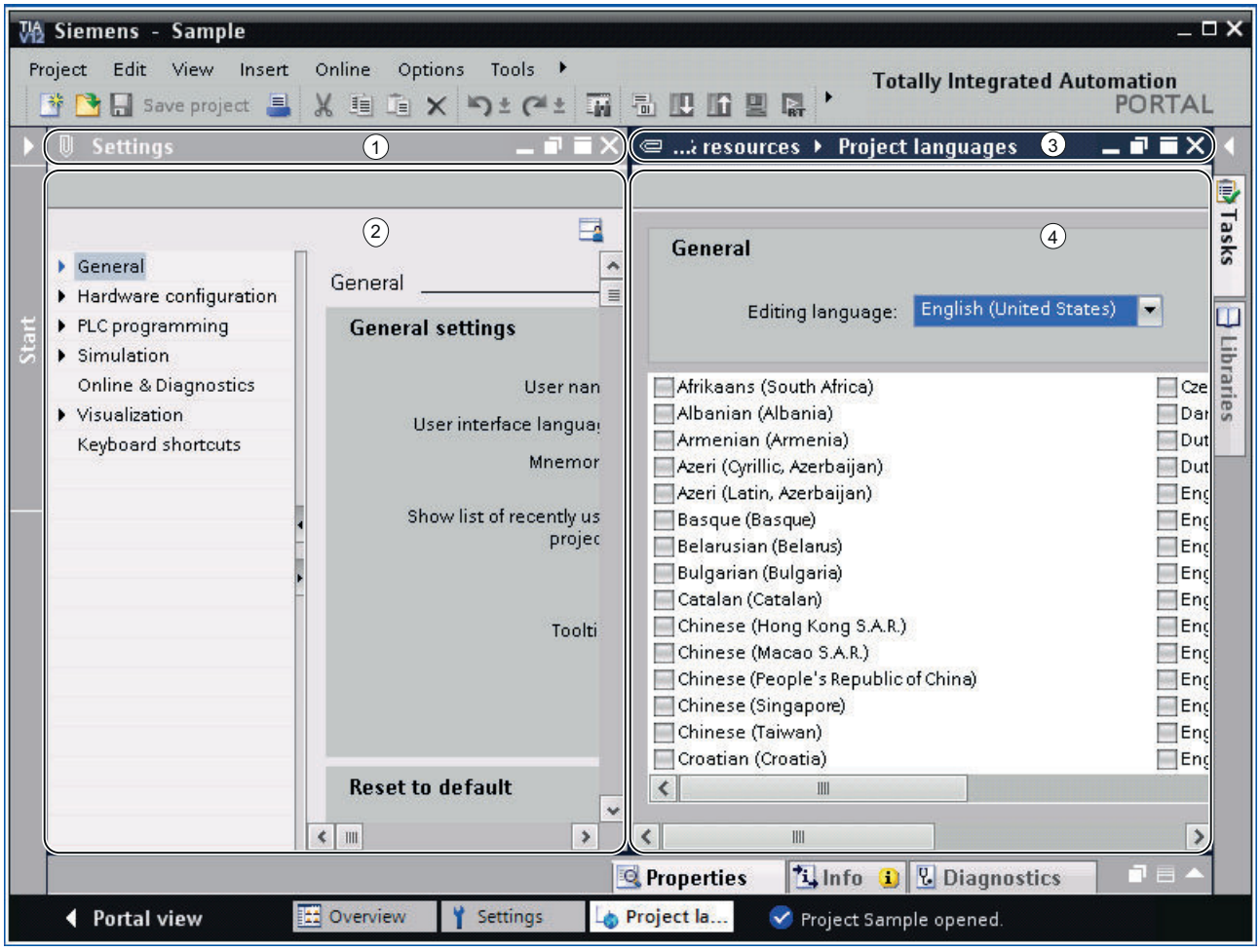
The objects that you can open for editing purposes are displayed in the work area. These objects include, for example:

- Editors and views
- Tables

You can open several objects. However, normally it is only possible to see one of these at a time in the work area. All other objects are displayed as tabs in the Editor bar. If, you would like to view two objects at the same time when performing certain tasks, you can tile the work area either horizontally or vertically or undock elements of the work area. If no objects are open, the work area will be empty.

Layout of the work area

The following figure shows an example of a vertically split work area:



① Title bar of left-hand editor
② Work area of left-hand editor
③ Title bar of right-hand editor
④ Work area of right-hand editor

See also

Maximizing and minimizing the work area (Page 318)

Splitting the work area (Page 319)

Floating the work area elements (Page 320)

Using grouped elements of the work area (Page 321)

Minimizing and maximizing elements of the work area (Page 322)

Switching between the elements in the work area (Page 323)

Saving a layout of editors and tables (Page 337)

Save user interface layout (Page 335)

Maximizing and minimizing the work area

You have the option to adapt the work area to make it as large as possible. You can use the following function for this:

- Maximizing the work area
  You can close the task cards, project tree and inspector window with a single click. This increases the size of the work area. You can minimize the work area again at any time in order to return to the previous view.
- Collapsing task cards, project tree, and Inspector window automatically
  You can use the "Collapse automatically" option for the task cards, project tree, and Inspector window. This function causes these items to collapse automatically when you don't need them.

Maximizing and minimizing the work area

To maximize the work area, follow these steps:

1. Open an element such as an editor or a table.
   The element appears in the work area.
2. Click the "Maximize" button in the title bar of the element.
   The task cards, project tree and inspector window collapse, and the work area is shown with its maximum dimensions.

To minimize the work area again, follow these steps:

1. Click the "Embed" button in the title bar of the displayed element.
   This restores the view that existed before the work area was maximized. That is, if the task cards, project tree, or Inspector window were expanded before, they will be expanded again.

Collapsing task cards, project tree, and Inspector window automatically

To collapse the task cards automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the task cards.
   The task cards collapse when you click anywhere outside the task cards.
2. To use the task cards, click the collapsed task cards.
3. The task cards expand and are available for use. The "Collapse automatically" option remains enabled.

To collapse the project tree automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the project tree.
   The project tree collapses when you click anywhere outside the project tree.
2. To use the project tree, click the collapsed project tree.
   The project tree expands and is available for use. The "Collapse automatically" option remains enabled.

To collapse the Inspector window automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the Inspector window.
   The Inspector window collapses when you click anywhere outside the Inspector window.
2. To use the Inspector window, click the collapsed Inspector window.
   The Inspector window expands and is available for use. The "Collapse automatically" option remains enabled.

To disable the automatic collapse option, follow these steps:

1. Click "Expand permanently" again in the relevant window.
   The Collapse automatically" option is disabled, and the window remains expanded.

See also

Basics of the work area (Page 316)

Splitting the work area (Page 319)

Floating the work area elements (Page 320)

Using grouped elements of the work area (Page 321)

Minimizing and maximizing elements of the work area (Page 322)

Switching between the elements in the work area (Page 323)

Saving a layout of editors and tables (Page 337)

Splitting the work area

You can split the work area vertically or horizontally.

Procedure

To split the work area vertically or horizontally, follow these steps:

1. In the "Window" menu, select the "Split editor space vertically" or "Split editor space horizontally" command.
   The element you have clicked and the next element in the Editor bar will be displayed either next to one another or one above the other.

Note

If no elements are open in the work area, the "Split editor space vertically" and "Split editor space horizontally" functions will not be available.

See also

Basics of the work area (Page 316)

Maximizing and minimizing the work area (Page 318)

Floating the work area elements (Page 320)

Using grouped elements of the work area (Page 321)

Minimizing and maximizing elements of the work area (Page 322)

Switching between the elements in the work area (Page 323)

Saving a layout of editors and tables (Page 337)

Floating the work area elements

You can float work area elements in their own separate window:

- Editors
- Tables
- Setting windows
- Task cards
- Inspector window

You can embed floating elements again in the work area at any time.

Note

Properties of elements in a floating window

The properties of elements that you have selected in a floating window are only displayed in the Inspector window if the Inspector window is floating as well.

Floating the work area elements

To float work area elements, follow these steps:

1. Click the "Float" button in the title bar of the element.
   The element will be released from the work area and displayed in its own window. You can now place the window wherever you wish. If you have minimized the window, you can restore it via the editor bar.

Embedding elements in the work area

To embed elements in the work area again, follow these steps:

1. Click the "Embed" button in the title bar of the element.
   The element will appear in the work area again.

See also

Basics of the work area (Page 316)

Maximizing and minimizing the work area (Page 318)

Splitting the work area (Page 319)

Using grouped elements of the work area (Page 321)

Minimizing and maximizing elements of the work area (Page 322)

Switching between the elements in the work area (Page 323)

Saving a layout of editors and tables (Page 337)

Using grouped elements of the work area

If you open more than five elements of the same type, e.g., editors or tables, they are grouped in the editor bar. You can use these groups as follows:

- Displaying individual elements of a group
- Displaying all elements of a group in separate windows
- Embedding all displayed elements of a group in the work area
- Minimizing all displayed elements
- Closing all elements of a group

Displaying individual elements of a group

To display individual elements of a group, follow these steps:

1. In the editor bar, click the group containing the element you want to display.
   All list of all available elements of the group is displayed.
2. Click the element that you want to display.

Displaying all elements of a group in separate windows

To display all elements of a group in separate windows, follow these steps:

1. In the editor bar, right-click the group whose elements you want to display.
2. Select "Restore group" in the shortcut menu.
   All elements of the group are displayed in separate, overlapping windows. Move the windows in order to see the individual element, or choose an element via the group in the editor bar.

Embedding all displayed elements of a group in the work area

To embed all elements of a group displayed in separate windows in the work area again, follow these steps:

1. In the editor bar, right-click the group whose elements you want to embed.
2. Select "Embed group" in the shortcut menu.
   All elements of the group are embedded in the work area again.

Minimizing all displayed elements

To minimize all elements of a group, follow these steps:

1. In the editor bar, right-click the group whose elements you want to minimize.
2. Select "Minimize group" in the shortcut menu.
   All elements of the group are minimized. However, the minimized elements remain open and can be quickly maximized again via the group in the editor bar.

Closing all elements of a group

To close all elements of a group, follow these steps:

1. In the editor bar, right-click the group whose elements you want to close.
2. Select "Close group" in the shortcut menu.
   All elements of the group are closed. The group is removed.

See also

Basics of the work area (Page 316)

Maximizing and minimizing the work area (Page 318)

Splitting the work area (Page 319)

Floating the work area elements (Page 320)

Minimizing and maximizing elements of the work area (Page 322)

Switching between the elements in the work area (Page 323)

Saving a layout of editors and tables (Page 337)

Minimizing and maximizing elements of the work area

You can minimize the elements that are open in the work area, such as editors or tables, as needed. However, an element remains open even if it has been minimized, and can quickly be maximized again using the editor bar.

Minimizing elements in the work area

To minimize elements in the work area, follow these steps:

1. Click the "Minimize" button in the title bar of the element.
   The element is minimized, but can still be accessed via the editor bar.

To minimize all elements at the same time, follow these steps:

1. In the "Window" menu, select the "Minimize all" command.

Maximizing elements in the work area

To maximize elements in the work area again, follow these steps:

1. Click the required element in the editor bar.
   The element is maximized and appears in the work area.

See also

Basics of the work area (Page 316)

Maximizing and minimizing the work area (Page 318)

Splitting the work area (Page 319)

Floating the work area elements (Page 320)

Using grouped elements of the work area (Page 321)

Switching between the elements in the work area (Page 323)

Saving a layout of editors and tables (Page 337)

Switching between the elements in the work area

You can switch between the elements in the work area at any time.

Switching between the elements in the work area

To switch to the previous or next editor, follow these steps:

1. In the "Window" menu, select the "Next editor" or "Previous editor" command.
   The next or previous editor will be displayed.

See also

Basics of the work area (Page 316)

Maximizing and minimizing the work area (Page 318)

Splitting the work area (Page 319)

Floating the work area elements (Page 320)

Using grouped elements of the work area (Page 321)

Minimizing and maximizing elements of the work area (Page 322)

Saving a layout of editors and tables (Page 337)

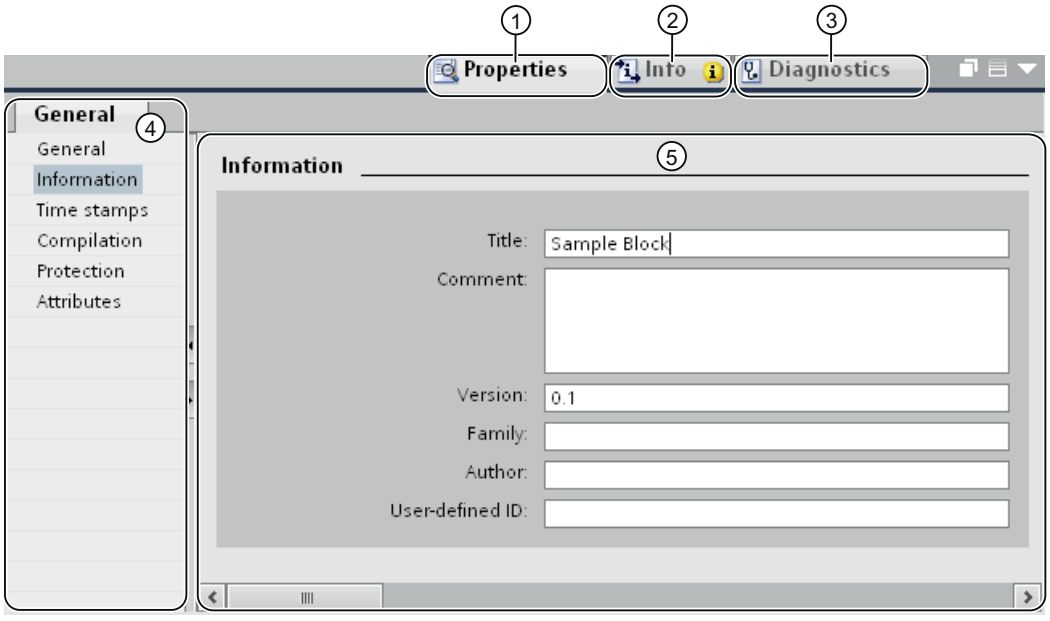
8.1.2.7 Inspector window

Function of the Inspector window

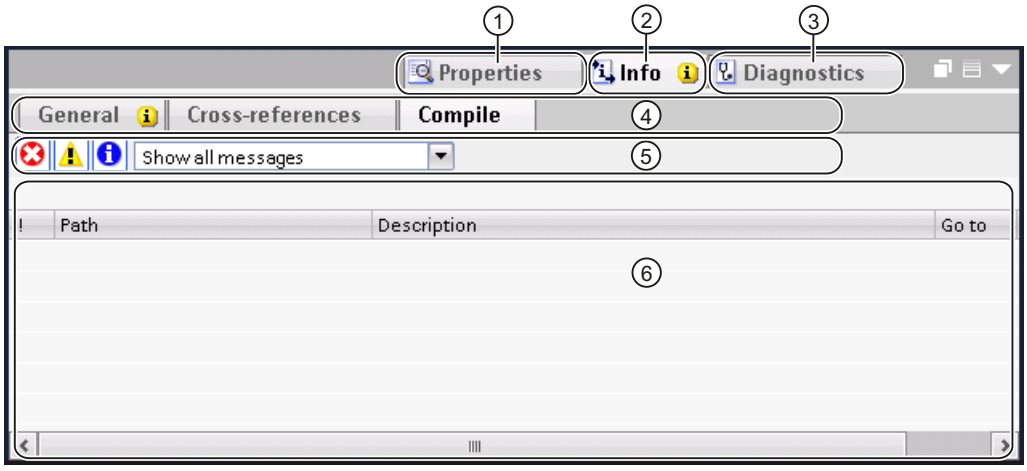
Additional information on an object selected or on actions executed are displayed in the inspector window.

Layout of the Inspector window

The following figures show the components of the Inspector window:



① "Properties" tab
② "Info" tab
③ "Diagnostics" tab
④ Area navigation within the "Properties" tab
⑤ Content of the "Properties" tab



① "Properties" tab
② "Info" tab
③ "Diagnostics" tab
④ Navigation through additional tabs within a tab (only available in the "Info" and "Diagnostics" tabs)
⑤ Toolbar (only available in the secondary "General" and "Compile" tabs of the "Info" tab)
⑥ Content of the "Compile" tab in the "Info" tab

"Properties" tab

This tab displays the properties of the object selected. You can change editable properties here.

"Info" tab

This tab displays additional information on the object selected, as well as alarms on the actions executed (such as compiling).

"Diagnostics" tab

This tab provides information on system diagnostics events, configured alarm events, and connection diagnostics.

Navigation within the tabs

You can use area navigation and the lower-level tabs to display the information you require within the tabs.

Toolbar

You can use the toolbar in the "General" and "Compile" tabs within the "Info" tab to specify which types of alarms are to be displayed. You can enable or disable the display for the following alarm types:

- Errors
- Warnings
- Information

See also

Project tree (Page 312)

Basics of the work area (Page 316)

Portal view (Page 307)

Project view (Page 309)

Basics on task cards (Page 326)

Details view (Page 330)

8.1.2.8 Task cards

Basics on task cards

Function of task cards

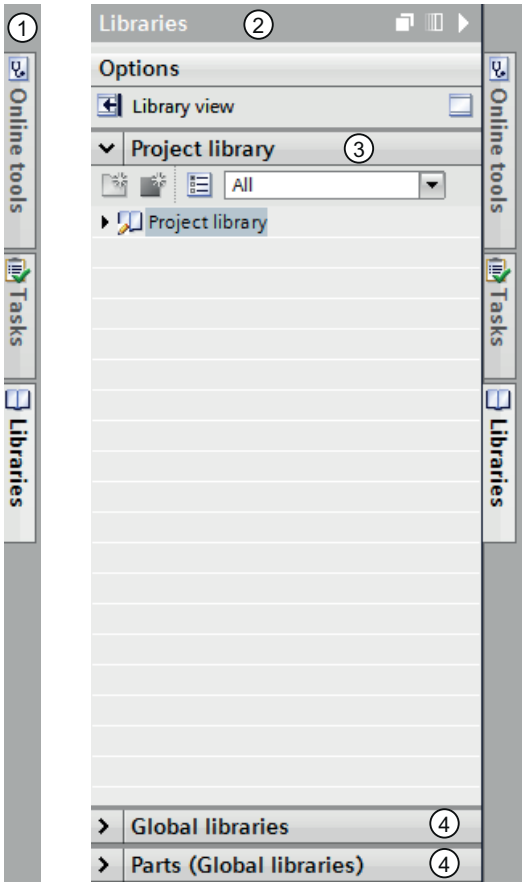
Depending on the edited or selected object, task cards that allow you perform additional actions are available. These actions include:

- Selecting objects from a library or from the hardware catalog
- Searching for and replacing objects in the project
- Dragging predefined objects to the work area

The task cards available can be found in a bar on the right-hand side of the screen. You can collapse and reopen them at any time. Which task cards are available depends on the products installed. More complex task cards are divided into panes that you can also collapse and reopen.

Layout of task cards

The following figure shows an example of the bar with the task cards:



① Task cards closed
② Task card open
③ Opened palette of a task card
④ Closed palette of a task card

See also

Changing the pane mode (Page 328)

Project tree (Page 312)

Basics of the work area (Page 316)

Inspector window (Page 324)

Portal view (Page 307)

Project view (Page 309)

Details view (Page 330)

Changing the pane mode

You can choose between two pane modes:

- Single pane mode:
  Only one pane is open at any given time. If you open another pane, the previously opened pane is closed automatically.
- Multi-pane mode:
  You can open several panes at the same time.

Procedure

To change the pane mode, follow these steps:

1. Click the "Change pane mode" button above the panes inside a task card.

See also

Basics on task cards (Page 326)

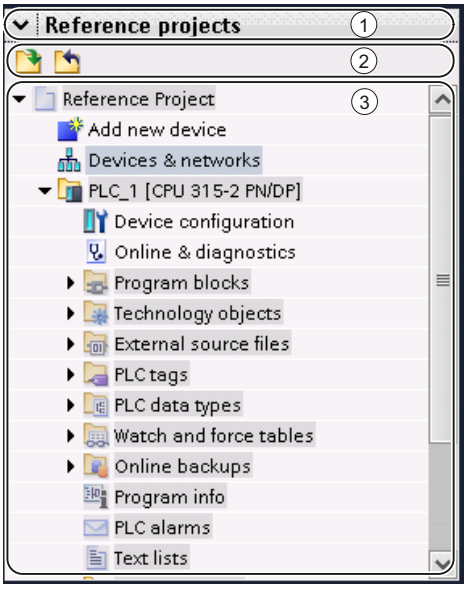
8.1.2.9 Reference projects

Function of reference projects

In the "Reference projects" palette, you can open other projects in addition to the current project. These reference projects are write-protected and cannot be edited. However, you can drag the objects of a reference project into your current project and further edit them there. You can also compare the objects of a reference project to the objects of your current project.

Layout of the "Reference projects" palette

The following figure shows the layout of the "Reference projects" palette:



① Title bar
② Toolbar
③ Opened reference projects

Title bar

The arrow for closing the palette is located in the title bar of the "Reference projects" palette. Once it is closed, the direction in which the arrow is pointing changes from downwards to right. It can now be used to reopen the palette.

Toolbar

The toolbar contains buttons for opening and closing reference projects.

Opened reference projects

Opened reference projects are displayed as read-only with their objects and their hierarchical structure.

See also

Basics of reference projects (Page 390)

Opening and closing a reference project (Page 390)

8.1.2.10 Details view

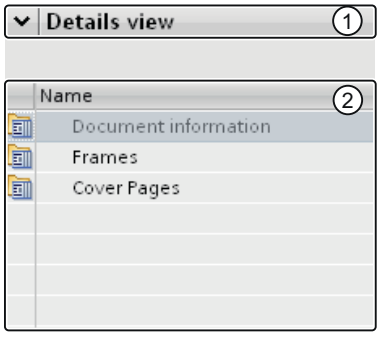
Purpose of the details view

The detail view shows certain content of the selected object is in the overview window or in the project tree. This might include text lists or tags.

The content of folders is not shown, however. To display the content of folders, use the project tree or the Inspector window.

Layout of the details view

The following figure shows an example of the details view:



① Title bar
② Content of the selected object

Title bar

The arrow for closing the details view is located in the title bar of the details view. After it has closed, the direction in which the arrow is pointing changes from left to right. It can now be used to reopen the details view.

Objects

The displayed content varies depending on the selected object. You can move the content of objects from the details view to the required location using drag-and-drop.

See also

Project tree (Page 312)

Basics of the work area (Page 316)

Inspector window (Page 324)

Basics on task cards (Page 326)

Portal view (Page 307)

Project view (Page 309)

8.1.2.11 Overview window

Overview window

Functions of the Overview window

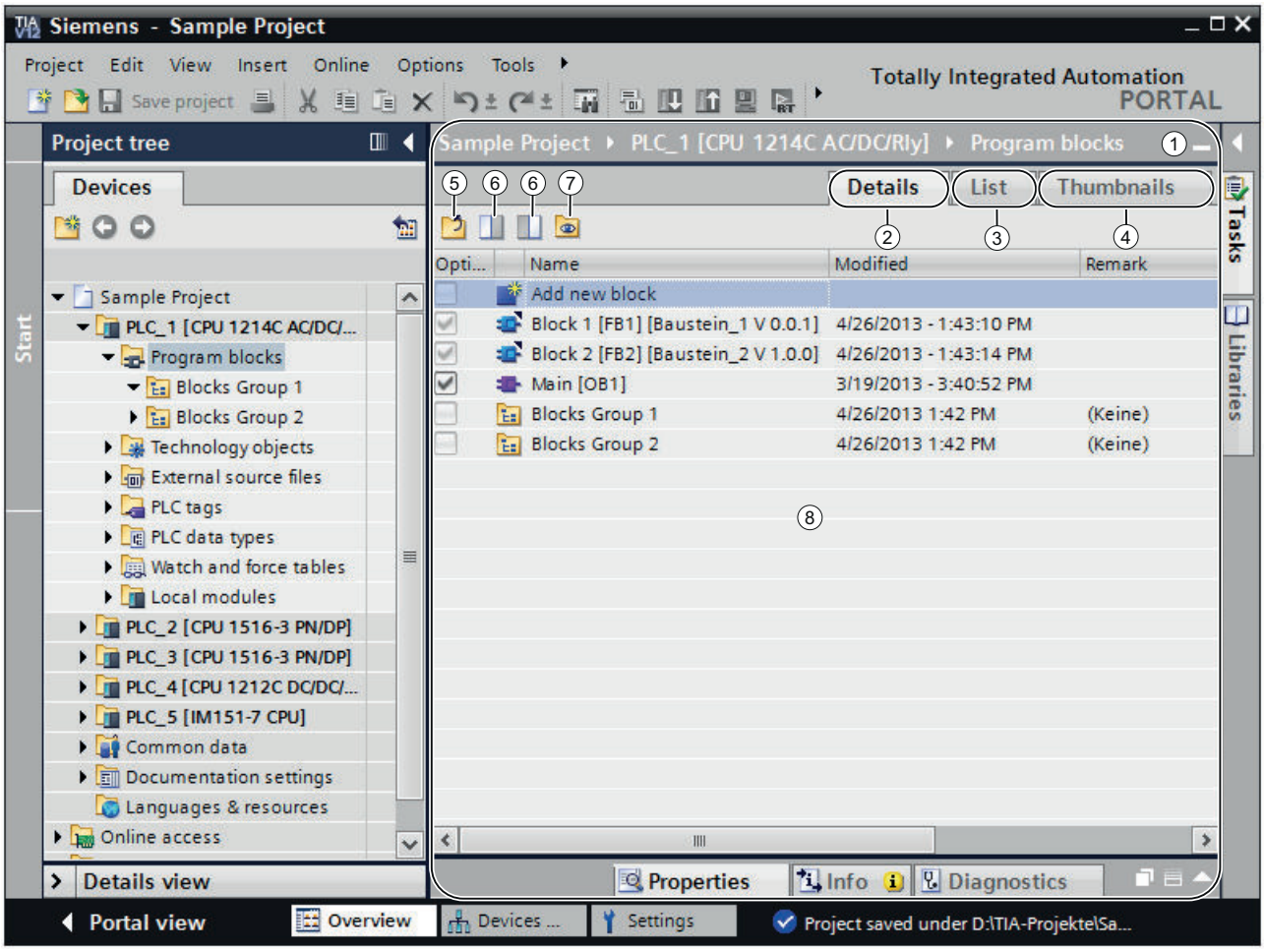
The Overview window supplements the project tree. The Overview window shows the contents of the folder currently selected in the project tree.

In addition, you can perform the following actions in the Overview window:

- Open objects
- Display and edit the properties of objects in the Inspector window
- Rename objects
- Call object-specific actions from the shortcut menu
- Compare objects side by side
- Perform various object operations, such as inserting objects from the library via drag-and-drop and moving, copying, pasting, and deleting objects

Layout of the Overview window

The following figure shows the components of the Overview window:



① Overview window
② Switch to the Details view
③ Switch to the List view
④ Switch to the Icon view
⑤ Move to higher level
⑥ Split the overview window in two. Either the right or left half of the overview window is synchronized. Clicking again cancels the split.
⑦ All elements within a selected folder are displayed even if these are located in lower-level groups. This option is only available in details view.
⑧ Contents of the object selected in the project tree.

Display forms of the Overview window

The content of the Overview window can be displayed as follows:

- Details view
  The objects are displayed in a list with additional information, such as the date of the last change.
- List view
  The objects are displayed in a simple list.
- Icon view
  The objects are displayed as icons according to category.

See also

Comparing objects in the overview window (Page 333)

Sorting the details view of the overview window (Page 334)

Overview of the library view (Page 475)

Comparing objects in the overview window

You can display the contents of two folders or objects side by side in the Overview window. The Overview window is split in half and you can display different contents on the left and right sides.

In addition, you can use a drag-and-drop operation to move objects between the split windows. Thus, for example, you can move contents from one window to the other.

Procedure

To split the Overview window in half or cancel the split, follow these steps:

1. In the toolbar, click on the "Synchronize left side" or "Synchronize right side" icon to split the overview window. Either the left or the right side of the overview window synchronized with the contents of the selected object in the project tree.
2. To cancel the split, click again on the previously selected icon.

See also

Overview window (Page 331)

Sorting the details view of the overview window

You have several options for adapting the display in the details view of the overview window:

- Adding additional columns
  Some columns are hidden by default to increase clarity. You can display hidden columns if needed. The columns available depend on the selected object.
- Display of folder contents in a flat hierarchy
  Folder contents can be displayed in a flat hierarchy. All the content is displayed at once even if it is located in different groups.
- Sorting the table columns
  You can sort individual columns of the table in ascending or descending order.

Showing or hiding columns

To show or hide additional table columns, follow these steps:

1. Right-click the title bar of the table.
2. Select the "Show/Hide" command in the shortcut menu, and select the columns you want to display.

Displaying folder contents in flat hierarchy

To display the content of a folder in a flat hierarchy, follow these steps:

1. Select the required folder in the project tree or in the library navigation of the library view.
2. Click the "Show subordinate elements" icon in the toolbar.
   All elements are displayed at once in the table even if they are located in subfolders.

Sorting a table in ascending or descending order

To sort the table by a column in ascending or descending order, follow the steps below:

1. Click the table header of a column if you want to sort the column in ascending order.
2. Click again on the same column of the table header to sort the column in descending order.
3. Click a third time on the table header of the same column to cancel the sorting.

See also

Overview window (Page 331)

8.1.2.12 User interface layout

Save user interface layout

Options for backing up the user interface layout

When you make a change to the user interface, this is retained even after a restart of the TIA Portal. A change to the user interface layout includes, for example, moving a window or adjusting the size of an editor.

In addition to the automatic saving of the user interface layout, you have the option of manually backing up specific layouts:

- Save the window layout
  You can save the layouts of the windows and editors of the TIA Portal manually and restore these at a later time. It is possible to call five window layouts using a key combination. Use this function, for example, if you are working with a notebook to which you connect an external monitor when necessary. You can create a window layout for mobile use on the notebook display and another layout for when you work at the office with an external monitor.
- Save the layout within editors
  With some editors, you can adjust the display. You can, for example, adjust the width of tables or show or hide individual table columns.

See also

Save window layout (Page 335)

Load window layout (Page 336)

Managing window layouts (Page 337)

Saving a layout of editors and tables (Page 337)

Resetting the user interface layout (Page 338)

Basics of the work area (Page 316)

Save window layout

You can save the current window layout in order to call it again in the same form at a later time.

Procedure

To save a window layout, follow these steps:

1. Arrange all windows in the way in which you want to save them.
2. In the "Window" menu, select the "Save window layout as" command.
   The "Save window layout" dialog box appears.
3. Enter a name for the window layout in the "Name" field.

4. Enter a description of the window layout in the "Description" field in order to be able to identify the window layout more easily later.
5. Click "Save".

Result

The new window layout is saved in the last position after the existing saved window layouts. The first five window layouts can be called using a key combination.

See also

Save user interface layout (Page 335)

Load window layout

If you have already saved a window layout, you can load this, allowing you to quickly adjust your work environment to the respective conditions. You can load the first five window layouts using quick access via the "Window" menu or via a key combination.

If you load a window layout and then make changes to the arrangement of the window, you can restore the originally saved window layout.

Using quick access to load window layouts 1 to 5

To load one of the first five saved window layouts, follow these steps:

1. In the "Window" menu, select a window layout or select the key combination <Alt+Shift+[1 ... 5]>.

Loading additional window layouts

To load a window layout that is not among the first five window layouts, follow these steps:

1. In the "Window" menu, select the "Additional window layouts" command.
   The "Manage window layouts" dialog box appears.
2. Select the desired window layout.
3. Click "OK".

Restore window layout

To go back to a saved window layout, follow these steps:

1. In the "Window" menu, select the "Restore window layout" command or select the key combination <Alt+Shift+0>.

See also

Save user interface layout (Page 335)

Managing window layouts

You can carry out the following actions with existing window layouts:

- Changing the order of window layouts
  The order of the window layouts is important, as the first five window layouts can be called directly via the "Window" menu and via a key combination.
- Select a window layout
  If a window layout is not one of the first five window layouts, you can call it using the "Manage window layouts" dialog box.
- Deleting window layouts

Procedure

To manage the existing window layouts, follow these steps:

1. In the "Window" menu, select the "Manage window layouts" command.
   The "Manage window layouts" dialog box appears.
2. Select the window layout which you want to modify.
3. Click the "Up" or "Down" symbol to move the window layout up or down.
4. Click the "Delete" symbol to delete the selected window layout.
5. Click "OK".
   The selected window layout is activated.

See also

Save user interface layout (Page 335)

Saving a layout of editors and tables

You have the option of adapting editors and tables to meet your requirements. For example, you can hide columns in tables that you don't need. You can then save your customized view.

Procedure

To save the layout of editors and tables in the work area, follow these steps:

1. Adapt the editor or table according to your requirements.
2. Click the "Remember Layout" button in the editor or table.

Result

The layout is saved. When you reopen the editor or table, this layout will be used.

See also

Basics of the work area (Page 316)

Maximizing and minimizing the work area (Page 318)

Splitting the work area (Page 319)

Floating the work area elements (Page 320)

Using grouped elements of the work area (Page 321)

Minimizing and maximizing elements of the work area (Page 322)

Switching between the elements in the work area (Page 323)

Save user interface layout (Page 335)

Resetting the user interface layout

Every change you make to the layout of the user interface is saved. The changes are thus available even after a restart of the TIA Portal. For example, if you change the height and width of a text editor or the division of a table, your changes are retained so that you don't have to re-customize elements every time.

In some cases, however, it may be helpful to restore the original layout settings; for example, if another user prefers a different arrangement of the user interface.

Procedure

To reset the user interface settings to the default, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation.
3. Click the "Reset to default" button under "Reset to default > Editor layout".

Result

The default settings for the user interface are restored.

See also

Overview of the program settings (Page 302)

Save user interface layout (Page 335)

8.1.3 Keyboard operation in the TIA Portal

8.1.3.1 Operation of the TIA Portal with the keyboard

You can navigate through the TIA Portal using the keyboard, for example if you do not have a mouse available at the given moment. Many functions are also accessible via keyboard shortcuts. You can find an overview of all keyboard shortcuts in the settings for the TIA Portal.

In the following sections, you will learn how to navigate in the TIA Portal using the keyboard, edit objects, and customize the TIA Portal to your needs.

See also

Displaying an overview of all keyboard shortcuts (Page 339)

8.1.3.2 Displaying an overview of all keyboard shortcuts

You can display an overview of all keyboard shortcuts.

Procedure

To display an overview of all available keyboard shortcuts, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Open the "Keyboard shortcuts" entry in the area navigation.
   You can see an overview of all keyboard shortcuts, which are valid for the currently installed products.

8.1.3.3 Basic functions of the TIA Portal

Below, you will learn how you can use the basic functions of the TIA Portal using only your keyboard.

Using the basic functions of the TIA Portal with the keyboard

The following table shows how you can access basic functions of the TIA Portal with keyboard shortcuts:

Function	Keyboard shortcut	Menu command
Change between the project view and the portal view	<Alt+F7>	
Open the Help system If you need help on the TIA Portal, press <F1>.	<F1>	Help > Show help
Cancel the current action	<Esc>	
Find	<Ctrl+F>	
Replace an object You can replace found objects when searching in the editor.	<Ctrl+H>	

Function	Keyboard shortcut	Menu command
Find next If you have started a search in the editor, you can jump to the next hit with <F3>.	<F3>	
Print object	<Ctrl+P>	Project > Print
Run default action of the object	<Enter>	
Scroll horizontally to the right	<Ctrl+Arrow right>	
Scroll horizontally to the left	<Ctrl+Arrow left>	

Operating menus

The following table shows how you can navigate through menus using the keyboard:

Function	Keyboard shortcut
Start keyboard shortcuts in the menu You can access the menu using the <Alt> key and then continue to navigate with the arrow keys to scroll through the menu. Confirm your selection of the menu command with <Enter>.	<Alt>
Go directly to a specific menu You can go directly to an individual menu command by holding down the <Alt> key. There is an underlined letter for each menu command. Press the <Alt> key along with the underlined letter.	<Alt+underlined letter in respective menu>
Open shortcut menu of an object With the shortcut menu key (on Microsoft Windows compatible keyboards), you can open the shortcut menu of the selected object. Alternatively, you can use <Shift+F10> if you are not using a Microsoft Windows compatible keyboard. You can use the arrow keys to scroll through the shortcut menu and select a menu command with <Enter>.	<Shortcut menu key> Alternative: <Shift+F10>

Operating expandable elements

The following table shows how you can operate expandable elements using the keyboard:

Function	Keyboard shortcut
Expand a folder in a tree With <Arrow right>, for example, you expand a folder in the project tree.	<Arrow right>
Close a folder in a tree With <Arrow left>, for example, you collapse a folder in the project tree.	<Arrow left>
Open a drop-down list You can open drop-down lists with <F4> and then navigate with the arrow keys to scroll through the drop-down list. Finally, press the <Enter> key to confirm your selection.	<F4>
Open autocompletion	<Ctrl+Space> <Ctrl+I>
Show object selection	<Ctrl+J>

8.1.3.4 Using project-related functions

Editing a project

Function	Keyboard shortcuts	Menu command
Open a project	<Ctrl+O>	Project > Open
Close a project	<Ctrl+W>	Project > Close
Save a project	<Ctrl+S>	Project > Save
Save a project under a different name	<Ctrl+Shift+S>	Project > Save as
Delete a project	<Ctrl+E>	Project > Delete project
Print project	<Ctrl+P>	Project > Print
Undo last action	<Ctrl+Z>	Edit > Undo
Redo last action	<Ctrl+Y>	Edit > Redo

Calling up the help function

Function	Keyboard shortcuts	Menu command
Calling up the help function	<F1> or <Shift+F1>	Help > Show help

8.1.3.5 Arranging windows

Below, you will learn how to open and close individual windows of the TIA Portal and work with saved window layouts using the keyboard.

Opening and closing windows

The following table shows how you can open and close windows with keyboard shortcuts:

Function	Keyboard shortcut	Menu command
Open/close project tree	<Ctrl+1>	View > Project tree
Opening/closing the detailed view	<Ctrl+4>	View > Details view
Opening/closing the overview	<Ctrl+2>	View > Overview
Opening/closing a task card	<Ctrl+3>	View > Task card
Open libraries	<Ctrl+Shift+L>	
Open hardware catalog If you are in the device or network view, the hardware catalog opens.	<Ctrl+Shift+C>	
Open/close inspector window	<Ctrl+5>	View > Inspector window
Open the "Properties" tab in the inspector window	<Ctrl+6>	
Open the "Info" tab in the Inspector window	<Ctrl+7>	
Open the "Diagnostics" tab in the Inspector window	<Ctrl+8>	
Display or hide reference projects	<Ctrl+9>	

Function	Keyboard shortcut	Menu command
Display the on-screen keyboard You can display a keyboard on the screen, for example, to operate with a touch screen.	<Ctrl+Shift+K>	
Close all editors	<Ctrl+Shift+F4>	Window > Close all

Using saved window layouts

You can save individual window arrangements and restore them at a later point in time. The following table shows how you to access saved window layouts with keyboard shortcuts:

Function	Keyboard shortcut	Menu command
Restore active window layout If you use a saved window layout and have made changes to the program interface in the meantime, you can restore the original state of the active window layout with <Alt+Shift+0>.	<Shift+Alt+0>	Window > Restore window layout
Load window layout You can use <Alt+Shift+[number of the window layout]> to activate the first of the five saved window layouts.	<Shift+Alt+[Number of the window layout]>	Window > Window layout 1 to 5

8.1.3.6 Navigating through the program interface

The TIA Portal is divided into various user interface areas, for example, individual windows, toolbars, and editors. If you want to work with the keyboard within an interface area, you first have to place the focus on it. Below, you will learn how to place the focus on individual interface areas using the keyboard. You will also learn how to move within an interface area in the TIA Portal using the keyboard.

Switching between interface areas and editors

The following table shows how to move between individual interface areas of the TIA Portal:

Function	Keyboard shortcut
Move clockwise between the interface areas You can use the <F6> key to move clockwise between the individual interface areas of the TIA Portal. The interface area currently in focus is highlighted with a blue title bar. If you are in the project tree, for example, and press the <F6> key, you jump to the currently open editor. If you press <F6> again, the task cards are in focus. If you press <Shift+F6>, on the other hand, you move counterclockwise between the work areas.	<F6>
Move counterclockwise between the interface areas With <Shift+F6> you move counter-clockwise between the interface areas of the TIA Portal.	<Shift+F6>
Go to the next open editor With <Ctrl+Alt+Arrow right> you go to the next open editor. You can see the open editors in the editor toolbar.	<Ctrl+Alt+Arrow right> Alternative: <Ctrl+F6>
Go to the previous open editor With <Ctrl+Alt+Arrow left>, you go to the last opened editor.	<Ctrl+Alt+Arrow left> Alternative: <Ctrl+Shift+F6>

Function	Keyboard shortcut
Go to the next higher section of the interface area With <Shift+Esc>, you move to the next higher section of the program interface. If, for example, you have selected a device in the project tree and you press <Shift+Esc>, the entire project navigation is put into focus.	<Shift+Esc> Alternative: <Alt+Arrow up>
Go to the next lower section of the interface area With <Enter>, you place the focus on the next lower section of the program interface. For example, if you have just opened the properties of a device in the Inspector window in order to assign parameters to the device, press <Enter> to go one level deeper in the program interface. You can then navigate to the desired parameter using the Tab key.	<Enter> Alternative: <Alt+Arrow down>

Navigation within interface areas and editors

The following table shows how you can navigate within an interface area using the keyboard:

Function	Keyboard shortcut
Jump to the next element within an interface area You can use the Tab key to jump from one element to the next within a work area. If, for example, you have opened the properties of a device and want to jump from one field to the next, press the Tab key. Any changes you have made to the current text box are applied in this case.	<Tab>
Jump to the previous element within an interface area With <Shift+Tab>, you can jump to the previous element in a work area, for example, a previous text box. Any changes you have made to the current text box are applied in this case.	<Shift+Tab>
Jump to the higher-level element within an interface area With <Shift+Home> you can jump to the higher-level element of a work area, e.g., to the higher-level folder in the project tree.	<Shift+Home>
Move to the next tab within an interface area If an interface area is divided into separate tabs, you can move between the tabs with the shortcut keys <Ctrl+Tab>. If, for example, you are in the "Properties" tab of the Inspector window and you want to jump to the "Info" tab, press the shortcut keys <Ctrl+Tab>.	<Ctrl+Tab>
Go to the previous tab With <Ctrl+Shift+Tab>, you move to the most recently open tab within an interface area.	<Ctrl+Shift+Tab>
Jumping to the toolbar of an editor You can use the <Alt+F10> key to jump to the toolbar of an editor. For example, if you have opened the print preview and want to switch to the next page of the printout in the toolbar, press <Alt+F10>. Then, use the arrow keys to navigate to the appropriate icon in the toolbar and confirm the selection with <Enter>.	<Alt+F10>
Use arrows on the divider to display or hide user interface components The table in the work area can be minimized and maximized. First, navigate to the work area and use the Tab key to place the focus on one of the little arrows on the separator line above the table. The arrows have the focus as soon as they are highlighted in blue. Then, press the space bar to minimize or maximize the table.	<Space>

8.1.3.7 Customizing editors

Below, you will learn how to arrange editors using the keyboard. You will also learn how to select the display size and the area within a graphical editor.

Arranging and customizing editors

The following table shows how to arrange open editors above and below each other or side-by-side, and how to close an open editor:

Function	Keyboard shortcuts	Menu command
Close active editor	<Ctrl+F4>	
Split editor space vertically If, for example, you have opened the overview window and the network view and want to display them side-by-side, press the <F12> key.	<F12>	Window > Split editor space vertically
Split editor space horizontally You can display two open editors in the work area above and below each other.	<Ctrl+F12>	Window > Split editor space horizontally
Remove window split If you have displayed two editors in the work area horizontally or vertically in split mode, you can remove the split with <Alt+Shift +F12>.	<Alt+Shift+F12>	Window > Unsplit editor space

Customizing the display in an editor

The following table shows how you how to zoom in and out in graphical editors and how to move the area selection in an editor:

Function	Keyboard shortcuts
Zoom in step-by-step in an editor With <Ctrl> and the <Plus> key on the numeric keypad of the keyboard, you can zoom in on the display of the editor.	<Ctrl+Plus> Alternative: <Ctrl+mousewheel up>
Zoom out step-by-step in an editor Use the <Ctrl> and the <Minus> key on the numeric keypad of the keyboard to zoom out of the display of the editor.	<Ctrl+Minus> Alternative: <Ctrl+mousewheel down>
Set view in editor to 100% Press <Ctrl+0> to reset the current view in a graphical editor to 100%.	<Ctrl+0>
Move the area selection of the editor If you hold down the spacebar, you can move the displayed section of an editor using the mouse.	<Space>

8.1.3.8 Editing objects

Selecting objects

The following table shows how to select individual objects, for example, devices in the project tree:

Function	Keyboard shortcut	Menu command
Select an object located at the left, right, above or below	<Arrow keys>	
Jump to the first object within the current interface area The first object in the interface area currently in focus is selected. In the project tree, this would be the project node at the top, for example.	<Home>	
Jump to the last object within the current interface area The last object in the interface area currently in focus is selected, for example, the last item in the project tree.	<End>	
Select all objects in an area All objects in the work area currently in focus are selected.	<Ctrl+A>	Edit > Select all
Select multiple objects If you want to select several objects that are not located directly next to each other, you first have to move the focus (gray outline of an object) to the next desired object using <Ctrl+Arrow keys>. The current selection is maintained. Then, press the space bar to select the new focused object as well. Repeat this process until all desired objects are selected.	<Ctrl+Arrow keys> + <Space>	

Editing objects

The following table provides an overview of all the keyboard shortcuts required for editing objects:

Function	Keyboard shortcut	Menu command
Insert new object A new object is inserted depending on your current context. If, for example, you are in the device view, the "Add Device" dialog opens for creating a new device.	<Ctrl+N>	
Open object	<Enter>	
Rename an object	<F2>	Edit > Rename
Copy an object	<Ctrl+C> Alternative: <Ctrl+Ins>	Edit > Copy
Cut an object	<Ctrl+X> Alternative: <Shift+Del>	Edit > Cut
Paste an object	<Ctrl+V> Alternative: <Shift+Ins>	Edit > Paste
Delete an object	<Del>	Edit > Delete

Function	Keyboard shortcut	Menu command
Compile an object	<Ctrl+B>	Edit > Compile
Open properties of an object Many objects in the TIA Portal have editable properties. Press the shortcut keys <Alt+Enter> to display the properties of an object.	<Alt+Enter>	-

8.1.3.9 Text editing

Below, you will learn how to operate text editing functions using only the keyboard.

Editing text

The following table shows the basic editing functions for text:

Function	Keyboard shortcuts
Switch to insert or overwrite mode	<Insert>
Exit edit mode	<Esc>
Delete	<Del>
Delete characters	<Backspace>
Confirm entry in a text box and leave the text box	<Return>
Line break in a multiline text box In a multiline text box, hold down the <Shift> button to create a line break.	<Shift+Return>
Reset input in a text box If you are in an text box and press <Esc>, you exit the box and the changes are discarded.	<Esc>

Navigating within a text area

The following table shows how to navigate in a text area with the keyboard:

Function	Keyboard shortcuts
Jump to start of line	<Home>
Jump to end of line	<End>
Jump to start of text	<Ctrl+Home>
Jump to end of text	<Ctrl+End>
Jump to the previous page	<PgUp>
Jump to the next page	<PgDn>
Confirm entry in a text box and leave the text box	<Return>
Line break in a multiline text box	<Shift+Return>
Reset input in a text box If you are in an text box and press <Esc>, you exit the box and the changes are discarded.	<Esc>

Selecting text

The following table shows how to select text with the keyboard:

Function	Keyboard shortcuts
Expand selection to the word at the left or right The text or current text selection is marked up to the end of word. If you are at the beginning or end of a word, the previous or next word is selected.	<Ctrl+Shift+Arrow left or Arrow right>
Expand selection to beginning of line	<Shift+Home>
Expand selection to end of line	<Shift+End>
Expand selection to beginning of text The text is selected up to the beginning or the end.	<Ctrl+Shift+Home>
Expand selection to end of text The text is selected up to the beginning or the end.	<Ctrl+Shift+End>

8.1.3.10 Editing tables

Below, you will learn how to navigate with the keyboard in tables, edit individual fields, and select parts of tables.

General keyboard operation in tables

The following table shows how you can edit tables using only the keyboard:

Function	Keyboard shortcuts
Place a cell in edit mode	<F2> or <Return>
Confirm entry and exit edit mode	<Return>
Cancel editing and discard changes	<Esc>
Open drop-down list in a cell Open the drop-down list with <F4>. Use the arrow keys to select the desired entry and then confirm the selection with <Return>.	<F4>
Close drop-down list in a cell and discard changes	<Esc>

Navigate in tables

The following table shows how you can navigate within a table using the keyboard:

Function	Keyboard shortcuts
Go to the next cell	<Arrow keys>
Go to the next editable cell on the right	<Tab>
Go to the next editable cell on the left	<Shift+Tab>
Move a screen upwards	<PgUp>
Move a screen downwards	<PgDn>
Go to the first cell in the row	<Home>
Go to the last cell in the row	<End>
Go to the first cell in the table	<Ctrl+Home>

Function	Keyboard shortcuts
Go to the last cell in the table	<Ctrl+End>
Go to the top cell in the column	<Ctrl+Arrow up>
Go to the bottom cell in the column	<Ctrl+Arrow down>

Selecting areas in tables

The following table shows how you can select areas within a table using the keyboard:

Function	Keyboard shortcuts
Select column	<Ctrl+Space>
Select line	<Shift+Space>
Select all cells	<Ctrl+A>
Expand selection by one cell	<Shift+arrow keys>
Extend selection up one page	<Shift+PgUp>
Extend selection down one page	<Shift+PgDn>
Expand selection up to the first row	<Ctrl+Shift+Arrow up>
Expand selection down to the last row	<Ctrl+Shift+Arrow down>
Expand selection to the first cell in the row	<Ctrl+Shift+Arrow left>
Expand selection to the last cell in the row	<Ctrl+Shift+Arrow right>

8.1.3.11 Using online functions

Controlling online functions with the keyboard

The following table provides an overview of the shortcut keys that you can use for the online functions of the TIA Portal:

Function	Keyboard shortcut	Menu command
Establish an online connection	<Ctrl+K>	Online > Go online
Go offline	<Ctrl+M>	Online > Go offline
Download project data to the device	<Ctrl+L>	Online > Download to device
Show accessible devices This opens a dialog showing all devices that are connected to the PG/PC interface of the PG/PC.	<Ctrl+U>	Online > Show accessible devices
Start CPU The CPU is set to "RUN" mode. The CPU must be online for this.	<Ctrl+Shift+E>	Online > Start CPU
Stop CPU The CPU is set to "STOP" mode. The CPU must be online for this.	<Ctrl+Shift+Q>	Online > Stop CPU
Start simulation The hardware and software of the project can be tested in a simulated online environment, without the modules actually being online.	<Ctrl+Shift+X>	Online > Simulation > Start

8.1.3.12 Using the on-screen keyboard

Introduction

When working with the TIA Portal, you also have the Microsoft on-screen keyboard available.

Displaying the on-screen keyboard

To display the on-screen keyboard, follow these steps:

1. In the "View" menu, select the "Screen keyboard" command.

Exiting the on-screen keyboard

To exit the on-screen keyboard, follow these steps:

1. In the "File" menu of the on-screen keyboard, select the "Exit" command.

8.1.4 Special features specific to the operating system

8.1.4.1 Influence of user rights

Restrictions when user rights are limited

The software provides several functions that require direct access to the hardware of the programming device / PC and therefore also to the installed operating system. To make full use of the range of functions, the software must cooperate closely with the operating system. To ensure problem-free interaction, you should therefore be logged on to the operating system with adequate user rights.

In particular, you may not be able to use functions requiring an online connection or those that change the settings of interface cards if you work with limited user rights.

Recognizing restricted functions

You can recognize functions requiring special rights as follows:

- A shield icon is displayed beside the function.

  The function can be used but is regulated by the user account control.
- A box is grayed out and cannot be accessed.
  You require administrator privileges to access the box. In some operating system environments, you can obtain administrator privileges by entering an administrator password.

Note

A box being grayed out does not necessarily mean a lack of rights. You should also check the additional information in the tooltip cascades to find out the conditions for editing the box.

8.1.4.2 Expanding user rights

Counteracting restrictions due to user rights

Certain functions may not be available if you are not logged on to the operating system with adequate rights. You can counteract these restrictions in the following ways:

- Enabling of extended rights using Windows user account control
- Logging on to the operating system with administrator privileges
- Using temporary administrator rights

Enabling extended rights using the Windows user account control

To be able to use a function indicated by the shield icon of the Windows user account control, follow these steps:

1. Click on the box or button with the shield icon.
   The security prompt of the Windows user account control opens.
2. Follow the instructions of the Windows user account control and, when prompted enter an administrator password, if possible.

The function can now be used once without restrictions.

Logging on to the operating system with administrator privileges

To be able to use a function that is disabled due to lack of user rights, follow these steps:

1. Close the software.
2. Log off from the operating system.
3. Log on to the operating system with administrator privileges.
4. Restart the software.

Using temporary administrator rights

To obtain administrator privileges temporarily, follow these steps:

1. Click the "Change settings" button. You will find this button in dialogs that allow the temporary assignment of administrator privileges.
   An operating system dialog box for entering an administrator password opens.
2. Enter an administrator password.

The settings can be temporarily changed. When you call the dialog again, the procedure must be repeated.

Note

This function is not supported by all operating systems. If no "Change settings" button is present or the button is grayed out, you will need to log on to the operating system with administrator privileges instead.

8.2 Help on the information system

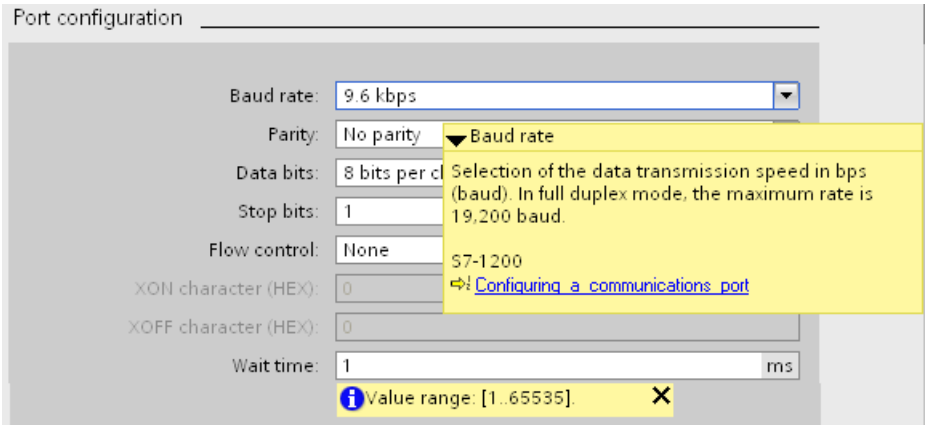
8.2.1 General remarks on the information system

Quick answers to your questions

A comprehensive help system is supplied with the TIA Portal for solving your tasks. It describes basic concepts, actions, and functions. While working with the program, you also receive the following support:

- Roll-out for correct inputs in dialog boxes
- Tooltips for information on elements of the user interface, for example text boxes, buttons and icons. Some of the tooltips are supplemented by cascades containing more precise information.
- Help on the current context, on menu commands for example when you click on the keys <F1> or <Shift+F1>.
- Help on alarms

The following figure shows an example of a cascading tooltip (top) and a roll-out (bottom):



Help

The supplied help system describes concepts, instructions and functions. It also contains reference information and examples. The help opens in a separate window.

A navigation pane appears on the left side of the help window. You can also hide the navigation pane to make room on the screen. The navigation pane provides you with the following functions:

- Table of contents
- Search in the index
- Full text search of the entire help
- Favorites

Identification of the topics in the help according to the type of information

The help topics are identified by different symbols depending on the type of information they contain.

Symbol	Information type	Explanation
	Operating instructions	Describes the steps to follow in order to carry out a particular task.
	Example	Contains a concrete example to explain the task.
	Factual information	Contains background information that you need to know to carry out a task.
	Reference	Contains comprehensive reference information to refer back to.

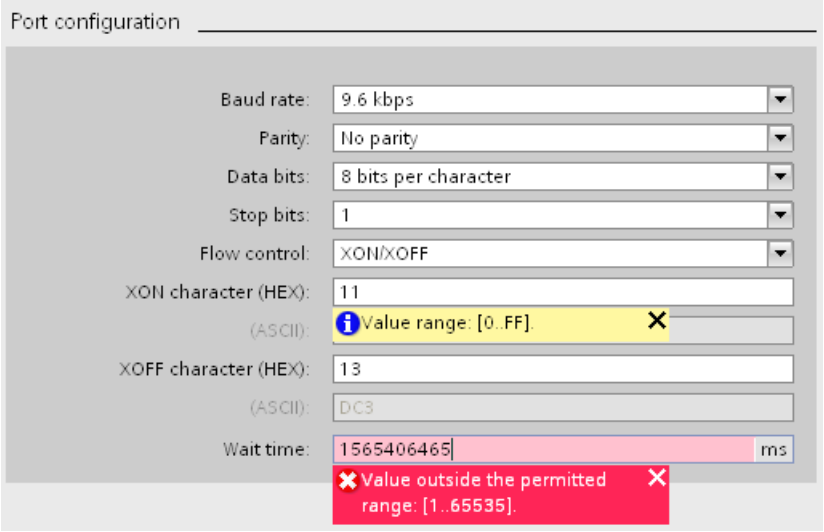
Identification of the topics in the help according to the target system

Depending on the products that are installed, the help system may contain sections that apply to specific devices only. To be able to recognize such sections at first glance, you will find a note in brackets in the table of contents. The search results in the full text search and in the index are marked in the same way if they only apply to a specific device.

Roll-out

Certain text boxes offer information that rolls out and helps you to enter valid parameters and values. The roll-out informs you about permitted value ranges and data types of the text boxes.

The following figure shows a roll-out (yellow) and a roll-out error message (red), which indicates an invalid value:



The roll-out is closed as soon as you exit the window or click on the "x" in the upper right-hand corner.

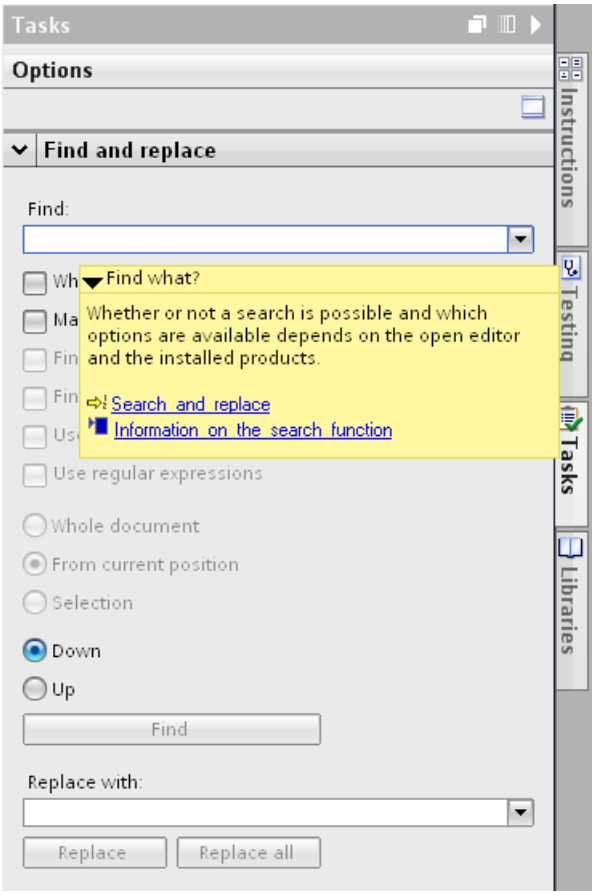
Tooltip

Interface elements offer you a tooltip for easier identification.

Tooltips, which have an arrow icon on the left, contain additional information in tooltip cascades. If you position the mouse pointer briefly over the tooltip or click the arrow icon, this information is displayed. The automatic display of tooltip cascades can be disabled.

If additional information is contained in the help system, a link to the corresponding help topic appears in the cascade. If you click on the link, the corresponding topic opens in help.

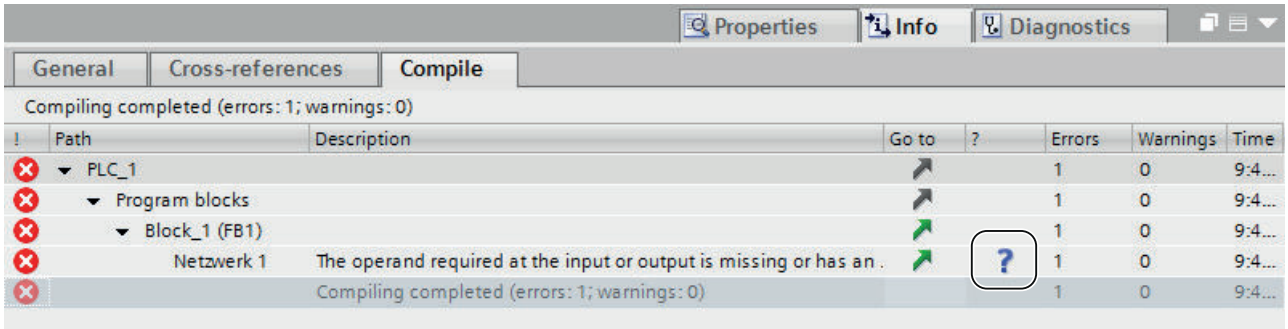
The following figure shows a tooltip with opened cascade:



Help on alarms

Numerous actions are accompanied by alarms in the Inspector window of the TIA Portal. The alarms give information about whether or not an action was successful. In addition, you see which changes have been made in the project. Further help is available for some alarms. If further help is available for an alarm, you access the help by clicking the question mark symbol.

The following figure shows the "Info" tab in the Inspector window with several alarms and a question mark symbol:



See also

Configuring the display of tooltips and tooltip cascades (Page 358)

Using user-defined documentation (Page 361)

8.2.2 Opening the Help system

You access the help via the menu, using links in the tooltip cascades, or with the <F1> key.

Procedure

To open the supplied help system, follow these steps:

1. In the "Help" menu, select the "Show help" command or press <F1> to display the help matching the current context.

Or:

1. Click on the link in a tooltip cascade to go directly to a point in the help system that contains more detailed information.

See also

Calling user-defined documentation (Page 370)

8.2.3 Searching the Help system for keywords

Searching for keywords in the help text

To search the help topics for predefined keywords, follow these steps:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.
   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Index" tab.
3. Enter the search term in the input box or select the search term from the list of key words.
4. Click "Display".

8.2.4 Full-text searches

Full-text searches

To search the entire text for specific words, follow these steps:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.
   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Search" tab.
3. Type in your search term in the text box.
4. Refine your search if necessary using additional criteria:
   - Select "Search previous results" to start an additional search operation of your last search results only.
   - Select "Search for similar words" to find words that differ only slightly from your search term.
   - Select "Search titles only" to obtain only results that contain your search term in the title. The contents of the Help topics are ignored during the search.
5. Click on the arrow button to the right of the search field to use logic operations. The following logic operations are available:
   - Combine two or more search terms using the "AND" operator to find only Help topics that contain all the search terms in the text.
   - Combine two or more search terms using the "OR" operator to find only Help topics that contain one or more of the search terms in the text.
   - Combine two or more search terms using the "NEAR" operator to find only Help topics that contain terms in close proximity to each other (eight words).
   - Precede a word with the "NOT" operator to exclude Help topics from the search that contain this word.
6. Click on "List topics" to start the search.
   The results are now listed with title, position and ranking. The "Position" column shows the section in which the Help topic found is located. Sorting according to ranking is based on the position of the Help topics found in the table of contents and based on the number of hits in the Help topics.

8.2.5 Using favorites

Using favorites

You can save individual help topics as favorites. This saves you searching for the help topic a second time.

Saving favorites:

To save a page as a favorite, follow these steps:

1. Open the help topic or the chapter you want to save as a favorite.
2. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.
   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
3. Open the "Favorites" tab.
4. Click the "Add" button.
   The help topic or chapter is saved as a favorite and is available the next time you open the help system.

Calling up favorites:

To call up a page from the favorites, follow these steps:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.
   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Favorites" tab.
3. Select the topic you want to open from the list.
4. Click the "Display" button.

Deleting favorites

To delete an entry from the favorites, proceed as follows:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.
   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Favorites" tab.
3. Select the topic you want to remove from the list.
4. Click the "Remove" button.

8.2.6 Printing help topics

Printing information

You can either print all the contents of the Help system or individual topics only.

Procedure

To select the topics you would like to print, follow these steps:

1. Click the "Display printing dialog" button.
   The table of contents opens in a separate window.
2. Select the check boxes for the folders and help topics to be printed in the "Print help topics" dialog.
3. Click the "Print" button to print the selected information.
   The "Print" dialog opens.
4. Select the printer on which you want print the help topics.
5. Click "Properties" if you want to make additional printer settings.
6. Confirm your entries with "OK".
   The help topics are printed out on the selected printer.

8.2.7 Configuring the display of tooltips and tooltip cascades

Configuration options for tooltips and tooltip cascade

You can customize the display of tooltips and tooltip cascades to suit your needs. You can make the following settings:

- Display or hide truncated text
  Sometimes texts may be too long for a text field. The texts are then fully displayed in a tooltip when you hover your mouse over the text field. You can enable or disable this function.
- Enable or disable tooltips
  Tooltips provide more detailed information about an element of the user interface. You can also have tooltips displayed in a cascade. If you disable the tooltips, the cascade with context-sensitive help is also no longer displayed. However, you have the option of manually displaying the tooltip for the currently active interface element by pressing <F1>.
- Enable or disable automatic opening of tooltip cascades
  By keeping the mouse pointer over a tooltip for a brief time, any available cascades are displayed automatically. You can enable or disable the automatic display of cascades. When automatic display is disabled, you must open the cascade manually if necessary. To do this, click on the arrow icon in the tooltip.

Procedure

To configure the display of tooltips and tooltip cascades, follow these steps:

1. Select the "Settings" command in the "Options" menu.
2. Select the "General" group in the area navigation.
3. Select or clear the individual check boxes in the "Tooltips" area to suit your needs. The "Open cascade automatically in tooltips" check box is only available if you have enabled the display of tooltips.

See also

General remarks on the information system (Page 351)

8.2.8 Safety Guidelines

Safety guidelines

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

Danger

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

Warning

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

Caution

with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

Notice

without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

Note

indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

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

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by qualified personnel. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Prescribed Usage

Note the following:

Warning

This device may only be used for the applications described in the catalog or the technical description and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens. Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

Trademarks

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

Disclaimer of Liability

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

8.2.9 Assembling customized documentation

Customized documentation

In the Siemens Industry Online Support, you can assemble customized documentation that is tailored to your needs. All configurable manuals and operating instructions of the Siemens Industry Online Support are available to you for this purpose. You can select the parts that are of interest to you and combine them into in a library to form personal documentation. You can organize the documentation using folders in the library. The folders will later become the individual chapters of your custom documentation.

You can open your personal library here (https://www.automation.siemens.com/mdm/?guiLanguage=en).

Requirements

- The manuals or operating instructions used must be configurable. You can recognize configurable manuals by the suffix "configurable" in their name.
- To use all the functions, you have to register in the Siemens Industry Online Support and log on.

Documentation in different languages

You can change the language of the assembled documentation to German, French, Spanish, Italian and Chinese. This gives you the possibility, for example, to gather relevant information for a specific project and make it available to colleagues who speak other languages.

Export function in the documentation

You can perform an export at any part of your library in various formats (PDF, XML, RTF).

Help on creating documentation

You can find more help on creating and using custom documentation in the Siemens Industry Online Support (https://www.automation.siemens.com/mdm/_help/en/mdm_reference_manual_de-DE.htm).

8.3 Providing user-defined documentation

8.3.1 Using user-defined documentation

User-defined documentation for project or library contents

Over time, you create your own contents in a project or a library. Your own contents include, for example, blocks, tags or library types. While the functionality of the TIA Portal is described in the supplied help system, there is no help for the contents you have created yourself. You can create your own user-defined documentation to explain to other employees how your project works or how to use individual library types.

You can provide user-defined documentation in the available user interface languages. The default user interface languages in the TIA Portal are English, German, French, Spanish, Italian, and Chinese. You need to observe a few conventions when you create user-defined documentation so that the help matching an object can be opened.

You create the user-defined documentation either in one of the supported Office formats or as compiled HTML help in CHM format.

Possible areas for user-defined documentation

You can offer user-defined documentation in the following areas of the TIA Portal, for example:

- Project tree
- "Libraries" task card and library view
- Some editors, depending on the products installed, for example:
  - Program editor
    The programming languages LAD, FBD, STL, SCL and GRAPH are supported. Block calls also support the connection of user-defined documentation.
  - "Screens" editor
    Uses of screens and instances of HMI faceplates are supported in the "Screens" editor.

Directories for user-defined documentation

Save the user-defined documentation in one of the following directories:

- Project folder
  If you create user-defined documentation for objects within a project, save this help in the project folder. The user-defined documentation is also included when you pass on the project.
- Directory of a global library
  If you create user-defined documentation for objects within a global library, save the user-defined documentation in the directory of the global library. The user-defined documentation is also included when you pass on the global library.
- Central directory on the hard drive or a network drive
  You can store the user-defined documentation in a central directory on the hard disk or on a network drive. In this way, you have access to the user-defined documentation in each project or you use the documentation on a network drive together in the team. You specify the central file directory for the user-defined documentation using an XML file or in the settings of the TIA Portal.

Homepage for the user-defined documentation

You can create a separate homepage for each language version of the user-defined documentation. The homepage for the user-defined documentation can contain general help for a project or for a library. The homepage must be saved in the central storage directory for user-defined documentation.

Calling the user-defined documentation

If user-defined documentation is available for an object, you call it with the keyboard shortcut <Shift+F1>. The user-defined documentation is always opened with the standard program specified for the respective file format in Microsoft Windows.

Once you have pressed <Shift+F1>, certain directories are searched in a fixed order for user-defined documentation. The search order is given below:

1. Search in the central directory for user-defined documentation
   1.1 Search for a CHM file
   1.2 Search for documentation in other file formats
2. Search in the project or library directory
   2.1 Search for a CHM file
   2.2 Search for documentation in other file formats

The search is initially performed in the language directory for the currently set user interface language of the TIA Portal. If no help is contained in this language directory, the search for user-defined documentation is performed in the same order in the English language directory.

As soon as user-defined documentation is found in one location, the user-defined documentation is opened and the search is canceled. If no user-defined documentation is found in any of the directories, a search for a homepage for user-defined documentation is performed in the order presented above. The search for a homepage is again initially performed in the language directory for the currently set user interface language. If no homepage is found there, the English language directory is searched.

Call log

You can display a call log for the user-defined documentation for easier connection of the user-defined documentation. The alarms within the log indicate the directories in which documentation is searched for and whether the call of the user-defined documentation is successful. In addition, the file name that is expected for the file is indicated. This allows you to identify how you must name your documentation and the directories in which you must save the user-defined documentation. The call log has the same sequence as the one used to search for user-defined documentation or a homepage.

The log is displayed in the Inspector window in the "Info" tab. Before you can display the call log, you must first enable the call log in the settings of the TIA Portal or using an XML file.

See also

Conventions for the creation (Page 367)

Specifying settings with an XML file (Page 365)

Creating a homepage (Page 366)

Creating user-defined documentation (Page 371)

Calling user-defined documentation (Page 370)

General remarks on the information system (Page 351)

8.3.2 Specifying settings in the TIA Portal

You specify the following settings for user-defined documentation in the settings of the TIA Portal:

- Display call log in the Inspector window
  A log of the call of user-defined documentation is displayed on the 'Info > General' tab of the Inspector window. The log helps you to adhere to the conventions for calling user-defined documentation.
- Search for user-defined documentation in a central file directory
  You can save user-defined documentation in a directory outside the current project directory in order, for example, to make documentation available across projects.
- Central directory for user-defined documentation
  You store cross-project documentation in the central file directory for user-defined documentation.

Note

XML configuration file takes precedence over the settings of the TIA Portal

If you use an XML configuration file and have specified settings for user-defined documentation there, the settings in the XML file take precedence. As soon as you refresh the XML configuration file or restart the TIA Portal, the settings from the XML file are applied. The settings that you have made in the TIA Portal lose their validity.

Procedure

To specify a central storage location for user help, follow these steps:

1. Select the "Settings" command in the "Options" menu.
2. Open the "General > General" area.
3. Navigate to the "User documentation" section.
4. Select the "Display call log for user-defined documentation" check box in order to display a log of the call of the user-defined documentation in the Inspector window.
5. Select the "Search for user-defined documentation in a central directory" check box in order to store user-defined documentation in a cross-project directory.
6. Specify the path to where you save the cross-project documentation in the "Central directory for user-defined documentation" field.

See also

Specifying settings with an XML file (Page 365)

8.3.3 Specifying settings with an XML file

As an alternative to settings in the TIA Portal, you can make settings for the user-defined documentation in an XML file. The XML file is the same file you use for integrating corporate libraries.

If you use an XML configuration file and have specified settings for user-defined documentation there, the settings in the XML file take precedence. As soon as you refresh the XML configuration file or restart the TIA Portal, the settings from the XML file are applied. The settings that you have made in the TIA Portal lose their validity.

You can set the following options in the XML configuration file:

- Display call log in the Inspector window
  A log of the call of user-defined documentation is displayed on the 'Info > General' tab of the Inspector window. The log helps you to adhere to the conventions for calling user-defined documentation.
- Search for user-defined documentation in a central file directory
  You can save user-defined documentation in a directory outside the current project directory in order, for example, to make documentation available across projects.
- Central directory for user-defined documentation
  You store cross-project documentation in the central file directory for user-defined documentation.

Procedure

To specify settings for the user-defined documentation, follow these steps:

1. Create an XML file named "CorporateSettings.xml", if you are not yet using an XML configuration file for the integration of company libraries. If you are already using a configuration file, proceed with step 3.
   The configuration file must be saved with "UTF-8" coding.
2. Save the file in the following directory on your computer:
   C:\ProgramData\Siemens\Automation\Portal V13\CorporateSettings\
3. Enter the content listed below into the XML configuration file.
4. Adapt the attributes for display of the user-defined documentation. The meaning of the individual elements is available in the comments in the XML configuration file. Use the value "true" to activate a function. Use the value "false" to deactivate a function.

Content of the XML configuration file

The XML configuration file must have the following content:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
 <Settings.Settings ID="0">
  <ObjectList>
   <Settings.General ID="1" AggregationName="General">
    <!-- Here you find the settings for global company libraries, if available. -->
    <ObjectList>
     <Settings.UserDocumentation ID="2" AggregationeName="UserDocumentation">
      <!-- In the following section, you specify the values for display of the user-defined
documentation. -->
      <AttributeList>
       <!-- Activates or deactivates the display of the access log. -->
       <DisplayLogInformation>
        <Value>true</Value>
       </DisplayLogInformation>
       <!-- Activates or deactivates the search for user-defined documentation in a central
directory. -->
       <EnableLookupFromCentralStorageLocation>
        <Value>true</Value>
       </EnableLookupFromCentralStorageLocation>
       <!-- Specifies the central directory for user-defined documentation. -->
       <CentralStorageLocation>
        <Value>D:\CorporateDocumentation\UserDocumentation\</Value>
       </CentralStorageLocation>
      </AttributeList>
     </Settings.UserDocumentation>
    </ObjectList>
   </Settings.General>
  </ObjectList>
 </Settings.Settings>
</Document>
```

See also

Using user-defined documentation (Page 361)

Specifying settings in the TIA Portal (Page 364)

Calling user-defined documentation (Page 370)

Creating a homepage (Page 366)

Creating a configuration file for corporate libraries (Page 496)

8.3.4 Creating a homepage

You can design a homepage for user-defined documentation. The homepage can be an HTML page that you save either within a CHM or in the directory of the respective language. You can

also use other file formats approved for user-defined documentation. You design the homepage of the user-defined documentation outside the TIA Portal.

Procedure

To create a homepage, follow these steps:

1. Design a file in HTML format or in any other file format approved for user-defined help.
2. Name the file "Home".
3. Copy the file to the central directory for user-defined documentation on the hard disk or on a network drive:
   <Central directory for user-defined documentation>\<Folder for the respective language>
4. If the respective language folder does not exist yet, create the folder now.
   Alternative: If you are creating the homepage for a CHM file, place the homepage in the main directory of the CHM file.

Sample configuration for the homepage

Below you see the correct path for the following conditions:

- The user-defined documentation is in Spanish.
- The homepage is an HTML file.

The path for these conditions is as follows:

<Central directory for user-defined documentation>\es-ES\Home.html

See also

Specifying settings with an XML file (Page 365)

Using user-defined documentation (Page 361)

Conventions for the creation (Page 367)

Calling user-defined documentation (Page 370)

Displaying the call log (Page 371)

Creating user-defined documentation (Page 371)

8.3.5 Conventions for the creation

You must observe some conventions to ensure that user-defined documentation is called at the correct location:

- the user-defined documentation must be saved in the correct directory.
- The file name must be exactly the same as the object name in the TIA Portal.

To prevent malicious code from being executed on your computer, only file formats that are considered as relatively safe are permitted.

Supported file formats

Create the user-defined documentation in one of the following file formats:

- Microsoft Word (.docx)
- Microsoft Excel (.xslx)
- Microsoft PowerPoint (.pptx and .ppsx)
- HTML pages (.htm or .html)
- Microsoft XPS (.xps)
- Rich Text Format (.rtf)
- Text documents (.txt)
- Compiled HTML help (.chm)
- PDF documents (.pdf)

You save the homepage of the user-defined documentation in HTML format or save the homepage within a CHM file.

Notice

Infection of the computer with malicious code

If the user-defined documentation contains malicious code, it can infect your computer. Especially HTML pages and CHM files can contain malicious code.

Make sure that the user-defined documentation comes from a trustworthy source. You should also use the standard security measures, such as the use of a firewall and an up-to-date virus scanner.

Directories for user-defined documentation

Save the user-defined documentation in one of the following directories:

- Project folder:
  UserFiles\UserDocumentation\<Folder for the respective language>\<Object category>
- Directory of a global library:
  UserFiles\UserDocumentation\<Folder for the respective language>\<Object category>
- Central directory on the hard drive or a network drive:
  <Central directory for user-defined documentation>\<Folder for the respective language>
  \<Object category>\

The user-defined documentation must be located in the suitable subfolder for the respective language. The table below shows the respective language folders for the user languages installed as default:

Language	Subfolder
German	\de-DE
English	\en-US
Spanish	\es-ES
French	\fr-FR

Language	Subfolder
Italian	\it-IT
Chinese	\zh-CN

The language folder must contain a separate subfolder for each object category. Create the corresponding subfolders for objects for which you are providing user-defined documentation. Always use the English designation of the object category. The table below shows the English designations of the most important object categories in the TIA Portal:

Object category	English designation
HMI screens	Screens
Organization blocks (OB)	Organization Blocks
Function blocks (FBs)	Function Blocks
Functions (FCs)	Functions
Data blocks	Data Blocks
Types in the library	Library Types
Master copies in the library	Master Copies
The project node in the project tree	Projects
All types of folders in the project tree, in the project library or in global libraries	Folders
All types of links in the project tree, for example, "Add new block", "Add new device", etc.	ShortCut
Libraries in the "Libraries" task card or in the library view	Libraries

If you are not sure of the English designation for an object category, change the user interface language of the TIA Portal to English. Alternatively, open the user-defined documentation for an object with <Shift+F1> and check in the call log which designation is expected for the object category.

Permitted file names

The file name must be exactly the same as the object name in the TIA Portal.

There are, however, restrictions for file names under Microsoft Windows. The same restrictions apply to the file system used to format the hard drive. The file name may only include certain characters and must not exceed a specific length. The restrictions for file names differ depending on the Windows version and the file system used for the hard drive.

To ensure that the help call works, read up on possible restrictions in the Microsoft Windows documentation.

Special features of CHM files

You store CHM files directly in the respective language folder. The folder for the respective object category must be included in the actual CHM file. Within the compiled CHM file, the names of the individual HTML files must also be exactly the same as the object names in the TIA Portal.

Note

Opening CHM files on network drives

If CHM files are saved on a network drive, the CHM files are not displayed correctly in more recent versions of Microsoft Windows. This behavior is determined by the security guidelines of the operating system. All versions of Microsoft Windows as of Windows Server 2003 SP1 are affected.

You can bypass the security guidelines by changing the registry database in Microsoft Windows.

To not compromise the security of your computer, save the CHM files only locally on your computer and do not change the registry database.

See also

Using user-defined documentation (Page 361)

Creating a homepage (Page 366)

8.3.6 Calling user-defined documentation

The user-defined documentation is opened in the language that is currently set as the user interface language. If there is no user-defined documentation available in the currently set user interface language, the English version of the user-defined documentation opens. If no user-defined documentation exists, a homepage is searched for.

Requirement

You have already saved user-defined documentation or a homepage according to the conventions.

Procedure

To open the user-defined documentation, follow these steps:

1. Select the object for which you want to display the user-defined documentation.
2. Press <Shift+F1>.
   The suitable user-defined documentation or the homepage opens.

See also

Using user-defined documentation (Page 361)

Opening the Help system (Page 355)

Creating a homepage (Page 366)

Specifying settings with an XML file (Page 365)

Displaying the call log (Page 371)

8.3.7 Displaying the call log

Use the call log to check whether the user-defined documentation is connected correctly. The call log shows the directories in which the search is performed for user-defined documentation or a homepage. The call log also displays the names that the individual files must have in order to call the user-defined documentation.

Requirement

The call log is enabled in the settings of the TIA Portal or using an XML configuration file.

Procedure

To display the call log, follow these steps:

1. Open the "Info" tab in the Inspector window.
2. Open the "General" tab.
3. Select the object for which you want to call the help.
4. Press <Shift+F1>.
   If possible, the matching user-defined documentation or the homepage of the user-defined documentation is opened. In any case, you will be informed in the Inspector window about which user-defined documentation is opened. You may be shown the directories in which no user-defined documentation was found.

See also

Calling user-defined documentation (Page 370)

Creating a homepage (Page 366)

8.3.8 Creating user-defined documentation

You create user-defined documentation for individual elements within a project or global library outside the TIA Portal. You can create the user-defined documentation in all available user interface languages.

If you create the user-defined documentation as CHM file, the procedure for creating the help is somewhat different to the creation process for other file formats.

Note the information provided in chapter "Conventions for the creation (Page 367)".

Creating user-defined documentation as single file

To create user-defined documentation as a single file, follow these steps:

1. Create a file in a valid file format.
2. Name the file identically to the object for which you want to call the user-defined documentation.
   If you are offering help for a library type, for example, name the help file identical to the type.
3. Depending on whether you are creating the user-defined documentation for project contents or for contents of a global library, copy the file to one of the following storage locations:
   - project folder under "UserFiles\UserDocumentation\<Folder for the respective language>\<Object category>"
   - Directory of a global library under "UserFiles\UserDocumentation\<Folder for the respective language>\<Object category>"
   - Central directory on the hard drive or a network drive:
     <Central directory for user-defined documentation>\<Folder for the respective language>\<Object category>\

   If the respective language folder or the folder for the object category does not exist yet, create the required folders before copying the file.

Sample configuration for user-defined documentation

Below you see the correct path for the following conditions:

- The user-defined documentation is intended for a type in a global library.
- The user-defined documentation is in French.
- The type is called "commande de moteur".
- The user-defined documentation is supplied with the global library.
- The user-defined documentation is created in Microsoft PowerPoint format.

The path for these conditions is as follows:

<Folder of the global library>\UserFiles\UserDocumentation\fr-FR\Library Types\commande de moteur.pptx

Creating user-defined documentation as CHM file

To provide user-defined documentation in CHM format, follow these steps:

1. Create a folder in Windows Explorer for each object category for which you want to create user-defined documentation. Use the English designation for the object category.
2. Create an HTML file for each object for which you want to provide user-defined documentation. Name the HTML file identically to the object for which you want to call the user-defined documentation. If you want to provide user-defined documentation for a library type, for example, name the HTML file identically to the type.
3. Store the HTML files in the corresponding folders of the respective object category.
4. Use the Microsoft HTML Help Workshop to create the CHM. Use the prepared folder structure with the HTML files.
5. Copy the CHM file to one of the following storage locations:
   - Project folder under "UserFiles\UserDocumentation\<Folder for the respective language>"
   - Directory of a global library under "UserFiles\UserDocumentation\<Folder for the respective language>"
   - Central directory on the hard drive or a network drive:
     <Central directory for user-defined documentation>\<Folder for the respective language>

   If the respective language folder does not exist yet, create the language folder before you copy the CHM file.

See also

Creating a homepage (Page 366)

Conventions for the creation (Page 367)

Editing projects

9

9.1 The basics of projects

Introduction

Projects are used to organize the storage of data and programs resulting from the creation of an automation solution. The data that makes up a project includes the following:

- Configuration data on the hardware structure and parameter assignment data for modules
- Project engineering data for communication over networks
- Project engineering data for the devices
- Logs for important events in the life cycle of the project

Project hierarchy

Data is stored in a project in the form of objects. Within the project, the objects are arranged in a tree structure (project hierarchy).

The project hierarchy is based on the devices and stations along with the configuration data and programs belonging to them.

Common data of the project and online access, for example, are also displayed in the project tree.

See also

Using logs (Page 376)

Creating a new project (Page 376)

Compatibility of projects (Page 377)

Opening projects (Page 379)

Upgrading projects (Page 380)

Displaying properties of the project (Page 382)

Saving projects (Page 383)

Closing projects (Page 384)

Removing projects (Page 384)

Deleting projects (Page 385)

9.2 Using logs

For some operations within the TIA Portal, logs are created automatically in the background. These logs document changes in the project. Logs are created automatically, for example, when you migrate projects and programs or when you update instances from the library.

Logs are displayed in the "Common data" folder in the project tree. They are stored together with the project in the project folder and can therefore be read independently of the programming device/PC used as soon as you have opened the project. The log can be filtered for errors, warnings, and information.

In addition to displaying them in the TIA Portal, logs can also be printed.

Displaying logs

To open a log, follow these steps:

1. Open the "Common data > Logs" folder in the project tree.
2. Double-click the desired log in the list.
   The contents of the log are displayed in the work area.
3. Optional: To show or hide a particular category of alarms, activate or deactivate the button for "Errors", "Warnings", or "Information" in the toolbar.

Deleting logs

To delete a log, follow these steps:

1. Select the log in the project tree.
2. Press <Del>.
   The selected log is deleted from the project directory and removed from the project tree.

9.3 Creating and managing projects

9.3.1 Creating a new project

Procedure

To create a new project, follow these steps:

1. Select the "New" command in the "Project" menu.
   The "Create a new project" dialog opens.
2. Enter your project name and path or accept the proposed settings.
3. Click the "Create" button.

Result

The new project is created and displayed in the project tree.

See also

The basics of projects (Page 375)
Compatibility of projects (Page 377)
Opening projects (Page 379)
Upgrading projects (Page 380)
Displaying properties of the project (Page 382)
Saving projects (Page 383)
Closing projects (Page 384)
Removing projects (Page 384)
Deleting projects (Page 385)

9.3.2 Compatibility of projects

You can use the TIA Portal to open projects that were created with an older version of the TIA Portal or with a different installation package. In the following, you will learn what to consider in this case.

Opening projects from older product versions

The table below describes the behavior of the TIA Portal when opening projects from older product versions:

Product version of the TIA Portal (file extension of the respective version)	Behavior when opened with the current product version of the TIA Portal
V10.5 (.ap10) V11.x (.ap11)	The project is automatically upgraded to the latest product version when opened, following your confirmation. The upgraded project is a copy of the original project. The original project is retained unchanged.
V12 (.ap12)	You can select between the following options: • Upgrade project to product version V12 SP1 and open in compatibility mode • Upgrade project to the current product version and open The upgraded project is a copy of the original project in both cases. The original project is retained unchanged.
V12 SP1 (.ap12) V13 (.ap13)	You can select between the following options: • Open project in compatibility mode • Upgrade project to the current product version and open A copy of the original project is created when you upgrade the project. The original project is retained unchanged.

Compatibility mode

Compatibility mode is available for projects that were created with TIA Portal V12 SP1 and V13. If you open a project in the V12 SP1 or V13 project format and do not upgrade it, the project is in compatibility mode. The range of functions of the TIA Portal is limited to the range of functions of the earlier product version. The project remains backward compatible and can still be opened and edited with the earlier version of the TIA Portal.

Components supplied subsequently for the earlier product version as part of a Hardware Support Package (HSP) can be added to projects in compatibility mode. To continue working with the project in the earlier version of the TIA Portal, you simply need to install the Hardware Support Package as well.

Global libraries are always created in the most recent format and are not backwards compatible, even if the project is opened in compatibility mode.

You need to upgrade the project to use the full range of functions of the current version. By upgrading the project you convert it to the current project format. The full range of functions of the current version becomes available.

Backward compatibility with the current product version

Projects that have been saved with the current version of the TIA Portal are not backward compatible with older versions due to their enhanced functionality. Projects saved with TIA Portal V13 SP1 can only be opened with TIA Portal V13 SP1 or later.

Opening projects created with add-on products

The project you want to open may include data that were created with an optional software. The following cases may occur if you did not install the optional software:

- Software components which are not absolutely required are missing:
  A dialog appears listing the missing software components. After the project is opened, its properties are displayed. You now have the opportunity to install missing components. All the devices contained in the project are available even if you do not install the missing components. However, you can only work with the devices that are supported by the currently installed software.
  Devices which are not supported because software components are missing are identified by the following symbol in the project tree:

- Essential software components are missing:
  A dialog appears listing the missing software components. The essential software components are marked. The project can only be opened if you install the missing software components.

See also

The basics of projects (Page 375)

Creating a new project (Page 376)

Opening projects (Page 379)

Upgrading projects (Page 380)

Displaying properties of the project (Page 382)

Saving projects (Page 383)

Closing projects (Page 384)

Removing projects (Page 384)

Deleting projects (Page 385)

Compatibility of global libraries (Page 484)

9.3.3 Opening projects

You can open all projects from the current version and earlier versions in the TIA Portal.

Projects from earlier versions of the TIA Portal may first have to be upgraded to a more recent project format. You are prompted to upgrade the project when you open it.

You recognize projects of the TIA Portal by their file name extension ".ap[version number]". Projects of TIA Portal V13 SP1 have the file name extension ".ap13".

Procedure

To open an existing project, follow these steps:

1. Select the "Open" command in the "Project" menu.
   The "Open project" dialog box opens and the list of most recently used projects is displayed.
2. Select a project from the list and click "Open".
3. If the project you require is not included in the list, click the "Browse" button. Navigate to the desired project folder, and open the project file.
   Projects in the current project format are opened in the project view. If you have selected a project from an older version of the TIA Portal, the "Upgrade project" dialog opens. More information on upgrading the project can be found in chapter "Upgrading projects (Page 380)".

See also

The basics of projects (Page 375)

Creating a new project (Page 376)

Compatibility of projects (Page 377)

Upgrading projects (Page 380)

Displaying properties of the project (Page 382)

Saving projects (Page 383)

Closing projects (Page 384)

Removing projects (Page 384)

Deleting projects (Page 385)

Compiling project data (Page 393)

Opening a global library (Page 486)

9.3.4 Upgrading projects

Projects from older versions of the TIA Portal can be edited with the current version of the TIA Portal. Depending on the product version that was used to create the project, you either upgrade the project or open the project in compatibility mode. To find out which options you have depending on the product version of the project, go to chapter "Compatibility of projects (Page 377)".

When you open projects from older product versions, you are prompted to upgrade the project. You can manually upgrade projects that are already open in compatibility mode. After the project is upgraded to the current product version, it can no longer be opened in older versions of the TIA Portal. But the original project is still available. The upgraded project is saved as a copy and receives the name extension "_V13_SP1".

Upgrading projects from V11.x or lower

To upgrade a project from TIA Portal V11.x or earlier, follow these steps:

1. Open the project.
   The "Upgrade project" dialog box opens.
2. Click "OK".
3. Compile the hardware and software of all devices in the project.

Upgrading projects from V12

To upgrade a project from TIA Portal V12, follow these steps:

1. Open the project.
   The "Upgrade project" dialog box opens.
2. Select the target version for the upgrade:
   - Click "Yes" to upgrade the project to the current product version.
   - Click "No" to upgrade the project to product version V12 SP1 and to work in compatibility mode
3. Compile the hardware and software of all devices in the project.

Upgrading V12 SP1 and V13 projects or using them in compatibility mode

To upgrade a project from TIA Portal V12 SP1 or V13 or to use it in compatibility mode, follow these steps:

1. Open the project.
   The "Upgrade project" dialog box opens.
2. Select how you want to proceed with the project:
   - Click "Yes" to upgrade the project to the current product version.
   - Click "No" to use the project in compatibility mode.
3. Optional: If you have upgraded the project to the current product version, compile the hardware and software of all devices in the project.

Manually upgrading projects in compatibility mode to V13 SP1

For this procedure, a project must already be open in compatibility mode.

To manually upgrade a project in compatibility mode, follow these steps:

1. Select the "Upgrade" command in the "Project" menu.
   A security prompt appears.
2. Click "Yes" to confirm.
   The project is closed, and the upgraded project is opened.
3. Compile the hardware and software of all devices in the project.

Notes

Pay attention to the following notes after upgrading a project:

Note

Upgrading global libraries

Because global libraries are independent of projects, they are not upgraded automatically together with the project. If you want to continue using global libraries from older versions of the TIA Portal, you upgrade these global libraries as well. More information on upgrading global libraries can be found in the chapter "Compatibility of global libraries (Page 484)".

Note

Upgrading know-how-protected blocks

The block is only upgraded and loaded after it has been opened once with the password. This means you should open know-how-protected blocks once after upgrading the project to also upgrade the blocks. If you have protected numerous know-how-protected blocks with the same password, you can select and open all of them at once.

See also

Compatibility of global libraries (Page 484)

Upgrading global libraries (Page 487)

The basics of projects (Page 375)

Creating a new project (Page 376)

Compatibility of projects (Page 377)

Opening projects (Page 379)

Displaying properties of the project (Page 382)

Saving projects (Page 383)

Closing projects (Page 384)

Removing projects (Page 384)

Deleting projects (Page 385)

Compiling project data (Page 393)

9.3.5 Displaying properties of the project

You can display the properties of a project. Properties include the following:

- Metadata for the project
  This includes the following information: creation time, author, file path, project size, copyright, project languages, etc. Many of the properties can be changed.
- Project history
  The project history contains an overview with important events in the project life cycle. Here, for example, you can see the version of the TIA Portal used to create a project and whether it has been converted to another version in the meantime. If a project was created during a migration, for example, this is also indicated in the project history table with the date and time of the migration. If a log was created for an event, you can also call the log directly.
- Support packages in the project
  An overview of the add-on software needed to work with all devices in the project is displayed. In addition, installed GSD files are listed (device description files for other devices in the hardware catalog).
- Software products in the project
  You can display an overview of all installed software products needed for the project.

Procedure

To display the project properties, follow these steps:

1. Select the open project in the project tree.
2. Select "Properties" in the shortcut menu of the project.
   The dialog with the properties of the project opens.
3. Select the project properties in the area navigation that you want to have displayed.

See also

The basics of projects (Page 375)

Creating a new project (Page 376)

Compatibility of projects (Page 377)

Opening projects (Page 379)

Upgrading projects (Page 380)

Saving projects (Page 383)

Closing projects (Page 384)

Removing projects (Page 384)

Deleting projects (Page 385)

9.3.6 Saving projects

You can save the project at any time either under the same or a different name. You can even save a project when it still contains elements with errors.

Saving a project

To save a project, follow these steps:

1. Select the "Save" command in the "Project" menu.
   All changes to the project are saved under the current project name. If you are editing a project from an earlier version of the TIA Portal, the file extension of the project is also retained and you can continue to edit the project in the earlier version of the TIA Portal.

Project Save as

To save a project under another name, follow these steps:

1. Select the "Save as" command in the "Project" menu.
   The "Save current project as" dialog opens.
2. Select the project folder in the "Save in" box.
3. Enter the new project name in the "File name" box.
4. Confirm your entry with "Save".
   The project is saved under the new name and opened.

Note

Undoing actions

Keep in mind that you cannot undo actions once you have saved the project.

See also

The basics of projects (Page 375)

Creating a new project (Page 376)

Compatibility of projects (Page 377)

Opening projects (Page 379)

Upgrading projects (Page 380)

Displaying properties of the project (Page 382)

Closing projects (Page 384)

Removing projects (Page 384)

Deleting projects (Page 385)

9.3.7 Closing projects

Procedure

To close a project, follow these steps:

1. Select the "Close" command in the "Project" menu.
   If you have made changes to the project since the last time you saved it, a message is displayed.
2. Decide whether or not you want to save the changes.

See also

The basics of projects (Page 375)

Creating a new project (Page 376)

Compatibility of projects (Page 377)

Opening projects (Page 379)

Upgrading projects (Page 380)

Displaying properties of the project (Page 382)

Saving projects (Page 383)

Removing projects (Page 384)

Deleting projects (Page 385)

9.3.8 Removing projects

You can remove projects from the list of recently used projects. The project data is retained on the storage medium.

Procedure

To remove a project from the list of recently used projects, follow these steps:

1. Select the "Delete project" command in the "Project" menu.
   The "Delete project" dialog opens and includes the list of most recently used projects.
2. Select a project from the list.
3. Click the "Remove" button.
4. Click "Yes" to confirm the prompt in order to remove the project from the list.

Result

The project is no longer displayed in the list of recently used projects. If you open the project again, it will be added to the list again.

See also

The basics of projects (Page 375)
Creating a new project (Page 376)
Compatibility of projects (Page 377)
Opening projects (Page 379)
Upgrading projects (Page 380)
Displaying properties of the project (Page 382)
Saving projects (Page 383)
Closing projects (Page 384)
Deleting projects (Page 385)

9.3.9 Deleting projects

Note

When you delete a project, the entire project data is removed from the storage medium.

Requirement

The project you want to delete is not open.

Procedure

Follow the steps below to delete an existing project:

1. Select the "Delete project" command in the "Project" menu.
   The "Delete project" dialog opens and includes the list of most recently used projects.
2. Select a project from the list.
   If the project you require is not included in the list, click the "Browse" button. Navigate to the desired project folder, and open the project file.
3. Click the "Delete" button.
4. Click "Yes" to confirm. This starts the deletion of the project.

Result

The entire project folder is deleted from the file system.

See also

The basics of projects (Page 375)
Creating a new project (Page 376)
Compatibility of projects (Page 377)
Opening projects (Page 379)
Upgrading projects (Page 380)
Displaying properties of the project (Page 382)
Saving projects (Page 383)
Closing projects (Page 384)
Removing projects (Page 384)

9.3.10 Archiving and retrieving projects

9.3.10.1 Working with project archives

Archiving and transferring projects

If you work for a long time with a project, large files may result, especially with extensive hardware configurations. Therefore, you may want to reduce the size of the project, for example, when you archive it to an external hard drive, or when you send it via e-mail and require a smaller file size.

Options for reducing the size of the project

There are two ways to reduce the size of the project:

- Creating a project archive
  TIA Portal project archives are compressed files, each containing an entire project including the entire folder structure of the project. Before the project directory is compressed into the archive file, all files are reduced to their essential components to further decrease the size of the project. Project archives are therefore well suited for sending via e-mail.
  Project archives have the file extension ".zap[version number of the TIA Portal]". Projects created with TIA Portal V13 SP1 have the file extension ". zap13".
  To open a project archive, retrieve the project archive. By retrieving it, the archive file with the included project files is extracted to the original project directory structure.
- Minimizing a project
  You can skip additional compression in an archive file, and instead create a copy of the project directory. The included files are reduced to the essential elements of the project. This minimizes the required storage space. The full functionality of the project is maintained and you can open the project as usual.
  A minimized project is especially well suited for archiving, for example, on an external medium.

See also

Retrieving compressed project (Page 389)

Creating compressed project archive (Page 387)

Minimizing project (Page 388)

9.3.10.2 Creating compressed project archive

You can reduce the required storage space of the currently open project by archiving the project in a compressed file.

Note

The most recently saved state of the project is used for archiving. Therefore, save the project before using the archiving function. This will ensure that your most recent changes are included in the archived project.

Procedure

To archive a project, follow these steps:

1. In the "Project" menu, select the "Archive > Compressed archive" command.
   The "Archive current project as..." dialog opens.
2. Select the directory in which you want to save the archive file.
3. Enter a file name in the "File name" box.
4. Click "Save".

Result

A compressed file with the extension ".zap13" is generated. Project archives of projects in compatibility mode for product version V12 SP1 receive the file extension ".zap12" and are backward compatible with TIA Portal V12 SP1. The archive file contains the complete project directory. The individual files of the project are also reduced to the essential components in order to save space.

See also

Working with project archives (Page 386)

Retrieving compressed project (Page 389)

Minimizing project (Page 388)

9.3.10.3 Minimizing project

You can reduce the required storage space of the currently open project by reducing the project files to their essential components. The "Minimize" function creates a copy of the original project directory.

Note

The most recently saved state of the project is used for minimizing. Therefore, save the project before using the minimizing function. This will ensure that your most recent changes are included in the project copy.

Procedure

To minimize a project to its essential components, follow these steps:

1. In the "Project" menu, select the "Archive > Minimize project" command.
   The "Minimize current project under..." dialog opens.
2. Select the directory in which you want to save the project copy.
3. Enter the name of the new project directory in the "Directory name" box.
4. Click "Save".

Result

A copy of the original project directory is created at the designated location. The files contained within it are reduced to their essential components in order to save space.

See also

Working with project archives (Page 386)

Creating compressed project archive (Page 387)

9.3.10.4 Retrieving compressed project

You extract project archives of the TIA Portal with the "Retrieve" function. This restores the project directory structure including all project files.

Requirement

No project is open.

Procedure

To extract a project archive, follow these steps:

1. Select the "Retrieve" command in the "Project" menu.
   The "Retrieve archived project" dialog opens.
2. Select the project archive.
3. Click "Open".
4. The "Find folder" dialog opens.
5. Select the target directory to which the archived project should be extracted.
6. Click "OK".

Result

The project is extracted to the selected directory and opened immediately. If you extract a project archive that includes a project from a product version earlier than V12 SP1, you may have to upgrade the project. You will automatically receive the corresponding prompt as soon as you open the project. The same rules apply that are described in the chapter "Compatibility of projects (Page 377)".

See also

Working with project archives (Page 386)

Opening projects (Page 379)

Compatibility of projects (Page 377)

Upgrading projects (Page 380)

9.4 Using reference projects

9.4.1 Basics of reference projects

Introduction

You can open other projects as a reference in addition to the current project. You can use these reference projects as follows:

- You can drag individual objects from a reference project into the current project and then edit them.
- You can open specific objects, for example, code blocks from a reference project as read-only. But this is not possible for all elements.
- You can use an offline/offline comparison to compare devices of the reference project to devices from the current project.

Note that reference projects are read-only. You cannot change the objects of a reference project.

Projects that were created with an older version of the TIA Portal or with a different installation package can also be opened as reference projects. The same compatibility rules apply here as for normal opening of a project from an older version of the TIA Portal.

See also: Compatibility of projects (Page 377)

See also

Comparing reference projects (Page 391)

Opening and closing a reference project (Page 390)

Reference projects (Page 328)

9.4.2 Opening and closing a reference project

Opening a reference project

To open a reference project, follow these steps:

1. In the "Reference projects" palette of the project tree, click on "Open reference project" in the toolbar.
   The "Open reference project" dialog box opens.
2. Navigate to the desired project folder, and open the project file. TIA Portal V13.x projects have the extension ".ap13". Older projects of the TIA Portal have the extension ".ap[version number]".
3. Click "Open".
   The selected project is opened as a read-only reference project.

Closing a reference project

To close a reference project, follow these steps:

1. In the "Reference projects" palette of the project tree, select the reference project you want to close.
2. Click on "Close reference project" in the toolbar.
   The selected reference project is closed.

See also

Basics of reference projects (Page 390)

Comparing reference projects (Page 391)

Reference projects (Page 328)

9.4.3 Comparing reference projects

Introduction

You can compare devices from reference projects with devices from both the current project as well as from the same or another reference project or from a library.

Note

Please note the following:

- You cannot specify actions for the comparison objects, since the reference projects are write-protected.
- You can perform a detailed comparison for the comparison objects, if the type of comparison object generally allows a detailed comparison.
- When comparing reference projects, you can always switch between automatic and manual comparison.

Procedure

To compare the objects of a reference project to the device data of the current project, follow these steps:

1. In the project tree, select the device whose data you want to compare to the data of a reference project and which allows offline/offline comparison.
2. Select "Compare > Offline/Offline" from the shortcut menu.
   The compare editor opens with the selected device displayed in the left area.
3. Open the "Reference projects" palette in the project tree.

4. Select the device of a reference project that you want to compare to the device data from the current project.
5. Drag the device from the reference project into the right drop area of the compare editor. You can identify the status of the objects based on the symbols in the status and action area. When you select an object, the object's properties and the corresponding object of the associated device is clearly shown in the properties comparison.
   You can drag a library or other devices from a reference project from the current project into drop areas at any time and thus start a new comparison. It does not matter which device you drag into the drop area.

See also

Basics of reference projects (Page 390)

Reference projects (Page 328)

Opening and closing a reference project (Page 390)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

9.5 Editing project data

9.5.1 Compiling and loading project data

9.5.1.1 Compiling project data

General information on compiling project data

Compiling project data

During compilation, project data is converted so that it can be read by the device. Hardware configuration data and program data can be compiled separately or together. You can compile the project data for one or more target systems at the same time.

The following project data must be compiled prior to loading:

- Hardware project data, for example, configuration data of the devices or networks and connections
- Software project data, for example, program blocks or process screens

Note

While a device is being compiled, no additional compiling process can be started. Note in this regard that you can not only perform a compiling process manually, but you can also trigger it automatically for HMI devices.

Scope of the compilation

When you compile project data, you have the following options depending on the device involved:

- Hardware and software (only changes)
- Hardware (only changes)
- Hardware (rebuild completely)
- Software (only changes)
- Software (rebuild all blocks)
- Software (reset memory reserve)

See also

Compiling project data (Page 393)

Compiling project data

The following section describes the general procedure for compiling project data in the project tree. You will find details of how certain objects are compiled and any special points to note in the online help of the product.

Procedure

To compile project data, follow these steps:

1. In the project tree, select the devices whose project data you want to compile.
2. Select the option you require in "Compile" submenu of the shortcut menu.

   Note

   Note that the options available to you depend on the selected device.

   The project data is compiled. You can check whether or not the compilation was successful in the Inspector window with "Info > Compile".

See also

General information on compiling project data (Page 392)

9.5.1.2 Loading project data

General information on loading

Introduction

In order to set up your automation system, you need to load the project data you generated offline on the connected devices. This project data is generated, for example when configuring hardware, networks, and connections or when programming the user program or when creating recipes.

The first time you download, the entire project data is downloaded. During later loading operations, only changes are downloaded.

You can download the project data to devices and memory cards.

Note

While a device is being compiled, no additional download process can be started. Note in this regard that you can not only perform a compiling process manually, but you can also trigger it automatically for HMI devices.

Possible options for downloading

Depending on the object you want to download, you have the following options:

- Hardware and software (only changes)
  Both the hardware configuration and software are downloaded to the destination if differences exist between the online and offline versions.
- Hardware configuration
  Only the hardware configuration is downloaded to the destination.
- Software (only changes)
  Only the objects that differ online and offline are downloaded to the destination.
- Load PLC program to device and reset
  All the blocks are loaded to the destination and all values are reset to their initial state. Be aware that this also applies to retentive values.

You can also upload project data already contained in a device back to your project. You have the following options:

- Upload entire device as new station
  The hardware configuration of the device and the software on the device is uploaded to the project.
  All relevant data of the device is uploaded to the project.
- Upload software of a device
  Only the blocks and parameters from the device are uploaded to an existing CPU in the project.

In both cases, during loading all instances of library types are connected again with the corresponding version of the type in the project library. If no suitable type is yet available for

a loaded instance or the correct version of the type does not exist in the project library, the type or the version is added to the project library.

Downloading with synchronization

In team engineering, it is possible for several users to work on one project with several engineering systems at the same time and to access one S7-1500 CPU. To ensure consistency within the shared project, it is necessary to synchronize the changed data prior to loading so that nothing gets overwritten unintentionally.

If differences are detected between the online and offline data management within the shared project that were caused by a different engineering system, automatic synchronization of the data to be loaded is offered when loading.

In this case, the "Synchronization" dialog displays the data to be synchronized with the current status (online-offline comparison) and the possible actions.

Use case	Recommendation	Synchronization
One or more blocks on the CPU (online) are more recent than in the engineering system (offline).	These blocks should be uploaded from the CPU to the engineering system before downloading.	Automatic synchronization is possible: The blocks in the engineering system are updated prior to loading.
One or more new blocks have been created and exist only on the CPU (online).	These blocks should be uploaded from the CPU to the engineering system before downloading.	Automatic synchronization is possible: The new blocks are added to the engineering system prior to the download.
One or more blocks on the CPU have been deleted.	The blocks should also be deleted prior to the download in the engineering system.	Automatic synchronization is not possible. The blocks deleted on the CPU should be manually deleted in the offline project in the engineering system.
One or more blocks on the CPU and in the engineering system are different. This is the case when a different user has changed blocks to which you have also made corrections and has already downloaded them to the CPU.	These blocks with competing changes must be adapted manually. You decide in this case which changes you are going to accept. If the blocks on the CPU are to be retained, you should adopt these blocks from the CPU in your engineering system before downloading to the CPU. If the blocks that you have changed are to be applied, you can continue with the download without synchronization.	Automatic synchronization is not possible: The affected blocks on the CPU or in the engineering system must be adapted manually. One of the existing block versions (online or offline) will be overwritten in the process.
There are differences in the hardware configuration on the CPU (online) and in the engineering system (offline).	Differences in the hardware configuration must be adapted manually. You decide in this case which hardware configuration you are going to accept. If the existing hardware configuration on the CPU is to be retained, you should adopt this in your engineering system prior to downloading. If you want to apply the hardware configuration you changed, you can continue downloading without synchronization.	Automatic synchronization is not possible: The hardware configuration must be adapted manually. One of the existing hardware configurations (online or offline) will be overwritten.

If required, you can use the "Force download to device" command to download blocks without synchronization.

See also

Downloading project data to a device (Page 396)

Downloading project data to a memory card (Page 397)

Uploading project data from a device (Page 399)

Downloading project data to a device

The following section describes the general procedure for downloading project data to a device. You will find details of how certain objects are downloaded and any special points to note in the online help of the product.

Requirement

- The project data is consistent.
- Each device to which you want to download is accessible via an online access.

Procedure

To download the project data to the selected devices, follow these steps:

1. Select one or more devices systems in the project tree.
2. Right-click on a selected element.
   The shortcut menu opens.
3. Select the option you require in the shortcut menu of the "Download to device" submenu.

   Note

   Note that the options available to you depend on the selected device.

   When necessary, the project data is compiled.

   – If you had previously established an online connection, the "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for downloading.

   – If you had not previously established an online connection, the "Extended download to device" dialog opens, and you must first select the interfaces via which you want to establish the online connection to the device. You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command.
   See also: Auto-Hotspot

4. Check the messages in the "Load preview" dialog, and select the actions in the "Action" column, if necessary.

Warning

Preventing personal injury and material damage

Performing the proposed actions while the plant is in operation can cause serious bodily injury and property damage in the event of malfunctions or program errors.

Make sure that no dangerous situations can arise before you start the actions.

As soon as loading becomes possible, the "Load" button is enabled.

5. Click the "Load" button.
   The loading operation is performed. If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are required for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
6. Click the "Finish" button.

Result

The selected project data was downloaded to the devices.

See also

General information on loading (Page 394)

Downloading project data to a memory card (Page 397)

Uploading project data from a device (Page 399)

Downloading project data to a memory card

You have the option of downloading project data to a memory card. For CPUs of the S7-300/400 series you can also explicitly download the user program to a memory card inserted in the CPU.

To download project data to a memory card, you have the following options:

- Dragging project data to a memory card
- Writing project data to a memory card
- Downloading user program to a memory card that is inserted in a CPU of the S7-300/400 series

Requirement

A memory card is displayed.

See also: Accessing memory cards (Page 468)

Downloading project data to a memory card

To download project data to a memory card, follow these steps:

1. In the project tree, drag the project data you want to download to the memory card.
   If necessary, the project data is compiled. The "Load preview" dialog then opens. This dialog displays alarms and recommends actions needed for the loading operation.
2. Check the alarms and select the actions in the "Action" column, if necessary.
   As soon as downloading becomes possible, the "Load" button is enabled.
3. Click the "Load" button.
   The loading operation is performed.

Or:

1. In the project tree, select the project data that you want to download.
2. To do this, right-click the selection and select the "Copy" command from the shortcut menu. You can also use the key combination <Ctrl+C>.
3. Right-click the memory card and select the "Paste" command from the shortcut menu. You can also use the key combination <Ctrl+V>.
   If necessary, the project data is compiled. The "Load preview" dialog then opens. This dialog displays alarms and recommends actions needed for the loading operation.
4. Check the alarms and select the actions in the "Action" column, if necessary.
   As soon as downloading becomes possible, the "Load" button is enabled.
5. Click the "Load" button.
   The loading operation is performed.

Or:

1. In the project tree, select the project data that you want to download.
2. Select the "Card Reader/USB memory > Write to memory card" command in the "Project" menu.
   The "Select memory card" dialog opens.
3. Select a memory card that is compatible with the CPU.
   A button with a green check mark is activated at the bottom of the dialog.
4. Click the button with the green check mark.
   If necessary, the project data is compiled. The "Load preview" dialog then opens. This dialog displays alarms and recommends actions needed for the loading operation.
5. Check the alarms and select the actions in the "Action" column, if necessary.
   As soon as downloading becomes possible, the "Load" button is enabled.
6. Click the "Load" button.
   The loading operation is performed.

Downloading the user program to a memory card in the CPU (S7-300/400 only)

To download the user program to a memory card in a CPU of the S7-300/400 series, follow the steps below:

1. Select a CPU of the S7-300/400 series in the project tree.
2. Select the "Download user program to memory card" command in the "Online" menu.
   The "Load preview" dialog opens. This dialog displays alarms and recommends actions needed for the loading operation.
3. Check the alarms and select the actions in the "Action" column, if necessary.
   As soon as downloading becomes possible, the "Load" button is enabled.
4. Click the "Load" button.
   The loading operation is performed and the "Load results" dialog is displayed. This dialog displays alarms and suggests possible actions.
5. Check the alarms and select the actions in the "Action" column, if necessary.
6. Click the "Finish" button.

See also

General information on loading (Page 394)

Downloading project data to a device (Page 396)

Uploading project data from a device (Page 399)

Uploading project data from a device

The following section describes the general procedure for uploading project data from a device. Which project data you can upload from a device depends on the products installed.

You have the following options for uploading project data from a device to your project:

- Upload as new station
  With this option you upload existing project data of a device to your project as a new station.
- Upload project data of a device
  With this option you upload project data from the device to an existing CPU in the project. You will find the project data that can be loaded in the online help of the product.

In both cases, all instances of library types are connected again with the corresponding version of the type in the project library during loading. If no suitable type is yet available for a loaded instance or the correct version of the type does not exist in the project library, the type or the version is added to the project library.

Requirement

- A project is open.
- The hardware configuration and software to be uploaded must be compatible with the TIA Portal. If the data on the device was created with a previous program version or with a different configuration software, please make sure they are compatible.

Uploading as a new station

To upload the complete device to your project, follow these steps:

1. Select the project name in the project tree.
2. Select "Upload device as new station (hardware and software)" in the "Online" menu.
   The "Upload device to PG/PC" dialog opens.
3. Select the type of interface you want to use for the uploading operation in the "Type of the PG/PC interface" drop-down list.
4. Select the interface to be used from the "PG/PC interface" drop-down list.
5. Click the "Configure interface" button to the right of the "PG/PC interface" drop-down list to adapt the settings for the selected interface.
   See also: Auto-Hotspot
6. Display all compatible devices by selecting the relevant option and clicking the "Start search" command. In the accessible devices table, select the device from which you want to upload project data.
7. Click on "Load".
   Depending on the selected device, a dialog appears in which you have to enter additional information, such as the position of the module rack.
   The project data of the device is uploaded to the project. You can edit it offline and then download it to the device again.

Uploading project data of a device

To upload only project data from one device to your project, follow these steps:

1. Establish an online connection to the device from which you want to download the project data.
   See also: Auto-Hotspot
2. Select the device in the project tree.
   The "Upload from device (software)" command in the "Online" menu is enabled.
3. Select the "Upload from device (software)" command in the "Online" menu.
   The "Upload preview" dialog box opens.
4. Check the alarms in the "Upload preview" dialog, and select the necessary actions in the "Action" column.
   As soon as uploading becomes possible, the "Upload from device" button is enabled.
5. Click the "Upload from device" button.
   The loading operation is performed.

See also

General information on loading (Page 394)

Downloading project data to a device (Page 396)

Downloading project data to a memory card (Page 397)

Loading project data from a memory card

You have the following options for uploading project data from a memory card to your project:

- Upload project data of the memory card as a new station
  With this option you upload the project data of a memory card to your project as a new station.
- Upload project data of the memory card to an existing device
  With this option you upload project data of a memory card to an existing device in your project. You will find the project data that can be loaded in the online help of the product.

In both cases, all instances of library types are connected again with the corresponding version of the type in the project library during loading. If no suitable type is yet available for a loaded instance or the correct version of the type does not exist in the project library, the type or the version is added to the project library.

Requirement

- A project is open.
- The memory card is displayed.
  See also: Accessing memory cards (Page 468)
- The hardware configuration and software to be uploaded must be compatible with the TIA Portal. If the data on the memory card was created with a previous program version or with different configuration software, please make sure they are compatible.

Uploading project data as a new station

To upload project data from a memory card to your project, follow these steps:

1. In the project tree, select the project data you want to upload.
2. Select "Upload device as new station (hardware and software)" in the "Online" menu.

Or:

1. In the project tree, drag the memory card folder to the project.

Or:

1. Right-click on the memory card.
2. Select "Copy" in the shortcut menu.
3. Right-click on the project.
4. Select the "Paste" command in the shortcut menu.

Uploading project data to an existing device

To upload the project data of a memory card to an existing device, follow these steps:

1. In the project tree, drag the folder of the memory card to a device in the project or copy the memory card and paste the data into a device.
   The "Upload preview" dialog box opens.
2. Check the alarms in the "Upload preview" dialog, and select the necessary actions in the "Action" column.
   As soon as uploading becomes possible, the "Upload from device" button is enabled.
3. Click the "Upload from device" button.
   The loading operation is performed.

9.5.2 Comparing project data

9.5.2.1 Basics of project data comparison

Function

You can compare project data of the same type in order to determine possible differences. In principle, the following comparison methods are available:

- Online/offline comparison
  With this comparison method you can compare the software of objects of a device with the objects of a project. This is only possible when you establish an online connection to the device.
- Offline/offline comparison
  With this comparison method you can either compare the software or the hardware. When you compare the software, you can compare objects from projects or libraries. The hardware comparison is available for devices from the currently open project or from reference projects. You can decide for the software as well as the hardware comparison whether the comparison should be performed automatically for all objects or whether you want to compare individual objects manually.
- Detailed comparison
  For some objects, for example, blocks, you can also perform a detailed comparison in addition to the online/offline and offline/offline comparison. This involves opening the blocks to be compared beside each other and highlighting the differences.

A simple online/offline comparison is performed as soon as you establish an online connection. During this process, comparable objects in the project tree are marked with icons that represent the result of the comparison. You can also run a more comprehensive online/offline and offline/

offline comparison in the compare editor. When you compare software, you can also select actions for non-identical objects in the comparison.

Note

- Not all objects allow all types of comparison. Which comparison method you can use for which project data depends on the products installed.
- Compile your user program before you start a comparison or detailed comparison. After each change of the program during a comparison, repeat this step before you update the result of the comparison. This ensures that the comparison shows the current status.

See also

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Running a detailed comparison (Page 412)

9.5.2.2 Carrying out an online/offline comparison

The online/offline comparison lets you compare objects of a device with objects of a project.

Requirement

The project tree is open.

Procedure

To perform an online/offline comparison, follow these steps:

1. Select a device in the project tree that allows online/offline comparison.
2. Select the "Compare > Offline/online" command in the shortcut menu.
3. If you have not already established an online connection to this device, the "Go online" dialog opens. In this case, set all the necessary parameters for the connection and click "Connect".
   The online connection is established and compare editor opens.

Result

All objects that exist online and offline are displayed. The symbols in the compare editor and in the project tree show you the status of the objects. In the compare editor, you can now define certain actions for the objects, depending on their status.

See also

Basics of project data comparison (Page 402)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Running a detailed comparison (Page 412)

9.5.2.3 Carrying out offline/offline comparisons

With offline/offline comparison you have the option to compare the project data of two devices. You can carry out a software as well as a hardware comparison. When you compare the software, you can compare objects from projects or libraries. The hardware comparison is available for devices from the currently open project or from reference projects. You can decide whether the comparison should be performed automatically for all objects or whether you want to compare individual objects manually.

You can drag any other device to the drop area at any time to perform further comparisons.

Requirement

The project tree is open.

Procedure

To perform an offline/offline comparison, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane.
   All existing objects of the selected devices are displayed depending on the settings of the compare editor in the "Software" tab and an automatic comparison is carried out. You can identify the status of the objects based on the symbols in the compare editor. You can define certain actions depending on the status of the objects. You can select an object in order to additionally display the property comparison for the object.
4. If you want to carry out a manual comparison, click on the button for switching between automatic and manual comparison in the status and action area. Then select the objects that you want to compare.
   The properties comparison is displayed. You can identify the status of the objects based on the symbols. You can define certain actions depending on the status of the objects.
5. If you want to carry out a hardware comparison, open the "Hardware" tab. You can once again carry out a manual comparison, if necessary. But you cannot specify any action.

See also

Basics of project data comparison (Page 402)

Carrying out an online/offline comparison (Page 403)

Using the compare editor (Page 405)
Running a detailed comparison (Page 412)

9.5.2.4 Using the compare editor

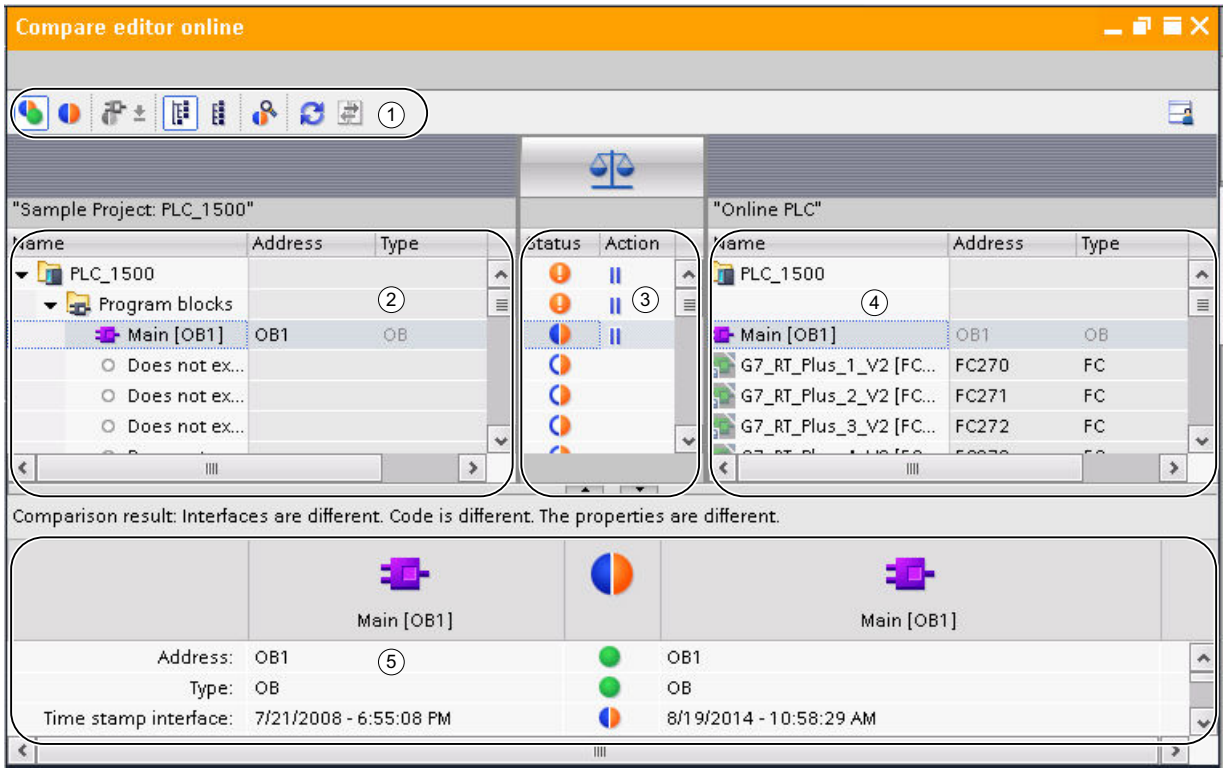
Overview of the compare editor

Function

The compare editor gives an overview of the results of a comparison in a table. The appearance varies slightly depending on whether it is an online/offline comparison or a hardware/software comparison.

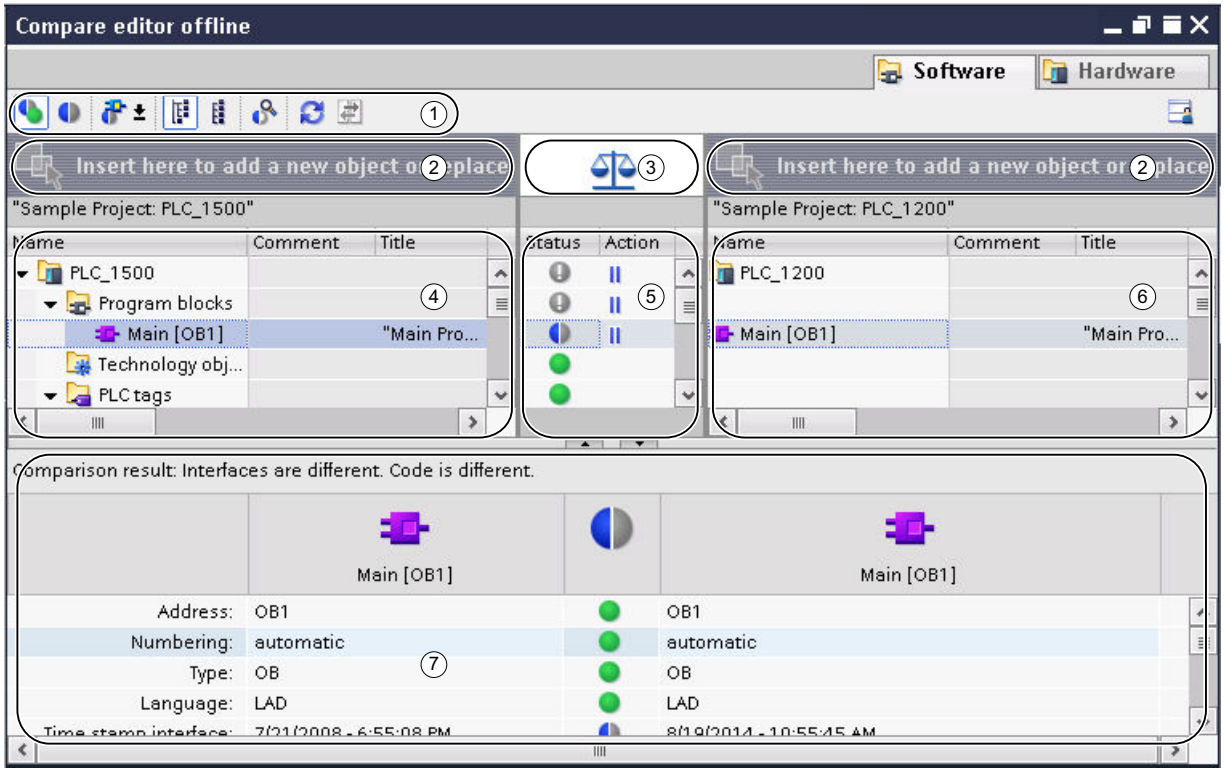
Layout of the compare editor

The following figure shows the layout of the compare editor for an online/offline comparison:



① Compare editor toolbar
③ Left comparison table
③ Status and action area
④ Right comparison table
⑤ Property comparison

The following figure shows the layout of the compare editor for an offline/offline comparison (software):



① Compare editor toolbar

② Drop areas

③ Button to toggle between automatic and manual comparison

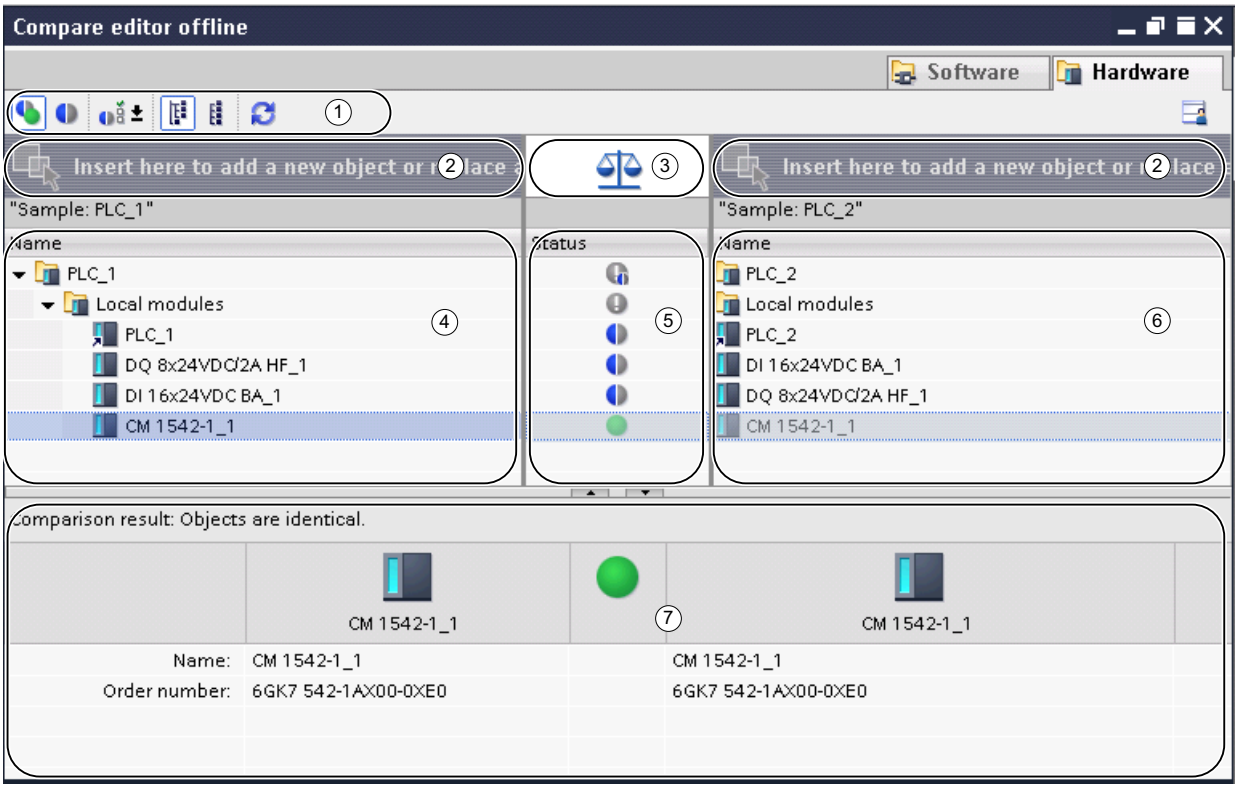
④ Left comparison table

⑤ Status and action area

④ Right comparison table

⑦ Property comparison

The following figure shows the layout of the compare editor for an offline/offline comparison (hardware):



① Compare editor toolbar

② Drop areas

③ Button to toggle between automatic and manual comparison

④ Left comparison table

⑤ Status area

④ Right comparison table

⑦ Property comparison

Compare editor toolbar

With the toolbar, you can access the following compare editor functions:

- Show identical and different objects
  You can show identical objects if you want to view the comparison in full.
- Show only objects with differences
  You can hide identical objects to make the comparison easier to follow.
- Show additional filters available (only online/offline comparison and offline/offline comparison for software)
  You can define which objects are to be compared.
- Display available assignment criteria (only offline/offline comparison for hardware)
  You can specify the criterion according to which the modules are to be assigned to each other for the comparison.

- Changing the view
  You can choose between a hierarchical and a flat view. In the hierarchical view, the devices are shown in their structure; in the flat view, the objects of the devices are listed without structure.
- Start detail comparison (only online/offline comparison and offline/offline comparison for software)
  You can start a detailed comparison for objects to show the individual differences. This function is, however, not available for every object.
- Refresh the view
  After you have modified objects, you can update the comparison results using this function.
- Execute actions (only online/offline comparison and offline/offline comparison for software)
  You can synchronize non-identical objects using specific actions.

Drop areas

In the case of an offline/offline comparison, you can drag the devices you want to compare into the drop areas. In the case of a software comparison, the devices to be compared can originate from the opened project, from reference projects, from the project library or from global libraries. However, note that you can only drop complete libraries into the right drop area. In the case of a hardware comparison, you can compare devices from the opened project or from reference projects.

Button to toggle between automatic and manual comparison

With an offline/offline comparison, you can switch between automatic and manual comparison. In the case of automatic comparison, the objects to be compared are assigned automatically to each other. In the case of a manual comparison, you can select the objects that are to be compared.

Comparison tables

Comparison tables show the objects of the devices being compared to one another.

The following table shows the meaning of the columns of the comparison table:

Column	Description
Name	Name of the compare object
Comment	Comment on the compare object
Title	Title of the compare object
Address	Address of the compare object
Numbering	Type of numbering for the comparison object
Type	Type of compare object
Language	Programming language set for the compare object.
Time stamp interface	Time of the last modification to the block interface
Time stamp code	Time of the last modification to the source code
Author	Name of the author of the compare object
Version	Version of the compare object

Column	Description
Family	Name of the object family
Load memory	Memory usage of the load memory of the compare object
Work memory	Memory usage of the work memory of the compare object
Modified on	Time of last modification
Optimized block access	Indicates whether "Optimized block access" is enabled for a block.
Signature	Signature of the compare object (SIMATIC Safety)
Interface signature	Signature of the block interface of the compare object (SIMATIC Safety)

Not all columns are available in every comparison type. For the hardware comparison, for example, the comparison tables contain only the "Name" column.

Not all columns are shown in the default setting. However, as in all table editors, you can show or hide the columns as required and sort according to individual columns.

Status and action area

The status and action area offers the following options:

- You can view the results of automatic comparison. The results are displayed with symbols.
- In an online/offline comparison and an offline/offline comparison for software, you can specify actions for non-identical objects.

Status and action symbols

The following table shows the comparison results symbols for an online/offline comparison:

Symbol	Description
	Folder contains objects whose online and offline versions differ
	Comparison results are not known
	Online and offline versions of the object are identical
	Online and offline versions of the object are different
	Object only exists offline
	Object only exists online

The following table shows the comparison results symbols for an offline/offline comparison:

Symbol	Description
	Reference program
	Version compared
	Folder contains objects of which the versions compared differ

Symbol	Description
	Results of the offline/offline comparison are not known
	The versions of the object compared are identical
	The versions of the object compared differ
	Object only exists in the reference program
	Object only exists in the version compared
	Hardware comparison only: Although the lower-level objects of the container are identical, there are differences between the containers themselves. Such a container may be a rack, for example.
	Hardware comparison only: The lower-level objects of the container are different. There are also differences between the containers. Such a container may be a rack, for example.

The following table shows the symbols for possible actions in a software comparison:

Symbol	Description
	No action
	Overwrite the object of the compared version with the object from the reference program
	Overwrite the object of the output program with the object from the compared version
	Different actions for the compare objects in the folder

Property comparison

The property comparison compares the properties of the selected compare objects. The result is displayed with symbols. Only the property comparison is made with a manual comparison so that the status and action area remains empty. With automatic comparison, you can perform the property comparison in addition to the comparison in the comparison tables.

See also

Basics of project data comparison (Page 402)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Changing the view (Page 416)

Show and hide table columns (Page 411)

Filtering the compare editor view (Page 411)

Updating the comparison results (Page 413)

Synchronizing non-identical objects (Page 414)

Show and hide table columns

In a software comparison, you can show or hide the columns of comparison tables as required.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.
5. In order to show or hide multiple columns, click "More" and select or clear the check boxes of the corresponding columns in the "Show/Hide" dialog.

Result

The columns in both the left and the right comparison table are shown or hidden.

See also

Basics of project data comparison (Page 402)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Overview of the compare editor (Page 405)

Filtering the compare editor view (Page 411)

Running a detailed comparison (Page 412)

Updating the comparison results (Page 413)

Synchronizing non-identical objects (Page 414)

Changing the view (Page 416)

Filtering the compare editor view

You can improve the clarity of the compare editor using the following filters:

- Hiding identical comparison objects
  You can hide comparison objects which have identical online/offline or offline/offline versions. Any such comparison objects you have hidden can also be shown again at any time.
- Displayed objects
  In an online/offline comparison or an offline/offline comparison for software, you can specify the objects for which the comparison results are to be shown.

Requirement

The compare editor is open.

Hiding identical comparison objects

To hide identical objects, follow these steps:

1. Click on the "Show only objects with differences" button in the toolbar.
   Only the elements that differ online and offline are displayed.

Showing identical comparison objects

To show identical objects again, follow these steps:

1. Click on the "Show identical and different objects" button in the toolbar.
   All elements will be displayed.

Selecting displayed objects

To select the objects for which comparison results should be displayed, follow these steps:

1. Perform an online/offline or an offline/offline comparison for software.
2. Click the arrow of the button "Show additional available filters" in the toolbar.
3. Select the required filter.

See also

Basics of project data comparison (Page 402)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Overview of the compare editor (Page 405)

Changing the view (Page 416)

Show and hide table columns (Page 411)

Running a detailed comparison (Page 412)

Updating the comparison results (Page 413)

Synchronizing non-identical objects (Page 414)

Running a detailed comparison

Note

Not all objects allow a detailed comparison. The project data for which you can perform a detailed comparison depends on the products installed. A detailed comparison of the hardware components is not available for hardware comparison.

Procedure

Proceed as follows to perform a detailed comparison:

1. First, perform an online/offline or an offline/offline comparison for software.
   The compare editor opens.

   Note

   You can only perform a detailed comparison for objects that are listed in the left as well as the right comparison table.

2. In the compare editor, select the object for which you want to perform a detailed comparison.
3. Click the "Start detailed comparison" button in the toolbar.

See also

Basics of project data comparison (Page 402)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Overview of the compare editor (Page 405)

Show and hide table columns (Page 411)

Changing the view (Page 416)

Filtering the compare editor view (Page 411)

Updating the comparison results (Page 413)

Synchronizing non-identical objects (Page 414)

Updating the comparison results

As soon as you change an object, the comparison results are no longer valid and must be updated.

Note

For online/offline comparisons, you should note that changes in the device may result in the system automatically updating the comparison editor if objects in the comparison are affected by the change. This can have the following results:

- Some of the actions you have defined may become invalid, for example if the device no longer contains the object in question. Objects with such invalid actions will be highlighted so you can define new, valid actions.
- The selection you made before the automatic update may also be cancelled.

Requirement

The comparison editor is open.

Procedure

To update the comparison results, follow these steps:

1. Click the "Refresh view" button in the toolbar.
   The comparison results are updated.

Note

Please note that the "Refresh view" button will not be available while the comparison editor is loading or synchronizing content.

See also

Basics of project data comparison (Page 402)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Overview of the compare editor (Page 405)

Show and hide table columns (Page 411)

Changing the view (Page 416)

Filtering the compare editor view (Page 411)

Running a detailed comparison (Page 412)

Synchronizing non-identical objects (Page 414)

Synchronizing non-identical objects

Specifying actions

If you have performed a comparison, you can specify the actions to be performed for non-identical objects in the compare editor. You cannot select any actions for identical objects. Note that you cannot execute any actions during hardware comparison.

In the case of an online/offline comparison, only synchronization actions in one direction are permitted, in order to retain program consistency. Thus, for example, you can load multiple blocks to a device or from a device, but you cannot perform a combination of loading actions in one synchronization action. In this case, the first action you set in the compare editor determines the synchronization direction. For example, if you specify for a block that the offline block is to be loaded to the device, then the other objects can also only be loaded to the device via a synchronization action. To load objects from the device again, first select the "No action"

option. You can then specify the action settings again as required. Or, you can perform a new comparison.

Note

Please note the following CPU-specific aspects when defining actions:

- S7-300/400:
  - You can define actions for the "Program blocks" folder, for folders you have created yourself or for individual blocks.
  - Neither SCL nor GRAPH blocks can be loaded from the device to the offline project.
- S7-1200/1500:
  - You can define actions for the "Program blocks" folder, for folders you have created yourself or for individual blocks. If you have performed an online/offline comparison and select download to the device as action, a consistent download is executed. If you upload the object from the device to the project, however, you can also upload individual blocks.
  - SCL blocks cannot be loaded from the device to the offline project.

Requirement

The compare editor is open.

Procedure

To select an action for a non-identical object, follow these steps:

1. In the status and action area, click in the "Action" column on the cell of the object for which you want to define an action.
   The cell changes to a drop-down list.
2. Click on the drop-down list.
3. Select the action you want.
   The action set will be carried out for the object in question the next time synchronization is performed.
   If you have accidentally changed the action you had selected, you can undo the change before the next synchronization.
4. To restore the previously set action selection, right-click the action in the status and action area that you want to restore.
5. Select the "Restore last selection" command in the shortcut menu.

See also

Basics of project data comparison (Page 402)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Overview of the compare editor (Page 405)

Show and hide table columns (Page 411)

Filtering the compare editor view (Page 411)

Updating the comparison results (Page 413)

Synchronizing objects (Page 416)

Synchronizing objects

Synchronization executes the actions you have specified for non-identical objects. Note, however, that in the case of an online/offline comparison you can only perform actions in one direction in one synchronization action.

Requirement

- The compare editor is open.
- The desired actions have been selected.

Procedure

To synchronize objects, follow these steps:

1. Click the "Execute actions" button in the toolbar.

Result

The actions you specified for the objects are performed.

See also

Basics of project data comparison (Page 402)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Overview of the compare editor (Page 405)

Show and hide table columns (Page 411)

Filtering the compare editor view (Page 411)

Updating the comparison results (Page 413)

Specifying actions (Page 414)

Changing the view

You can choose between a hierarchical and a flat view for the left comparison table. In the hierarchical view, the devices are shown in their structure; in the flat view, the objects of the devices are listed without structure. In the right comparison table, the objects are always displayed flat.

Setting the hierarchical view

To set the hierarchical view, follow these steps:

1. Click the "Display in hierarchical view" button in the toolbar of the compare editor.

Setting the flat view

To set the flat view, follow these steps:

1. Click the "Display in flat view" button in the toolbar of the compare editor.

See also

Basics of project data comparison (Page 402)
Carrying out an online/offline comparison (Page 403)
Carrying out offline/offline comparisons (Page 404)
Overview of the compare editor (Page 405)
Show and hide table columns (Page 411)
Filtering the compare editor view (Page 411)
Running a detailed comparison (Page 412)
Updating the comparison results (Page 413)
Synchronizing non-identical objects (Page 414)

9.5.3 Protecting project data

9.5.3.1 Protection concept for project data

Introduction

You can protect your project data from unauthorized access. These include, for example:

- Access protection for devices
- Copy and display protection of objects
- Restrictions for printouts of know-how-protected objects

For objects with know-how protection, this protection is also retained after the object is pasted into a library. Note that every protection mechanism is not available for all objects. How to protect specific objects is described in the online help of the product.

Revoking access rights for devices

If you want to execute a function that is password-protected by means of the device protection level, you are prompted to enter a password. When the password is entered correctly, you can execute the required function. The access right is retained on the device until you close the TIA Portal.

If you want to reactivate password protection while the TIA Portal is open, you can explicitly revoke the access rights for a device. As a result, certain functions for the protected device cannot be executed until the correct password is entered again. You specify the functions for which a password must be entered when you assign the device protection level.

See also

Printing project data (Page 436)

9.5.3.2 Revoking access rights for devices

Requirement

- A protection level has been set for the device.
- A protected function for the device has been enabled by entering the password.

Procedure

To revoke the access rights for the device, follow these steps:

1. Select the device for which you want to revoke access rights in the project tree.
2. Select the "Delete access rights" command in the "Online" menu.

Result

The access rights are revoked, and starting from now the user will be prompted to enter the password again to execute a password-protected function on the device. The function can only be executed if the correct password is entered.

If the device has an online connection, it will be disconnected.

See also

Protection concept for project data (Page 417)

9.5.4 Printing project contents

9.5.4.1 Printing project documentation

Documentation settings

Introduction

Once a project is created, the contents can be printed in an easy-to-read format. You may print the entire project or individual objects within the project. A well-structured printout is helpful when editing the project or performing service work. The printout can also be used for your customer presentations or as full system documentation.

You can prepare the project in the form of standardized circuit manuals and print it in a uniform layout. You can limit the scope of the printout. You have the option to print to the entire project, individual objects along with their properties, or a compact overview of the project. In addition, you can print the contents of an open editor.

Improving the printout with frames and cover pages

You can design the appearance of the printed pages according to your own requirements, for example, to add your own company logo or the corporate design of your company in the project documentation. You can create any number of design variants as frames and cover pages. The frames and cover pages are stored in the project tree under the item "Documentation settings" and are part of the project. You can insert placeholders for data from previously entered document information within the frames and cover pages. These will be filled automatically with the appropriate metadata during printing.

If you want to avoid designing your own template, there are ready-made frames and covers pages available. These include templates complying with the ISO standard for technical documentation.

Modular structure of a printout

An printout generally consists of the following components:

- Cover page (only when printing from the project tree)
- Table of contents (only when printing from the project tree)
- Name and path of an object within the project tree
- Object data

Printout of the cover page or the table of contents can be deactivated in the "Print" dialog.

See also

Creating frames (Page 425)

Creating a cover page (Page 425)

Editing cover pages and frames (Page 427)

Entering document information (Page 423)

Print function for module labels (Page 438)

Printout of project contents

Availability of print function

The following contents can be printed:

- An entire project in the project tree
- One or more project-related objects in the project tree
- Contents of an editor
- Tables
- Libraries
- Diagnostics view of the Inspector window

It is not possible to print in the following areas:

- Portal view
- Detailed view
- Overview window
- Compare editor
- All tabs of the Inspector window, except the diagnostics view
- All task cards, except the libraries
- Most of the dialogs
- Properties and devices of the programming device/PC not related to the project, for example online portals and connected card readers.

Scope of printout

To be able to print, at least one printable element has to be selected.

If a selected object is printed, all subordinate objects are also printed. For example, if a device is selected in the project tree, all of its data is also printed. If you select the entire project in the project tree for printing, all project contents are printed with the exception of the graphical views. These have to be printed separately. Items in the project tree that are not part of the project cannot be printed. For example, this includes online portals and connected card readers and USB memory devices.

When table contents are printed, all lines in the table in which a cell is selected are printed. In order to print one or more table columns, the desired columns must be selected. If no individual cells or columns are selected, the entire table is printed.

Limitations when printing

In general, it is possible to print all objects that can be visualized on the user interface. Conversely, this means that you cannot print objects that you do not have access to. If a printout fails, possible reasons may include the following:

- A valid license does not exist for displaying an object.
- There is no device description for an object.
- A software component needed to display an object is not installed.

See also

Printing project data (Page 436)

Changing the print settings

Changing the print settings

You can specify general print settings that are retained even after the TIA Portal is closed and re-opened. Some settings are dependent on the products installed. The following settings are possible in every case:

Always print table data as pairs of values

If this option is selected, tables are not printed in tabular format but rather as a pairs of key and value.

Example:

Object name	Property 1	Property 2
Object A	Value A1	Value A2
Object B	Value B1	Value B2

In this case, the printout has the following appearance:

Object A

Property 1: Value A1

Property 2: Value A2

Object B

Property 1: Value B1

Property 2: Value B2

Printing mask editors

- Always print data in tables
  All parameters of technology objects are printed in tabular format.
- Print mask graphics if possible
  If the utilized editor supports this function, the contents of the editor are not printed as a table but rather as a complete graphic as it appears on the screen.

Procedure

To change the print settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group.
3. Select the desired default settings in the "Print settings" area.
   The changes are applied immediately and are retained for all projects, even after the TIA Portal is closed.

See also

Overview of the print settings (Page 305)

Specifying the print layout

Specifying the print layout

If you do not want to rely on ready-made print templates, you can specify your own cover page or your own layout for the individual pages. Your designs are saved together with the respective project.

Your designs for the cover page and your templates for the page layout can be found in the project tree under the "Documentation information" group. You will also find metadata on the project there under the entry "Document information". For subsequent print operations, you can customize the appearance of the printout in the "Print" dialog using the saved cover pages and page layout templates and the available metadata.

Designing the cover page

The cover page can be customized. You can insert a background graphic and provide placeholders for text on the page. The placeholders are automatically filled with data from a documentation information during printing.

Cover pages are located in the project tree under the "Documentation information > Cover pages" group.

Designing the content page

The regular pages of a printout can contain the following elements:

- Frame with static content, such as a company logo
- Placeholders for text, such as the name of the project, the page number, and the time the printout was started
  Several different values for the individual placeholders can be specified in the document information. Other values, such as the project name, are preassigned and are inserted automatically during printing.
- Footnote
  The footnote is always output below the content area.
- Content area
  You can specify an area where the printed content is to be embedded.

The design of the content pages is saved in Frames. The individual frames are located in the project tree under the "Documentation information > Frames" group.

Entering document information

You can enter metadata in the document information for every project. In addition, a print frame and a cover page are specified in the document information. You can create different information, if required, to enable you to quickly switch between different document information containing different information, frames, cover pages, page sizes, and page orientations when printing. For example, this is useful if you want to generate printouts in different languages and different document information is provided for each language.

In the documentation editor, you can specify placeholders on the cover page or in the frame of the regular pages. These placeholders can be automatically replaced with metadata from the documentation information during printing.

The various document information are therefore part of the printing function and specify the print layout and print content.

Procedure

To add metadata, follow these steps:

1. To create new document information, double-click "Add new document information" under "Documentation information > Document information" in the project tree.
   The new document information is created and opened immediately.
2. Enter a name for the set in the "Name" field.
3. Fill in the individual fields with the metadata for the project.

Managing cover pages and frames

Using cover pages and frames

Uses for cover pages

You can give your plant documentation printouts a professional appearance by adding a cover page. You can design your own cover page or use ready-made cover pages. Ready-made cover pages can be adapted and stored again as a template.

Cover pages can be saved in global libraries where they are available for use across projects.

Cover pages are designed for use as a right printed page only.

Uses of frames

You can embed the regular pages of your plant documentation inside a consistently uniform page frame. The frame can contain placeholders for project metadata, which is stored in the document information. It can also contain graphic elements that you design yourself.

You can create your own frames or rely on ready-made page frames. You can adapt a ready-made page frame and then store it again as a new frame.

Like cover pages, frames can be saved in global libraries where they are available for use across projects.

Frames are designed for use on right printed pages only.

Cover pages and templates in the project tree

Cover pages and frames associated with the project are stored in the project tree under the entry "Documentation information". There are separate folders here for frames and cover pages.

The following actions are available in the project tree for cover pages and frames.

- Creating your own subfolders
- Copying and pasting
- Inserting cover pages and frames from the "Documentation templates" system library
- Copying cover pages and templates to a global library

Cover pages and templates in libraries

The "Documentation templates" system library contains a few cover pages and templates that are available in every project. The cover pages and templates can be moved from there to the project tree using a drag-and-drop operation. You can then adapt the cover pages and templates in the project tree according to the requirements of your project.

Cover pages and templates can be moved from the project tree to a global library. Afterwards, these are available in every project.

See also

Library basics (Page 470)

Overview of the "Libraries" task card (Page 472)

Designing cover pages and frames (Page 427)

Using ready-made frames and cover pages (Page 426)

Creating frames

You can create any number of frames for each project. The frames are stored in the project tree below the "Documentation information > Frames" group. You can assign a frame to all document information. When you select document information for printing, its associated frame is used.

Procedure

To create a new frame, follow these steps:

1. Double-click the entry "Add new frame" below the "Documentation information > Frames" group in the project tree.
   The "Creating frames" dialog opens.
2. Enter a name for the frame in the "Name" field.
3. Choose the paper size from the "Paper type" drop-down list.
4. Choose whether the page is to be created in portrait or landscape format in the "Orientation" drop-down list.

Click the "Add" button.

Result

A new frame is created. The frame is then opened automatically in the documentation editor where it can be edited.

See also

Editing cover pages and frames (Page 427)

Creating a cover page (Page 425)

Creating a cover page

You can create any number of cover pages for the printout for each project. The cover pages are stored in the project tree below the the "Documentation information > Cover pages" group. You can assign a cover page to all document information. When you select specific document information for printing, its associated cover page is used.

Procedure

To create a new cover page, follow these steps:

1. Double-click the entry "Add new cover page" below the "Documentation information > Cover pages" group in the project tree.
   The "Add new cover page" dialog box opens.
2. Enter a name for the cover page in the "Name" field.
3. Choose the paper size from the "Paper type" drop-down list.
4. Choose whether the page is to be created in portrait or landscape format in the "Orientation" drop-down list.

Click the "Add" button.

Result

A new cover page is created. The cover page is then opened automatically in the documentation editor where it can be edited.

See also

Editing cover pages and frames (Page 427)

Creating frames (Page 425)

Using ready-made frames and cover pages

The TIA Portal comes with some ready-made frames and cover pages. These can change according to your wishes.

Procedure

To create and edit the ready-made frames and cover pages, follow these steps:

1. Open the "Global libraries" pane in the "Libraries" task card.
2. In the "Templates" folder, open the "Cover Pages" or "Frames" folder.
3. Drag a cover page or a frame from one of the folders into the project tree and drop it into one of the following folders:
   - For frames: "Document information > Frames"
   - For cover pages: "Document information > Cover pages".

   The ready-made frame or cover page can now be used in the project.
4. Double-click on the new entry in the project tree click to edit the frame or the cover page.

See also

Using cover pages and frames (Page 424)

Editing cover pages and frames (Page 427)

Designing cover pages and frames

Editing cover pages and frames

The documentation editor is a graphical editor which allows you to design frames and cover pages for your plant documentation. You can place images or text elements on the frame and the cover pages in the document editor. The text elements are either static or they are automatically filled during printing with the data from the document information that you have selected in the print dialog.

Procedure

To edit a cover page or a frame in the documentation editor, follow these steps:

1. In the project tree, double-click on the entry for an existing cover page or frame under the "Documentation information > Frames " or "Documentation information > Cover pages" group.
   The documentation editor opens.
2. Design the cover page or frame as desired.
3. Close the documentation editor.
   The changes to the cover page or frame are applied automatically.

See also

Creating a cover page (Page 425)

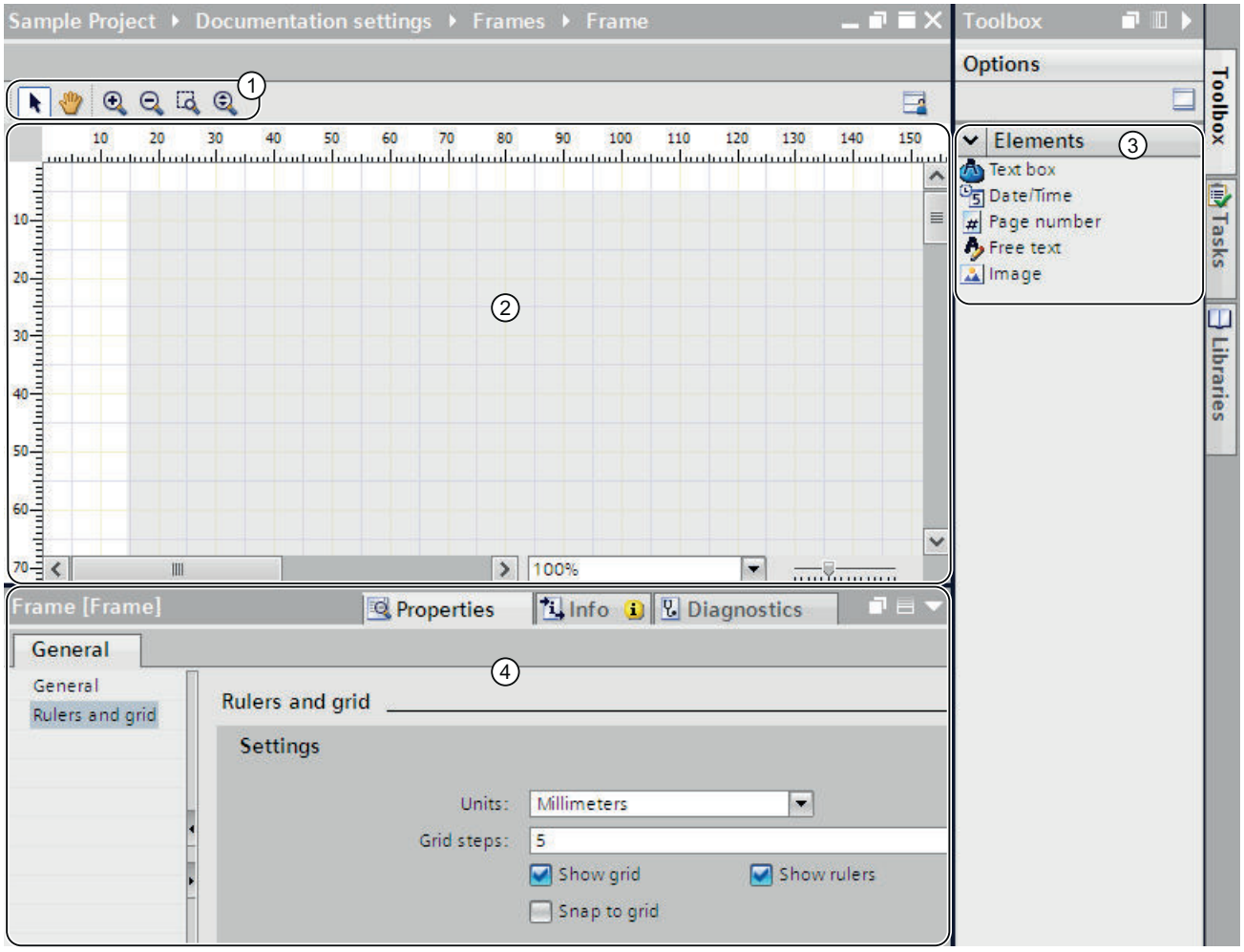
Creating frames (Page 425)

General operation of the documentation editor (Page 428)

General operation of the documentation editor

Components of the documentation editor

The following figure provides an overview of the components of the documentation editor:



① Toolbar

The toolbar provides the following tools (from left to right):

- Arrow tool
  Enables object selection.
- Navigation tool
  Allows shifting of the partial page.
- Zoom-in button
  Magnifies the page display incrementally.
- Zoom-out button
  Reduces the page display incrementally.
- Zoom with selection
  Adapts the page size to a selected work area.
- Dynamic zoom
  Adapts the page width to the work area.

② Work area

You can design the cover page or frame in the work area.

③ "Toolbox" task card

The "Toolbox" task card contains various types of placeholders that you can use on the cover sheet or frame. The placeholders can be placed in the work place using a drag-and-drop operation.

④ Properties in the Inspector window

You can display and modify the properties of the currently selected object in the "Properties" tab of the Inspector window. For example, you can modify the properties of the page, format text, specify the position of objects on the page, etc.

Operation in the documentation editor

The following basic functions are available in the documentation editor:

- Drag-and-drop functionality
  The documentation editor is a graphic editor. which means you can place objects anywhere with the mouse. An image of the page is displayed in the work area. This image corresponds to the ultimate print layout.
  If you want to select objects on the page in order to move them or modify their properties, the arrow tool must be activated in the toolbar.
- Zoom function
  You can use the zoom function to change the size of the page display. You have two options for changing the page size:
  - Via the buttons in the toolbar
    Select the "Zoom in" or "Zoom out" magnifying glass button in the toolbar of the documentation editor. Then click on the page in order to magnify (zoom in) or reduce (zoom out) the page incrementally.
    To zoom in on a particular area, select the "Zoom with selection" tool and use the mouse to drag an outline around the area you want to focus on.
    To continuously zoom in or zoom out of the work area, use the "Dynamic zoom" tool. To magnify the page display, click anywhere on the work area, and then hold down the mouse button while dragging the mouse toward the top of the page. To reduce the page display, drag the mouse toward the bottom of the page.
  - Via the zoom bar
    You can also use the zoom bar (located in the bottom right corner of the work area) to change the display size. Select a percentage value from the drop-down list or enter a percentage value. Alternatively, you control the display size using the slider.
- Navigation over the page
  In addition to scrolling, you the option of changing the partial page with the navigation tool. To change the partial page with the navigation tool, select the Hand button in the toolbar. Then, click anywhere on the page and hold the mouse button down while moving the page to the desired position.

Using and adapting the positioning aids

You have various aids at your disposal to help you position elements on the page:

- Rulers
  Rulers are affixed to the page margins in the work area.
- Page grid
  A grid is placed underneath the page in the work area.

You can display, hide or adapt the positioning aids in the Inspector window under "Properties > Rulers and grid". You can make the following settings:

- Units:
  Specify the unit of measurement for the grid and the rulers.
- Grid steps:
  Specify the width of the grid.
- Show grid:
  Specify whether the grid is to be displayed or hidden.

- Snap to grid:
  Specify whether objects are to be aligned automatically to the grid. If the option is selected, the grid lines function like a "magnet".
- Show rulers:
  Specify whether the rulers are to be displayed.

See also

Editing cover pages and frames (Page 427)

Specifying the print area (Page 431)

Inserting placeholders for metadata (Page 431)

Specifying the print area

An area within the frame is provided for the actual printed contents. The project data is then always inserted inside this defined and uniformly consistent area within the frame. You can adjust the size of the print area.

Requirement

A frame is open in the documentation editor.

Procedure

To define an area for the printed contents, follow these steps:

1. Click on the slightly darker area within the page display in the documentation editor to select the area for the print content.
   This opens the properties of the area to be printed in the Inspector window.
2. Enter the position of the print area on the X and Y axes in the Inspector window.
3. Specify the width and height of the print area in cm in the Inspector window.

Alternatively, you can change the width and position of the print area in the graphic display of the page. To do so, use the mouse to drag the margins of the print area until the desired size and position are achieved.

See also

Creating frames (Page 425)

General operation of the documentation editor (Page 428)

Inserting placeholders for metadata

You can provide placeholders on the cover page and in a frame. The placeholders are automatically filled with metadata from documentation information during printing, if they are placeholders for text. Alternatively, you can add non-modifiable data, such as free text or an image.

All elements are arranged in numbered Z-Orders. If objects overlap, you can determine in which sequence these are to be arranged.

Types of placeholders

The following types of placeholders are available to you:

- Text field
  The text field stands as a placeholder for a text element from a document information. In the properties of the text field, you set which text from a document information should be automatically inserted during printing.
- Field for date and time
  A date and time is inserted instead of the placeholder when printing. This can be the date of creation or the point in time when the last change was made to the project. In the properties of the Inspector window, you specify which date or time is printed.
- Page number
  The correct page number is automatically applied when printing.
- Free text
  You can enter freely selectable text in the properties of the text field. The text is static and is not influenced by the document information selected at the time of printing.
- Image
  Select the image file in the properties of the placeholder in the Inspector window. Images in the formats BMP, JPEG, PNG, EMF or GIF are possible.

Requirement

An cover page or frame is open in the documentation editor.

Procedure

To insert placeholders for metadata on the cover sheet or in a frame, follow these steps:

1. Drag a field from the "Toolbox > Elements" task card to the work area of the documentation editor.
   The placeholder is inserted. The placeholder properties are shown in the Inspector window and can be edited there.
2. Select the metadata to be inserted during printing from the "Text" drop-down list in the Inspector window under "Properties > General > Text box". Alternatively, you have the option of entering free text or selecting an image depending on the type of placeholder.
3. In the Inspector window under "Properties > General > Position and size", specify the position of the placeholder on the X and Y axis and enter the width and height of the text box in cm. You specify the sequence of the objects in the "Z-Order" field, if these overlap. The smaller the value, the further down an object is located.
4. In the Inspector window, go to "Properties > View" and select the font formatting and the orientation of the text as well as the alignment of the text. You cannot make this setting for images.

See also

General operation of the documentation editor (Page 428)

Displaying print preview

Creating a print preview

Creating a print preview

You can create a preview of the printout. Document information can be chosen for this, in the same way as for the actual printout. In this way, you preview the selected frame and, if applicable, the cover sheet. The settings are retained for later printing.

Procedure

To create a print preview and to set the scope of the later printout, follow these steps:

1. Select the "Print preview" command in the "Project" menu.
   The "Print preview" dialog opens.
2. Select the frame layout you want to use for the printout.
   - In the "Document information" drop-down list, select the documentation information you want to use later for the printout.
   - Select the "Print cover page" check box to print the cover page, which is specified in the selected document information.
   - Select the "Print table of contents" check box to add a table of contents to the printout.

   The check boxes for printing the cover page and the table of contents can only be selected if you have started the printout in the project tree.
3. Under "Print objects/area", select what is to be printed. The selection is only possible if you have started the printout from an editor that supports this function.
   - Choose "All" to print out the entire content of the editor.
   - Choose "Selection" to print only the objects currently selected in the editor.

4. Select the print scope under "Properties".
   - Choose "All" to print all configuration data of the selected objects.
   - Choose "Visible" to print the information of an editor that is currently visible on the screen. This option can only be chosen if you have started the printout from an editor that supports this function.
   - Choose "Compact" to print out an abbreviated version of the project data.
5. Click "Preview" to generate the preview.
   A print preview is created in the work area.

Note

Wait time for extensive documents

It can take several minutes to generate the print preview in the case of very extensive projects. You can continue working normally in the meantime on systems with adequate resources. The progress of the print preview is shown in the status bar.

See also

Operation within the print preview (Page 435)

Operation within the print preview

Functions within the print preview

The print preview shows an exact image of the subsequent printout. You can use the buttons in the toolbar to modify the print preview display. The following functions are available (from left to right):

- Navigation mode
  Allows shifting of the partial page.
  To change the partial page with the navigation tool, select the arrow button in the toolbar. Then, click anywhere on the page and hold the mouse button down while moving the page to the desired position.
- Zoom function
  - "Zoom in" and "Zoom out"
    Magnifies or reduces the page display.
    To zoom in or zoom out the display incrementally, select the corresponding button. Then click on the page in order to magnify (zoom in) or reduce (zoom out) the page incrementally.
    To zoom in on a particular area, enable the "Zoom with selection" icon and use the mouse to drag an outline around the area you want to focus on.
    To zoom dynamically through the page, select the button "Zoom in / zoom out dynamically". With pressed mouse button, scroll down over the page to zoom in. Scroll up to zoom out.
  - Percentage value in the drop-down list
    Specifies the display size of the page in percent.
    Enter a percentage value or select a percentage value from the drop-down list. Alternatively, choose the "Fit to page" option from the drop-down list to adapt the page size to the work area. Or, choose "Fit to width" to adapt the page width to the work area.
- "Forward" and "Backward"
  Each change in the partial page, the page count, or the display size is saved in a history in the background. You can use the "Forward" or "Backward" button to return to the previous view or the next view.
- Page navigation
  - "First page"
    Jumps to the first page
  - "Previous page"
    Goes to the previous page.
  - "Page number" input field
    Shows the current page. To jump directly to a page, enter the page number of the page you want to view.
  - "Next page"
    Goes to the next page.
  - "Last page"
    Jumps to the last page.

See also

Creating a print preview (Page 433)

Printing project data

You have two options for printing out project data:

- Print immediately using default settings by means of the "Print" button in the toolbar.
  The button is only active if a printable object is selected.
- Printout with additional setting options with the "Project > Print" menu command.
  For example, you can choose a different printer or specific documentation information or you can specify whether a cover page and table of contents are to be printed. In addition, you can specify the print scope or display a print preview prior to printing.

Requirement

- At least one printer is configured.
- The objects to be printed are not protected.
  The print scope for protected objects is limited. Disable the know-how protection to print the objects in full.

Printing project data

To print out data from the current project or the entire project with additional setting options, follow these steps:

1. Select the entire project in the project tree in order to print out the entire project. To print only individual elements within a project, select them in the project tree.
2. Select the "Print" command in the "Project" menu.
   The "Print" dialog opens.
3. Select the printer in the "Name" box.
4. Click "Advanced" to modify the Windows printer settings.
5. Select the frame layout you want to use for the printout.
   - Select the documentation information in the "Document information" drop-down list.
     The frame stored in the document information is used for the printout. All placeholders within the chosen frame are filled with the metadata from the selected document information.
   - Select the "Print cover page" check box to print the cover page, which is stored in the selected document information.
   - Select the "Print table of contents" check box to add a table of contents to the printout.

   The check boxes for printing the cover page and the table of contents can only be selected if you have started the printout in the project tree.

6. Under "Print objects/area", select what is to be printed. The selection is only possible if you have started the printout from an editor that supports this function.
   - Select "All" to print out the entire content of the editor.
   - Select "Selection" to print only the objects currently selected in the editor.
7. Select the print scope under "Properties".
   - Select "All" to print all configuration data of the selected objects.
   - Select "Visible" to print the information of an editor that is currently visible on the screen. This option can only be chosen if you have started the printout from an editor.
   - Select "Compact" to print out an abbreviated version of the project data.
8. Click "Preview" to generate a print preview in advance.
   A print preview is created in the work area.
9. Click "Print" to start the printout.

Note

Scope of the "Print" dialog

The options available in the "Print" dialog vary depending on the elements to be printed.

Result

The project data is prepared in the background for printing and then printed on the selected printer. The status bar shows the progress of the print operation. You can continue working normally while data is being prepared for printing.

The print results and any errors or warnings are listed in the Inspector window under "Info" at the conclusion of the print job.

Canceling a print job

To cancel an active print job, follow these steps:

1. Click the blue cross in the status bar next to the progress display for the printout.
   The printout is aborted.

See also

Protection concept for project data (Page 417)

Revoking access rights for devices (Page 418)

Printout of project contents (Page 420)

Designing cover pages and frames (Page 427)

9.5.4.2 Printing module labels

Print function for module labels

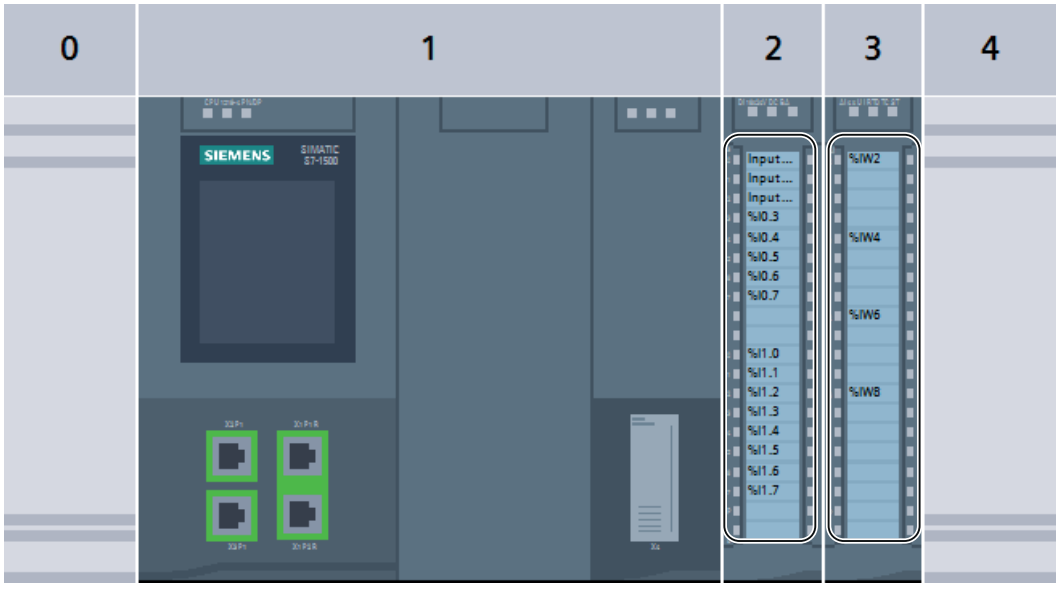
Printing of module labeling strips for hardware modules

You can print labeling strips for the modules in your project with the help of the TIA Portal. The labeling strips are custom-fit to each module and can contain the following printed information:

- Symbolic name of the input or output
- Absolute address of the input or output
- Symbolic name and additionally the absolute address of the input or output. The order of the information can be specified.

The modules are displayed graphically in the device view. If you set the zoom level in the device view to at least 200%, the labels for the individual modules will be visible. The printout on the labeling strip corresponds to the representation of the labeling in the device view.

The following figure shows an example of two modules in the device view on which the labeling of the inputs and outputs is visible:



Export and further editing as Microsoft Word file

The labeling strips are first exported as a Microsoft Word file (.docx) before they are printed. The file can be further edited with commonly available word processing programs such as Microsoft Word. The individual labeling strips are represented as a table in the .docx file.

The character spacing of the text within the table is adapted by default, so that texts are not truncated. If you want to prevent this from stretching or compressing the text too much, change the character spacing of the text in the table cell properties.

Printing the labeling data to an XML file

As an alternative to printing the labeling strips, you can print the addresses of the inputs and outputs of a module to an XML file. You can use the export to an XML file for devices for which no ready-made labeling strips are available. You can also use the export to an XML file to create labeling strips with another program. The program must be able to transform the data of the XML file into an input format that is suitable for the labeling system. The schema of the XML file is available in the section "XML schema of the export file (Page 442)".

Print media

You can print the labeling strips either on ready-made print sheets or on standard DIN A4 paper. You can separate the individual labeling strips from the ready-made print sheets and insert them in the designated labeling areas of your modules. If you print on standard paper, the individual labeling strips must be cut out. Cut marks are automatically included on the printout and serve as aids.

Because the paper feeds of printers differ slightly, the printout may be slightly offset on the paper. When the labeling strips are printed on ready-made sheets, printing that is accurate to the millimeter is important. Otherwise, the text will not be fit exactly inside the stamped area. In addition, if the printing is imprecise, the labeling of an input or output may no longer be congruent with the channel status displays of the module. To ensure precise printing, you can enter an offset value for your printer in the TIA Portal. For information on how to determine the proper offset value for your printer, refer to Chapter "Determining the print area offset (Page 445)".

See also

Exporting labeling data as XML (Page 441)

XML schema of the export file (Page 442)

Printing labels (Page 439)

Determining the print area offset (Page 445)

Documentation settings (Page 419)

Printing labels

You can print labeling strips for the modules in your project if provision has been made for attaching labels to the utilized modules. The labels are first exported to a Microsoft Word file (.docx). A separate .docx file is created for each module family (for example, for all selected S7-1500 modules). The labels are always printed from the word processing program.

Requirement

The following requirements apply to printing of labeling strips:

- The chosen modules must support the printing of labels. Otherwise, the data can only be output to an XML file.
- A word processing program that supports Microsoft Word DOCX files must be installed, e.g., Microsoft Word 2010 or later.
- You need the ready-made labels for your modules or commercially available DIN A4 paper.

Procedure

To print labeling strips for hardware modules, follow these steps:

1. In the project tree, select the modules for which you want to print labeling strips.
   - You can select one or more stations in order to print out labeling strips for all modules plugged into these stations.
   - Alternatively, select the desired modules below the stations in the "Local modules" folder.
2. Right-click one of the devices, and select the "Export labeling strips" command from the shortcut menu.
   The "Export labeling strips" dialog opens.
3. In the "Content of the labeling strip" area, select the data to be printed on the labeling strip:
   - Select "Symbolic name" in order to print the symbolic name of the input or output (corresponds to the contents of the "Name" column in the IO tag table).
   - Select "Absolute address" in order to print the absolute address of the input or output (corresponds to the contents of the "Address" column in the IO tag table).
   - Select "Absolute and symbolic address" or "Symbolic and absolute address" in order to print both addresses. Printing takes place according to the specified order.
4. In the "Export format" area, define how the labeling data will be output.
   - Select "Print on SIEMENS labeling sheet" if you are printing on a ready-made label sheet for your modules.
   - Select "Print on plain DIN A4 page" if you are printing on standard DIN A4 paper.
5. Select correction values for your printer in the "Offset print area", if required. The correction values are used for correct alignment of the print area. Correction values are only necessary if you are printing on ready-made labeling strips.
   - Enter a correction value, in millimeters, in the "Vertical offset" field. A negative value shifts the print area upward. A positive value shifts the print area downward.
   - Enter a correction value, in millimeters, in the "Horizontal offset" field. A negative value shifts the print area to the left. A positive value shifts the print area to the right.
6. In the "Path" field, select the path where the exported files will be stored.
7. Click the "Export" button to start the export.
   The export files are created.
8. Open the DOCX files with a conventional word processing program, such as Microsoft Word, and change the design of the labeling strips if necessary.

9. Print out the labeling strips from your word processing program. Use the paper that you specified in the Export dialog for this.
10. If you are using ready-made sheets, separate the labeling strips at the stamped lines provided for that purpose. When standard DIN A4 paper is used, you must cut out the labeling strips.

See also

Determining the print area offset (Page 445)

Exporting labeling data as XML (Page 441)

Exporting labeling data as XML

The TIA Portal supports a large number of different modules and can be continually expanded using Hardware Support Packages, for example. Ready-made labeling strips are not available for every supported module. However, you can still use the TIA Portal to label inputs and outputs of modules that are not supported. First, you export the absolute and symbolic addresses of the inputs and outputs to a standardized XML file. Then, you import the XML file to an external program for printing the labels. In this program, you prepare the data to suit your modules and print out the labels.

Procedure

To export labeling data for hardware modules as XML, follow these steps:

1. In the project tree or network view, select the modules for which you need labeling strips.
   - You can select one or more stations in order to export the input and output addresses of all modules plugged into these stations.
   - Alternatively, select the desired modules below the stations in the "Local modules" folder.
2. Right-click one of the devices and select the "Export module labeling strips" command from the shortcut menu.
   The "Export labeling strips" dialog opens.
3. In the "Export format" area, select the "Export to XML file" option.
4. In the "Path" field, select the path where the XML file will be stored.
5. Click the "Export" button to start the export to an XML file.
   The XML file is created with the name "<Project name>_IO_Channels.xml".

See also

XML schema of the export file (Page 442)

Print function for module labels (Page 438)

Printing labels (Page 439)

XML schema of the export file

XML schema of an export file

The XML file for module labeling strips is structured according to the following schema:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://tempuri.org/XMLSchema.xsd"
elementFormDefault="qualified"  xmlns:mstns="http://tempuri.org/
XMLSchema.xsd"  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Stations">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Station" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
        <xs:element name="Rack" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
          <xs:element name="Module" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
            <xs:element name="IOChannel" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Address" type="xs:string"></
xs:element>
               <xs:element name ="Tag" type="xs:string"></xs:element>
              </xs:sequence>
            <xs:attribute name="Number" type="xs:int"></xs:attribute>
            </xs:complexType>
            </xs:element>
            </xs:sequence>
            <xs:attribute name="Name"></xs:attribute>
          </xs:complexType>
          </xs:element>
          </xs:sequence>
          <xs:attribute name="Name" type="xs:string"></xs:attribute>
        </xs:complexType>
        </xs:element>
        </xs:sequence>
        <xs:attribute name="Name" type="xs:string"></xs:attribute>
      </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  </xs:element>
</xs:schema>
```

Example of an XML file

The following example shows an XML file that contains the labeling data for an S7-1500 CPU with a digital input module and an analog input module:

```
<?xml version="1.0" encoding="UTF-8"?>
<Stations>
  <!-- The CPU is specified first -->
  <Station Name="S71500/ET200MP-Station_1">
    <Rack Name="Rack_0"> <!-- Name of the rack -->
      <Module Name="Sample S7-1500" /> <!-- Name of the CPU -->
     <Module Name="DI 16x24VDC BA_1"> <!-- Name of the digital input
module -->
       <!-- The individual channels of the digital input module are
specified -->
        <IOChannel Number="0">
          <Address>%I0.0</Address>
          <Tag>Input Value 1</Tag> <!-- Symbolic address of input 0
-->
        </IOChannel>
        <IOChannel Number="1">
          <Address>%I0.1</Address>
          <Tag>Input Value 2</Tag>
        </IOChannel>
        <IOChannel Number="2">
          <Address>%I0.2</Address>
          <Tag>Input Value 3</Tag>
        </IOChannel>
        <!-- All other channels follow -->
      </Module>
      <Module Name="AI 4xU/I/RTD/TC ST_1"> <!-- Name of the analog
input module -->
        <!-- The individual channels of the analog input module are
specified -->
        <IOChannel Number="0">
          <Address>%IW2</Address>
          <Tag> <!-- Symbolic addresses are not specified for the
analog input module. -->
          </Tag>
        </IOChannel>
        <IOChannel Number="1">
          <Address>%IW4</Address>
          <Tag>
          </Tag>
        </IOChannel>
        <IOChannel Number="2">
          <Address>%IW6</Address>
          <Tag>
          </Tag>
        </IOChannel>
        <IOChannel Number="3">
          <Address>%IW8</Address>
          <Tag>
          </Tag>
```

```
        </IOChannel>
      </Module>
      <Module Name="Sample S7-1500" />
      <Module Name="DI 16x24VDC BA_1">
        <IOChannel Number="0">
          <Address>%I0.0</Address>
          <Tag>Input Value 1</Tag>
        </IOChannel>
        <IOChannel Number="1">
          <Address>%I0.1</Address>
          <Tag>Input Value 2</Tag>
        </IOChannel>
        <IOChannel Number="2">
          <Address>%I0.2</Address>
          <Tag>Input Value 3</Tag>
        </IOChannel>
        <!-- All other channels follow -->
      </Module>
      <Module Name="AI 4xU/I/RTD/TC ST_1">
        <IOChannel Number="0">
          <Address>%IW2</Address>
          <Tag>
          </Tag>
        </IOChannel>
        <IOChannel Number="1">
          <Address>%IW4</Address>
          <Tag>
          </Tag>
        </IOChannel>
        <IOChannel Number="2">
          <Address>%IW6</Address>
          <Tag>
          </Tag>
        </IOChannel>
        <IOChannel Number="3">
          <Address>%IW8</Address>
          <Tag>
          </Tag>
        </IOChannel>
      </Module>
    </Rack>
  </Station>
</Stations>
```

See also

Exporting labeling data as XML (Page 441)

Determining the print area offset

If you are using a ready-made label sheet, the printing on it must be applied precisely so that the text is properly oriented on the prestamped labels and will have the proper fit relative to the channel status displays of the module. However, the paper feeds vary slightly from one printer to another. For this reason, you must enter a suitable correction value for your printer in the TIA Portal, if necessary. The print area is then shifted in the exported .docx file in such a way that the printing fits precisely on the ready-made label sheet.

The settings for shifting the print area are stored for the specific Windows user. If you log on to Windows using a different user name, you have to enter the correction values again.

The procedure for determining the correction value for your printer is described below.

Requirement

- You require a ready-made label sheet.
- You must have access to the actual printer that you will use subsequently for the printout. The printer must be made ready for printing on standard DIN A4 paper.

Procedure

To determine the correction value for your printer, follow these steps:

1. Print out a label sheet on standard DIN A4 paper, as described in Chapter "Printing labels (Page 439)".
2. Compare the printout on the DIN A4 paper with the ready-made label sheet.
3. If the print area is offset, you must use correction values.
   - Using a ruler, measure the horizontal offset relative to the ready-made label sheet. This will be entered later in the "Horizontal offset" field of the Export dialog box for the printing. If the print area is offset to the right, a negative correction value must be entered. If the print area is offset to the left , a positive correction value must be entered.
   - Using a ruler, measure the vertical offset relative to the ready-made label sheet. This will be entered later in the "Vertical offset" field of the Export dialog box for the printing. If the print area is offset downward, a negative correction value must be entered. If the print area is offset upward, a positive correction value must be entered.

9.6 Undoing and redoing actions

9.6.1 Basics of undoing and redoing actions

Function

You can undo performed actions at any time. For this purpose, every action you perform is saved in an action stack. When undoing actions, the stack is processed from top to bottom. In other words, if you undo an action that lies further down in the stack, all actions located above it in the stack will also be undone automatically.

You can redo previously undone actions until you execute a new action. Once you execute a new action, it is no longer possible to redo previously undone actions.

Particularities for undoing

There are a few actions that empty the action stack. You cannot undo these actions or the actions performed before these actions. The following actions empty the action stack:

- Saving
- Project management (creating a new project, opening project, closing a project, deleting a project)
- Establish and disconnect an online connection

Keep in mind that the action stack is emptied every time you establish and terminate an online connection. This means that the actions executed offline cannot be undone once the online connection is established. All actions which you execute while online can be undone until you terminate the online connection. Once you have disconnected the online connection, only the offline actions executed afterward can be undone.

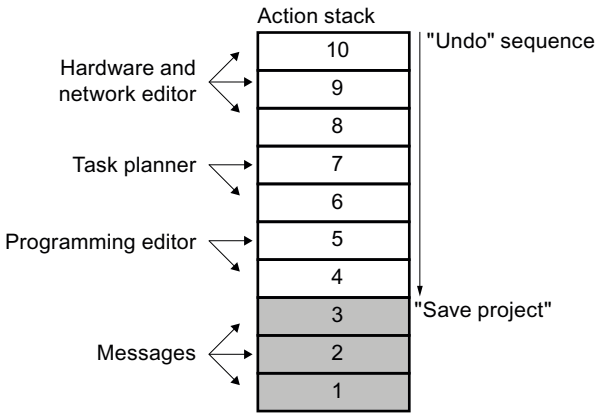
Displaying the action stack

The "Undo" button in the toolbar is enabled as soon as you perform an action that can be undone. This button is split; you can use the arrow down portion to open a drop-down list. containing all actions of the action stack that you can undo. If you had performed actions in an editor other than the currently displayed editor, the corresponding editor is also displayed as a subheading. This allows you to always identify the point at which the undo operation will be applied. The subheadings are removed from the list when the editor responsible can no longer undo actions.

Actions you have undone are entered in the action stack from where they can be redone. Here, you can redo actions you have undone. The display of actions you can redo it is analogous to the display of the actions that you can undo.

Example of undoing actions

The figure below shows how actions performed in various editors and tables are undone:



In this example, you cannot undo actions 1 to 3 because the project was saved. You can undo actions 4 to 10 in the order indicated by the direction of the arrow. In other words, you must undo action 10 first. Once you have undone action 8, you cannot then undo action 5. You must first undo actions 7 and 6. As the final step in the sequence, you can then undo action 4. You also have the option of undoing several actions in a single step by undoing an action located further down in the action stack. All actions located above it in the stack will be undone automatically.

The same principle also applies to redoing of actions.

See also

Undoing an action (Page 447)

Redoing an action (Page 448)

9.6.2 Undoing an action

The following options are available for undoing actions:

- Undoing the last action only
  Only the last action performed is undone.
- Undoing as many actions as required
  Multiple actions in the action stack are undone in a single step.

Undoing the last action only

To undo the last action performed, follow the steps below:

1. Click the "Undo" button in the toolbar.
   - If the action was not performed in the currently displayed editor, a confirmation prompt appears.
   - If the undo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
2. Click "Yes" to confirm.
3. Enter the password, if necessary.
   The editor in which the action was performed is displayed and the action is undone.

Undoing as many actions as required

To undo multiple actions in the action stack in a single step, follow these steps:

1. Click the Down arrow next to the "Undo" button in the toolbar.
   This opens a drop-down list containing all actions you can undo. Actions performed in other editors are identified by the editor name in the subheading.
2. Click the action you want to undo.
   The chosen action and all actions in the stack located above the chosen action are undone. If the undo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
3. Enter the required passwords, if necessary.
   The editors in which the actions were performed are displayed and the actions are undone.

See also

Basics of undoing and redoing actions (Page 446)

Redoing an action (Page 448)

9.6.3 Redoing an action

You have the option of redoing an action that has been undone, so that you can return to the status present before "Undo" was performed. However, this is only possible until you perform a new action. The following options are available for redoing actions:

- Redoing the last undone action only
  Only the last undone action is redone.
- Redoing as many undone actions as required
  Multiple undone actions in the action stack are redone in a single step.

Redoing the last undone action only

To redo the last undone action, follow the steps below:

1. Click the "Redo" button in the toolbar.
   - If the action is not being redone in the currently displayed editor, a confirmation prompt appears.
   - If the redo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
2. Click "Yes" to confirm.
3. Enter the password, if necessary.
   The editor in which the action was undone is displayed and the action is redone.

Redoing as many undone actions as required

To redo multiple undone actions in the action stack in a single step, follow these steps:

1. Click the Down arrow next to the "Redo" button in the toolbar.
   This opens a drop-down list containing all actions you can redo. Actions performed in other editors are identified by the editor name in the subheading.
2. Click the action you want to redo.
   The chosen action and all actions in the stack located above the chosen action are redone. If the redo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
3. Enter the required passwords, if necessary.
   The editors in which the actions were undone are displayed and the actions are redone.

See also

Basics of undoing and redoing actions (Page 446)

Undoing an action (Page 447)

9.7 Find and replace

9.7.1 Basics of searching

Introduction

You can use the following search options within TIA Portal:

- Searching and replacing in the editor
- Browsing the hardware catalog

In addition, individual products in TIA Portal offer more search options.

Searching and replacing in the editor

You can search for texts in the editors. The search function finds all texts containing the search key in the currently opened editor. The results are selected in sequence in the opened editor.

You also have the following options:

- Narrowing down the search with additional options
- Replacing found texts

The additional options and the type of texts for which you can search depend on the installed products and the currently open editor.

Browsing the hardware catalog

You can search for a specific hardware component in the hardware catalog.

See also: Browsing the hardware catalog

See also

Searching and replacing in the editor (Page 452)

9.7.2 Searching and replacing in the editor

9.7.2.1 Basics for search in open editors

Introduction

You can start a search which is limited to the editor currently open in the workspace. You can customize the search as follows to suit requirements:

- You can optimize the search with additional options.
- You can specify the search area.
- You can specify the search direction.

Additional options for searching

You can refine the search with the help of the following additional options:

- Whole words only
  Only whole words are taken into account for the search. Compound words that contain the search key as a part are ignored.
- Match case
  Upper- and lowercase letters are taken into account in the search.
- Find in substructures
  The search also includes texts contained in another object.

- Find in hidden texts
  Texts that are assigned to another text but that are currently hidden are also included in the search.
- Use wildcards
  Enter an asterisk as the wildcard for any number of characters. Example: You want to search for all words starting with "Device". Type in "Device*" in the search key box. Enter a question mark as the wildcard, however, if you only want to exclude a single character.
- Use regular expressions (for searching in scripts only)
  A regular expression is a character string used to describe sets of values and for filtering. This allows you to create complex search patterns.

The additional options available depend on the installed products and the editor opened.

Search area and search direction

You have the following options for define the search area:

- Whole document
  The entire is searched regardless of the current selection and position.
- From current position
  The search starts at the current position. Depending on the search direction, all texts above or to the left or below or to the right of the current position are not included in the search.
- Selection
  The search term is only searched for in the current selection.

You can define the search direction independent of the search area:

- Down
  Search takes place in editor from top to bottom or from left to right.
- Up
  Search takes place in editor from bottom to top or from right to left.

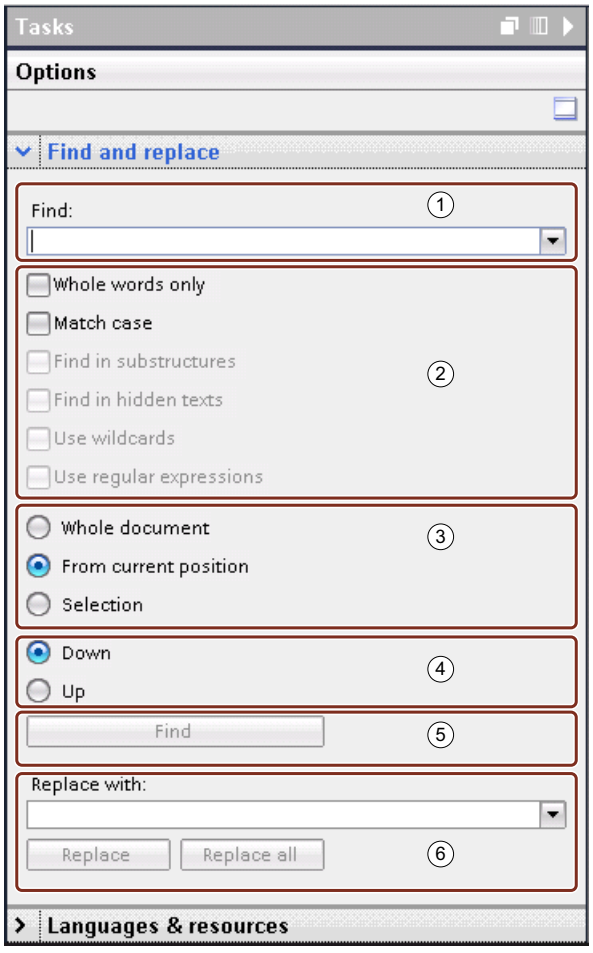
9.7.2.2 Overview of the "Find and replace" palette

Function of the "Find and replace" palette

The "Find and replace" palette in the "Tasks" task card lets you search in an open editor. It includes all options that you need for an efficient search.

Layout of the "Find and replace" palette

The following figure shows the components of the "Find and replace" palette:



① Search field

② Additional options

③ Search area

④ Search direction

⑤ Start search

⑥ Replace

9.7.2.3 Searching and replacing in the editor

You have the option to search for text in an editor or to replace it.

Start search

Follow these steps to start the "Find and replace" function:

1. Select the "Find and replace" command in the "Edit" menu or open the "Find and replace" pane in the "Tasks" task card.
   The "Find and replace" pane opens.
2. Enter a term in the "Find" drop-down list.
   As an alternative, you can select the most recent search key from the drop-down list.
3. Select the options desired for the search.
4. Using the option buttons, select the starting point for the search and the search direction.
5. Click "Find".
   The first hit is marked in the editor.
6. Click "Find" again to display the next hit.
   The next hit is marked in the editor. Repeat this procedure as needed until you reach the last hit.

Replacing the search key

You have the option of replacing hits individually or automatically replacing all the found texts, if the respective editor supports this function. Follow these steps to replace terms:

1. Enter a term in the "Find" drop-down list.
   As an alternative, you can select the most recent search key from the drop-down list.
2. Select the options desired for the search.
3. Click the "Find" button to start a search for the specified search key.
   The first hit is displayed in the editor.
4. In the "Replace" drop-down list, enter the text you wish to use to replace the search key.
   As an alternative, you can select one of the last entered texts from the drop-down list.
5. Click the "Replace" button to replace the selected hit with the specified text.
   The found text is replaced and the next hit is marked in the editor.
   Repeat this procedure until you have replaced all the desired hits. To skip to the next hit without replacing the marked word, click the "Find" button instead of "Replace".
6. Click "Replace all" to automatically replace all hits at once.

See also

Basics of searching (Page 449)

9.8 Working with multi-language projects

9.8.1 Project text basics

Texts in different languages in the project

When you enter texts while working on a project, you would normally do this in your own language. If you then pass on the project to someone else who does not know this language, this person will require a translation of the relevant texts to a language they know. This is why all texts can be translated. In this way, you can ensure that anyone who is subsequently confronted with the texts sees the texts in his/her language of choice.

Project language

Project languages are all languages in which a project will later be used. Based on the editing language, all the texts can be translated to the various project languages. You specify the languages that will be available in the project tree under "Languages & Resources > Project languages".

Editing language

Every project has an editing language. When you enter texts, these are always created in the editing language. You should therefore make sure that the editing language set is the language in which you enter the texts. This avoids problems if you translate the texts later.

The editing language does not depend on the language of the user interface. You could, for example, set English as the user interface language, but use Italian as the editing language. If you enter texts, these will be created in the project language "Italian" in this case, although the user interface of the TIA Portal is displayed in English.

You set the editing language in the project tree under "Languages & Resources > Project languages > Editing language".

Reference language

The reference language is used as a template for translation. The text is displayed in the reference language for each text box in the "Tasks > Languages and resources" task card. You therefore know which text that belongs in a text box, even when no text is entered in the currently selected editing language.

User texts and system texts

For clarification purposes, a distinction is made between user texts and system texts:

- User texts are texts that the user created.
- System texts are texts that are created automatically according to the configuration in the project.

You manage the project texts in the project tree under "Languages & Resources > Project texts".

Examples of multilingual project texts

You can, for example, manage the following project texts in more than one language:

- Block titles and block comments
- Network titles and network comments
- Comments in tables
- Alarm texts
- Operator-relevant texts
- Text lists
- Labels of buttons
- Display names of recipes

Translating texts

The following procedures are available to translate texts.

- Translate all texts used in the project in tabular form
  You can enter the translations for the individual project languages directly in the "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts".
- Specify text assigned to individual objects in the Inspector window
  In the Inspector window, you can translate the texts that are assigned to the currently selected objects. Columns are displayed in a table for all available project languages. You can enter the translations for each text in the columns.
- Translating texts using reference texts
  You can change the editing language for shorter texts. All the text cells are filled again with the default values and can be filled in the current language. As orientation, you can display what you last entered in the box in the reference language. To do this, select the "Tasks" task card and open the "Languages & resources".
- Exporting texts and translating them externally
  With larger volumes of text, you can export the texts to an Office Open XML file and translate them in a conventional table calculation program. You then import the translated list again into the TIA Portal.

Note

Using East Asian project languages

You need Microsoft Windows at least in the Professional version or higher to display East Asian project languages. Microsoft Windows in the Professional version must be installed in the local language. With the "Ultimate" or "Enterprise" versions, it is sufficient to install the appropriate language pack.

See also

Overview of the program settings (Page 302)

Changing the settings (Page 306)

Application examples for multilanguage projects (Page 462)

9.8.2 Select project languages

All the texts can be displayed within a project in the same language that you selected for your software user interface. This means that all project texts must exist in the corresponding language. You can select the available project languages yourself.

Requirement

- You are in the project view.
- A project is open.

Procedure

To select the project languages, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree.
   The elements below this are displayed.
2. Double-click on "Project languages".
   In the work area, you will see a list of languages that you can select.
3. Select the required languages.

Result

All texts can be displayed in the activated languages if there is already a translation for these languages.

9.8.3 Setting the editing language

All the texts in the project are created in the editing language when they are entered. If you change the editing language, all future text input will be stored in the new editing language.

Requirement

- You are in the project view.
- A project is open.

Procedure

To change the editing language, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree. The lower-level elements are displayed.
2. Double-click on "Project languages". The possible settings for the project languages are displayed in the work area.
3. Select the editing language in "General > Editing language".

9.8.4 Translating all project texts in tabular form

You can display and edit all project text used in the currently open project in a list. User and system texts are separated into two different lists for clarity. Both lists contain a column for each project language. Enter the translations of the texts in the respective columns.

Requirement

- You are in the project view.
- You have selected at least one further project language.

Procedure

To translate text in the project-wide list, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree. The elements below this are displayed.
2. Double-click "Project texts". A list with the user texts in the project is displayed in the work area.
3. Click on "System texts" if you you want to edit the list of system texts rather than the user texts.
4. You can improve the clarity of the lists if you have a lot of texts.
   - To group identical texts and to translate them all at once, click the "Switch on/off grouping" button in the toolbar.
   - To hide texts that do not have a translation, click the "Filter for empty texts on/off" button in the toolbar.
   - To further limit the displayed project texts to certain devices, select the devices for which you want to display project texts in the drop-down list.
5. Enter the translation of the project texts in the relevant column.

9.8.5 Translating text associated with individual objects

If you want to edit the text of individual objects, it would be too difficult to locate the matching text in the table with all project texts. For this reason, there is a table in the Inspector window in which only the texts assigned to the currently selected objects are displayed. In the table, you can add missing translations for individual project languages or change existing texts.

Requirement

Enter a text in at least one project language for the texts to be translated.

Procedure

To edit the text of the currently selected object, follow these steps:

1. Select the object whose text you want to edit.
2. Open the "Properties" tab in the Inspector window.
3. Open the lower-level "Texts" tab in the inspector window.
   A table with all the texts that belong to the selected objects is displayed. The table contains one column for the currently selected editing language and the reference language, as well as additional columns for the other project languages.
4. Add or change the entries in the table for each project language.

See also

Application examples for multilanguage projects (Page 462)

9.8.6 Translating texts using reference texts

Introduction

After changing the editing language, all texts are shown in input boxes in the new editing language. If there is not yet a translation available for the newly set language, the input boxes are empty or filled with default values.

If you enter text in an input box, this is saved in the current editing language. Following this, the texts exist in two project languages for this input field, in the previous editing language and in the current editing language. This makes it possible to create texts in several project languages.

You can display existing translations for an input box in other project languages. These serve as a comparison for text input in the current editing language and they are known as the reference language.

Note

The display of reference texts depends on the installed products and is not supported by every editor.

Requirement

There is at least one translation into a different project language for an input field.

Procedure

To display the translation of an input cell in a reference language, follow these steps:

1. In the "Tasks" task card, select the "Languages & Resources" pane.
2. Select a reference language from the "Reference language" drop-down list.

Result

The reference language is preset. If you click in a text box, translations that already exist in other project languages are shown in the "Tasks > Languages & Resources" task card.

9.8.7 Exporting project texts

You can export project texts for translation and then reimport them. The texts are exported to an Office Open XML file with the extension ".xlsx". This can be edited in Microsoft Excel or a number of other spreadsheet programs.

The following export options are available:

- Exporting individual project texts
  You can select individual texts in the project texts editor and then export the selected texts.
- Exporting project texts of a device
  When you have selected a device, the "Properties > Texts" tab of the Inspector window includes all texts that are part of the respective device. Here you can export all texts that are part of the respective device.
- Exporting all user texts or system texts at once
  You can either export all texts in the project or further limit the export by categories.

Note

Row limit in Microsoft Excel

Note that spreadsheet programs may be able to process only a certain number of rows. Microsoft Excel 2003 supports a maximum of 65536 rows, for example. Later versions of Microsoft Excel support significantly more rows.

Exporting individual project texts

To export individual project texts, follow these steps:

1. Open the "Languages & Resources" folder in the project tree.
   The lower-level elements are displayed.
2. Double-click "Project texts".
   The project texts editor opens.
3. Choose the "User texts" or "System texts" tab in the editor, depending on which texts you want to export.
4. Select the project texts you want to export.
5. Click "Export project texts" on the editor toolbar.
   The "Export" dialog box opens.
6. Select the language you want to translate from in the "Source language" drop-down list.
7. Select the language you want to translate to in the "Target language" drop-down list. The drop-down list contains the project languages you specified previously. If the required language is missing, you must first specify it in the project languages editor.
8. Specify a file path and a file name for the export file in the "Select file for export" input box.
9. Click "Export".

Exporting project texts of a device

1. To export all project texts that are part of a specific device, follow these steps:
2. Select the device and open the device properties in the Inspector window.
3. Open the "Texts" tab in the Inspector window.
4. On the toolbar, click the "Export project texts" icon.
   The "Export" dialog box opens.
5. Select the language you want to translate from in the "Source language" drop-down list.
6. Select the language you want to translate to in the "Target language" drop-down list. The drop-down list contains the project languages you specified previously. If the required language is missing, you must first specify it in the project languages editor.
7. Specify a file path and a file name for the export file in the "Select file for export" input box.
8. Click "Export".

Exporting all system or user texts

To export all project texts, follow these steps:

1. Select the "Export project texts" command in the "Tools" menu.
   The "Export" dialog box opens.
2. Select the language you want to translate from in the "Source language" drop-down list.
3. Select the language you want to translate to in the "Target language" drop-down list. The drop-down list contains the project languages you specified previously. If the required language is missing, you must first specify it in the project languages editor.
4. In "Select content", select the check box "User texts" to export user texts. To export system texts, select "System texts". To export both user texts and system texts, select both check boxes.
5. In "Select content", select the required text categories for the user texts or the system texts.
6. In the "Export file" input field, specify a file name for the export file.
7. In the "Path" input field, select a path in the data system to which the export file is to be saved.
8. Click "Export".

See also

Application examples for multilanguage projects (Page 462)

Importing project texts (Page 461)

9.8.8 Importing project texts

After external compilation in a table calculation program, you import the project texts into the TIA Portal. You can import the project texts at the following locations:

- The "Tools" menu
- In the toolbar of the project texts editor
- In the properties of a device
  When you have selected a device, the "Properties > Texts" tab of the Inspector window includes all texts that are part of the respective device. You can also import the texts of the device at this location.

Importing project texts

To import a file containing project texts, follow these steps:

1. Select the "Import project texts" command in the "Tools" menu.
   Alternatives:
   - Click the "Import project texts" icon in the toolbar of the project texts editor.
   - Select a device and open its properties in the Inspector window. Open the "Texts" tab and click the "Import project texts" icon in the toolbar.
2. The "Import" dialog box opens.
3. Select the path and the file name of the import file from the "Select file for import" field.
4. Select the "Import source language" check box if you have made changes to the source language in the export file and you want to overwrite the entries in the project with the changes.
5. Click "Import".

See also

Exporting project texts (Page 459)

9.8.9 Application examples for multilanguage projects

Let us assume you are working in a team with colleagues some of whom speak English, some French and some German. You have created a project with the TIA Portal and have already created a functioning configuration.

To allow your other colleagues to be able to keep track of the project, you would like all devices being used to have comments in English and German. First, you would like to enter the comments in German. Following this, to save time and costs, you want to have the texts translated into English in a spreadsheet program by an external translation office.

In addition to this, you also want a single comment for a particular device in French so that your French-speaking colleague can continue working on this device.

The section below describes an example of how you can achieve this with the tools of the TIA Portal.

Translating the project into English

To enter the comments in German and to have them translated into English later, follow these steps:

1. Set the editing language to "German" and fill all the comment boxes with the relevant texts in German.
   On the device selected from the French-speaking colleague, enter, for example "Unser neues Gerät" in German first.
   All the comments are now stored in German.
2. Export all user texts to an Office Open XML file with the extension ".xlsx".

3. Have the user texts contained in the file translated into English in a spreadsheet program such as Microsoft Excel.
4. Import the file into the TIA Portal after it has been translated.
   All texts are now available in German and English.

Translating a single comment field to French

To translate an individual comment field to French, follow these steps:

1. Open the comment box for the device on which the French-speaking colleague will be working.
2. Open the "Languages & Resources" pane in the "Tasks" task card.
3. Set "French" as the editing language in the "Languages & Resources" pane. As the reference language, set, for example, "English".
   Since no translation has yet been installed in French, the comment box is empty. In the "Languages & Resources" pane, the English translation "Our new device" is displayed as a reference.
4. Orientating yourself on the English reference text enter "Notre nouvel appareil" in the comment box.
   The comment for this device is now available in the languages German, English and French.

See also

Project text basics (Page 454)

Exporting project texts (Page 459)

Translating text associated with individual objects (Page 458)

9.9 Working with text lists

9.9.1 Text lists

Introduction

You can centrally manage texts to be referenced in alarms. All the texts are stored in text lists. Each text list has a unique name with which you can call up its content. A range of values is assigned to each text in a text list. If a value from a range of values occurs, the corresponding text is called up.

All the texts can be translated to all project languages. Here, you have two options available:

- You can enter the translation of the texts in a list. You will find the list in the project tree under "Languages & Resources > Project texts".
- You can export all texts to a file in Office Open XML format and enter the translation in a spreadsheet program. The translations can then be imported again. Only export the data to areas that are protected with appropriate access mechanisms. Only import files that originate from trusted sources.

The texts are translated into the other project languages within the framework of the project texts. In the text lists editor, you only have to manage the assignment of individual texts to a text list.

Text lists editor

Each device in the project has its own text lists. For this reason, these lists are arranged under the devices in the project tree. In addition, there are text lists that apply to all devices. These can be found in the project tree under "Common data > Text lists".

The text lists editor is divided into a bottom and top area. The top area displays the individual text lists. As soon as you select a text list, the included texts and the associated value ranges are displayed in the bottom area. You can sort the table columns in the text lists editor in ascending and descending order by clicking the table header of the respective column.

User-defined and system-defined text lists

There are two types of text lists:

- User-defined text lists
  You can create user-defined text lists yourself and fill them with texts; in other words, you can specify value ranges and the corresponding texts yourself. With user-defined text lists, the name of the text list begins with "USER" as default. You can change this name to any suitable name.
- System-defined text lists
  System-defined text lists are created by the system. These always involve texts relating to devices. They are automatically created as soon as you insert a device in the project. With system alarms, the name of the text list begins with "SYSTEM". The name of the text list and the ranges of values it contains cannot be modified. You can only edit texts assigned to individual value ranges.

User-defined text lists	System-defined text lists
A user-defined text list can only be assigned to one device.	System-defined text lists can be assigned both to a device as well as to the entire project.
You can create new text lists and delete existing text lists.	You cannot create new text lists or delete text lists.
You can add and delete value ranges in the text lists.	You cannot add or delete value ranges in the text lists.
You can specify both the value ranges as well as the associated texts.	You can only edit the text associated with one value range.

Device-specific and cross-device text lists

Device-specific text lists relate to only one device in the project and are therefore only valid for this device. For this reason, they are arranged under a device in the project tree. Device-specific text lists can be used-defined or created by the system.

If system-defined text lists are generally valid for several devices or not intended uniquely for one device, these text lists are grouped together in the project tree under "Common data". Text lists in the "Common data" folder are available for all devices. Cross-device text lists are always created by the system and are used solely for system diagnostics alarms. For this reason, you cannot store any user-defined text lists under "Common data".

See also

Exporting project texts (Page 459)

9.9.2 Creating user-defined text lists

You can create user-defined text lists for individual devices.

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Procedure

To create user-defined text lists, follow these steps:

1. Click on the arrow to the left of a device in the project tree.
   The elements arranged below the device are displayed.
2. Double-click on "Text lists".
   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Double-click on the first free row in the table.
   A new user-defined text list is created.
4. Enter a name for your new text list in the "Name" column.
5. Select if you want to specify the value ranges in decimal, binary or in bits from the drop-down list in the "Selection" column. Depending on the device, there may be further options available at this point.
6. Enter a comment in the "Comment" column.
   A new user-defined text list is created and you can now enter the value ranges and texts.

9.9.3 Editing user-defined text lists

You can enter value ranges and the corresponding texts in user-defined text lists. User-defined text lists are always located below a device in the project tree.

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Procedure

To add to user-defined text lists with value ranges and texts, follow these steps:

1. Click on the arrow to the left of a device in the project tree.
   The elements arranged below are displayed.
2. Double-click on "Text lists".
   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Select a text list in the table.
   The contents of the selected text list are displayed in the work area. There, you can enter a value range and assign texts to the individual value ranges.
4. Enter the value ranges you require in the "Range from" and " Range to" columns. The entry must be made in the numeric format selected for the text list.
5. Enter a text for each value range in the "Entry" column.

9.9.4 Editing system-defined text lists

In system-defined text lists, you can modify the individual texts assigned to a value range.

System-defined text lists are located in the project tree either below a device or under "Common data".

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Procedure

To edit texts in system-defined text lists that are assigned to a value range, follow these steps:

1. Click on the arrow to the left of a device in the project tree or the "Common data" element. The elements arranged below are displayed.
2. Double-click on "Text lists".
   All the text lists assigned to the device or used in common are displayed in the work area listed in a table.
3. Select a text list in the table.
   The contents of the selected text list are displayed in the work area. Here, you can add to or edit the texts assigned to a value range.
4. Enter a text for each value range in the "Entry" column.

9.10 Using memory cards

9.10.1 Basics about memory cards

Introduction

Memory cards are plug-in cards that come in a variety of types and can be used for a variety of purposes. Depending on the device type or device family, memory cards can be used for purposes, such as:

- Load memory of a CPU
- Storage medium for projects, firmware backups, or any other files
- Storage medium for performing a firmware update
- Storage medium for the PROFINET device name

For information regarding the technical variants of the respective memory cards and general information on their handling, refer to the respective documentation for the device. For information on handling memory cards in the TIA Portal, refer to the online help under the keyword "Memory Card".

Notice

Memory card is unusable for SIMATIC devices

If you use a SIMATIC memory card for non-SIMATIC purposes or you format it incorrectly, you will overwrite the internal structure of the SIMATIC memory card. The structure is not recoverable and the SIMATIC memory card becomes unusable for SIMATIC devices.

Do not use memory cards for non-SIMATIC-related purposes, and do not format SIMATIC memory cards with third-party devices or Windows tools.

See also

Adding a user-defined card reader (Page 468)

Accessing memory cards (Page 468)

Displaying properties of memory cards (Page 469)

9.10.2 Adding a user-defined card reader

Introduction

If your card reader is not detected automatically, you can add it manually.

Requirement

The project view is open.

Procedure

To add a card reader, follow these steps:

1. Open the project tree.
2. Select the "Card Reader / USB memory > Add user-defined Card Reader" command in the "Project" menu.
   The "Add user-defined Card Reader" dialog opens.
3. In the drop-down list box, select the path for the card reader.
4. Confirm your entries with "OK".

See also

Basics about memory cards (Page 467)

Accessing memory cards (Page 468)

Displaying properties of memory cards (Page 469)

9.10.3 Accessing memory cards

Requirement

- A memory card is inserted in the card reader.
- The project view is open.

Note

You cannot work with multiple memory cards at one time. Only insert one memory card into the card reader.

Procedure

To access memory cards, follow these steps:

1. Open the project tree.
2. Select the "Card Reader / USB memory > Card Reader / Show USB memory" command in the "Project" menu.
   The "Card reader / USB memory" folder is displayed in the project tree.
3. Open the "Card Reader / USB memory" folder.
   You can now access the memory card.

   Note

   If data from a non-installed product is stored on the memory card, the folders that contain these data are shown in gray. You receive an error message when you attempt to access such a folder. Install the corresponding product if needed.

See also

Basics about memory cards (Page 467)

Adding a user-defined card reader (Page 468)

Displaying properties of memory cards (Page 469)

9.10.4 Displaying properties of memory cards

You can display the properties for the utilized memory cards. Note that different memory cards with different properties must be used, depending on the device.

Requirement

- A memory card is inserted in the card reader.
- The project view is open.

Procedure

To display the properties of a memory card, follow these steps:

1. Right-click on the memory card for which you want to display the properties.
2. Select the "Properties" command in the shortcut menu.
   The "Memory Card <name of the memory card>" dialog opens. The properties are displayed in this dialog.

See also

Basics about memory cards (Page 467)

Adding a user-defined card reader (Page 468)

Accessing memory cards (Page 468)

9.11 Using libraries

9.11.1 Library basics

Introduction

You can store objects you want to reuse in libraries. For each project there is a project library that is connected to the project. In addition to the project library, you can create any number of global libraries that can be used over several projects. Since the libraries are compatible with each other, library elements can be copied and moved from one library to another. Libraries are used, for example, to create templates for blocks that you first paste into the project library and then further develop there. Finally, you copy the blocks from the project library to a global library. You make the global libraries available to other employees working on your project. The other employees continue to use the blocks and adapt them to their individual requirements as needed.

Both the project library and global libraries distinguish between two different types of objects:

- Master copies
  Almost every object can be saved as a master copy and pasted into the project again later. You can, for example, save entire devices with their contents or cover sheets for plant documentation as master copies.
- Types
  Elements that are required to run user programs, for example, blocks, PLC data types, user-defined data types or faceplates are suitable as types. Types can be versioned and therefore support professional further development. Projects using the types can be updated as soon as new versions of the types are available.

Project library

Each project has its own library, the project library. In the project library, you store the objects you want to use more than once in the project. The project library is always opened, saved, and closed together with the current project.

Global libraries

In addition to the project library, you use global libraries if you want to use libraries over several projects. Global libraries exist in three versions:

- System libraries
  Siemens supplies global libraries for its own software products. These include off-the-peg functions and function blocks that you can use within your project. The supplied libraries cannot be changed. The supplied libraries are loaded automatically matching the project. If you are working with the project in compatibility mode, the corresponding libraries for the respective product version of the TIA Portal are loaded. For all other projects, the supplied libraries of the latest TIA Portal version are loaded.
- Corporate libraries
  Corporate libraries are made available centrally by your organization, for example, in a central folder on a network drive. The TIA Portal manages the corporate libraries automatically. As soon as a more recent version of an existing corporate library becomes available, you receive a prompt to update the corresponding corporate library to the more recent version.
- User libraries
  Global user libraries are independent of a specific project and can therefore be passed on to other users. Shared access to global user libraries is also possible, for example on a network drive, if all users open the global user library with write protection.
  Global user libraries from older versions of the TIA Portal that you created yourself can still be used. To continue using global user libraries from older versions of the TIA Portal, they must first be upgraded.

Comparing library objects

You can compare blocks and PLC data types with the objects of a device. This allows you to determine, for example, whether certain blocks or PLC data types have been used in a project and whether they have been modified.

See also

Overview of the "Libraries" task card (Page 472)

Overview of the library view (Page 475)

Overview of the library management (Page 478)

Basics on master copies (Page 497)

Basics on types (Page 501)

9.11.2 Using the "Libraries" task card

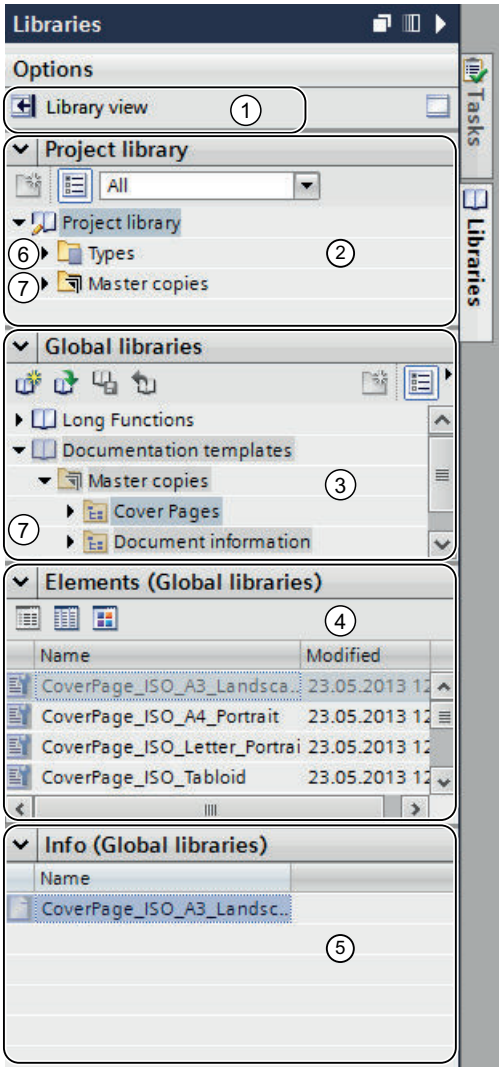
9.11.2.1 Overview of the "Libraries" task card

Function of the "Libraries" task card

The "Libraries" task card enables you to work efficiently with the project library and the global libraries.

Layout of the "Libraries" task card

The "Libraries" task card consists of the following components:



① "Library view" button
② "Project library" pane

③ "Global libraries" pane
④ "Elements" pane
⑤ "Info" pane
⑥ "Types" folder
⑦ "Master copies" folder

"Library view" button

The "Library view" button is used to change to the library view. The "Libraries" task card and the project tree are hidden with this action.

See also: Using the library view (Page 475)

"Project library" pane

In the "Project library" pane, you can store the objects that you want to use more than once in the project.

"Global libraries" pane

In the "Global libraries" pane, you manage the global libraries whose elements you want to reuse over several projects.

The "Global libraries" pane also lists libraries that were shipped together with the products you purchased. These libraries provide you with ready-made functions and function blocks, for example. The supplied global libraries cannot be edited.

"Elements" pane

In this pane, you can display the contents of folders in the library. The "Elements" pane is not displayed by default. If you want to display the "Elements" pane, you have to enable it first. Three view modes are available in the "Elements" palette:

- Details mode
  The properties of folders, master copies and types are shown in table form in details mode.
- List mode
  In list mode, the contents of folders are listed.
- Overview mode
  In overview mode, the contents of folders are displayed with large symbols.

See also: Using the element view (Page 474)

"Info" pane

You can display the contents of the library elements in the "Info" pane. The individual versions of types and the last revision date of the version are also displayed.

"Types" folder

In the "Types" directories, you can manage types and type versions of objects that you use as instances in the project.

See also: Using types (Page 501)

"Master copies" folder

In the "Master copies" directories, you can manage master copies of objects that you can use as copies in the project.

See also: Using master copies (Page 497)

See also

Library basics (Page 470)

Comparing library elements (Page 532)

9.11.2.2 Using the element view

Introduction

When you open the "Libraries" task card the first time, the "Project library" and "Global libraries" palettes are opened and the "Info" palette is closed. You can display the "Elements" palette if needed.

The elements view shows the elements of the selected library. Three view modes are available in the elements view:

- Details
  The properties of folders, master copies and types are shown in table form in details mode.
- List
  In list mode, the contents of folders are listed.
- Overview
  In overview mode, the contents of folders are displayed with large symbols.

The "Info" palette shows the contents of the selected library element. If you select a type in the elements view, for example, the type versions are displayed in the "Info" palette.

Requirement

The "Libraries" task card is displayed.

Procedure

To use the element view, follow these steps:

1. Click "Open or close the element view" in the "Project library" or "Global libraries" pane.
2. To change the view mode from the details view to the list mode or overview mode, click the corresponding icon on the toolbar.

See also

Library basics (Page 470)

Overview of the "Libraries" task card (Page 472)

Using global libraries (Page 483)

Comparing library elements (Page 532)

9.11.3 Using the library view

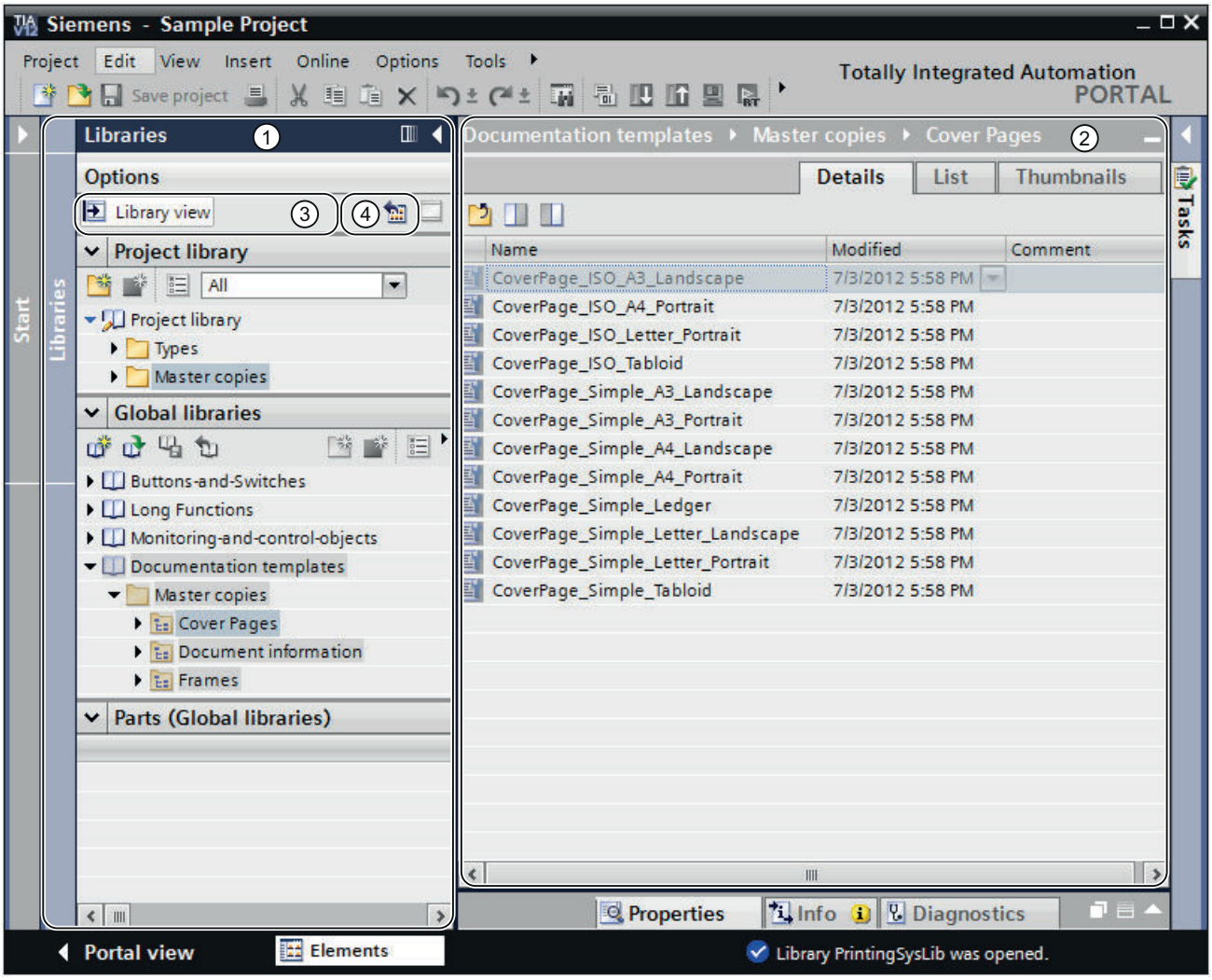
9.11.3.1 Overview of the library view

Function of the library view

The library view combines the functionality of the "Libraries" task card and the overview window. In the library view, the elements of a library are displayed in various views. In the details view, for example, you see additional properties of the individual elements. You can also edit and version the types in the library view.

Layout of the library view

The following figure shows the components of the library view:



① Library tree
② Library overview
③ "Library view" button
④ "Open or close library overview" button

Library tree

The library tree is similar to the "Libraries" task card, apart from a few minor differences. In contrast to the task card, there is no "Elements" palette, because the elements are displayed in the library overview. In addition, you can close the library view in the library tree, or open and close the library overview.

See also: "Libraries" task card (Page 472)

Library overview

The library overview corresponds to the overview window and displays the elements of the currently selected object in the library tree. You can display the elements in three different views:

- Details view
  The objects are displayed in a list with additional information, such as the date of the last change.
- List view
  The objects are displayed in a simple list.
- Icon view
  The objects are displayed as icons according to category.

In addition, you can perform the following actions in the library overview:

- Renaming elements
- Deleting elements
- Copying elements
- Moving elements
- Editing type instances
- Versioning types
- WinCC only: Editing faceplates and HMI user data types

See also: Overview window (Page 331)

See also

Basics on master copies (Page 497)

Basics on types (Page 501)

Opening and closing the library view (Page 477)

Library basics (Page 470)

Comparing library elements (Page 532)

9.11.3.2 Opening and closing the library view

The library view is opened automatically in some cases, for example, when you edit the test instance of a type or when you edit faceplates and HMI user data types. You can also open the library view manually.

Opening the library view

To open the library view manually, follow these steps:

1. Open the "Libraries" task card.
2. Click the "Open library view" button in the "Libraries" task card.
   The library tree opens. The "Library" task card and the project tree are closed.
3. If the library overview is not displayed, click "Open/close library overview" button in the library tree.
   The library overview opens.

Exit library view

To exit the library view, follow these steps:

1. Click the "Close library view" button in the library tree.
   The library tree closes. The "Libraries" task card and the project tree are opened.

See also

Overview of the library view (Page 475)

Library basics (Page 470)

Using the "Libraries" task card (Page 472)

Using global libraries (Page 483)

Comparing library elements (Page 532)

9.11.4 Using library management

9.11.4.1 Overview of the library management

Function of the library management

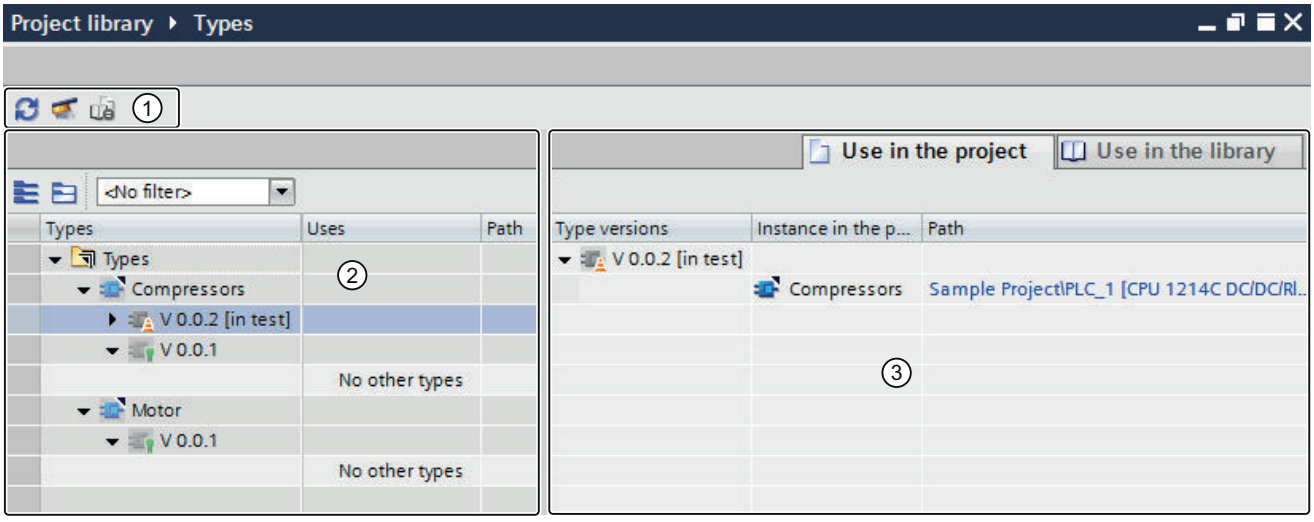
Master copies and types with dependencies to other library elements are subject to some functional restrictions. They cannot be deleted, for example, as long as dependencies still exist. This prevents other library elements from becoming useless. The library management is used to identify the dependencies and to create an overview of the work progress.

The library management offers the following functions:

- Display of the correlations of types and master copies
  If a type is referenced in other types or master copies, the correlations are displayed in the library management. You will also be able to see which library elements reference a type or a master copy.
- Display of points of use of types in the project
- Use of filters to narrow down the displayed types

Layout of the library management

The following figure shows the components of the library management:



① Toolbar of the library management

② "Types" area

③ "Uses" area

Toolbar of the library management

You can perform the following tasks in the toolbar of the library management:

- Update view
  If the project was changed, you can update the view of the library management.
- Clean up library
  By cleaning up a library, you delete all types and type versions that are not linked to an instance in the project.
- Harmonize project
  By harmonizing a project, you adapt the names and the path structures of type uses in the project to the corresponding names and path structures of the types within a library.

"Types" area

The "Types" area displays the contents of the folder that you have selected in the library view. For each type, the types that it references are displayed. You can expand or collapse all type nodes by using the buttons in the toolbar of the "Types" area. You can also filter the view with the "Filter" drop-down list.

"Uses" area

The "Uses" area gives you an overview of the points of use of the selected types and master copies. The "Uses" area is divided into two tabs:

- "Use in the project" tab
  The "Use in the project" tab is used to show the instances of type versions and their respective point of use in the project. When you select an instance, you can show the cross references of the instance in the project in the Inspector window.
- "Use in the library" tab
  The "Use in the library" tab is used to show all points within the library at which a type or a master copy is used.

See also

Opening library management (Page 480)

Filtering the display of types (Page 481)

Displaying cross references of an instance (Page 482)

Displaying instances in the project (Page 481)

Displaying relations to other library objects (Page 483)

Library basics (Page 470)

Basics on master copies (Page 497)

Basics on types (Page 501)

9.11.4.2 Opening library management

Procedure

To open the library management, follow these steps:

1. Open the library view.
2. Select a type or any folder that contains types.
3. Select the "Library management" command from the shortcut menu.

Result

The library management opens and the types are displayed with their versions.

See also

Overview of the library management (Page 478)

9.11.4.3 Filtering the display of types

Use filters to obtain a better overview of types in extensive libraries. A filter provides you with the option of limiting the displayed types. The following filters are available:

- Display all types that include a version with the "In test" or "In progress" status
- Display all released types
- Display all types that have no instances in the project
- Display all types that have more than one version

Requirement

The library management is open.

Procedure

To filter the displayed types, follow these steps:

1. In the "Types" area, select the folder whose contents you want to filter.
2. Select the required filter in the "Filter" drop-down list.
   In the "Types" area, types are displayed that correspond to the selected filter criteria.

See also

Overview of the library management (Page 478)

State of type versions (Page 503)

Creating a test version of a type (Page 511)

Editing a test version of a type (Page 513)

Creating an editing version of a type (Page 514)

9.11.4.4 Displaying instances in the project

In the library management, you can have the instances of all versions of a type or an individual type version displayed. You can jump directly to each instance in the project.

Requirement

The library management is open.

Procedure

To display the instances of a type or its versions, follow these steps:

1. Select a type or one of its versions in the "Types" area.
2. Open the "Uses in the project" tab in the "Uses" area.
   The instances in the project are displayed for each type version. The "Path" column shows the path at which the respective instance is located in the project.
3. Optional: Click the path to jump directly to the respective instance in the project tree.
   The library management is hidden and the instance is selected in the project tree.

See also

Displaying cross references of an instance (Page 482)

Using types (Page 509)

Overview of the library management (Page 478)

Displaying relations to other library objects (Page 483)

9.11.4.5 Displaying cross references of an instance

You can display the cross references of an instance without exiting the library management.

Requirement

The library management is open.

Procedure

To display the cross references of an instance in the project, follow these steps:

1. In the "Types" area, select a type version whose instances you want to display.
2. Select the instance of the required type version in the "Uses > Uses in the project" area.
3. Open the "Info > Cross-references" tab in the inspector window.
   The cross reference of the instance are displayed in the project.

See also

Using cross-references (Page 533)

Overview of the library management (Page 478)

Displaying instances in the project (Page 481)

9.11.4.6 Displaying relations to other library objects

You can display relations between individual library objects in the library management. The references of the individual type versions to the other library objects are automatically displayed in the "Types" area. In the "Uses" area, you can also view the other library objects in which the respective type version is referenced.

Requirement

The library management is open.

Procedure

To display the other library objects from which a type version is referenced, follow these steps:

1. Select a folder, an individual type or an individual version in the "Types" area.
2. Open the "Uses in the library" tab in the "Uses" area.
   In the "Uses" area, you can now see which other library objects are referenced by the individual type versions.
3. Optional: To jump to the referenced library object, click on the respective path in the "Path" column.

See also

Displaying instances in the project (Page 481)

Overview of the library management (Page 478)

9.11.5 Using global libraries

9.11.5.1 Creating a global library

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To create a new global library, follow these steps:

1. Click the "Create new global library" icon in the toolbar of the "Global libraries" palette or select the command "Global libraries > Create new library" in the "Options" menu.
   The "Create new global library" dialog opens.
2. Specify the name and the storage location for the new global library.
3. Confirm your entries with "Create".

Result

The new global library is generated and pasted into the "Global libraries" palette. A folder with the name of the global library is created in the file system at the storage location of the global library. The actual library file is given the file name extension ".al13".

See also

Library basics (Page 470)

Opening a global library (Page 486)

Displaying properties of global libraries (Page 488)

Saving a global library (Page 490)

Closing a global library (Page 491)

Deleting a global library (Page 491)

9.11.5.2 Compatibility of global libraries

You can use the TIA Portal to open global libraries that were created with an older version of the TIA Portal. However, the product version of global libraries must correspond to the product version of the opened project if you want to use objects from the global library in the project. Consequently, you may need to upgrade the global libraries.

Opening global libraries from older product versions

The table below illustrates the behavior of the TIA Portal when global libraries from older product versions are opened:

Product version of the TIA Portal (file extension of the respective global library)	Behavior when global library is opened
V10.5 (.al10) V11.x (.al11)	The global library is automatically upgraded to the latest library version when opened, following your confirmation. The upgraded global library is a copy of the original library. The original global library is retained unchanged. Alternatively, load the global library for viewing as a write-protected library.
V12 (.al12)	You can select between the following options: • Upgrade global library to product version V12 SP1 and open in compatibility mode • Upgrade global library to the current product version and open If you upgrade the global library, a copy of the original global library is created. The original global library is retained unchanged.
V12 SP1 (.al12) V13 (.al13)	You can select between the following options: • Open global library in compatibility mode If you choose compatibility mode, the project should also be opened in compatibility mode. Otherwise, you cannot use or edit contents of the global library. • Upgrade global library to the current product version and open If you upgrade the global library, a copy of the original global library is created. The original global library is retained unchanged. You can select between the following options:

Compatibility mode

If a project is open in compatibility mode for product version V12 SP1 or V13 of the TIA Portal, the global library must also be in compatibility mode. You can use objects from the global library in the project opened in compatibility mode. You can also save new objects in the global library if the objects originate from a project in compatibility mode. A global library in compatibility mode can still be opened and edited with the earlier version of the TIA Portal.

Backward compatibility of the current library version

Global libraries saved with the library format of the current TIA Portal product version are not backward compatible with older versions due to their enhanced functionality. Global libraries in the current library format can only be used in connection with TIA Portal V13 SP1 projects.

See also

Compatibility of projects (Page 377)

Opening a global library (Page 486)

Upgrading global libraries (Page 487)

Upgrading projects (Page 380)

9.11.5.3 Opening a global library

Global libraries can be further developed centrally and used over several projects. Several persons can open a global library simultaneously from a central storage location, provided all users open the global library in write-protected mode.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To open a global library, follow these steps:

1. Click the "Open global library" icon in the toolbar of the "Global libraries" pane or select the command "Global libraries > Open library" in the "Options" menu.
   The "Open global library" dialog box is displayed.
2. Select the global library you want to open. Library files are identified by the file name extension ".al[version number]". This means global libraries that were saved with current TIA Portal product version have the file name extension ".al13".
3. Write protection is activated for the library. If you want to modify the global library, disable the "Open as read-only".
4. Click "Open".
   The global library is opened and pasted into the "Global libraries" pane if the library version matches the project version. The "Upgrade global library" dialog opens if you have selected a global library from an older version of the TIA Portal. More information on upgrading the global library can be found in the chapter "Upgrading global libraries (Page 487)".

See also

Retrieving global libraries (Page 494)

Upgrading global libraries (Page 487)

Opening projects (Page 379)

Compatibility of global libraries (Page 484)

Library basics (Page 470)

Creating a global library (Page 483)

Displaying properties of global libraries (Page 488)

Saving a global library (Page 490)

Closing a global library (Page 491)

Deleting a global library (Page 491)

9.11.5.4 Upgrading global libraries

If you want to use objects from a global library in a project, the library version must match the product version of the project. For example, if you want to edit a project in V12 SP1 compatibility mode, the global libraries must be present in the library version of the TIA Portal V12 SP1. If you want to edit a project in product version V13 SP1, the global libraries must be present in the library version of the TIA Portal V13 SP1.

Depending on the library version, the following options are available:

- Upgrading the global library to the product version V13 SP1
  You can upgrade each global library from an older product version to the current product version. You are prompted accordingly when you open the global library. You can manually upgrade global libraries that have already been opened in compatibility mode to product version V13 SP1.
- Upgrading the global library to the product version V12 SP1 (compatibility mode)
  If you open a global library from version V12 of the TIA Portal, you also have the option of upgrading the global library to the library version V12 SP1. In this case, however, the global library can only be used in connection with projects from TIA Portal V12 SP1.
- Using a global library in compatibility mode
  If you are using a project in compatibility mode, the global library must also be opened in compatibility mode for the corresponding product version.

Upgrading global libraries from V11.x or lower

To upgrade a global library from TIA Portal V11.x or earlier, follow these steps:

1. Open the global library.
   The "Upgrade global library" dialog box opens.
2. Click "OK".
   A copy of the global library is created and upgraded. The copy of the global library receives the name extension "_V13_SP1". The global library is opened.

Upgrading global libraries from V12

To upgrade a global library from TIA Portal V12, follow these steps:

1. Open the global library.
   The "Upgrade global library" dialog box opens.
2. Select the target version for the upgrade:
   - Click "Yes" to upgrade the global library to the current product version.
   - Click "No" to upgrade the global library to product version V12 SP1 and to work in compatibility mode.

   A copy of the global library is created and upgraded. The copy of the global library receives the name extension "_V13_SP1" or "_V12SP1". The global library is opened.

Upgrading V12 SP1 libraries or using them in compatibility mode

To upgrade a global library from TIA Portal V12 SP1 or to use it in compatibility mode, follow these steps:

1. Open the global library.
   The "Upgrade global library" dialog box opens.
2. Select how you want to proceed with the global library:
   - Click "Yes" to upgrade the global library to the current product version.
     A copy of the global library is created and upgraded. The copy of the global library receives the name extension "_V13_SP1". The global library is opened.
   - Click "No" to use the global library in compatibility mode.

Manually upgrading libraries in compatibility mode to V13 SP1

For this procedure, a global library must already be open in compatibility mode.

To manually upgrade global libraries to the current library version, follow these steps:

1. Right-click on the global library you want to upgrade.
2. Select the "Upgrade library" command in the shortcut menu.
   The "Upgrade" dialog opens.
3. Confirm with "Yes".
   A copy of the global library is created and upgraded. The copy of the global library receives the name extension "_V13_SP1". The upgraded global library is opened. The original global library is closed.

See also

Opening a global library (Page 486)

Compatibility of global libraries (Page 484)

Upgrading projects (Page 380)

9.11.5.5 Displaying properties of global libraries

Global libraries contain properties for describing the respective library in more detail. Properties include the following:

- General information about the library
  This includes the following information: creation time, author, file path, file size, copyright, etc. Many of the attributes can be changed.
- Library history
  The library history contains an overview of the migrations performed. Here you can also call the log file for the migrations. The library history also contains information on updates of the global library.

- Support packages in the library
  You can display an overview of additional software. The additional software is needed to process all devices of the project.
- Software products in the library
  You can display an overview of all installed software products needed for the project.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To display the properties of a global library, follow these steps:

1. Right-click the global library whose properties you want to display.
2. Select the "Properties" command in the shortcut menu.
   A dialog containing the properties of the global libraries opens.
3. Select the properties in the area navigation that you want to have displayed.

See also

Opening a global library (Page 486)

Library basics (Page 470)

Creating a global library (Page 483)

Saving a global library (Page 490)

Closing a global library (Page 491)

Deleting a global library (Page 491)

9.11.5.6 Displaying logs of global libraries

Logs are created when you update or clean up global libraries or assign a shared version to several types. The logs list all changes you have made to the global library. The logs are stored together with the global library and are always available once you have opened the global library.

Procedure

To open the logs of a global library, follow these steps:

1. Open the global library in the "Libraries" task card or in the library view.
2. Open "Common data > Logs" in the lower-level folder.
3. Double-click the required log.
   The log opens in the work area.

See also

Updating a library with the contents of another library (Page 528)

9.11.5.7 Saving a global library

After you have changed a global library, you need to save it. You can save a global library under another name using the "Save library as" command.

Note

Backward compatibility with older versions of the TIA Portal

Note that global libraries can no longer be opened in older versions of the TIA Portal once they have been saved in the current version.

Requirement

The "Libraries" task card is displayed or the library view is open.

Saving changes

To save a global library, follow these steps:

1. Right-click on the global library you want to save.
2. Select the "Save library" command in the shortcut menu.

Saving a global library under another name

To save a global library under another name, follow these steps:

1. Right-click the global library that you want to save under a different name.
2. Select the "Save library as" command in the shortcut menu.
   The "Save global library as" dialog opens.
3. Select the storage location and enter the file name.
4. Confirm your entries with "Save".
   The library is saved in the specified location under the new name. The original library is retained.

See also

Working with archives of global libraries (Page 492)

Archiving global libraries (Page 493)

Library basics (Page 470)

Creating a global library (Page 483)

Opening a global library (Page 486)

Displaying properties of global libraries (Page 488)

Closing a global library (Page 491)

Deleting a global library (Page 491)

9.11.5.8 Closing a global library

Global libraries are independent of projects. This means that global libraries are not closed together with your project. You must therefore close global libraries explicitly.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To close a global library, follow these steps:

1. Right-click on the global library you want to close.
2. Select the "Close library" command in the shortcut menu.
   If you have made changes to the global library, select whether or not you want to save the changes.
   The global library is closed.

See also

Creating a global library (Page 483)

Opening a global library (Page 486)

Displaying properties of global libraries (Page 488)

Saving a global library (Page 490)

Library basics (Page 470)

Deleting a global library (Page 491)

9.11.5.9 Deleting a global library

If you no longer require a global library, you can delete it. Libraries supplied by Siemens cannot be deleted.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To delete a global library, follow these steps:

1. Right-click the global library you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. Click "Yes" to confirm.

Result

The global library is removed from the "Global libraries" palette. The entire library for the global library is deleted from the file system.

See also

Library basics (Page 470)

Creating a global library (Page 483)

Opening a global library (Page 486)

Displaying properties of global libraries (Page 488)

Saving a global library (Page 490)

Closing a global library (Page 491)

9.11.5.10 Archiving and disabling global libraries

Working with archives of global libraries

If you want to back up global libraries on an external hard drive or send them via e-mail, for example, use the archiving function to reduce the storage space of the library.

Options for reducing the size of the project

There are two ways to reduce the space required to store global libraries:

- Creating a compressed archive of global libraries
  Archives of global libraries are compressed files, each containing an entire global library, including the entire folder structure of the library. Before the directory with the global library is compressed into the archive file, all files are reduced to their essential components to further decrease the storage space. Compressed archives of global libraries are therefore well suited for sending via e-mail.
  Compressed archives of global libraries of the current product version have the file extension ".zal13". Archives from TIA Portal V12.x have the file extension ".zal12".
  To open a compressed archive of a global library, retrieve the archive. The archive file is extracted to a location you have selected with the entire folder structure and all files.
- Minimizing global libraries
  You can skip additional compression in an archive file, and instead create a copy of the global library directory. The included files can be reduced to the essential elements. This minimizes the required storage space. The full functionality of the global library is retained and the global library can be loaded as usual.
  Minimized global libraries are especially well suited for archiving, for example, on an external medium.

See also

Archiving global libraries (Page 493)

Retrieving global libraries (Page 494)

Archiving global libraries

You reduce the storage space of a global library by packing it into a compressed file. You can also reduce the storage space by saving the global library reduced to the essential elements. You can do both of these with the archiving function for global libraries. When you archive a library, the original library version is retained. This means that libraries in compatibility mode for TIA Portal V12 SP1 are still compatible with TIA Portal V12 SP1.

Requirement

The global library is loaded.

Procedure

To archive a global library, follow these steps:

1. Select the global library that you want to archive.
2. Right-click the global library and select the "Archive" command from the shortcut menu.
   The "Archive global library as..." dialog opens.
3. Select the directory where you want to save the archive file or the new directory of the global library.
   The directory may not be located in a project directory or within the directory of a global library.

4. Select the file type from the "File type" drop-down list:
   - Global libraries archive, if you want to create a compressed file of the library.
   - Minimized global library, if you only want to create a copy of the library directory with minimal storage requirement.
5. Enter a file name in the "File name" field if you are creating an archive file. If you are creating a minimized global library, enter the name of the new library directory to be created in the "File name" box.
6. Click "Save".

Result

A compressed file with the extension ".zal13" or ".zal12" is generated. The file extension depends on whether you have archived a library in V12 SP1 compatibility mode or in the library format of TIA Portal V13.

The file includes the complete directory of the global library. The individual files of the global library are also reduced to the essential components in order to save space.

If you have minimized the global library, only a copy of the original directory of the global library is created at the required location. The files contained within it were reduced to their essential components in order to save space.

See also

Working with archives of global libraries (Page 492)

Retrieving global libraries (Page 494)

Retrieving global libraries

Before you can use an archived global library, you have to retrieve it. The global library is extracted and then opened in the TIA Portal.

Procedure

To extract the archive of a global library, follow these steps:

1. Select the "Global libraries > Retrieve library" command in the "Options" menu.
   The "Retrieve archived global library" dialog opens.
2. Select the archive file.
3. Select the check box "Open read-only" if you want to load the global library write-protected.
4. Click "Open".
5. The "Find folder" dialog opens.
6. Select the target directory to which the archived global library should be extracted.
7. Click "OK".

Result

The global library is extracted to the selected directory and opened immediately.

If the archive file contained a global library from TIA Portal V12 SP1, the global library is opened in compatibility mode. Manually upgrade the library to the library version of TIA Portal V13, if necessary.

See also

Working with archives of global libraries (Page 492)

Archiving global libraries (Page 493)

Opening a global library (Page 486)

Compatibility of global libraries (Page 484)

Upgrading global libraries (Page 487)

9.11.5.11 Using global corporate libraries

Basics of corporate libraries

Introduction

Corporate libraries are global libraries made available by an administrator and assigned to the TIA Portal. The administrator can assign new libraries or change libraries at any time. New libraries are automatically loaded in the TIA Portal following your confirmation. As soon as more recent versions of corporate libraries are available, the existing corporate libraries are automatically updated to the more recent version, also following your confirmation.

The corporate libraries are located just like normal global libraries in the "Global libraries" palette of the "Libraries" task card.

Providing corporate libraries

Corporate libraries can be saved in any directory on the hard drive of the computer or on a network drive. You use an XML file to assign corporate libraries to the TIA Portal. The XML file includes the directories and names of the assigned corporate libraries. The XML file must be saved in the following directory on the hard drive of the computer:

`C:\ProgramData\Siemens\Automation\Portal V13\CorporateSettings\`

The XML file must be named "`CorporateSettings.xml`".

You either copy the configuration file yourself to the corresponding directory or the configuration file is assigned to you through the company network. The valid configuration is automatically applied when the TIA Portal is started. When the TIA Portal is started, it continuously monitors the directory for configuration files. If the configuration file has changed, you receive a prompt to apply the new configuration. You can reject this prompt twice. You will always receive the next prompt three hours later. You must apply the new configuration at the third prompt. You receive a changed configuration file from the project administrator, for example, if corporate libraries have been added or deleted.

Options as project administrator

You can automatically assign the configuration file or the corporate libraries to the computers of the team members or distribute updates to the team members. This function is not part of the TIA Portal and requires a corresponding IT infrastructure in your company. If you want to administer the configuration file centrally, discuss this approach with the IT managers at your company.

Creating a configuration file for corporate libraries

You use a configuration file in XML format to make the corporate libraries available in the TIA Portal. The configuration file includes the directories and file names of the libraries to be loaded. Below you will learn how you create the XML configuration file and where to save it.

Procedure

To provide a configuration file for corporate libraries, follow these steps:

1. Create an XML configuration file with the content listed below. Use the coding "UTF-8".
2. Save the XML file under the name "CorporateSettings.xml".
3. Save the file in the following directory on your computer:
   C:\ProgramData\Siemens\Automation\Portal V13\CorporateSettings\

Content of the XML configuration file

The XML configuration file must have the following content:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
 <Settings.Settings ID="0">
  <ObjectList>
   <Settings.General ID="1" AggregationName="General">
    <AttributeList>
     <CorporateLibraryPaths>
     <!-- Example of an entry -->
     <Item>D:\CorporateLibraries\Corporate_Library_1.al13</Item>
     <!-- Here you enter additional global libraries, if any. -->
     </CorporateLibraryPaths>
    </AttributeList>
   </Settings.General>
  </ObjectList>
 </Settings.Settings>
</Document>
```

Result

As soon as you have saved the XML configuration file in the respective directory, you will receive a prompt in the TIA Portal to load the corporate libraries.

See also

Specifying settings with an XML file (Page 365)

9.11.6 Creating folders in a library

The library elements are stored in the libraries according to their type in the "Types" and "Master copies" folders. Create additional folders below "Types" and "Master copies" to further organize master copies and types.

Requirements

- The "Libraries" task card is displayed or the library view is open.
- If you want to create new folders within a global library, you have to open the global library with write permission.

Procedure

To create a new folder, follow these steps:

1. Right-click any folder within the library.
2. Select "Add folder" from the shortcut menu.
   A new folder is created.
3. Enter a name for the new folder.

See also

Working with types in the project library (Page 506)

Filtering master copies (Page 500)

9.11.7 Using master copies

9.11.7.1 Basics on master copies

Master copies are used to create standardized copies of frequently used elements. You can create as many elements as needed and insert them into the project based on a master copy. The elements inherit the properties of the master copy.

You store master copies either in the project library or in a global library. Master copies in the project library can only be used within the project. When you create the master copy in a global library, it can be used in different projects.

The following elements can be created as master copies in the library, for example:

- Devices with their device configuration
- Tag tables

- Instruction profiles
- Watch tables
- Elements from the documentation settings, for example, cover sheets and frames
- Blocks and groups containing multiple blocks
- PLC data types and groups containing multiple PLC data types
- Text lists
- Alarm classes
- Technology objects

In many cases, the objects you add as master copy contain additional elements. A CPU, for example, can contain blocks. If the included elements are uses of a type version, the used versions of the types are automatically created in the library. The elements contained therein are then used as an instance and linked to the type.

See also

Adding master copies (Page 498)

Using master copies (Page 500)

Basics on types (Page 501)

Filtering master copies (Page 500)

9.11.7.2 Adding master copies

If you want to use objects multiple times, copy them as master copies in the project library or in a global library. You can choose from the following methods for creating master copies:

- You select one or more elements and create individual master copies from them
- You select multiple elements and create a single master copy from them that contains all selected elements.

Requirement

- The "Libraries" task card is displayed.
- If you add a device as a master copy, this device meets the following requirements:
  - The device is compiled and consistent.
  - The device contains no test instance of a type.
- If you add the master copy to a global library, the global library is opened with write permission.

Creating master copies from one or more elements

To create a master copy for one or more elements, follow these steps:

1. Open the library in the "Libraries" task card.
2. Select the desired elements.
3. Using a drag-and-drop operation, move the elements to the "Master copies" folder or any subfolder of "Master copies".

Alternative:

1. Select the desired elements.
2. Copy the elements to the clipboard and paste the elements at the required location.
   Each element is pasted in the library as a master copy. A type is created automatically in each case from any objects contained (e.g., referenced blocks).

Creating a single master copy from multiple elements

To create a single master copy for all elements from multiple elements, follow these steps:

1. Open the library in the "Libraries" task card.
2. Copy to the clipboard the elements that you want to create as master copies.
3. Right-click on the "Master copies" folder or any of its subfolders in the library.
4. In the shortcut menu, select "Paste as a single master copy" command.

Alternative:

1. Select the desired elements.
2. Using a drag-and-drop operation, move the elements to the "Master copies" folder or any subfolder of "Master copies". Meanwhile, keep the <Alt> key pressed.
   The elements are pasted in the library as a single master copy. The single master copy contains all selected elements. A type is created automatically in each case from any objects contained (e.g., referenced blocks).

Note

Avoiding complex structures of master copies

To prevent name conflicts and conflicts regarding the folder structure during subsequent use of master copies, avoid complex master copies. Complex master copies are, for example, master copies that consist of multiple elements and nested folders.

See also

Basics on master copies (Page 497)

Using master copies (Page 500)

Library basics (Page 470)

Adding types to the project library (Page 506)

9.11.7.3 Filtering master copies

To make it easier to keep track of a large number of master copies, you can filter the display according to the type of master copy.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To filter the view, follow these steps:

1. Open the "Master copies" folder in the project library or a global library.
2. In the drop-down list of the toolbar, select the type of objects you want to display under "Master copies".

Result

Only the selected type of master copies is displayed. You can set the filter to "All" to return to an unfiltered view.

See also

Library basics (Page 470)

Creating folders in a library (Page 497)

Basics on master copies (Page 497)

Using master copies (Page 500)

Using the element view (Page 474)

9.11.7.4 Using master copies

Master copies are contained either in the project library or in a global library. You can paste one or more master copies in the project at the same time. If you insert multiple master copies at the same time, ensure that all master copies are compatible with the desired point of use.

Requirement

The "Libraries" task card is displayed.

Procedure

To paste a master copy into the project, follow these steps:

1. Open the "Master copies" folder or any subfolder of "Master copies" in a library.
2. Using a drag-and-drop operation, move the desired master copies or whole folders to the point of use.

Or:

1. Open the element view.
2. Using a drag-and-drop operation, move the desired master copies or whole folders from the "Elements" pane to the point of use.

Result

A copy of the master copies is inserted. If incompatible master copies were included in a multiple selection, these are omitted and a copy is not created in the project.

See also

Basics on master copies (Page 497)

Adding master copies (Page 498)

Filtering master copies (Page 500)

Library basics (Page 470)

Using the element view (Page 474)

9.11.8 Using types and their versions

9.11.8.1 Basics on types

Using types

Types are elements that are required for the execution of user programs. Types can be versioned and further developed from a central location.

The following elements can be stored as type in the project library or the global library:

- Functions (FCs)
- Function blocks (FBs)
- PLC data types
- User data types
- Faceplates
- Screens
- Styles
- User-defined functions

Any number of instances can be derived from the versions of types in the project. The instances are then linked to the version of the type. If you are using types from a global library, the type is also created in the project library. If the type already exists in the project library, missing type

versions are added if necessary. The instance is then linked only to the respective type version in the project library.

Types and their instances are marked with a black triangle. The following figure shows an instance, marked with a black triangle, and an ordinary program block:

Basics on versioning of types

Type versioning provides you with the means to develop types centrally and then roll out the most recent version to the individual projects as an update. In this way, error corrections and added functions can be easily integrated into existing projects. If you have created a new version of a global library, you can update existing projects in an automatic process. This minimizes errors and reduces the amount of maintenance work for large automation solutions that contain numerous individual projects.

Versioning allows you to trace the development process of individual types. Before you release a version, you can try it out in a test environment to determine whether changes made to a type integrate smoothly into an existing project. You only release a version for productive use after you have made sure that everything operates without errors. You can view the history of individual instances in the project at any time and determine the version from which the instance was derived.

The TIA Portal also checks automatically whether there are associated objects with individual versions of a type. Associated objects can be, for example, PLC data types or other blocks referenced in a block. All associated objects are already taken into consideration during the creation of a type or during copying between libraries. Before being released, versions of types are checked for their consistency to ensure that no inconsistencies arise in the project.

Versions of types

Versions are assigned to each type. The version number is displayed in both the "Libraries" task card as well as the library view next to the respective type. The version number is also displayed in the project tree next to instances of types. This allows you to see at all times which version of an instance is used in the project.

The version number consists of three numbers separated by periods. You can randomly assign the first two digits. Numbers from 1 to 999 are permitted as the first two numbers. The third digit is the build number. It is automatically incremented by one when you edit an instance related to the version. The build number is reset to 1 when you release the version.

The versions of types can have three states:

- In progress (faceplates and HMI user data types)
- In test (all sorts of types except faceplates and HMI user data types)
- Released

The following figure shows a type with two versions. One version is "in test" and one version is released:

See also

State of type versions (Page 503)

Basics on master copies (Page 497)

Adding types to the project library (Page 506)

Using types (Page 509)

Editing library elements (Page 526)

Duplicating types (Page 509)

9.11.8.2 State of type versions

The versions of types can be in three different states. The states can be determined from the instance or in the library.

"In progress" status

Only versions of faceplates and HMI user data types have the "In progress" status. If a version is in progress, "In progress" appears next to the version in the library.

When you create a new type or a new version of a released type, the type is assigned the status "In progress".

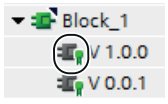
Types with the status "In progress" can be edited in the library view. A reference to an instance in the project does not have to exist. Upon release, the compatibility of the type is tested by a consistency check.

"In test" status

All types except for faceplates and HMI user data types can have the "in test" status. If a version is in test, "in test" appears next to the instance and in the library. A version in the test state is linked to a test instance in the project. This allows you try out the effects of your changes in a test environment including all online functions before you release a type for use in actual operations.

"Released" status

The "released" status is available for all types, regardless of the point of use. If a version has been released, the symbol of the version is marked with a seal in the library:



Released versions can be opened with write protection in their instance. If you want to edit a released version, you must first create a new "In progress" or "in test" version.

See also

Basics on types (Page 501)

Using types (Page 509)

Creating a test version of a type (Page 511)

Editing a test version of a type (Page 513)

Creating an editing version of a type (Page 514)

Performing a consistency check for a version (Page 514)

Discarding versions (Page 515)

Release versions (Page 516)

Assigning a version (Page 524)

Updating the project to the latest versions (Page 518)

Remove the link between instance and type (Page 520)

Filtering the display of types (Page 481)

9.11.8.3 Displaying a released type version

If you display a released version but do not want to edit it, open the instance with write protection. All types except faceplates and HMI user data types can be opened directly at the instance. Faceplates and HMI user data types can only be opened in the "Libraries" task card or in the library view.

Requirement

The released version has an instance in the project; except with types, this is a faceplate or an HMI user data type.

Opening a type version at an instance

To open a released version of a type with write protection starting from an instance, follow these steps:

1. Select the released version at the instance in the project tree.
2. To do this, right-click the instance and select the "Open" command from the shortcut menu. The instance is opened with write protection.

Opening a type version in the "Libraries" task card or library view

To open a released version of a type in the "Libraries" task card or in the library view, follow these steps:

1. Select the version.
2. To do this, right-click the version and select the "Open" command from the shortcut menu. If it is a faceplate or HMI user data type, it is opened directly in the library view. In this case, skip the remaining steps. If it is any other type, the "Open type" dialog opens.
3. Select the instance with the version that you want to display from the list of instances.
4. Click "OK" to confirm.
   The instance is opened with write protection.

9.11.8.4 Displaying properties of a type or version

You can display the properties of a type or version.

Procedure

To display the properties of a type or a version and to enter a comment, follow these steps:

1. Select a type or the version of a type in the "Libraries" task card or in the library view.
2. To do this, right-click on the type or one of its versions and select the "Properties" command from the shortcut menu.
   The "Properties" dialog box opens.
3. If needed, enter a comment on the type in the "Comment" field or edit an existing comment.

Visible and changeable properties

The following table shows the properties that you can see or change for a type or version:

Property	Description	Version	Types
Name	Name of the type	-	Visible and changeable
Version	Version number	Visible	-
Last change	If you create, release, duplicate or assign a version, this is recorded as a change to the type. The date and time of the change are entered in the "Last change" field.	Visible	-
Author	The creator of the version is indicated as the author.	-	-
Original library	The project and the library from which the current version of the type was generated are indicated. This information is important, for example, for finding the original of the type after it has been copied from another library.	Visible	-

Property	Description	Version	Types
Version GUID or Type GUID	The GUID is used to uniquely identify the type or the version of the type, even if, for example, there are types or versions with an identical name within the project library or the global library. The GUID cannot be changed and is assigned automatically.	Visible	Visible
Comment	Comment on the type or version	Visible and, for versions whose status is "in test" or "in work", changeable	Visible and changeable

9.11.8.5 Working with types in the project library

Adding types to the project library

Types for reuse in the project can be created in the project library for various elements. You can create the following elements as types, for example:

- Program blocks
- Faceplates
- PLC data types
- HMI user data type

If you add an element as a type to the project library and this element has dependencies to other elements, the dependent elements are automatically created as a type as well.

After a type has been added to the project library, the type is linked to the added element from the project.

Requirement

- The "Libraries" task card is displayed.
- The elements that you want to add as type are compiled.
- The elements are consistent.
- For blocks, all other requirements are met as described in section "Block requirements (Page 507)

Procedure

To add an existing element to the project library as a type, follow these steps:

1. Open the project library in the "Libraries" task card.
2. Drag-and-drop one or more elements into the "Types" folder or any subfolder of "Types".
   Alternative: Copy the elements in the project tree to the clipboard and add the elements in the desired project library folder.
   The "Generate type" dialog opens.

3. Enter the properties of the new type:
   - Enter a name for the new type in the "Name of the type" field.
   - Enter a version number for the new type in the "Version" field.
   - Enter the name of the editor who is responsible for the type in the "Author" field.
   - Enter a comment on the type in the "Comment" field.
4. Click "OK" to confirm.
   The new type is generated with a released version. The version is linked to the element that has been added.

See also

Basics on types (Page 501)

Duplicating types (Page 509)

Block requirements (Page 507)

Library basics (Page 470)

Adding master copies (Page 498)

Block requirements

Permitted blocks for creating a type

You can create types of the following blocks in the project library:

- Function blocks
- Functions
- User-defined data types

Consistency and compilation

To create a type from a block, the block must be consistent and compiled. When you create a type, the consistency of the block is checked automatically and the block is compiled if necessary.

Block requirements

To create a type from a block, the block must meet the following requirements:

- The block must match the type of CPU.
  A block which does not match a CPU can be identified by an incompatibility symbol to the right of the block in the project tree. This is the case, for example, when you copy a block from an S7-1500 CPU to an S7-300 CPU. A block of an S7-1500 is not compatible with an S7-300 CPU and cannot be compiled. This means that you cannot create a type from the block.
- The block is not a system data block.
- The block does not include global data access or single instance call of an instance data block.
  This also applies, for example, to the call of a data block within an STL block with the "OPN" command.

Special aspects for know-how-protected blocks

You can also create types of know-how-protected blocks. However, keep in mind the following additional requirements for know-how-protected blocks:

- Release of the block for use as type
  Know-how-protected blocks must be released to be used in a library. You make the necessary settings in the block properties. In the "Attributes" area, select the option "Block can be used as know-how protected library element".
- The block does not include access to data blocks, PLC tags or PLC constants.
  Because you cannot create types from data blocks, PLC tags or PLC constants, you also cannot create a type from know-how-protected blocks with access to the objects listed above. The instance of a know-how-protected block does not work without the associated data blocks, PLC tags or PLC constants.

Access to data blocks, PLC tags or PLC constants

Access to data blocks, PLC tags or PLC constants is permitted in blocks without know-how protection, but you can still create types from these blocks. While function blocks referenced in a block, for example, are also automatically created as type in the project library, this is not the case for access to data blocks, PLC tags and PLC constants. If data blocks, PLC tags or PLC constants are referenced in a block, only the block itself is created as type. The referenced objects are not created as type. If you create an instance from the type of the block later at another point of use (for example, a different CPU), the referenced objects are missing at the new point of use. This means that you should create referenced objects at the later point of use again. Make sure that the referenced objects have the correct name at the new point of use.

See also

Adding types to the project library (Page 506)

Duplicating types

Types in the project library can be duplicated. If you duplicate a type, the following applies to the duplicate:

- The duplicate is created in the same folder.
- The duplicate is created from the highest version of the type.
- The duplicate does not have an instance in the project.

Requirement

The type is released.

Procedure

To duplicate a type in the project library, follow these steps:

1. Right-click on a type.
2. Select the "Duplicate type" command in the shortcut menu.
   The "Duplicate type" dialog opens.
3. Enter the properties of the new type:
   - Enter a name for the new type in the "Name of the type" field.
   - Enter a version number for the new type in the "Version" field.
   - Enter the name of the editor who is responsible for the type in the "Author" field.
   - Enter a comment on the type in the "Comment" field.
4. Click "OK" to confirm.
   The new type is generated with a released version.

See also

Adding types to the project library (Page 506)

Editing library elements (Page 526)

Using types

Types from the project library can be used any number of times within a project. The use of types is always linked to a version of the respective type in the project library. If the type contains dependent elements, these are also created as a type of use at a suitable point in the project. Examples of dependent elements are PLC data types referenced in a block.

You can only assign a version of the same type to a device. If necessary, you can create uses of several types at the same time.

Uses of a type are called instances in the project tree.

Possible points of use for type versions

To use a type, create an instance of a particular version of the type at a suitable point in the project. Suitable points of use are:

- Folder in the project tree
  An instance of a type can be created in a folder in the project tree. The folder must be suitable for the particular type. If the type is a user data type, for example, you can only create the instance in the "PLC data types" folder.
- Editor
  An instance can be created from a type in a suitable editor. For example, you can create an instance from the type of a function block in another block. The type of the function block is called from another block in this way.

Options for creating an instance

You have two options for creating type instances:

- Drag the required type version from the project library and drop it at the location where you want to use it.
  Instances of the types and their dependent elements are generated and pasted at the location where you want to use them. The instances are connected to the respective type version in the project library. If you have created the instances in an editor, instances of the types are also created at the appropriate points in the project tree. By default, the folder structure from the library is reproduced in the project tree. If you want to select a different folder in the project tree, you will find the instances in the same folders as in the library.
- Copying and pasting type instances
  You can copy type instances to the clipboard and paste them at another point. This way you create another instance of the type version. The instance is still connected to the same type version in the project library. When you copy the instance of a type to the clipboard and paste it in another project, all required type versions are accepted in the project library of the other project.

Requirement

- The desired versions have been released.
- A device which supports the category of desired type is already available in the project
- The device is not yet assigned any further instance of the same type.

Procedure

To create an instance of a type, follow these steps:

1. In the project library, select the versions from which you want to create an instance.
2. Using a drag-and-drop operation, move the versions from the project library to the point of use in the project tree or an editor.
   Alternative: To automatically use the newest version, move the types themselves from the project library to the point of use using a drag-and-drop operation.
   For example, use a drag-and-drop operation to move the type of a function block to the block folder of the CPU in the project tree. To call the type directly from another block, for example, move the type from the project library to the point of use in the program editor using a drag-and-drop operation.

Alternative:

1. Copy one or more instances to the clipboard.
2. Paste the instances at a suitable point in the same project or in a different project.

See also

Basics on types (Page 501)

State of type versions (Page 503)

Displaying types for an instance (Page 511)

Library basics (Page 470)

Using master copies (Page 500)

Displaying types for an instance

In the project tree, you can jump to a type associated with an instance in the project library.

Procedure

To jump to the type associated with an instance in the project library, follow these steps:

1. Right-click the instance of the type in the project tree.
2. Select the "Go to type" command from the shortcut menu.
   The associated type is displayed in the project library.

See also

Using types (Page 509)

Creating a test version of a type

Before you release a type for productive use, you need to test the type within a project and on the automation system. The test is conducted in a specific test environment. This test environment can be a CPU, for example.

Create a version with "in testing" status for the test. The creation of a version "in testing" is suitable for all sorts of types except for faceplates and HMI user data types. On the other hand, versions "in progress" can be created from faceplates and HMI user data types.

There are two ways to create a test version of a type and define the test environment:

- In the "Libraries" task card or library view
  Generate the new version with "in testing" status in the "Libraries" task card or in the library view. You can generate the new version either directly from the type or from a specific version of the type.
- At an instance in the project tree
  You can also create the test version directly at the instance in the project tree. Since the instance is always used in a specific version in the project, a new version of the type is generated from the version used at the instance.

You can also create test versions from several types at the same time.

The following rules apply to a version "in testing":

- You can set only one version to "in testing" for each type at a given time.
- A version in testing may only be linked to a single instance in the project. Therefore, it is not possible to copy an instance to the clipboard, to duplicate it or to create an additional type from the instance as long as it has "in testing" status.

Requirement

- There is at least one instance of the type within the project in a given version.
- If you want to create the new version from a particular version of the type, the instance must be used in this version in the project.

Procedure

To generate a new test version of a type or the version of a type, follow these steps:

1. Select the type, a version of the type, or the instance.
   When you create the test version directly at the instance, you can select several elements or folders with multiple selection. You can skip steps 3 and 4 because the test environment is already defined by the selected instance.
2. Right-click the selected element and select the "Edit type" command from the shortcut menu.
   If you have started the editing in the "Libraries" task card or in the library view, the "Edit type" dialog opens. If you have started the editing at the instance in the project tree, the test instance is immediately opened for editing in the library view.

3. Select an instance of the type from the list in the project.
   If you have started editing at the type itself, the following applies:
   - The location at which the instance is used (for example, the CPU) is also used as test environment for the subsequent editing of the type.
   - Selecting the test instance also defines the version to be edited.

   The following applies to editing a specific version:
   If your starting point is a specific version, you can only select from the list instances that are used in the same version.
4. Click "OK" to confirm.

Result

A new version of the type is created. The new version is "in testing" and is identified as such in the user interface.

See also

Basics on types (Page 501)

State of type versions (Page 503)

Library basics (Page 470)

Editing a test version of a type

When you edit a version "in testing", a new version is not created. You can start the editing of the test version at the instance in the project tree, in the "Libraries" task card" or in the library view.

Note

Deleting and renaming interface parameters

You can add new parameters. However, if you rename or delete existing parameters, these parameters will not be supplied when the block is called.

Procedure

To edit the test version of a type, follow these steps:

1. Right-click the test version or the instance.
2. Select the "Edit type" command from the shortcut menu.
   The test instance is opened and can be edited.

See also

Basics on types (Page 501)

State of type versions (Page 503)

Library basics (Page 470)

Performing a consistency check for a version (Page 514)

Discarding versions (Page 515)

Release versions (Page 516)

Creating an editing version of a type

If you want to edit a type with faceplates or HMI user data types, create a new "in progress" version of the type. The new version is edited in the library view. To check the compatibility of the changes, a consistency check is automatically performed for the type prior to the release.

Requirement

The project library is opened in the "Libraries" task card or in the library view.

Procedure

To create a new version of a type in progress, follow these steps:

1. Right-click the type or the version of the type.
2. Select the "Edit type" command from the shortcut menu.
   A new "in progress" version is created and opened for editing in the library view.

See also

Basics on types (Page 501)

State of type versions (Page 503)

Library basics (Page 470)

Performing a consistency check for a version

A type version can unintentionally obtain an inconsistent state during editing. To notice errors in the development process in good time, you can regularly perform a consistency check. However, the consistency check always takes place automatically as soon as you release a version.

Details of how to start the consistency check manually for the version of a type are provided below.

Requirement

- The project library is opened in the "Libraries" task card or in the library view.
- The version is "in progress" or "in test".

Procedure

To perform a consistency check for the version of a type, follow these steps:

1. Right-click on the version that you want to check for consistency.
2. Select the "Consistency check" command in the shortcut menu.
   The consistency check is carried out. You receive a message with the result of the consistency check.

See also

Release versions (Page 516)

Discarding versions (Page 515)

Editing a test version of a type (Page 513)

Basics on types (Page 501)

State of type versions (Page 503)

Library basics (Page 470)

Discarding versions

Discard versions of a type "in test" or "In progress" when you no longer need the version. You can also select several types or folders and discard all test or editing versions contained therein. All uses of the deleted versions are reset to the last released status.

Requirement

- The version that you want to discard is in the state "In testing" or "In progress".
- Your are in the library view of the "Libraries" task card is open.

Procedure

To discard one version, follow these steps:

1. Right-click the version that you want to discard.
2. Select the "Discard changes and delete version" command from the shortcut menu.
   The version is deleted.

Alternative in the library view:

1. Click the "Discard changes and delete version" button in the toolbar while a version is opened for editing.
   The version is deleted.

See also

Basics on types (Page 501)

State of type versions (Page 503)

Library basics (Page 470)

Performing a consistency check for a version (Page 514)

Discarding all versions within a folder

You can discard all versions with the "In test" or "In progress" status at the same time within a folder. All uses of the deleted version are reset to the last released status.

Requirement

Your are in the library view of the "Libraries" task card is open.

Discarding a version of a single type

To discard all versions within a folder, follow these steps:

1. Right-click the folder.
2. Select the "Discard all" command from the shortcut menu.
   All "In test" or "In progress" versions are deleted.

Release versions

When you are finished editing a type version, release the version for productive use. Assign a version number for the release. You can also use a multiple selection to release several versions at the same time.

Requirement

- The "Libraries" task card is open or you are in the library view.
- The versions that you want to release are "in test" or "In progress" status.
- The versions are consistent.
  A consistency check is performed as soon as you start the release. If errors that prevent a release occur during the consistency check, a message is displayed indicating how you can correct the errors.

Procedure

To release type versions, follow these steps:

1. Select the versions you want to release.
2. Right-click your selection.
3. Select the "Release version" command from the shortcut menu.
   The "Release type version" dialog box opens.

4. If necessary, change the properties of the version:
   - Enter a name for the type in the "Name" field. If you have selected several versions for release, the "Name" field cannot be changed.
   - In the "Version" field, define a main and an intermediate version number for the version to be released. If you have selected several versions for release, the "Version" field cannot be changed and the last version number is used for the release.
   - In the "Author" field, enter the editor of the version to be released.
   - In the "Comment" field, enter a comment on the version to be released.
5. Optional: Select the "Delete unused type versions from the library" check box to delete all versions from the library that are not connected to any instance in the project. Versions with dependencies on other types or library elements are not deleted.
6. Click "OK" to confirm.

Alternative in the library view:

1. Click the "Release version" icon in the toolbar while a version is opened for editing.
2. Continue with steps 3 to 5 of the description above.

Result

The selected versions are released.

The properties are applied for the types themselves, the versions to be released, and for all future versions. Versions already released remain unaffected by the changes.

If needed, all instances with the same original version are updated to the most recent version and the unused versions of the type are deleted.

See also

Releasing all versions within a folder (Page 517)

Basics on types (Page 501)

State of type versions (Page 503)

Library basics (Page 470)

Performing a consistency check for a version (Page 514)

Assigning a version (Page 524)

Adding types to a global library (Page 520)

Updating the project to the latest versions (Page 518)

Releasing all versions within a folder

When you are finished editing all types within a folder, release all versions simultaneously.

Requirement

- Your are in the "Libraries" task card or in the library view.
- The folder includes versions "in test" or "in progress" status.
- All versions "in test" or "in progress" are consistent.
  A consistency check is performed as soon as you start the release. If errors that prevent release occur during the consistency check, a message is displayed indicating how to correct the errors.

Procedure

To release all type versions within a folder, follow these steps:

1. Right-click the required folder.
2. Select the "Release all" command from the shortcut menu.
   The "Release type version" dialog box opens.
3. If necessary, change the properties of the version:
   - In the "Author" field, enter the editor of the versions to be released.
   - In the "Comment" field, enter a comment on the versions to be released.
4. Select the "Delete all unused type versions from the library" check box to delete all versions from the library that are not connected to any instance in the project. Versions with dependencies on other types or master copies are not deleted.
5. Click "OK" to confirm.

Result

All type versions "in test" or "in progress" status within the selected folder are released.

The properties are applied for the versions to be released and for all future versions. Versions already released remain unaffected by the changes.

Versions of types not used in the project may be deleted.

See also

Release versions (Page 516)

Updating the project to the latest versions

After you have updated several types in the project library, update all instances in the project to the most recent version of the types from the project library. If you do not want to apply the changes to the entire project, restrict the update to individual devices in the project.

Each of the following elements can be selected as source for the update:

- The entire project library
- Individual folders within the project library
- Individual types
  You can select multiple types.

Requirement

Your are in the "Libraries" task card or in the library view.

Procedure

To update instances in a project with the contents from the project library, follow these steps:

1. Select the entire project library or elements from it.
2. Right-click the required elements and select the "Update > Project" command from the shortcut menu.
   The "Update project" dialog box opens.
3. Select either the entire project or individual devices for the update.
4. Select the options for the update process:
   - The "Update all instances of the affected types" check box is always selected during this process.
   - Select the "Delete all unused versions of affected types" check box to delete all older versions of the updated types from the project library.
5. Click "OK" to confirm.
   The update is performed.

Result

The following changes were made to the project:

- All older versions were deleted from the project library as needed.
- All instances within the selected devices were updated to the most recent version of the linked types.
- You can find a log of the update process in the projection tree under "Common data".

See also

Updating the project to the latest type versions (Page 523)

Using logs (Page 376)

Updating a library with the contents of another library (Page 528)

Basics on types (Page 501)

Library basics (Page 470)

State of type versions (Page 503)

Remove the link between instance and type

Instances of types are always connected to the version of the corresponding types. They cannot be edited like an ordinary object. If you edit the instance, a new version of the type is created automatically in the library and the changes therefore affect the entire project.

If you remove the link between the instance and its type, you then edit the object like an ordinary object in the project tree.

Requirement

The instance may not be "in test".

Procedure

To remove the link between instances and their type versions, follow these steps:

1. Select one or more instances in the project tree.
2. Right-click the selection and select the "Terminate connection to type" command from the shortcut menu.
3. The link to the corresponding type versions is removed.

See also

Basics on types (Page 501)

Library basics (Page 470)

State of type versions (Page 503)

9.11.8.6 Working with types in global libraries

Adding types to a global library

Global libraries are used as a central resource when working on multiple projects. Among the types, only the types in the project library can be edited directly. Therefore, use the project library if you want to work on types. When you are finished editing a type in the project library, you can add the type to a global library. Adding types from the project library corresponds to a normal copy process from the project library.

Requirement

- The "Libraries" task card is displayed or the library view is open.
- The global library to which you want to add types is opened and can be written to.

Procedure

To add types to a global library, follow these steps:

1. Open the required folder in the global library in the "Libraries" task card or in the library view.
2. Drag one or more types from the project library to the "Types" folder or any subfolder of the global library.

Alternative:

1. Copy the required types from the project library to the clipboard.
2. Open the required global library in the "Global library" palette of the "Libraries" task card.
3. Right-click the "Types" folder or any of its subfolders.
4. Select the "Paste" command in the shortcut menu.

Result

The types are inserted in the global library. Dependent types, such as types of HMI user data types or tags, are also copied to the global library, provided they do not already exist there. This ensures that all necessary elements for generating an instance are present in the global library.

If a type already exists in the global library, the described action corresponds to an update of the global library. In this case, the most recent released versions of the types are added to the global library.

See also

Basics on types (Page 501)

Release versions (Page 516)

Assigning a version (Page 524)

Updating the project to the latest type versions (Page 523)

Library basics (Page 470)

Using types

To use types from the global library, create an instance of a particular version of the type at a suitable point in the project. If necessary, you can create uses of several types at the same time. Uses of a type are called instances in the project tree.

Possible points of use for type versions

Suitable points of use for types from the global libraries are:

- Folder in the project tree
  An instance of a type can be created in a folder in the project tree. The folder must be suitable for the particular type. If the type is a user data type, for example, you can only create the instance in the "PLC data types" folder.
- Editor
  An instance can be created from a type in a suitable editor. For example, you can create an instance from the type of a function block in another block. The type of the function block is called from another block in this way.

Linking the instance with the project library

In the project, instances of types from a global library are not linked to the type in the global library. Instead, a copy of the type and its dependent elements is generated in the project library when an instance is created. Dependent elements, for example, can be PLC data types that are referenced in a block. In each case, the copy of the type and the dependent elements in the project library contains the version that you have linked to the instances. If the type or a dependent element already exists in the project library, only the missing version is added in the project library, if necessary.

The instance is finally linked to the copy of the type in the project library. You can only assign a type to a device once irrespective of the version.

Requirement

- A device which supports the category of the type is already available in the project.
- The device is not assigned to any further instances of the same type.

Procedure

To use the version of a type in the project, follow these steps:

1. In the global library, select the versions from which you want to create an instance.
2. Using a drag-and-drop operation, move the desired versions of the types to the point of use.
   Alternative: To automatically use the newest version, move the types themselves from the library to the point of use using a drag-and-drop operation.
   For example, use a drag-and-drop operation to move the type of a function block to the block folder of the CPU in the project tree. To call the type directly from another block, for example, move the type from the library to the point of use in the program editor using a drag-and-drop operation.

Result

Missing types or individual versions are added in the project library. If a type is not yet present in the project library, it is stored in the same folder as before in the global library. An instance is created from the types and their dependent elements and inserted at the point of use. The instances are connected to the respective type version in the project library.

If you have created the instances in an editor, instances of the types are also created at the appropriate points in the project tree. The folder structure from the library is reproduced in the project tree. You will therefore find the instances in the same folders as in the global library.

See also

Basics on types (Page 501)

Updating the project to the latest type versions (Page 523)

Library basics (Page 470)

Using the element view (Page 474)

Updating the project to the latest type versions

In large enterprises with numerous automation projects. the global libraries are frequently edited from a central location. The updated global libraries of the individual projects are made available after the completion of a new version. If you have received a more recent version of a global library, replace the outdated instances in your project with the most recent version. If you do not want to apply the changes to the entire project, restrict the update to individual devices in the project.

The project library is also updated with the new versions of the types in the global library during the updating of the project or individual devices.

The following elements can be selected as source for the updating:

- A global library
- Individual folders within a global library
- Individual types
  You can select multiple types.

Requirement

- Your are in the "Libraries" task card or in the library view.
- The updated global library is open.

Procedure

To update instances in a project with the contents from a global library, follow these steps:

1. Select the updated global library or the individual elements from it.
2. Right-click the global library or the required elements and select the "Update > Project" command from the shortcut menu.
   The "Update project" dialog box opens.
3. Select either the entire project or individual devices for the update.

4. Select the options for the update process:
   - The "Update all instances of the affected types" check box is always selected during this process.
   - Select the "Delete all unused versions of affected types" check box to delete all older versions of the updated types from the project library.
5. Click "OK" to confirm.
   The update is performed.

Result

The following changes were made to the project:

- The most recent version of the select types is present in the project library. All older versions were deleted if necessary.
- All instances within the selected devices were updated to the most recent version of the linked types.
- You can find a log of the update process in the projection tree under "Common data".

See also

Updating the project to the latest versions (Page 518)

Using logs (Page 376)

Updating a library with the contents of another library (Page 528)

Basics on types (Page 501)

Library basics (Page 470)

Adding types to a global library (Page 520)

9.11.8.7 Assigning a version

A library is more clearly structured if types related by content have the same version number. The identical version number reflects the work progress. When you have completed the work on multiple associated types, you can assign the same version number to these types.

You have the following options to assign a common version to types:

- The entire project library or a complete global library
- One or more folders in a library
- One or more types

Requirement

- The "Libraries" task card or the library view is open.
- Your selection must not contain types with "in test" or "In progress" status.

Procedure

To assign several types the same version, follow these steps:

1. Select the types to which you want to assign a common version.
2. Select the "Assign version" command from the shortcut menu.
   The "Assign version" dialog box opens.
3. If necessary, change the properties of the version:
   - In the "Version" field, determine the new version number. The version number must be higher than the highest version number of all selected types.
   - In the "Author" field, enter the person responsible for the version to be released.
   - In the "Comment" field, enter a comment on the version to be released.
4. Click "OK" to confirm.

Result

The selected type versions are changed as follows:

- A new version of all selected types is created with the specified version number.
- The properties are applied to all selected types, the new version and to all future versions. Lower versions remain unaffected by the changes. If you make no changes to the properties, the properties of the last released version of each type are applied.
- The build number of dependent types is incremented to the next free build number as long as the dependent types were not in your selection. If you had selected a dependent type as well, the version number you specified will be assigned.

A log of the changes is created. If you have versioned the types in the project library, you find the log in the project tree under "Common data > Logs". If you have versioned types in a global library, you find the log in the "Common data > Logs" folder in the level below the global library.

See also

Basics on types (Page 501)

State of type versions (Page 503)

Library basics (Page 470)

Release versions (Page 516)

Adding types to a global library (Page 520)

Using logs (Page 376)

Displaying logs of global libraries (Page 489)

9.11.9 Editing library elements

Types, master copies and folder can be cut, copied, pasted, moved, renamed or deleted in the usual way within the "Libraries" task card or the library view. Global libraries must be opened with write permission for each of the above-described processes.

Note

User-defined documentation for types and master copies

User-defined documentation is not affected by any of the operations within the library. If you move a master copy or a type to a different location, you also move the associated user-defined documentation in the file system to the corresponding location.

For additional help on using the user-defined documentation, refer to the chapter "Using user-defined documentation (Page 361)".

Copying types

The following rules apply when you copy types to the clipboard:

- Types are always copied to the clipboard with all associated versions. However, only versions that have previously been released are copied.
- Types are always copied to the clipboard with all dependent elements.
- Master copies are always copied to the clipboard with all type versions used in them.

Copying and pasting type versions

When you copy type versions to another library, the types must already exist in the target library.

Cutting elements

You can only paste previously cut library elements into the same library. In so doing, you can only paste master copies into the "Master copies" folder or any of its subfolders. Likewise, you can only paste types into the "Types" folder or any of its subfolders.

Pasting types

Pasting types in a different library corresponds to updating the target library.

The following rules apply when you have pasted a type into a different library:

- A type is always pasted with all its versions.
- If the type already exists in the target library, all versions that are more recent than the existing versions are added to the corresponding types in the target library.
- If there is already a version with released status in the target library, this version is not pasted again.

- If the same version exists with in test or in progress status in the target library, it is replaced by the released version.
- If a type needs other types, these are also added at the respective location.

Pasting master copies

When you paste master copies, all type versions contained in these copies are also pasted. If the corresponding types already exist in the library, only the missing versions are added to the individual types. If one of the types used does not yet exist, it is pasted at the highest level in the library. The type includes the type version that was used in the master copy.

Moving elements

When you move an element from one library to another, the element is copied and not moved. The same rules apply as described under "Pasting types" and "Pasting master copies".

Deleting of types and type versions

Note the following when you delete types or type versions:

- A type or a type version can only be deleted if there are no dependencies to other types.
- When you delete a type, all versions of the type are deleted.
- If you delete all versions of a type, the type is also deleted.
- If you delete a version that has instances in the project, the instances are also deleted from the project.
- If you delete a type that is also stored at the same time as a master copy, the master copy is also deleted.

Deleting instances

If you delete an instance that has dependencies to other instances, this instance is restored during the next compilation. The instance is then linked again to the original type version. This restores the consistency of the project.

See also

Library basics (Page 470)

Remove the link between instance and type (Page 520)

Updating a library with the contents of another library (Page 528)

Conventions for the creation (Page 367)

Using user-defined documentation (Page 361)

Duplicating types (Page 509)

9.11.10 Updating a library with the contents of another library

An existing library can be updated with the contents of another library. The following options are available for updating libraries:

- Updating a global library with types from another global library or from the project library
- Updating the project library with types from a global library

Each of the following elements can be selected as source for the update:

- An entire library
- Individual folders within a library
- Individual types
  You can select multiple types.

During the update, new versions are added to the types that already exist in the target library. Types that do not yet exist in the target library are copied together with all their versions to the target library.

Note

User help for copying types

User help is not copied together with a type to another library. You need to copy the user help for types to the corresponding directory.

Additional help on using the user help can be found in the chapter "Conventions for creating user help (Page 367)".

Requirement

If you want to update a global library, you have to open it with write permission.

Procedure

To update a library with the contents of a different library, follow these steps:

1. Select a library or individual elements from a library as source for the update.
2. Right-click the source and select the "Update > Library" command from the shortcut menu. The "Update library" dialog box opens.
3. Select the type of library you want to update:
   - Select "Update the project library" to update the project library with types from a global library.
   - Select "Update a global library" if you want to update a global library.
4. Optional: In the drop-down list, select the global library that you want to update, if you want to update a global library.

5. Select the options for the update:
   - The "Update all instances of the affected types" option is always disabled during this process.
   - Select the "Delete all unused versions of the affected types" check box to delete all older versions of types from the project library if these are not assigned to any instance in the project and if there are no dependencies to other types. This option cannot be selected for the update of a global library, since types of a global library never have a point of use in the project.
6. Click "OK" to confirm.
   The update is performed.

Result

The following changes were made to the target library:

- Types not yet available in the target library were copied together with all their versions. More recent versions were added to the types that already exist in the target library. If a more recent version of a type already existed in the target library, the latest version was nevertheless copied and automatically assigned a newer version number.
- If needed, all versions of types were deleted from the project library if these were not used in any instance in the project.
- A log listing all performed changes to the target library was created for the update process. If you have updated the project library, you can find the log in the project tree under "Common data > Logs".
  If you have updated a global library, you can find the log in the "Common data > Logs" folder in the level below the global library.

See also

Using logs (Page 376)

Updating the project to the latest versions (Page 518)

Updating the project to the latest type versions (Page 523)

Displaying logs of global libraries (Page 489)

Library basics (Page 470)

Using user-defined documentation (Page 361)

9.11.11 Harmonizing names and path structure

You can harmonize the project with a library. This helps you correct the following items:

- Names of the instances:
  Instances can be created during the development phase of a library, the names of which are appended by "_1", "_2", etc. due to an automatic correction. This extension is used to prevent duplicate names in the project. During harmonization, the instances once again receive the names of their associated types.
- Path structure:
  The original path structure can be lost due to parallel development or copying of dependent instances. This affects the clarity of the project. During harmonization, the path structure within the project is adapted to the path structure of the library.

Procedure

To harmonize the names and the path structure, follow these steps:

1. Open the library management.
2. Click "Harmonize project" in the toolbar.
   The "Harmonize project" dialog box opens.
3. Select the device with which you wish to harmonize the library.
4. Select the "Harmonize paths between project and library" check box if you want to restore the path structure.
5. Select the "Harmonize names between project and library" check box if you want to have the names corrected.
6. Confirm your entries with "OK".

Result

Depending on your settings, the names and the path structure in the project are harmonized with the library.

The changes to the project are logged. The log is available under "Common data > Logs" in the project navigation.

See also

Library basics (Page 470)

Overview of the library view (Page 475)

Overview of the library management (Page 478)

Using logs (Page 376)

9.11.12 Clean up library

You can clean up the project library or global libraries to remove types or versions that are not connected to any instance in the project. This step provides more clarity within the libraries and decreases the size of the library.

Cleaning up the project library

To clean up the project library, follow these steps:

1. Open the library management.
2. Click on "Clean up library" in the toolbar.
   The "Clean up project library" dialog box opens.
3. Select the scope in which types or versions are to be deleted:
   - to retain the version with the highest version number, even if this has no instance, select the option "Delete old type versions and retain the newest type version".
   - Select the option "Delete complete types" to delete the complete type if no version is connected to an instance.
4. Confirm your entry with "OK".
   Depending on your selection, either unused type versions or types are removed from the project library.
   The changes are logged. The log is available under "Common data > Logs" in the project navigation.

Clean up global library

To clean up a global library, follow these steps:

1. Open the library management.
2. Click on "Clean up library" in the toolbar.
   The "Clean up global library" dialog box opens.
3. Click "Continue".
   Unused type versions are deleted. The latest version of a type is always retained.
   The changes are logged. The log is available in the "Common data > Logs" folder in the level below the global library.

See also

Library basics (Page 470)

Overview of the library view (Page 475)

Overview of the library management (Page 478)

Using logs (Page 376)

Displaying logs of global libraries (Page 489)

9.11.13 Comparing library elements

Introduction

You can compare devices from libraries with devices from both the current project as well as from the same or another libraries or reference projects. Note, however, that reference projects are write-protected. You can also compare instances in a device with their type version in a library. Not all actions are available for the comparison with types. You cannot, for example, overwrite an instance of a newer version with an older type version from the library.

When comparing library elements, you can always switch between automatic and manual comparison.

Procedure

To compare library elements with the device data of a project, follow these steps:

1. In the project tree, select the device whose data you want to compare to a library element and which allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Open the "Libraries" task card.
4. Select the library element that you want to compare to the device data.
5. Drag the library element into the right drop area of the compare editor.
   You can identify the status of the objects based on the symbols in the status and action area. When you select an object, the object properties and the corresponding object of the assigned device are clearly shown in the property comparison.
   You can drag other devices into the drop areas from the current project, a library or from a reference point at any time to start a new comparison. It does not matter which device you drag into the drop area.

See also

Using the compare editor (Page 405)

Carrying out offline/offline comparisons (Page 404)

Using the library view (Page 475)

Library basics (Page 470)

Overview of the "Libraries" task card (Page 472)

Overview of the library view (Page 475)

Using the element view (Page 474)

Using global libraries (Page 483)

9.12 Using cross-references

9.12.1 Using cross-references

Introduction to cross-references

The cross-reference list provides an overview of the use of objects within the project. You can see which objects are interdependent and where the individual objects are located. Cross-references are therefore part of the project documentation.

You can also jump directly to the point of use of an object.

Which objects you can display and localize in the cross-reference list depends on the installed products.

See also

Displaying cross references of an instance (Page 482)

9.13 Simulating devices

9.13.1 Simulation of devices

Introduction

You can use the TIA Portal to run and test the hardware and software of the project in a simulated environment. The simulation is performed directly on the programming device or PC. No additional hardware is required.

The simulation software provides a graphical user interface for monitoring and changing the configuration. It differs according to the currently selected device.

Integration in the TIA Portal

The simulation software is fully integrated in the TIA Portal but is only supported by certain devices. Therefore, the button for calling the simulation software is only active if the selected device supports simulation.

The simulation software for some devices requires its own virtual interface to communicate with the simulated devices. The virtual interface can be found in the project tree under the "Online access" entry next to the physical interfaces of the programming device/PC.

Once you have opened the software, additional help on the simulation software is available via a separate link.

See also

Starting the simulation (Page 534)

9.13.2 Starting the simulation

Some devices can be simulated with additional software. You therefore do not have to have the actual devices to perform comprehensive testing of your project.

Procedure

To start the simulation software, follow these steps:

1. Select the device you want to simulate, for example, in the project tree.
2. Select the "Simulation > Start" command in the "Online" menu.
   This calls the simulation software.

See also

Simulation of devices (Page 533)

Editing devices and networks

10

10.1 Configuring devices and networks

10.1.1 Hardware and network editor

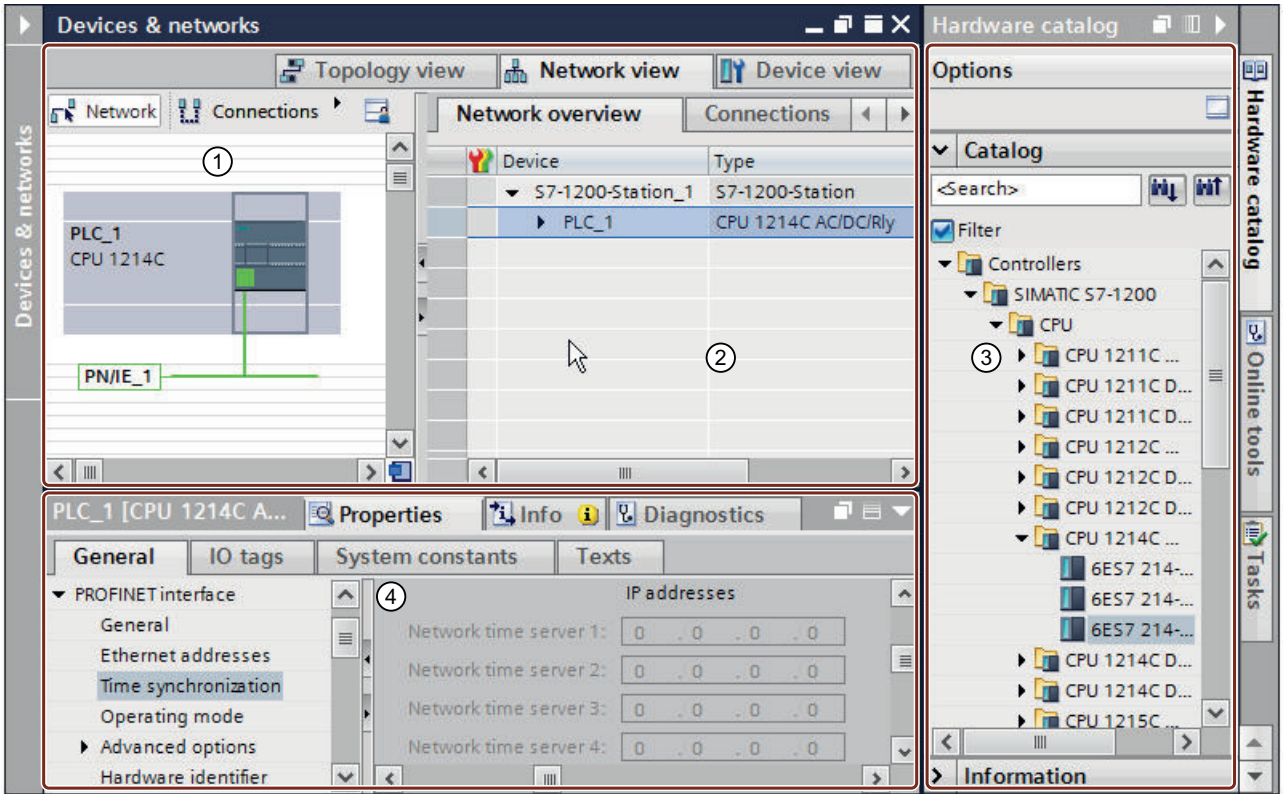
10.1.1.1 Overview of hardware and network editor

Function of the hardware and network editor

The hardware and network editor opens when you double-click on the "Devices and Networks" entry in the project tree. The hardware and network editor is the integrated development environment for configuring, networking and assigning parameters to devices and modules. It provides maximum support for the realization of the automation project.

Structure of the hardware and network editor

The hardware and network editor consists of the following components:



① Device view (Page 539), Network view (Page 537), Topology view (Page 542): Graphic area

② Device view (Page 539), Network view (Page 537), Topology view (Page 542): Table area

③ Hardware catalog (Page 548)

④ Inspector window (Page 547)

The hardware and network editor provides you with three views of your project. You can switch between these three views at any time depending on whether you want to produce and edit individual devices and modules, entire networks and device configurations or the topological structure of your project.

Drag the devices and modules you need for your automation system from the hardware catalog to the network, device ot topology view.

The inspector window contains information on the object currently marked. Here you can change the settings for the object marked.

10.1.1.2 Network view

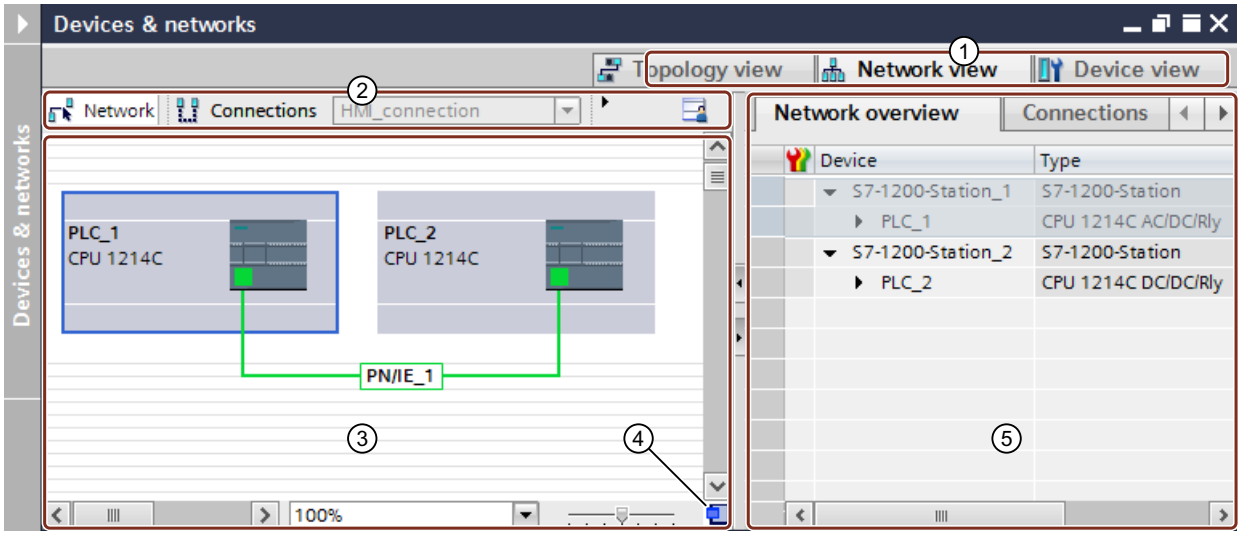
Introduction

The network view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Configuring and assign device parameters
- Networking devices with one another

Structure

The following diagram shows the components of the network view:



① Changeover switch: device view / network view / topology view
② Toolbar of network view
③ Graphic area of network view
④ Overview navigation
⑤ Table area of network view

You can use your mouse to change the spacing between the graphic and table areas of the network view. To do this, click between the graphic and the table areas and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize the table view, maximize the table view or restore the last selected split.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Mode to network devices.
	Mode to create connections. You can use the adjacent drop-down list to set the connection type.
	Mode to create relations.
	Opens the dialog for manual name assignment for PROFINET devices. For this purpose the IO device must be inserted and connected online with the IO system.
	Show interface addresses.
	Enables page break preview. Dotted lines are displayed at the positions where the pages break when printed.
	You can zoom in (+) or zoom out (-) the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
	Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the network view displays any network-related devices, networks, connections and relations. In this area, you add devices from the hardware catalog, connect them with each other via their interfaces and configure the communication settings.

The operator controls for view control are located at the bottom edge of the graphic area:

- Select the zoom leveling using the drop-down list. You can also enter a value directly into the field of the drop-down list.
- You can also set the zoom level using the slider.
- You can re-focus the window of the graphic area using the icon in the bottom right corner.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate in the overview navigation to the desired objects and display them in the graphic area.

Table area

The table area of the network view includes various tables for the devices, connections and communication settings present:

- Network overview
- Connections
- Relations

- IO Communication
- VPN

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Adding a device to the hardware configuration (Page 565)

Layout of the user interface (Page 307)

Displaying diagnostics status and comparison status using icons (Page 1356)

Networking devices in the network view (Page 582)

Tabular network overview (Page 585)

10.1.1.3 Device view

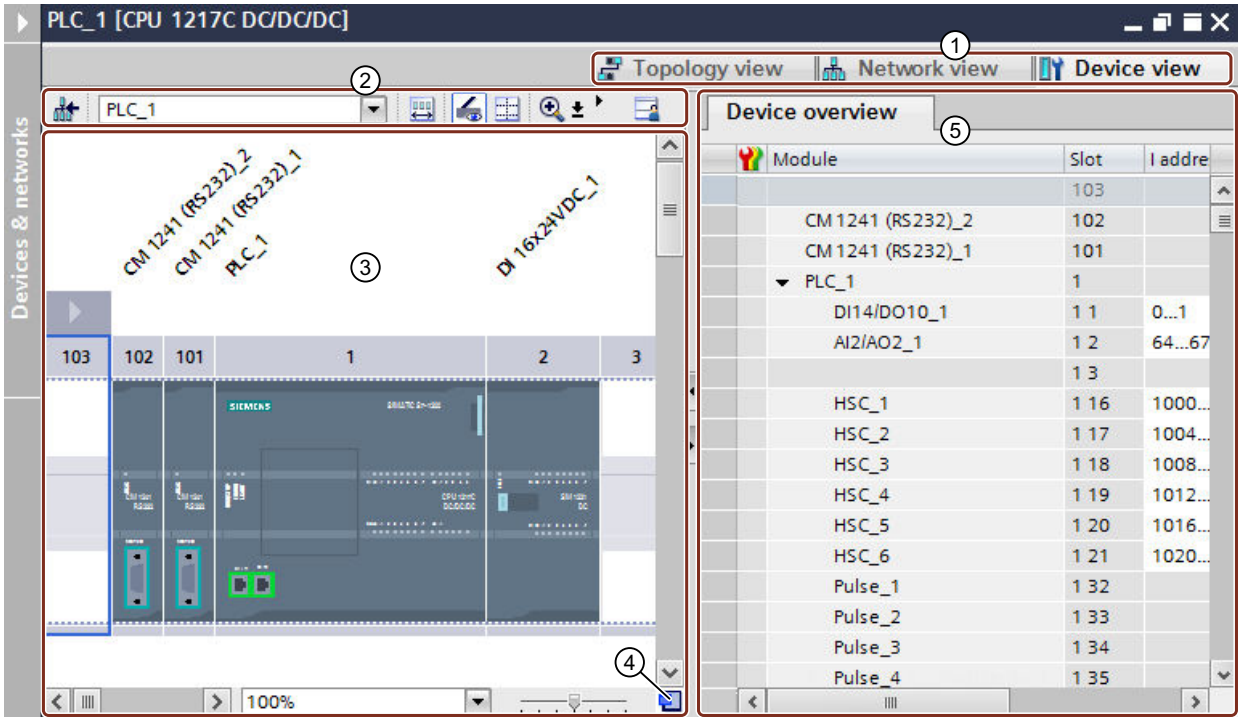
Introduction

The device view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Configuring and assign device parameters
- Configuring and assign module parameters

Structure

The following diagram shows the components of the device view:



① Changeover switch: device view / network view / topology view
② Toolbar of device view
③ Graphic area of the device view
④ Overview navigation
⑤ Table area of device view

You can use your mouse to change the spacing between the graphic and table areas of the device view. To do this, click between the graphic and the table areas and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize the table view, maximize the table view or restore the last selected split.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Switches to the network view. The device view can switch between the existing devices using the drop-down list.
	Show the area of unplugged modules.
	Opens the dialog for manual name assignment for PROFINET devices. For this purpose the IO device must be inserted and connected online with the IO system.
	Show module labels.

Icon	Meaning
	Enables page break preview. Dotted lines are displayed at the positions where the pages break when printed.
	You can use the Zoom icon to zoom in (+) or out (-) incrementally or to drag a frame around an area to be enlarged. With signal modules, you can recognize the address labels of the I/O channels from a zoom level of 200% or higher.
	Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the device view displays hardware components and if necessary the associated modules that are assigned to each other via one or more racks. In the case of devices with racks, you have the option of installing additional hardware objects from the hardware catalog into the slots on the racks.

The operator controls for view control are located at the bottom edge of the graphic area:

- Select the zoom leveling using the drop-down list. You can also enter a value directly into the field of the drop-down list.
- You can also set the zoom level using the slider.
- You can re-focus the window of the graphic area using the icon in the bottom right corner.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate in the overview navigation to the desired objects and display them in the graphic area.

Table area

The table area of the device view gives you an overview of the modules used and the most important technical and organizational data.

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Working with racks (Page 558)

Network view (Page 537)

Area for unplugged modules (Page 563)

Inserting a module into a rack (Page 568)

Objects in the device view (Page 560)

Layout of the user interface (Page 307)

Displaying diagnostics status and comparison status using icons (Page 1356)

10.1.1.4 Topology view

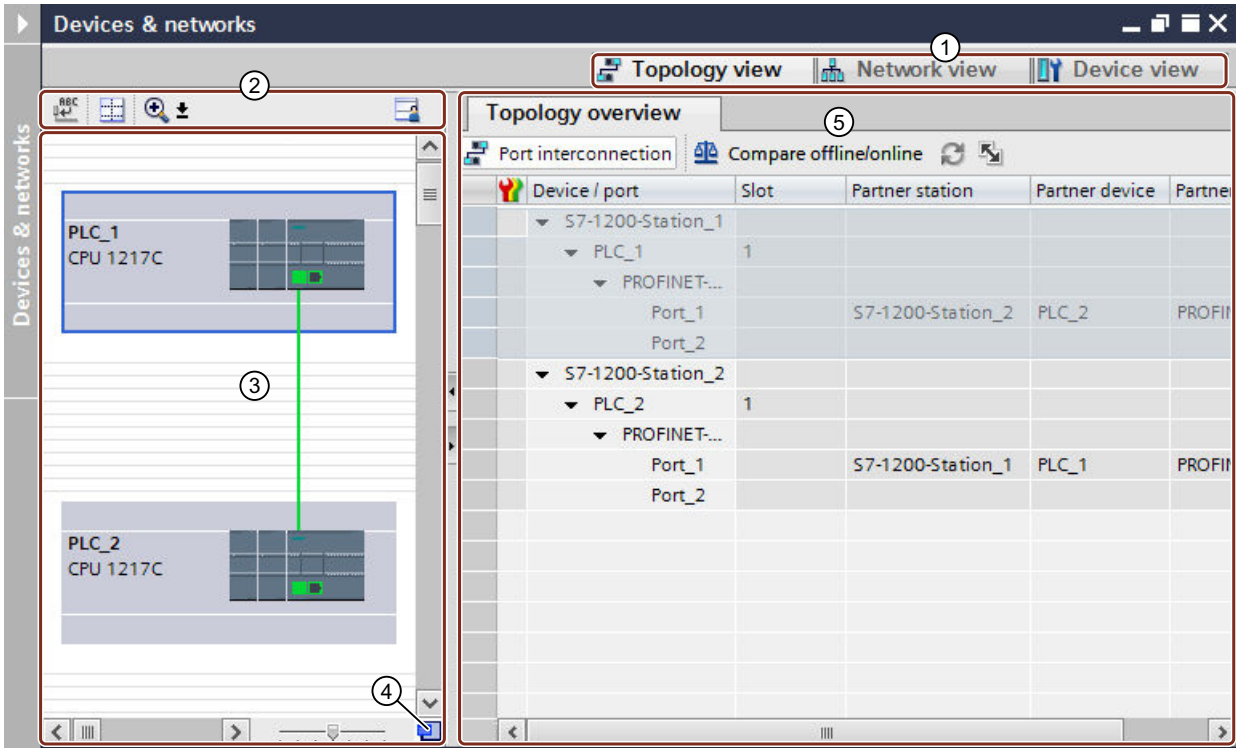
Introduction

The topology view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Displaying the Ethernet topology
- Configuring the Ethernet topology
- Identifying and minimizing differences between the desired and actual topology

Structure

The following figure provides an overview of the topology view.



① Changeover switch: device view / network view / topology view
② Topology view toolbar
③ Graphic area of the topology view
④ Overview navigation
⑤ Table area of the topology view

You can use your mouse to change the spacing between the graphic and table areas of the topology view. To do this, click between the graphic and the table areas and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize the table view, maximize the table view or restore the last selected split.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Opens the dialog for manual name assignment for PROFINET devices. For this purpose the IO device must be inserted and connected online with the IO system.
	Enables page break preview. Dotted lines are displayed at the positions where the pages break when printed.
	You can zoom in (+) or zoom out (-) the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
	Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the topology view displays Ethernet modules with their appropriate ports and port interconnections. Here you can add additional hardware objects with Ethernet interfaces. See: Adding a device to the hardware configuration (Page 565)

The operator controls for view control are located at the bottom edge of the graphic area:

- Select the zoom leveling using the drop-down list. You can also enter a value directly into the field of the drop-down list.
- You can also set the zoom level using the slider.
- You can re-focus the window of the graphic area using the icon in the bottom right corner.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate in the overview navigation to the desired objects and display them in the graphic area.

Table area

This displays the Ethernet or PROFINET modules with their appropriate ports and port interconnections in a table. This table corresponds to the network overview table in the network view.

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Layout of the user interface (Page 307)

Displaying diagnostics status and comparison status using icons (Page 1356)

10.1.1.5 Overview of settings for hardware configuration

You can make some hardware configuration settings in the TIA Portal settings.

Overview

The following table provides an overview of the settings for the hardware configuration:

Group	Setting	Description
Information on product support	Deactivated	Prevents access to the Siemens Industry Online Support
	Via Internet	Enables access to product information about individual devices in the hardware catalog via the Internet.
Topological overview	Temporarily assigning an IP address	Assigns a temporary IP address for topology discovery if a device does not have a valid IP address. Topology information (LLDP) cannot be read from a device without valid IP address.
	Show a warning if the option is activated	Shows a warning when a temporary IP address is assigned to a device during topology discovery.
Compiling	Download module comment	Transfers any existing comments in addition to the hardware configuration when loading the hardware configuration to the device. The comment is available after the device is loaded to a programming device.

See also

Changing the settings (Page 306)

10.1.1.6 Printing hardware and network configurations

Printout of hardware and network configurations

You can print out the following elements of the hardware and network view as part of the project documentation:

- Graphic network view
- Network overview table
- Graphic device view
- The device overview table
- The parameters of the object currently selected in the editor

Printout of editor content

If you start a printout within an opened editor and no module is selected, the content of the editor is always printed. This includes a graphic representation of the editor as well as the table for the editor. You can adapt the scope of the printout. You can specify whether only the graphic view, the table or both together are to be printed. Read section "Changing the print options (Page 546)" for more on this.

If the graphic is larger than the page layout you have selected, the printout is continued on the next page. No content is lost this way. Alternatively, you can change the zoom level of the graphic representation to fit the printout on one page. The printout is always made in the currently selected zoom setting.

To check that all content fits on one page, you can either use the print preview or activate the page break preview. When page break preview is activated, dashed lines are displayed within the graphic editor at the location where the page break is later made.

Printing very large tables

If a table is larger than the print area and therefore cannot be fully printed, the content of the table is not printed as a table, but instead as pairs between value and key.

Example:

Object name	Property 1	Property 2
Object A	Value A1	Value A2
Object B	Value B1	Value B2

In this case, the printout has the following appearance:

Object A

Property 1: Value A1

Property 2: Value A2

Object B

Property 1: Value B1

Property 2: Value B2

You can also preset this as a template so that tables are always printed as pairs between the key and the value. Read section "Changing the print settings (Page 421)" for more on this.

Printing module parameters

Parameters of selected modules are printed out along with the current value settings in text form. All parameters from corresponding modules are also printed. For example, if you have selected a CPU, the parameters of an inserted signal board, if present, are printed as well.

You can determine the scope of the module parameters to be printed. In the "Print" dialog, select whether all properties and parameters of a module are to be printed or whether to use the compact printout. If you select the compact form, only the entries in the "General" area of the module properties are printed. Comments on modules, as well as the author and module description, are excluded. In compact mode, the following module parameters are therefore printed, for example:

- Module specifications
  Name, module slot, short description, article number, firmware version
- Name of the PROFINET interface
- Subnet specifications
  Name of the subnet, ID of the S7 subnet

See also

Changing the print options (Page 546)

Documentation settings (Page 419)

Creating a print preview (Page 433)

Printing project data (Page 436)

Activating the page break preview for printout (Page 546)

10.1.1.7 Activating the page break preview for printout

You can activate the page break preview for the printout in the graphic editor. If this option is activated, dashed lines are shown within the graphic editor at the locations where page breaks are later made during printout.

Procedure

Proceed as follows to activate the page break preview:

1. Select the graphic area of the corresponding view.
2. Click on the "Show page break" symbol in the toolbar of the graphic editor.
   Dashed lines are displayed within the graphic editor at the location a page break is later made.
3. To modify the frame layout, select the "Print" command in the "Project" menu.
4. To disable page break preview, click again on the "Show page break" symbol in the toolbar of the graphic editor.

10.1.1.8 Changing the print options

Changing the scope of the printout

When printing from an editor, you can specify whether both graphics and tables are to be printed or just one of the two. Both are printed by default.

Procedure

To change the scope of the printout, proceed as follows:

1. In the "Options" menu, select the "Settings" command.
2. In the area navigation, open the "Print settings" parameter group under "General".
3. Scroll to the "Hardware configuration" group.
4. Select or clear the "Active graphic view" check box, depending on whether you want to print the graphics of the network and device view as well.
5. Select or clear the "Active table" check box, depending on whether you want to print the table for the editor as well.

See also

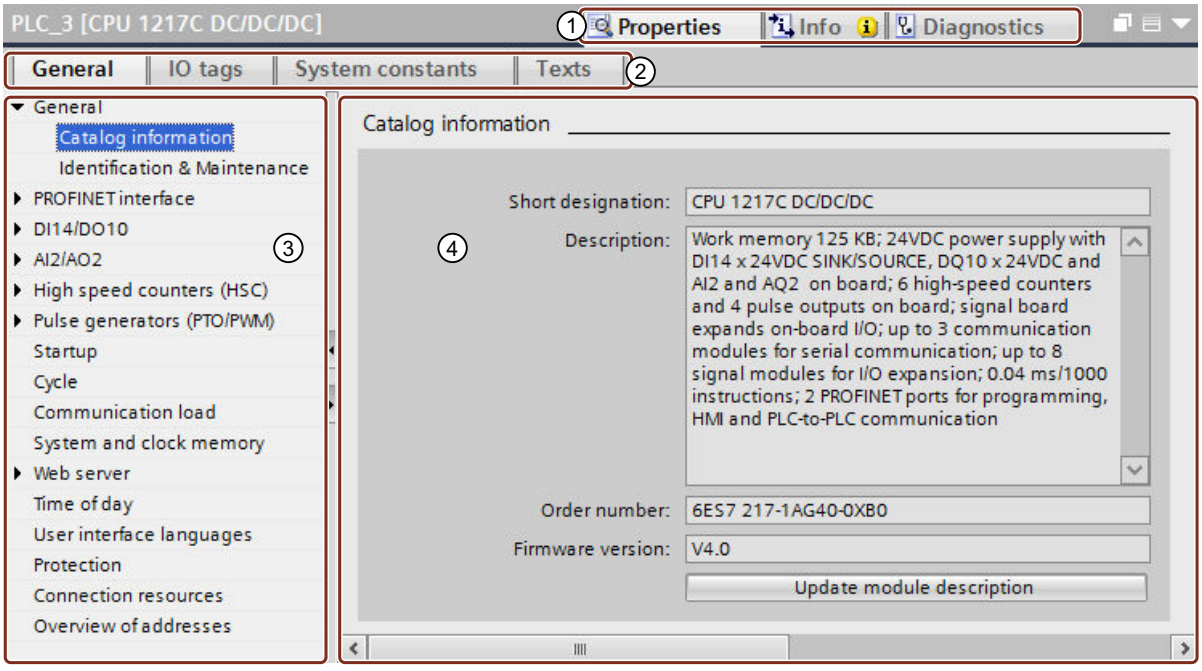
Printing hardware and network configurations (Page 544)

10.1.1.9 Inspector window

The properties and parameters shown for the object selected can be edited in the Inspector window.

Structure

The Inspector window consists of the following components:



① Switch between various information and work areas
② Switch between various tabs of the information and work areas
③ Navigation between various pieces of information and parameters
④ Display showing the selected information and parameters

Function

The information and parameters in the Inspector window are split into different types of information:

- Properties
- Info
- Diagnostics

To display the corresponding information and parameters, click in the relevant area. The "Properties" area is the most important one for configuring an automation system. This area is displayed by default and contains various tabs:

- General: Display the properties and settings of the device or module. Here you can edit the settings and parameters. The left pane of the Inspector window is used for area navigation. Information and parameters are arranged there in groups. If you click on the arrow symbol to the left of the group name, you can expand the group if sub-groups are available. If you select a group or sub-group, the corresponding information and parameters are displayed in the right pane of the Inspector window and can be edited there too.
- IO tags: Display the IO tags of the PLC. You can assign names for the tags, assign the tags to the user-defined tag tables via a drop-down list, and enter comments for the tags. The IO tags are also shown in the PLC tag table.
- System constants: Display the constants required by the system with the HW IDs of the modules. The system constants are also shown in the PLC tag table.
- Texts: Display the reference language and specify the text source for the project texts.

See also

Editing properties and parameters (Page 574)

Overview of hardware and network editor (Page 535)

Translating text associated with individual objects (Page 458)

Project text basics (Page 454)

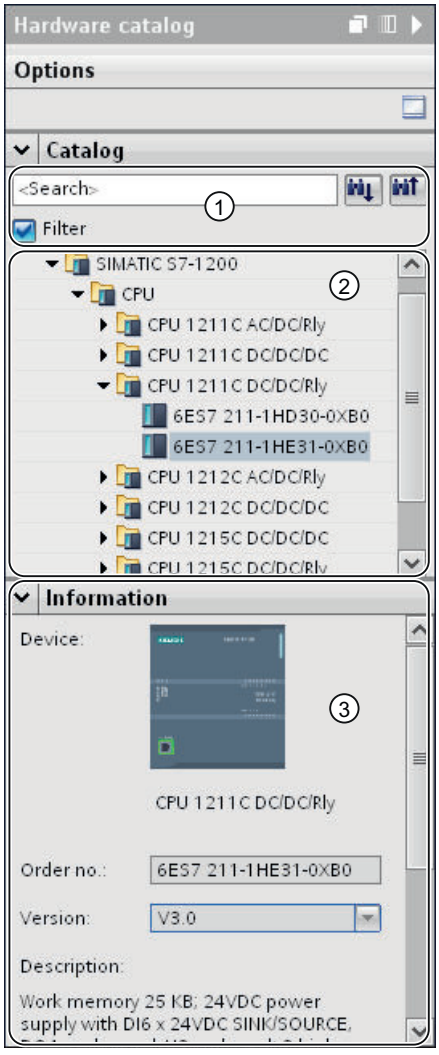
Addressing modules (Page 848)

10.1.1.10 Hardware catalog

The "Hardware catalog" task card gives you easy access to a wide range of hardware components.

Structure

The "Hardware catalog" task card consists of the following panes:



① "Catalog" pane, search and filter function
② "Catalog" pane, component selection
③ "Information" pane

Search and filter function

The search and filter functions of the "Catalog" pane make it easy to search for particular hardware components. You can limit the display of the hardware components to certain criteria using the filter function. For example, you can limit the display to objects that you can also place within the current context or which contain certain functions.

Objects that can be used in the current context include, for example, interconnectable objects in the network view or only modules compatible with the device in the device view.

Component selection

The component selection in the "Catalog" pane contains the installed hardware components in a tree structure. You can move the devices or modules you want from the catalog to the graphic work area of the device or network view.

Installed hardware components without a license are grayed out. You cannot use non-licensed hardware components.

Hardware components belonging to various components groups thematically are partially implemented as linked objects. When you click on such linked hardware components, a catalog tree opens in which you can find the appropriate hardware components.

Information

The "Information" pane contains detailed information on the object selected from the catalog:

- Schematic representation
- Name
- Article number
- Version number
- Description

See also

Browsing the hardware catalog (Page 557)

Overview of hardware and network editor (Page 535)

10.1.1.11 Enabling product support

For each device in the hardware catalog, you have the option of displaying additional information that is stored in the Siemens Industry Online Support. By default, the function is disabled. Instructions for enabling the function are given below.

Requirement

The TIA Portal has access to the Internet.

Procedure

To enable access to Siemens Industry Online Support, follow these steps:

1. In the "Options" menu, select the "Settings" command.
2. Open the "Hardware configuration" group in the area navigation.
3. Select the "Via Internet" check box.

Result

You can access product support, FAQs and manuals in the hardware catalog via the shortcut menu for the module.

See also

Displaying product support for hardware components (Page 551)

10.1.1.12 Displaying product support for hardware components

In the hardware catalog, you have direct access to information that is stored for each module in Siemens Industry Online Support. You can jump directly to the following pages in Siemens Industry Online Support:

- Information on product support
- FAQs
- Manuals

Requirement

- You have access to the Internet.
- Access to Product Support is enabled in the settings of the TIA Portal.
  For information on how to enable the function, refer to section "Enabling product support (Page 550)".

Procedure

To display the information for a particular module in Siemens Industry Online Support, follow these steps:

1. Navigate to the required module in the hardware catalog.
2. Right-click the module.
3. Select one of the following entries in the shortcut menu:
   - Information on product support
   - FAQs
   - Manuals

Result

The default browser set in the operating system is opened and the relevant page in the Siemens Industry Online Support is loaded.

See also

Enabling product support (Page 550)

10.1.1.13 Keyboard operation: Navigation in the editor

You can use shortcut keys in the network and device view to navigate between the components of the hardware and network editor and its objects.

Navigating between elements and functions

Function	Shortcut keys
Switch to the next lower selection level You can for example, use <Return> to switch from a selected rack to the lower selection level of the devices and modules that are snapped onto it. If a device is selected, you can use <Return> to switch to the lower selection level of the interfaces that are displayed on the device.	<Return>
Switch to the next higher selection level You can use <Esc>, for example, to switch from a selected interface to the higher selection level of the devices and modules. If a device is selected, you can use <Esc> to switch to the higher selection level of the rack.	<Esc>
Navigation between objects in the current selection level You can use the arrow keys to switch between the objects within a current selection level. To change the selection level, use the <Return> or <Esc> keys.	<Up arrow> <Down arrow> <Right arrow> <Left arrow>
Switches to the device view	<Ctrl+Shift+D>
Switches to the network view	<Ctrl+Shift+N>
Switches to the topology view	<Ctrl+Shift+T>
Switch between editor elements Use the <Tab> key to switch from one editor element to the next element. Use <Shift +Tab> to switch to the previous element. You can switch, for example, between the graphical view, Speedy Splitter, table view or underlying tabs.	<Tab> <Shift+Tab>
Switch between tabs Use the <Ctrl+Tab> keys to switch from one tab to the next tab on the right. Use <Ctrl +Shift+Tab> to switch to the next tab to the left. You can use these keys, for example, to switch between the device view, the network view and the topology view.	<Ctrl+Tab> <Ctrl+Shift+Tab>

Opening elements and functions

Function	Shortcut keys
Opening the online and diagnostics view When a device is selected, <Ctrl+D> opens the online and diagnostics view for the selected device.	<Ctrl+D>
Opening the download to device dialog When a device is selected, <Ctrl+L> opens the advanced download dialog.	<Ctrl+L>
Add new device <Ctrl+N> opens the dialog for adding a new device.	<Ctrl+N>
Opens the "Hardware catalog" task card	<Ctrl+Shift+C>
Opens "Online Tools" task card	<Ctrl+Shift+O>

See also

Keyboard operation in the TIA Portal (Page 339)

10.1.1.14 Keyboard operation: Editing objects

You can execute some of the functions of the network and device view directly with a combination of keyboard and mouse in the hardware and network editor. The keyboard operation in tables (Page 339) corresponds to standard characteristics. Here you find the keyboard operation for the graphic work area of the network and device view.

General keyboard operation

Function	Shortcut keys
Zoom in on view in frame Drag a frame in the graphical view in order to correspondingly change the size of the view.	<Ctrl+Space> + pressed mouse button
Move view Move the mouse pointer in order to move the view.	<Space> + pressed mouse button
Cancel current operation	<Esc>
Separate connection Use <Esc> or a double-click to exit connection mode when dragging a connection.	<Esc> or double-click
Zoom in graphic view The enlargement or reduction depends on the direction of rotation.	<Ctrl> + turn mouse wheel

Selected objects

Function	Shortcut keys
Select object	Mouse click
Cut an object The selected object is copied to the clipboard and deleted from the graphical view.	<Ctrl+X>
Copy object The selected object is copied to the clipboard.	<Ctrl+C>
Paste object The object from the clipboard is inserted into the selection.	<Ctrl+V>
Delete selected object	<Del>
Select several objects 1 You can add several objects to the selected objects by clicking on them individually. Alternatively, you can use <Shift> + pressed mouse key to drag a frame around the objects that are to be selected.	<Shift> + click
Select several objects 2 You can add several objects to the selected objects by clicking on them individually. Alternatively, you can use <Shift> + pressed mouse key to drag a frame around the objects that are to be selected. When holding down the <Ctrl> key, you can use a mouse click to deselect selected objects.	<Ctrl> + click

Function	Shortcut keys
Move selection When the mouse button is pressed, you can drag devices or modules to allowed slots on a rack.	Mouse button pressed
Copy selection Using <Ctrl> + pressed mouse button you can drag devices and modules to allowed slots on a rack. This copies the devices or modules.	<Ctrl> + pressed mouse button

10.1.2 Configuring devices

10.1.2.1 Basics

Introduction to configuring hardware

To set up an automation system, you will need to configure, assign parameters and interlink the individual hardware components. The work needed for this is undertaken in the device and network view.

Configuring

"Configuring" is understood to mean arranging, positioning, and networking devices and modules within the device or network view. Racks are represented symbolically. Just like "real" racks, they allow you to plug in a defined number of modules.

An address is automatically assigned to each module. The addresses can be subsequently modified.

When the automation system is started, the CPU compares the configuration that is preset by the software with the actual configuration of the system. Possible errors can be detected and reported straight away.

Assigning parameters

"Assigning parameters" is understood to mean the setting of properties of the components used. Parameter assignment is carried out for hardware components and for data exchange settings:

- Properties of modules with assignable parameters
- Settings for data exchange between components

The parameters are loaded to the CPU and transferred to the corresponding modules when the CPU starts up. Modules can be replaced with ease since the assigned parameters are automatically loaded into the new module during startup.

Adjusting the hardware to the project requirements

You need to configure hardware if you want to set up, expand or change an automation project. To do this, add hardware components to your configuration, connect them to existing components, and adapt the hardware properties to the tasks.

The properties of the automation systems and modules are preset in such a way that in many cases there is no need for additional parameter assignment. Parameter assignment is however needed in the following cases:

- You want to change the default parameter settings of a module.
- You want to use special functions.
- You want to configure communication connections.

See also

Changing properties of the modules (Page 1184)

Using existing configurations

Open existing projects

When opening existing projects, an automatic check is made to determine if the appropriate software is installed for all modules used within the project. If you try to open a project with modules that are not supported by the current scope of installation of the TIA portal, a message appears on opening the project informing you of the missing software components. If the software components are not absolutely required to open the project, the project can still be opened.

Reaction to missing software components

Projects that contain modules not supported by the current scope of the installation react as follows:

- Display the modules on the GUI
  - The non-supported modules are displayed in the project tree with all of their nested objects. However, the modules themselves cannot be processed in editors or in the inspector window. When possible, a replacement module is used that best matches the original module. Replacement modules are indicated by an exclamation mark.
  - Display of properties in tables is limited. This applies in particular to the display of network parameters, such as the IP address.
- Functional limitations
  - Non-supported modules cannot be printed out or compiled.
  - An online connection cannot be established to the module. It is therefore also impossible to download.
  - To change the device type, the device must first be deleted and then re-inserted. The "Change device type" function is not supported.
  - Copying and inserting nested objects, such as modules, is possible although the device itself cannot be copied and inserted.
  - The network configuration cannot be changed with replacement modules within the network view.
  - Cross-references can be displayed. However, the cross-references only reflect the state last saved within the project because on online comparison to the original module cannot be made.

See also

Opening projects (Page 379)

General slot rules

Introduction

Specific slot rules apply to each automation system and module.

If you select a module from the hardware catalog in the device view, all possible slots for the module selected are marked in the rack. You can only drag modules to marked slots.

If you insert, move or swap a module, the slot rules are also applied.

Consistency

Some slot rules depend on how the environment is configured. This means that you can sometimes plug modules into the rack although this would result in inconsistencies at the current time. If you change the configuration, for example by selecting different modules or module parameter settings, you can make the configuration consistent again.

In cases where inserting a module results in an inconsistency that can be corrected, this will be permitted. A consistency check is run when the configuration is compiled. Inconsistencies are displayed as alarms in the Inspector window under "Info". Based on the result of the consistency check, you can revise your configuration and make it consistent again.

Rules for arranging modules

The following rules apply generally to modules in racks:

- You can only plug modules into a rack.
- You can only plug interface modules into a module.
- You can only use modules of the same product or system family in one rack.

There are also other special rules for some modules:

- Can only be inserted in certain slots
- Insertion depends on other modules, CPUs or settings
- Limited number of uses in a rack

Browsing the hardware catalog

Introduction

Use the "Hardware catalog" task card to select the hardware components you want for a hardware configuration. Use the hardware catalog to select the interconnectable hardware components in the network and topology view and to select the modules you want in the device view.

Context filter

You can use the "Filter" option of the hardware catalog to restrict the number of displayed hardware components and the number of hardware components that can be found by searching.

If you select the filter, only those components are displayed that can be selected currently in the hardware catalog. If the do not select the filter, the entire hardware catalog is displayed.

If you switch between the various views, the view of the filter objects is adapted to the current context.

Search options

You can use the search function to search for specific entries in the hardware catalog. Note the following rules when entering search terms:

- No distinction is made between upper and lower case text.
- Dashes and blanks are ignored during the search.
- The search function considers parts of a search term.
- Several search terms must be separated by a space

You start the search from an object highlighted in the hardware catalog and either search upwards or downwards.

Icon	Meaning
	Downwards search
	Upwards search

Browsing the hardware catalog

If you want to browse the hardware catalog, proceed as follows:

1. Click in the entry field of the search function
2. Enter a search term. The search includes the following elements:
   - Name of device or module
   - Article number (MLFB)
   - Description in "Information" pane
3. Click on either the "Downwards search" or "Upwards search" buttons.

Note

To ensure the right search direction, note which point you have marked in the hardware catalog. To browse the entire catalog, click on the topmost object of the hardware catalog and start the search once you have entered the search term by clicking "Downwards search".

The first match with the search term found is displayed as the result. For more search results, again click on the "Downwards search" or "Upwards search" button.

Observe the context filter of the hardware catalog. If this is selected, the search in the hardware catalog is restricted to the displayed inserted hardware components.

See also

Hardware catalog (Page 548)

Working with racks

Introduction

To assign modules to a device, you need a rack, for example a mounting rail. Secure the modules on the rack and connect these via the backplane bus with the CPU, a power supply or other modules.

Creating a rack

If you insert a device in the network view, a station and a rack suitable for the device selected are created automatically. The rack and slots available are displayed in the device view. The number of slots available again depends on the type of device used.

Rack structure

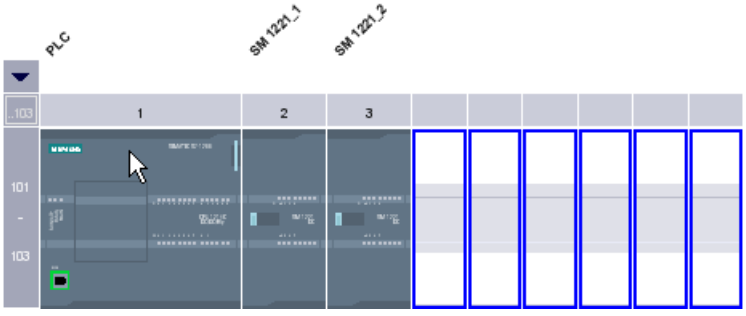
A rack always contains the device that has been inserted in the network view. The device is permanently assigned a slot which will depend on the type of device in question. There are additional slots on the right of the device and, if necessary, on left of the device; slot numbers are located above slots in which devices are plugged.

A corresponding short description is displayed above the plugged devices and modules. You show or hide this short description via the toolbar under "View" with the command "Display module titles" or the corresponding symbol in the toolbar of the device view (Page 539).

Symbol	Meaning
	Show module titles

When modules are selected in the hardware catalog, all the slots permitted for this module are marked. This allows you to see immediately the slot into which the selected module can be inserted.

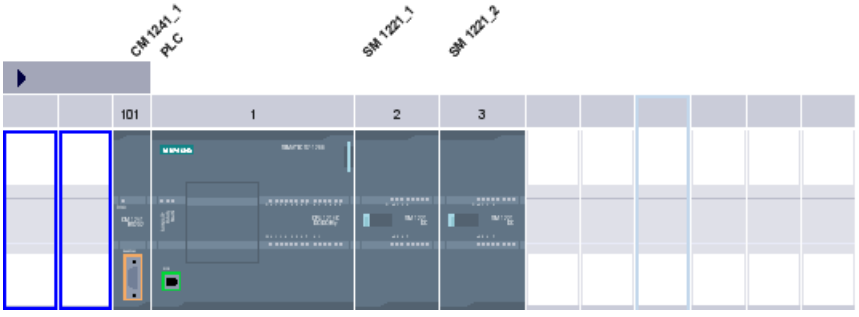
In the following screenshot, a signal module has been selected in the hardware catalog for a partially filled S7-1200 rack:



Since slots 101-103 are reserved for communications modules, only the other free slots are shown as available slots.

You can expand and collapse the front group of slots using an arrow symbol above the expandable slot. When the group of slots is collapsed, the first and last of the group's slot numbers are displayed.

The following figure shows the expanded slot group:



Groups of slots into which modules have already been plugged cannot be collapsed.

Multiple selection of modules and slots

There are various ways of selecting several modules or slots:

- By pressing <Shift> or <Ctrl>, you can select several modules or slots at the same time.
- Click outside the rack and then hold the mouse button and drag a frame to include the modules or slots you want to select.

Objects in the device view

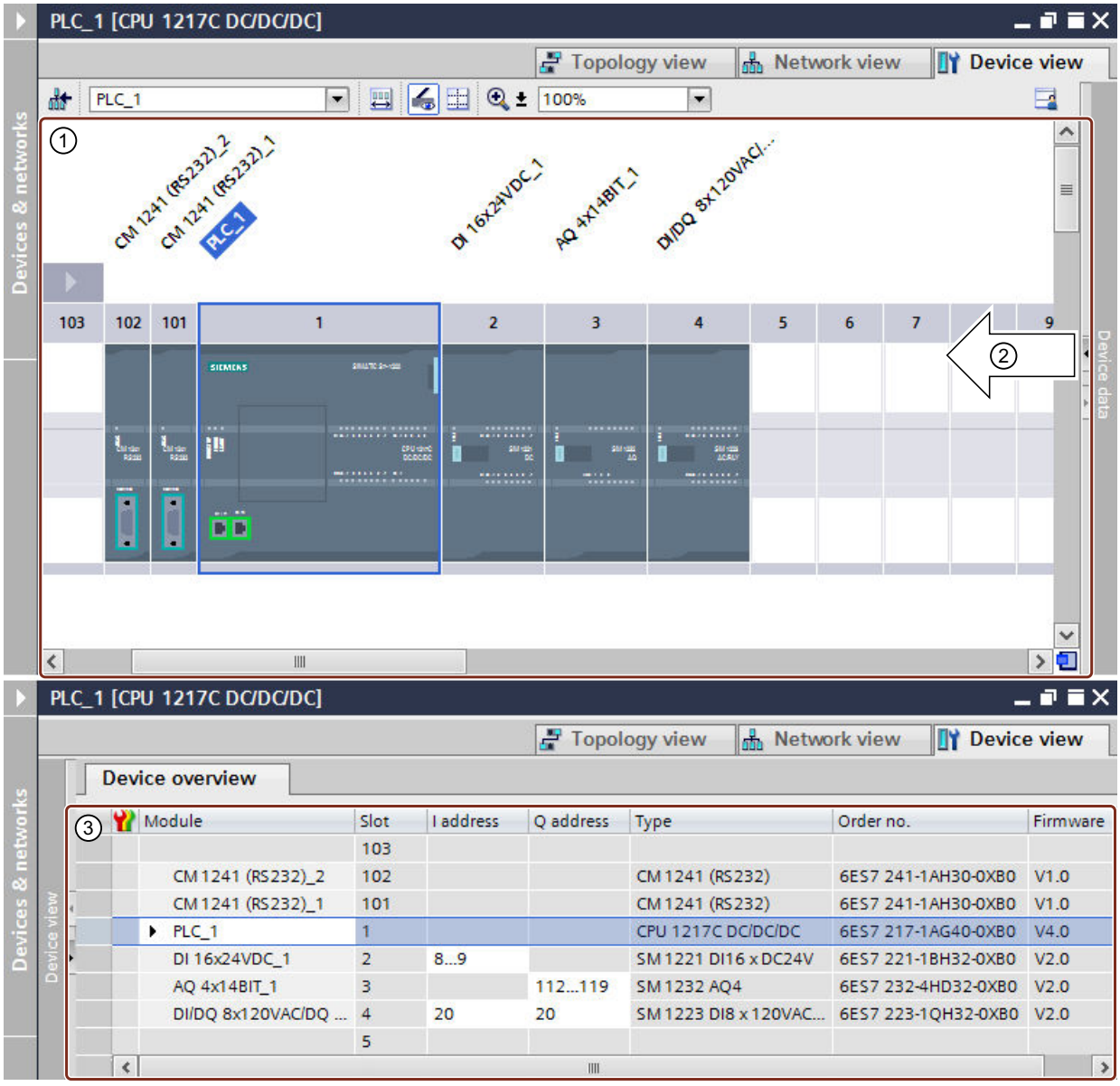
A graphic display of the rack and the devices plugged into it is shown in the left area of the device view. You can see the device overview in the right area of the device view. The device overview is a table showing you the most important information about the modules inserted in the rack. Both areas are displayed in a window. The size of both areas is adjusted using a divider. You can also use it to display one or the other area in full or to hide them.

Structure and content of device view

The offline configuration of the devices in the rack are displayed in the graphic device view. This is a symbolic representation of the configuration on the real rack.

The rack configuration is displayed as a table in the device view. Each line in the table contains the information for assigning a slot.

The following screenshot shows the device view with the configuration of a SIMATIC S7-1200 CPU.



① Graphic view of the allocation of the rack by the CPU and various modules in slots 1 to 4, as well as 101 and 102.

② You can use the divider to adjust the proportion of the device view between the left area (graphic view) and the right area (device overview). If you click on the arrows, you can quickly switch the division of the separate areas.

③ Device view with the tabular representation of the rack's slots and the inserted components. You can show additional columns and hide displayed columns using the shortcut menu of the column title.

Each line in the device overview represents one slot. The key information for each slot is displayed in the various columns:

Column	Meaning
Online status	Symbolic representation of the online status
Fail-safe	Symbolic representation for fail-safe module
Module	Name of module, can be edited in any way

Column	Meaning
Rack	Number of the rack
Slot	Slot number
I address	Input address area, can be edited in any way
Q address	Output address area, can be edited in any way
F-source address	F-source address when using fail-safe I/O
F-destination address	F-destination address when using fail-safe I/O
Type	Catalog name of module
Article no.	Article number of the module
Firmware	Firmware version of the module
Comment	Optional comment

Display in I/O channels

If you set the zoom level in the device view to at least 200%, you can see the individual I/O channels for I/O modules. If you have defined PLC tags for the channels, the names of the PLC tags are displayed.

The figure below shows the input channels of the digital input module with the two PLC tags "Test1" and "Test2" at a zoom level of 400 %.

You can select the individual I/O channels and have the following options for channels with PLC tags:

- You see the general properties of the selected PLC tag in the inspector window under "Properties".
- In the inspector window under "Info > Cross-references" you find the cross-reference information on the selected PLC tag. If you have selected the PLC tag, you can also open the cross-reference information using the shortcut menu.

See also

Device view (Page 539)

Area for unplugged modules

In some cases, the modules for a hardware configuration are not assigned a slot for short periods. Such unplugged modules are moved to the area of unplugged modules, a special area in the device view.

Adding modules to the storage area

The modules that, for example, should be assigned to a device using a copy action but for which the corresponding rack does not have a free compatible slot, are moved automatically into the area of unplugged modules.

Under the following conditions, modules are automatically added to the area of unplugged modules:

- In the network view, a module is moved to a device but the rack does not have a compatible free slot.
- In the device view, a module is moved or copied from the rack, the hardware catalog or the project tree straight into the storage area.

CPs and FMs that occupy a network resource can be moved into the area of unplugged modules but will lose the network resources they have been assigned.

You can add modules to the area of unplugged modules by means of drag-and-drop, for example. To do this, the area must be opened.

Using the area of unplugged modules

Use the corresponding button to open the area of unplugged modules.

You can find the area of unplugged modules in the device view.

You open the area of unplugged modules with the respective icon in the toolbar of the device view (Page 539).

Icon	Meaning
	Open area of unplugged modules

Note

To free up slots, move modules from your configuration into the storage area and plug the modules you want from the storage area into the freed up slots.

You can use this approach to temporarily move modules whose parameters have already been assigned out of the configuration without deleting them.

Treatment of modules in the storage area

The following rules apply to modules in the storage area:

- The modules appear in the project tree under the corresponding device in the "Local modules" folder.
- The modules retain all settings and parameters previously assigned.
- The modules are not taken into account during loading to a target system. This means that a consistency check is not undertaken for modules in the area of unplugged modules.
- Using the context menu, the modules can be copied, cut, or deleted, for example.

10.1.2.2 Configuring individual devices

Selecting a CPU

Introduction

Select a CPU from the hardware catalog and place it, together with a rack, in the network view. On this device drag the desired modules from the hardware catalog; they are arranged automatically on the rack.

Selecting the components in the hardware catalog

Each hardware component is displayed as a folder in the hardware catalog. When you open this folder, you see the different versions of the selected hardware component with its respective article numbers.

There will be an example of how to set up a CPU with a rack in network view.

Requirement

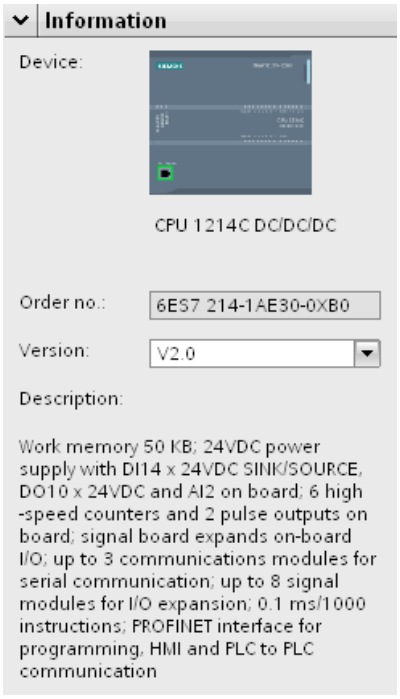
- The hardware catalog is open.
- You must be in the network view.

Procedure

To select a CPU from the hardware catalog, proceed as follows:

1. In the hardware catalog navigate to the folder with the desired CPUs.
2. Open the folder with the desired CPU type; you will see all article numbers for the selected CPU type.

3. Click on a CPU article number to get information about the selected CPU in the "Information" pane.



4. Set up the CPU and a rack. You have the following options:
   - Use drag-and-drop to drag the CPU from the hardware catalog into network view.
   - Use Copy & Paste to copy the CPU to the network view.
   - Double-click the CPU entry in the hardware catalog.

See also

Browsing the hardware catalog (Page 557)

Adding a device to the hardware configuration (Page 565)

Inserting a module into a rack (Page 568)

Working with racks (Page 558)

Creating an unspecified CPU (Page 567)

Adding a device to the hardware configuration

Introduction

There are various ways of adding a connectable device from the hardware configuration in the network view and the topology view:

- Command "Add new device" in the project tree
- Double-click device in hardware catalog

- Drag-and-drop from the hardware catalog in network view or in topology view:
  - Text entry from the "Catalog" pane
  - Preview graphic from the "Information" pane
- "Add > Device" command from menu bar in network view or topology view
- Shortcut menu of a device in the hardware catalog for copying and pasting

A suitable rack is created along with the new device. The selected device is inserted at the first permitted slot of the rack.

Regardless of the method selected, the added device is visible in the project tree and the network view of the hardware and network editor.

Adding device using the project tree

To use the project tree to add a device to the hardware configuration, proceed as follows:

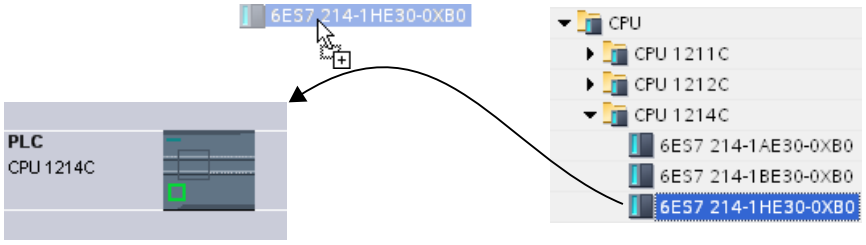
1. Click on the command "Add new device" in the project tree.
   The "Add new device" dialog box opens.
2. Display the required device in the tree structure:
   - Go to required device in the tree structure.
   - Enter a device name in the entry field.
3. Select the required device from the tree.
   More information about the device presently selected is displayed on the right-side of the dialog box.
4. If necessary, set the firmware version using the drop-down list in the dialog box.
5. Select the "Open device view" check box if you want to change to the device view immediately after adding the device.
   There you can immediately continue with device configuration and equipping the rack.
6. Click on "OK" to add the device selected.
   The dialog box closes.

Adding device from the hardware catalog

To add a device to the hardware configuration using the hardware catalog, proceed as follows:

1. Open the network view or the topology view.
2. Open the hardware catalog.
3. Go to the required device in the hardware catalog.
4. Click on the chosen device to select it.

5. If necessary, set the firmware version using the drop-down list in the hardware catalog.
6. Drag the device to the network view or the topology view.



You have now placed the device in the network view or in the topology view. The displayed rectangle (in other words "Station") symbolizes the plugged device together with its rack and any lower-level modules. Double-click on the device or station to open the device view and view the new rack and inserted device. In the next steps, you can configure the device in the device view and equip the rack with modules.

See also

Network view (Page 537)

Creating an unspecified CPU (Page 567)

Topology view (Page 542)

Creating an unspecified CPU

Introduction

If you have not yet selected a CPU but have already started programming or want to use an existing program, you have the option of using an unspecified CPU. You can also adjust some settings with unspecified CPUs. The setting options are restricted to parameters that all CPUs of the same CPU family have in common.

Creating an unspecified CPU in the portal view

To create an unspecified CPU in the portal view, follow these steps:

1. Now, click one of the following options:
   - "Devices & networks > Add new device"
   - "PLC programming" > "Device" button
2. For a device family, select an unspecified CPU from the tree structure of the "Add new device" dialog.
3. Click on "Add".

An unspecified CPU is created and the device view for this CPU appears.

Further options for creating unspecified CPUs

In the project view, you can create unspecified CPUs like specified CPUs:

- Using the "Add new device" button in the project tree
- In the "Hardware catalog" task card

You can also use these methods to create multiple unspecified CPUs.

Specifying unspecified CPUs

You have two options for specifying unspecified CPUs:

- Use drag-and-drop to assign an existing CPU from the hardware catalog to an unspecified CPU by means of module replacement (Page 574).
- Select a selected, unspecified CPU and then the menu command "Online > Hardware detection" and assign a CPU identified online. For this purpose, you assign an IP address using the "Add address for PG/PC" button.

Note

If you want to go online after the hardware detection, you have to first download the detected configuration to your project; otherwise, an error may occur due to inconsistent configurations. After hardware detection, the article numbers of the CPU in the project and the actually existing CPU are identical, but their parameters are not. The parameters of the CPU in the project have the default values; the parameters of the actually existing CPU have the values set by you.

See also

Selecting a CPU (Page 564)

Adding a device to the hardware configuration (Page 565)

Inserting a module into a rack

Introduction

Once you have added devices from the hardware catalog to your configuration in network view, you can add modules to the devices. There are various ways of adding a module to a rack in the device view:

- If there is an available valid slot, double-click a module in the hardware catalog.
- Use drag-and-drop to move the module from the hardware catalog to an available valid slot in the graphic or table area:
  - Text entry from the "Catalog" pane
  - Preview graphic from the "Information" pane
- Select "Copy" in the shortcut menu for a module in the hardware catalog and then select "Paste" in the shortcut menu on an available valid slot in the graphic or table area.

To access the device view from the network view, double-click a device or station in the network view or select the Device view tab. The device view contains an illustration of the device selected within a rack. The graphic illustration of the rack in the software corresponds to the real structure, i.e. you can see the same number of slots as exist in the real structure.

Note

You can also move a module to a rack in the network view. The filter function for the hardware catalog must be deactivated in this instance. The module is automatically plugged into a free and permitted slot. If there are no slots available, the module will be moved to the area of unplugged modules (Page 563).

Equipping a rack

Arrange the modules on a rack according to the applicable slot rules.

After a module has been inserted in a rack with an already inserted CPU, the address areas are checked automatically so that addresses are not assigned twice. After it has been inserted, each module then has one valid address range. To do so, DP slaves and IO devices must be networked with a CPU via the corresponding DP master or IO system.

Requirements

- You are in the device view.
- The hardware catalog is open.

Adding module from the hardware catalog

How to insert a module from the hardware catalog into a rack is illustrated based on the example of a signal module. To do this, follow these steps:

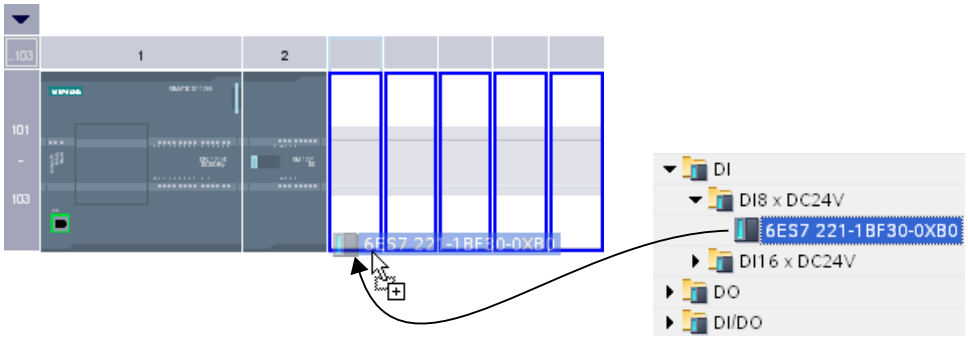
1. Go to the required module board in the hardware catalog.

   Note

   If you activate the filter function of the hardware catalog, only those modules which match the selected device type will be displayed.

2. Select the chosen module.

3. If necessary, set the firmware version using the drop-down list in the hardware catalog.
4. Drag the signal module to a free slot in the rack.

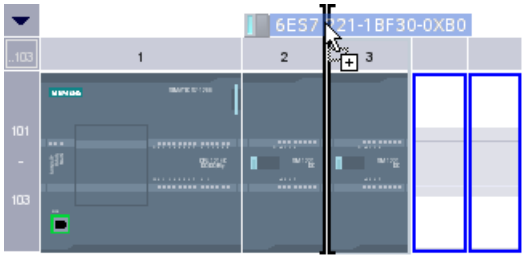


You have now inserted the digital signal module in a slot in the rack. Repeat these steps with the other modules.

The name of the module is displayed above the inserted modules. You can activate or deactivate module labeling in the menu bar with "View > Show module labels".

Inserting module

You can also drag modules and drop them between modules that have already been inserted. To do this, drag a module above and between the two existing modules while holding down the mouse button.



A mouse pointer appears. When you release the mouse button, all modules plugged to the right of the pointer are moved one slot to the right. Any redundant modules are moved to the area of unplugged modules. The new module is plugged at the point of the freed up slot.

See also

Device view (Page 539)

Area for unplugged modules (Page 563)

General slot rules (Page 556)

Deleting a hardware component

There are various ways of deleting hardware components. Deleted hardware components are removed from the system and assigned addresses made available again.

Rules

- CPUs or modules from the rack and from the area of unplugged modules can be deleted.
- When a rack is deleted in the device view, the plugged hardware components are moved to the area of unplugged modules.

Procedure

Proceed as follows to delete a hardware component:

1. Select the hardware components you want to delete.
   - Network view: Select devices or network relevant hardware components in the graphic view or in the network view.
   - Device view: In the graphic view or device overview, select racks or modules in racks or in the area of unplugged components.
   - Topology view: Select devices or hardware components with Ethernet interfaces in the graphic view or in the topology view.
   - Project tree: Select devices or individual hardware components from the tree structure.
2. Select "Delete" from the shortcut menu or press <Del>.
   If the "Delete" menu item is unavailable, your selection contains at least one component that cannot be deleted.

The selected hardware components are deleted.

Note

Deleting hardware components may result in inconsistencies in the project, such as violation of slot rules. Inconsistencies are reported during the consistency check. Correct the inconsistencies by taking appropriate action, for example, ensuring compliance with the slot rules.

See also

Keyboard operation: Editing objects (Page 553)

Copying a hardware component

You can copy hardware components in the device or network view. Copied hardware components are stored on a clipboard and can be pasted at another point from this clipboard. Copied stations are pasted as new stations in the network view. Copied devices and modules can be pasted into existing racks in the network and device view.

Rules

- Single objects as well as several objects can be copied at the same time.
- Modules inserted in the rack and in the area of unplugged modules can be copied.

- You can only copy devices and modules to free and valid slots in keeping with the slot rules.
- Racks with a CPU inserted cannot be copied individually, but only as complete units along with all inserted hardware components.

Procedure

Proceed as follows to copy a hardware component:

1. Select the hardware components you want to copy.
   - Device view: Select the module in a rack or put it in the area of unplugged modules.
   - Network view: Select the station or the relevant hardware component from the network view.
   - Project tree: Select the station or module.
2. Select "Copy" from the shortcut menu or press <Ctrl+C>.
   If the "Copy" menu item is unavailable, your selection contains at least one component that cannot be copied.
3. Select the location at which the content of the clipboard is to be pasted.
   - Device view: Select a free slot in the rack or area of unplugged modules.
   - Network view: Select a station where you want to insert devices or modules or move the mouse pointer to a free location in the network view to paste a copied station or a hardware component relevant to the network view.
4. Select "Paste" from the shortcut menu or press <Ctrl+V>.
   If the "Paste" menu item is unavailable, the clipboard is empty or contains at least one component that cannot be pasted at this point.

The selected object is pasted at the chosen point.

Once you have selected a station where you want to insert a module in the network view, the module is inserted in the first free and valid slot. If no free, valid slots are available, the object is inserted in the area of unplugged modules.

Note

You can also copy a module from one device to another:

To do so, copy a module in the hardware and network editor, select a different device in the network view or the drop down list of the device view and insert the module.

You can insert the copied object directly in a slot or place it in the area of unplugged modules in the device view. If you add the copied object in the network view of a device or a station, it will be inserted in the first available slot.

If there is no slot available for the object, it is automatically placed in the area of unplugged modules (Page 563).

Note

You can use <Ctrl> and drag-and-drop to directly copy a selected hardware component.

See also

Keyboard operation: Editing objects (Page 553)

Moving a hardware component

You can move hardware components in the device or network view.

Rules

- You can move devices and modules from the rack and the area for unplugged modules taking the slot rules into consideration.
- CPs can be moved in the network view. The CP is plugged in a free and valid slot in the target device. If there are no free slots available, the CP to be inserted is moved to the area for unplugged modules.
- In the network view, CPU and slave head modules can be moved between the devices; depending on CPU type also within the rack.

Note

Moved CPs are disconnected from their network but keep their network parameters and address. If you reconnect the CP to the network and its address has been assigned, use a dialog to assign a new unique address to the CP.

Procedure

Proceed as follows to move a hardware component:

1. Select the hardware component you want to move.
   - Device view: Select the module in a rack or put it in the area of unplugged modules.
   - Network view: Select the hardware component of relevance to the network view.
2. Select "Cut" from the shortcut menu or press <Ctrl+X>.
   If the "Cut" menu item is unavailable, your selection contains at least one component that cannot be cut.
3. Select the location to which the cut object is to be moved.
   - Device view: Select a free slot in the rack or area of unplugged modules.
   - Network view: Select a station where you want to insert devices or modules.
4. Select "Paste" from the shortcut menu or press <Ctrl+V>.
   If the "Paste" menu item is unavailable, the clipboard is empty or contains at least one component that cannot be pasted at this point.

The selected hardware component is moved to the target. If the hardware component being moved is a networked object, it is disconnected from the network.

Note

You can use drag-and-drop to directly move a selected hardware component.

See also

Keyboard operation: Editing objects (Page 553)

Replacing a hardware component

You can replace hardware components with others. This, for example, allows you to replace unspecified CPUs (Page 567) with available CPUs from the hardware catalog.

Rules

You can only replace hardware components if they support module replacement and if the two components are compatible.

Procedure

To replace one module with another, proceed as follows:

1. Select the module you want to replace.
2. Open the shortcut menu:
   - If the "Replace device" entry is enabled, the module can be replaced.
   - If the "Replace device" entry is disabled, a module cannot be replaced.
3. Click on "Replace device" in the shortcut menu. The "Replace device" dialog box appears.
4. Under "New device" in the tree structure, select the module with which you want to replace your current module.
5. Click "OK".

The existing module is replaced by the new one.

As an alternative, you can take a module by dragging it from the hardware catalog to the module you are replacing. If the module can be replaced by the selected module, this is indicated by the mouse pointer symbol.

Editing properties and parameters

Once you have inserted hardware components in your rack, you can edit their default properties, for example parameters or addresses in the network or device view.

Requirement

You are in the device view.

Note

You can also edit properties and parameters in the network view. In the graphic network view, you have access to the network-related hardware components and the station. You can access modules and hardware components not displayed in the graphic network view using the table network view.

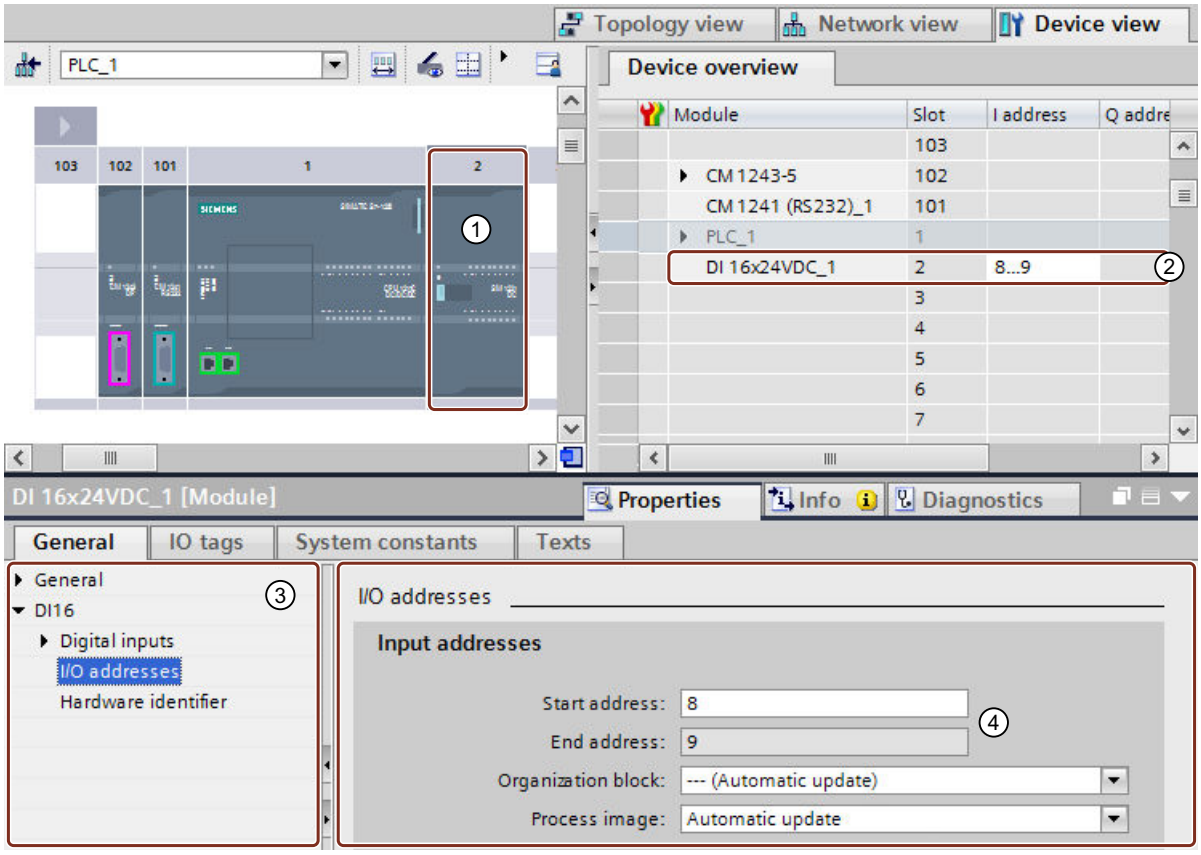
Procedure

To change the properties and parameters of the hardware components, proceed as follows:

1. In the graphic view, select the CPU, module, rack or interface you want to edit.
2. Edit the settings for the selected object:
   - Use the table view to edit addresses and names, for example.
   - In the Inspector window additional setting possibilities are available in "Properties".

Note that modules can only be fully parameterized if they are assigned to a CPU. Therefore, PROFIBUS or PROFINET interfaces modules must first be networked with the CPU or a centrally inserted communication module so that they form a master system or IO system. Only then is it possible, for example, to edit the addresses of the distributed components that are inserted.

Example of changing settings



① Selection of a module
② Editing option for addresses in the device overview
③ Selection options in the inspector window
④ Editing option for addresses in the inspector window

See also

Inspector window (Page 547)

Input and output addresses in the address overview

Introduction

The currently used input and output addresses can be displayed in the address overview in a table form. The address overview can be found in the Inspector window under "Properties" of the CPU.

Design of the address overview

With the different check boxes, you can set which objects should be displayed in the address overview:

- Inputs: Display of the input addresses
- Outputs: Display of the output addresses
- Address gaps: Display of open address spaces
- Slot: Display of the slot number

The following information is typically shown in the address overview:

Table header	Meaning
Type	Indicates whether the area is an input address area or an output address area.
Addr. from	Start address of the address range.
Addr. to	End address in the address range.
Module	Module using the address area.
PIP	Process image partition (see "PIP" below).
OB	Organization block that is assigned to the process image. This column is not available for every CPU.
DP	Number of the master system. You can use the number to determine which slaves are assigned to a master. The value in brackets specifies the PROFIBUS address of the hardware component.
PN	Number of the IO system. The value in brackets stands for the device number of the hardware component.
Rack	Number of the rack on which the hardware component is inserted.
Slot	Number of the slot in the rack in which the hardware component is inserted.

PIP

The "PIP" table column shows the assignment of the address to the cyclic process image or to a process image partition (PIP).

For S7-300/400:

- "OB1-PA": The address is assigned to the cyclic process image. The operating system updates this address automatically in each program cycle.
- "PIP x": The address is assigned to the process image partition x (for example PIP 1, no cyclic process image). The operating system updates this PIP when the assigned OB is executed. If this PIP is not assigned to an OB, the operating system does not update this PIP. You have the option to update the PIP yourself with the instructions "UPDAT_PI" and "UPDAT_PO" (for S7-400 and some S7-300 CPUs).

For S7-1200:

- "Automatic update": The address is assigned to the cyclic process image (PIP 0). The operating system updates this address automatically in each program cycle.
- "None": The address not assigned to any process image partition. You access this address directly in the user program (direct I/O access, no process image).

- "PIP x": The operating system updates this PIP when the assigned OB is executed. If this PIP is not assigned to an OB, the operating system does not update this PIP. You have the option to read inputs or write outputs in the user program via direct I/O access. The instructions "UPDAT_PI", "UPDAT_PO", "SYNC_PI" and "SYNC_PO" are not supported for S7-1200.
- "PIP OB servo": The process image partition "PIP OB Servo" is not assigned to an organization block (fixed setting for organization block: "---(None)"). The operating system does not update this PIP or any of the addresses contained in it: You access the addresses directly in the user program (direct I/O access).

For S7-1500:

- "Automatic update": The address is assigned to the cyclic process image (PIP 0). The operating system updates this address automatically in each program cycle.
- "None": The address not assigned to any process image partition. You access this address directly in the user program (direct I/O access, no process image).
- "PIP x" (PIP 1 to PIP 31): The operating system updates PIP x when the assigned OB is executed. If this PIP is not assigned to an OB, the operating system does not update PIP x. You have the option to update the PIP x in the user program with the instructions "UPDAT_PI" and "UPDAT_PO". If PIP x is assigned to an isochronous mode interrupt OB (OB 61 to OB 64), the operating system does not update PIP x. You have the option to update PIP x in your user program with the instructions "SYNC_PI" and "SYNC_PO".
- "PIP OB servo": The process image partition "PIP OB servo" is permanently assigned to the OB "MC-Servo". STEP 7 generates this OB automatically when you create a technology object in the Motion Control area. When the OB is executed, the PIP OB Servo is updated isochronously. All drives and encoders used by Motion Control are assigned to this process image partition.

See also

Specifying input and output addresses (Page 849)

Update module version

Explanation of terms

The terms "Module version" and "Firmware version" are explained in more detail in the following section.

- Module version: The specific version of the configuration software from which the module description stems.
  Example: V11.0.0.0
- Firmware version: The version of the firmware of the module whose parameters are assigned offline
  Example: V2.0

Requirement

- You have created a device configuration.
- You have installed an update or an optional package at a later date. As a result of this installation, the module version of at least one module type was updated in the hardware catalog, whereby the new version is incompatible with the previous version.
- You have used such modules in your device configuration and want to use the modified or added properties.

Procedure

Perform the following step for each affected module type.

1. Select the affected module in the device view.
2. In the Inspector window, go to "Properties > General > Catalog Information". Click the "Update module version" button there.
3. In the query that then appears, specify whether you want to update the module version only for the selected module or for all modules of this type in the current project.

Result

The selected modules are replaced by the same modules with updated module version in the current project.

In which cases is it unnecessary to update the module version?

An updating of the module version is not necessary in the following cases:

- You do not want to use the modified or added properties of the modules.
- When you open an existing project with a version of the configuration software that is more recent than the version with which you created the project, the system automatically performs a project conversion, for example, from TIA Portal V12 to V13. In this case, all older module versions are automatically adapted.

10.1.2.3 Comparing devices

Basics of device comparison

Function

You have the option of comparing the hardware components of two devices allowing you to identify any differences. You can perform an offline/offline comparison for this purpose. The devices to be compared can be from one project or different projects.

You can compare the central as well as the distributed I/O. The devices to be compared can be assigned either automatically or manually. The automatic assignment of central I/O is based

on the slot number. With the distributed I/O, the automatic assignment can be made according to the following criteria:

- Assignment using the address/HW ID: The assignment is made using the addresses or IDs of the devices. This criterion is suitable for comparing devices within a project.
- Assignment using the name: The assignment is made based on the device names. This criterion is suitable for comparing devices in different projects.

You can either specify the assignment yourself or let the system decide. In the latter case, the system selects the assignment itself depending on the context.

See also

Basics of project data comparison (Page 402)

Overview of the compare editor (Page 405)

Making a device comparison (Page 580)

Making a device comparison

Procedure

Follow the steps below to compare devices:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The comparison editor opens and the selected device is displayed in the left area.
3. Open the "Hardware" tab.
4. Drag an additional device to the drop area in the right-hand pane.
   All existing objects of the selected devices are displayed depending on the settings of the comparison editor in the "Hardware" tab and an automatic comparison is made. You can identify the status of the objects based on the symbols in the comparison editor.
5. If you want to change the matching criterion, click on the arrow of the "Available matching criteria" button in the toolbar. Then, select the matching criterion you want to use.
6. If you want to make a manual comparison, click the button for switching between automatic and manual comparison above the status area. Then select the objects you want to compare.
   The properties comparison is displayed. You can see the status of the objects based on the symbols.

See also

Basics of device comparison (Page 579)

Overview of the compare editor (Page 405)

10.1.3 Configure networks

10.1.3.1 Networking devices

Communication and networks

Communication between devices

The basis of all types of communication is always a previously configured network. The network configuration provides the necessary requirements for communication:

- All the devices in a network are provided with a unique address
- Communication of the devices with consistent transmission properties

Network configuration

The following steps are necessary when configuring networks:

- Connect devices to subnet
- Specify the properties/parameters for each subnet
- Specify the device properties for every networked module
- Download configuration data to the devices to supply interfaces with the settings resulting from the network configuration
- Document the network configuration

For Open User Communication, creating and configuring a subnet is supported by the assignment of connection parameters.

Relation between network configuration and project

Within a project, subnets and their properties are managed. Properties result mainly from adjustable network parameters and the quantity and communication properties of the connected devices.

The devices to be networked must be in the same project.

Subnet name and subnet ID

Within the project, subnets are clearly identified by a subnet name and ID. The subnet ID is saved in all components along with interconnectable interfaces. Components can then be clearly assigned to a subnet even after uploading into a project.

Networking options

In the project, you can create and network devices with components capable of communication. The following basic options are available for networking the devices:

- You link the interfaces of the components capable of communication with one another. A new subnet is created suitable for the type of interface.
- You connect the interface of the devices capable of communication with a new or existing subnet.
- You create an Open User Communication connection. When you assign parameters to the connection for Open User Communication, a subnet is created automatically between the communication partners.
- You use the graphic connection configuration to configure connections; missing networks are hereby recognized and are created either automatically or via dialog.

Due to the different tasks of the devices or the span of the plant, you may need to use several subnets. These subnets are managed in a project.

Networking devices in the network view

Options

In the graphic network view, you have an overview of the subnets of the entire system in the project. You can use the tabular network overview for additional support.

Depending on the starting situation, there are various ways of undertaking configuration to network the interface for a component capable of communication. The procedures are described in the following section:

- Creating an individual subnet
- Creating several subnets at one time
- Connecting two target devices via a new subnet
- Connecting devices to existing subnet
- Selecting an existing subnet from a list
- Automatic networking during the configuration of the connection;
  See also: Auto-Hotspot

Possible starting situations are:

- A suitable subnet is not yet available.
- The subnet with which you want to connect the component already exists.

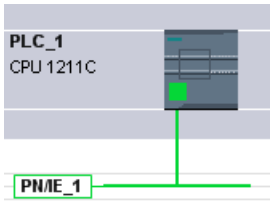
Procedure - creating a single subnet

To create a subnet and to connect it to an interface, proceed as follows:

1. Select the interface of a CPU / a CP.
2. Select the "Create subnet" command in the shortcut menu of the interface.

The selected interface is connected to a new subnet. Consistent address parameters are set automatically for the interface.

The following schematic shows an interface with outgoing line connecting to a subnet:



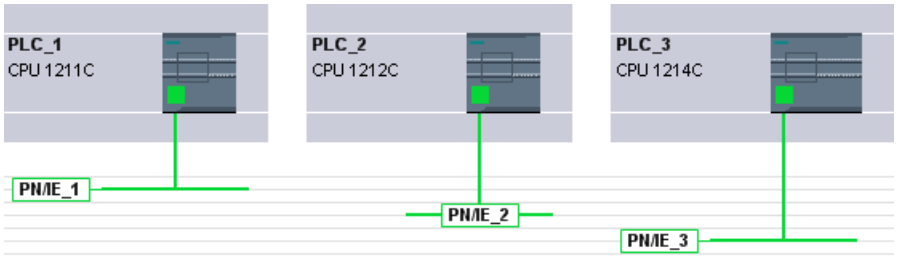
Procedure - creating several subnets at one time

To create several subnets at one time, proceed as follows:

1. Select several interfaces by clicking on them while pressing the <Ctrl> button.
2. Select the "Create subnet" command in the shortcut menu of the interface.

Each selected interface is connected to a new subnet. Consistent address parameters are set automatically for the interface.

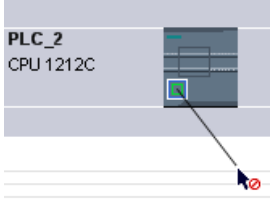
The following figure shows multiple subnets created by selecting multiple interfaces:



Procedure – Connecting two target devices via a new subnet

To connect an interface with another device via a subnet that does not yet exist, proceed as follows:

1. Position the mouse pointer over the interface for a component capable of communication requiring networking.
2. Click with the left mouse button and hold the button down.
3. Move the mouse pointer.
   The pointer now uses the networking symbol to indicate "Networking" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear when the pointer is on a valid target.



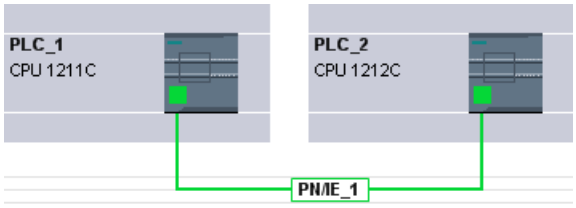
4. Now move the pointer in networking mode onto the interface of the target device. You can either keep the mouse button pressed or release it.
5. Now release the left mouse button or press it again (depending on your previous action).

Note

If you want to exit networking mode beforehand, press <Esc>, right-click or double-click in the background of the network view.

A new subnet is created. The interfaces are now connected via the new subnet. Consistent address parameters are set automatically for the interface.

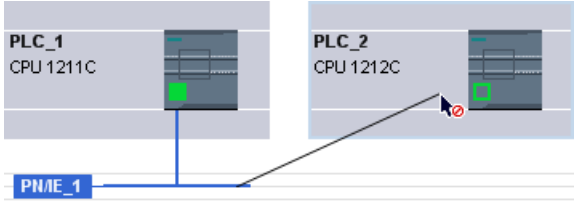
The following schematic shows two networked devices:



Procedure - Connecting devices to existing subnet

To connect an interface to an existing subnet, proceed as follows:

1. Position the mouse pointer on the interface of a communications-compliant component you want to network or on the existing subnet.
2. Click with the left mouse button and hold the button down.
3. Move the mouse pointer.
   The pointer now uses the networking symbol to indicate "Networking" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear once the pointer is moved to a valid target.
4. Now move the mouse pointer to the existing subnet or to the interface to be networked. You can either keep the mouse button pressed or release it.



5. Now release the left mouse button or press it again (depending on your previous action).
   If you want to exit networking mode beforehand, press <Esc>, or right-click.

Result:

The interface and selected subnet are now connected. Consistent address parameters are set automatically for the interface.

Procedure - selecting an existing subnet from a list

To link an interface with a subnet that has already been created, proceed as follows:

1. Select the interface of a CPU.
2. Select the "Assign to new subnet" command in the shortcut menu of the interface.
   A list box containing the available subnets appears.
3. Select a subnet from the list.

The interface and selected subnet are now connected. Consistent address parameters are set automatically for the interface.

Tabular network overview

Meaning

The tabular network overview adds the following functions to the graphic network view:

- You obtain detailed information on the structure and parameter settings of the devices.
- Using the "Subnet" column, you can connect components capable of communication with created subnets.

Basic functions for tables

The network overview supports the following basic functions for editing a table:

- Displaying and hiding table columns
  Note: The columns of relevance to configuration cannot be hidden.
- Optimizing column width
- Sorting table
- Displaying the meaning of a column, a row or cell using tooltips.

Networking devices in the device view

Networking in the device view

In the device view, you can check and set all the parameters of the components belonging to a device and the interfaces in detail. Here you can also assign the interfaces to the subnets created in the project.

Requirements

- The subnet with which you want to connect an interface has already been created.
- If the subnet has not yet been created, change to the network view and make the settings required for networking.

Procedure - connecting to an existing subnet

To connect an interface to an existing subnet, proceed in the device view as follows:

1. Select the entire communications-compliant component or the interface to be networked. The properties of the selected interface or component are displayed in the Inspector window.
2. In the Inspector window, select the parameter group for the selected interface; for example, the "Ethernet addresses" parameter group for a PROFINET interface.
3. Select the subnet to be connected from the "Interface connected with" drop-down list.

The interface and selected subnet are now connected. Consistent address parameters are set automatically for the interface.

Procedure - creating a new subnet

To create a subnet and to connect it to the interface, proceed as follows in the device view:

1. Select the entire communications-compliant component or the interface to be networked. The properties of the selected interface or component are displayed in the Inspector window.
2. In the Inspector window, select the parameter group for the selected interface; for example, the "Ethernet addresses" parameter group for a PROFINET interface.
3. In "Interface connected with", click the "Add new subnet" button.

The interface is connected to a new subnet of the appropriate subnet type. Consistent address parameters are set automatically for the interface.

Checking or changing network parameters and interface parameters

Introduction

Communication between networked devices requires the following parameters to be configured:

- Network parameters
  Network parameters identify the network within the system configuration, for example, using a name.
- Interface parameters
  Interface parameters define specific properties of a component capable of communication. Addresses and transmission characteristics are set automatically and are consistent with the network parameters.

Note

Network parameters and interface parameters are usually set during networking such that communication can take place for numerous applications without the parameters having to be changed.

Procedure - checking or changing network parameters

Proceed as follows to check or change network parameters:

1. Enter the network view.
2. Select the subnet from the network view.
   You can see the network parameters in the "Properties" tab in the inspector window.
3. If necessary, check or modify the network parameters in the relevant group of parameters.

Procedure - checking or changing interface parameters

You can check and modify interface parameters in the network and device view.

Proceed as follows to check or change interface parameters:

1. Enter the network view or device view.
2. Select the interface.
   You can see the interface parameters in the "Properties" tab in the inspector window.
3. If necessary, check or modify the interface parameters in the relevant group of parameters.

Changing networkings

Introduction

You can cancel an interface's network connection or assign it to another subnet of the same subnet type.

Consequences

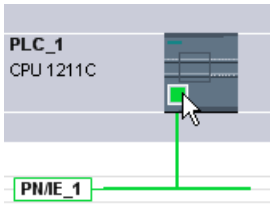
Depending on the version, a distinction should be made between:

- Canceling a network connection for an interface
  The configured parameters for the interface remain unchanged.
- Assigning a network connection to another subnet
  If the addresses in the assigned subnet are not unique, in other words, they already exist, they will be changed automatically to make them unique.

Procedure - disconnecting from a network

Proceed as follows to cancel the network connection for an interface:

1. Select the networked interface.



2. Select the "Disconnect from subnet" command in the shortcut menu of the interface.

The network connection is deleted, the interface addresses are, however, not changed.

Configured connections are retained; however these connections are marked red in the connection table because they are not networked. Specified connections remain specified.

See also

Networking devices in the network view (Page 582)

Copying, cutting or deleting subnets

Introduction

You can copy subnets as individual objects or copy them along with networked devices or other networks.

For example, you can create complex configurations to be used more than once in different variants within the project with no additional effort.

Effects on the copied subnet

Properties that have to be assigned explicitly within a project are re-assigned appropriately when the copied objects are copied.

For subnets, this means: The subnet ID and name are re-assigned to the copied subnet.

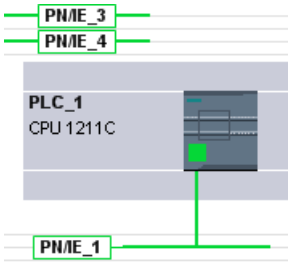
The configured properties are adopted in the copied subnet.

Procedure - copying a subnet

Proceed as follows to copy one or more subnets:

1. Select one or more subnets.
2. In the shortcut menu, select the "Copy" command.
3. Select the "Paste" command in the shortcut menu.

The copied subnets are shown as "orphaned" subnets in the top part of the network view.



Procedure - copying a subnet with connected devices

To copy one or more subnets with networked devices, proceed as follows:

1. Select one or more subnets with the connected devices, for example by drawing a lasso around them.
2. In the shortcut menu, select the "Copy" command.
3. Select the "Paste" command in the shortcut menu.

Complete copies of the subnets and connected devices are created.

Configured connections are adopted and remain within the copied devices. Connections to devices that have not been copied are interrupted and become unspecified.

MPI network configuration

Allocating MPI addresses

Note the following for devices with an MPI interface: All devices of a subnet must have a different device address.

CPUs with MPI address ship with the default MPI address 2. Since you can only use this address once in the MPI subnet, you will have to change the default address in all other CPUs.

The following applies to devices with the article no. 6ES7 3xx-xxxxx-0AB0:

When planning the MPI addresses for several CPUs, you have to fill "MPI address gaps" for FMs and CPs with separate MPI addresses to prevent addresses being assigned twice.

Only when all the modules in a subnet have different addresses and your actual structure matches that of the network configuration produced, should you load the settings across the network.

Rules for assigning the MPI address

- Allocate the MPI addresses in ascending order.
- Reserve MPI address 0 for a programming device.
- You can link up to 126 (addressable) devices with one another in an MPI subnet; up to 8 devices at a transfer speed of 19.2 KB/s.
- All MPI addresses in an MPI subnet must be different.

You will find other rules relating to the structure of a network in the manuals for setting up automation systems.

PROFIBUS network configuration

PROFIBUS addresses

Rules for the network configuration

All the devices in a subnet must have a different PROFIBUS address.

Only when all the modules in a subnet have different addresses and your actual structure matches that of the network configuration produced, should you load the settings across the network.

You can connect devices to the PROFIBUS subnet that communicate via configured connections or that belong to a PROFIBUS DP master system.

You can find more information on configuring a DP master system in the following sections.

Requirements

The 121xC CPU is PROFIBUS compatible as of firmware version 2.0.

Rules for assigning PROFIBUS addresses

- Allocate the PROFIBUS addresses in ascending order.
- Reserve the PROFIBUS address "0" for a programming device.
- Allocate a unique PROFIBUS address between 0 and 126 for each device on the PROFIBUS network and/or for each DP master and each DP slave in the PROFIBUS network.
- There are modules with which the smallest address that can be set has to be greater than 1.
- All PROFIBUS addresses of a PROFIBUS subnet must be different.

You will find additional rules relating to the structure of a network in the manuals for setting up automation systems, for example SIMATIC S7-1200.

Note

PROFIBUS address "0"

Reserve PROFIBUS address "0" for a programming device that you will briefly connect up to the PROFIBUS network at a later date for servicing.

See also

What you need to know about PROFIBUS bus parameters (Page 591)

What you need to know about PROFIBUS bus parameters

Matching parameters to one another

The PROFIBUS subnet will only function without problem if the parameters for the bus profile are matched to one another. You should therefore only change the default values if you are familiar with how to configure the bus profile for PROFIBUS.

Note

It may be possible for the bus parameters to be adjusted depending on the bus profile. If the bus parameters cannot be adjusted, they are grayed out. The offline values of the bus parameters are always shown even if you are online and linked to the target system.

The parameters shown apply to the entire PROFIBUS subnet and are briefly explained below.

Activating cyclic distribution of the bus parameters

If the "Enable cyclic distribution of the bus parameters" check box is selected under "Bus parameters" while PROFIBUS subnet is selected in the Inspector window, the bus parameters are transferred cyclically during operation by the modules that support this function. This allows a programming device, for example, to be easily connected to the PROFIBUS in runtime.

You should deactivate this function:

- For a heterogeneous PROFIBUS subnet (or more accurately: when external devices are connected whose protocol uses the DSAP 63 for Multicast)
- When in constant bus cycle time mode (minimize bus cycle)

Bus parameters for the bus profile of PROFIBUS subnets

Bus parameter	Adjustable?	Limit values
Tslot_Init	Yes	Max. Tsdr + 15 <= Tslot_init <= 16.383 t_bit
Max. Tsdr	Yes	35 + 2*Tset + Tqui <= Max. Tsdr <= 1.023 t_bit
Min. Tsdr	Yes	11 t_bit <= Min. Tsdr <= MIN(255 t_bit, ... Max. Tsdr - 1, 34 + 2*Tset + Tqui)
Tset	Yes	1 t_bit <= Tset <= 494 t_bit
Tqui	Yes	0 t_bit <= Tqui <= MIN(31 t_bit, Min. Tsdr - 1)
GAP factor	Yes	1 <= GAP factor <= 100
Retry limit	Yes	1 <= Retry limit <= 15
Tslot	No	---
Tid2	No	Tid2 = Max. Tsdr
Trdy	No	Trdy = Min. Tsdr
Tid1	No	Tid1 = 35 + 2*Tset + Tqui
Ttr	Yes	256 t_bit <= Ttr <= 16.777.960 t_bit

Bus parameter	Adjustable?	Limit values
Ttr typical	No	This time is provided for information only and is not transferred to the nodes.
Response monitoring		10 ms <= response monitoring (watchdog) <= 650 s

If you want to create a customized bus profile, we recommend the following settings:

- Minimum target rotation time (Ttr) = 5000 x HSA (highest PROFIBUS address)
- Minimum response monitoring (watchdog) = 6250 x HSA

Recalculating

You can use the "Recalculate" button to recalculate the parameters.

See also

PROFIBUS addresses (Page 590)

Description of the bus parameters (Page 592)

Description of the bus parameters

Detailed description of PROFIBUS bus parameters

Bus parameter	Meaning
Tslot_Init	The wait-for-reception (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner. If the influence of the line components on message frame run times is entered in the "Cable Configuration" parameter group, these components must also be taken into account. The component is added to the specified Tslot_Init and the total used as Tslot.
Max. Tsdr	The maximum protocol processing time defines the latest time by which the responding node should have replied.
Min. Tsdr	The minimum protocol processing time defines the earliest time by which the responding node may reply.
Tset	The trigger time is the time which may lapse between the reception of a data message frame and the response to it in the node.
Tqui	The modulator quiet time is the time which a sending node needs after the end of the message frame to switch from sending to receiving.
GAP factor	The GAP update factor defines the number of token rotations after which a newly added, active node can be added to the logical token ring.
Retry limit	This parameter defines the maximum number of attempts (message frame repeats) made to reach a node.
Tslot	The wait-for-reception time (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner. If the influence of the bus design components on message frame run times is entered in the "Cable Configuration" parameter group, these components must also be taken into account. The component is added to the specified Tslot_Init and the total used as Tslot.

Bus parameter	Meaning
Tid2	Idle time 2 defines the earliest time by which a sending node may send the next message frame after sending a message frame that has not been acknowledged.
Trdy	The ready time specifies the earliest time by which a sending node may receive a response message frame.
Tid1	Idle time 1 defines the earliest time by which a sending node may send the next message frame after receiving a response.
Ttr	The target rotation time is the maximum time made available for a token rotation. During this time, all active nodes (DP masters etc.) have the right to send (token) once. The difference between the target rotation time and the actual token holding time of a node determines how much time is left over for the other active nodes (programming device, other DP masters etc.) to send message frames.
Ttr typical	The typical data cycle time is the average response time on the bus if all configured slaves are exchanging data with the DP master. None of the slaves report diagnostics and there is no extra message frame traffic with programming devices or other active nodes etc. on the bus.
Response monitoring	The response monitoring time is only needed for PROFIBUS-DP bus systems. It defines the latest time by which a DP slave has to be addressed by its DP master with a new data message frame. If this does not happen, the DP slave assumes that the DP master has failed and resets its outputs to a secure mode.

See also

What you need to know about PROFIBUS bus parameters (Page 591)

Bus profiles with PROFIBUS

Introduction

Depending on the device types connected and protocols used on the PROFIBUS, different profiles are available. The profiles differ in terms of the setting options and calculation of bus parameters. The profiles are explained below.

Devices with different profiles on the same PROFIBUS subnet

The PROFIBUS subnet only functions without problem if the bus parameters of all devices have the same values. For example, if both DP and FMS services are being used on a subnet, the "slower" sets of bus parameters must always be set for all devices, i.e. even the "Universal (DP/FMS)" profile for DP devices.

Profiles and transmission rates

Profiles	Supported transmission speeds in Kbits/s
DP	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000
Standard	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000
Universal (DP-FMS)	9.6 19.2 93.75 187.5 500 1500
Customized	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000

Meaning of profiles

Profile	Meaning
DP	Select the "DP" bus profile when the only devices connected to the PROFIBUS subnet are those which satisfy the requirements of standard EN 50170 Volume 2/3, Part 8-2 PROFIBUS. The bus parameter setting is optimized on these devices. This includes devices with DP master and DP slave interfaces of the SIMATIC S7 and distributed I/Os of other manufacturers.
Standard	Compared to the "DP" profile, the "Standard" profile also offers scope for devices of another project or devices which have not been configured here to be taken into account when calculating the bus parameters. The bus parameters are then calculated following a simple, non-optimized algorithm.
Universal (DP/FMS)	Select the "Universal (DP/FMS)" bus profile when individual devices on the PROFIBUS subnet use the PROFIBUS-FMS service. This includes the following devices for example: • CP 343-5 (SIMATIC S7) • PROFIBUS-FMS devices of other manufacturers As with the "Standard" profile, this profile allows you to take other devices into account when calculating the bus parameters.
Customized	The PROFIBUS subnet will only function without problem if the parameters for the bus profile are matched to one another. Select the "Customized" profile when none of the usual profiles "match" a PROFIBUS device and you need to adapt the bus parameters to your special structure. Information on this can be found in the documentation for the PROFIBUS device. You should only change the default values if you are familiar with how to configure the bus profile for PROFIBUS. Not all combinations that can be theoretically set can be used even with this bus profile. The PROFIBUS standard specifies several parameter limits that depend on other parameters. For example, a responder must not respond (Min Tsdr) before the initiator can receive the message frame (Trdy). These standard specifications are also checked in the "Customized" profile. Tip: The bus parameters last valid on the PROFIBUS subnet are always automatically set as customized. For example, if the "DP" bus profile was valid for the subnet, then the bus parameters for "DP" are set in the "Customized" bus profile. The parameters can be modified on this basis. The monitoring times are not automatically recalculated so that you do not put at risk the consistency of set values, for example with configurations in other configuration tools without realizing that you have done so. You can also have the Ttr monitoring times and target rotation time calculated on the basis of parameters you have set: Click here on the "Recalculate" button.

Note

Both mono-master mode and multi-master mode are possible with all PROFIBUS profiles.

What you need to know about PROFIBUS line configuration

Cable configuration and bus parameters

Information regarding the cable configuration can be taken into consideration when calculating the bus parameters. For this purpose, you must select the "Consider cable configuration" check box in the properties for the PROFIBUS subnet.

The remaining information then depends on the type of cable used; the following settings are available:

- Copper cable
- Fiber-optic cable/optical ring

PROFIBUS line configuration, optical ring

The calculation depends on the OLM types used. The selection is made by activating the check box (multiple activation is possible, at least one OLM type must be selected):

- OLM/P12
- OLM/G12
- OLM/G12-EEC
- OLM/G12-1300

The following bus parameter adjustments are made:

- Configuration of a node not present

  Note

  The following restrictions apply to optical rings, even for passive nodes (DP slaves for example):

  A maximum of HSA-1 nodes may be connected to the PROFIBUS network. With an HSA of 126, addresses 126 and 125 must not be used. A maximum of 125 nodes are possible on the bus (No. 0 to 124).

  If an HSA is less than or equal to 125, the addresses HSA and greater may not be used. The address HSA-1 may not be used.

- Increase the retry value to 3
- Setting of minimum slot time needed for ring mode

  Note

  Short slot time values are needed for OLM/P12, average ones for OLM/G12 and OLM/G12-EEC and long ones for OLM/G12-1300. This results in high performance for small network lengths and average to low performance for average to large network lengths.

PROFIBUS communication load

Communication load - allowing for additional network stations

The bus parameters depend on the volume of communication between the active network nodes. There are differences between cyclic communication (DP) and connection-based, acyclic communication (S7 communication, Send/Receive (FDL), FMS). Unlike DP, the volume and size of communication tasks (communication load) depends on the user program. For this reason, the communication load cannot always be calculated automatically.

To calculate the bus times you can define a network configuration in the "Additional network stations" parameter group that differs from the network configuration.

Taking the profile into account

The network configuration can be defined for the "Standard", "Universal (DP/FMS)", and "User-defined" profiles. Parameters cannot be entered in the "Additional network stations" parameter group for the "DP" profile.

Quantification of the communication load

The following settings are available in order to make allowance for the communication load.

- Information regarding the number of unconfigured network stations;
- Information on the communication load resulting from the user programs for FDL or S7 communication. Here you can selected from the following settings:
  - Low
    Typically used for DP, no great data communication apart from DP.
  - Medium
    Typically used for mixed operations featuring DP and other communication services (such as for S7 communication), when DP has strict time requirements and for average acyclic volumes of communication.
  - High
    For mixed operations featuring DP and other communication services (such as for S7 communication), when DP has loose time requirements and for high acyclic volumes of communication.

Configuring Industrial Ethernet

Rules for the network configuration

The Ethernet interfaces of the devices have a default IP address that you can change.

IP address

The IP parameters are visible if the module capable of communication supports the TCP/IP protocol. This is usually the case for all Ethernet modules.

The IP address consists of 4 decimal figures in the range of 0 to 255. The decimal figures are separated from one another by a dot.

Example: 140.80.0.2

The IP address consists of:

- Address of IP subnet
- Address of the device (generally also called host or network node)

Subnet mask

The subnet mask splits these two addresses. It determines which part of the IP address addresses the network and which part of the IP address addresses the device.

The set bits of the subnet mask determine the network part of the IP address.

Example:

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

In the example given for the above IP address, the subnet mask shown here has the following meaning:

The first 2 bytes of the IP address identify the subnet - i.e. 140.80. The last two bytes address the device, i.e. 0.2.

It is generally true that:

- The network address results from AND logic operation of the IP address and subnet mask.
- The device address results from the AND NOT logic operation of the IP address and subnet mask.

Relation between IP address and default subnet mask

An agreement exists relating to the assignment of IP address ranges and so-called "Default subnet masks". The first decimal number (from the left) in the IP address determines the structure of the default subnet mask. It determines the number of "1" values (binary) as follows:

IP address (decimal)	IP address (binary)	Address class	Default subnet mask
0 to 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 to 191	10xxxxxx.xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

Note

Range of values for the first decimal point

A value of between 224 and 255 is also possible for the first decimal number of the IP address (address class D, etc). This is not recommended, however, because there is no address check for these values.

Masking other subnets

You can use the subnet mask to add further structures and form "private" subnets for a subnet that is assigned one of the address classes A, B or C. This is done by setting other lower points of the subnet mask to "1". For each bit set to "1", the number of "private" networks doubles and the number of devices they contain is halved. Externally, the network functions like an individual network as it did previously.

Example:

You have a subnet of address class B (for example IP address 129.80.xxx.xxx) and change the default subnet mask as follows:

Masks	Decimal	Binary
Default subnet mask	255.255.0.0	11111111.11111111.00000000.00000000
Subnet mask	255.255.128.0	11111111.11111111.10000000.00000000

Result:

All devices with addresses between 129.80.001.xxx and 129.80.127.xxx are on one IP subnet, all devices with addresses between 129.80.128.xxx and 129.80.255.xxx are on another IP subnet.

Router

The task of the routers is to connect the IP subnets. If an IP datagram is to be sent to another network, it first has to be conveyed to a router. To make this possible, you have to enter the address of the router for each device in the IP subnet.

The IP address of a device in the subnet and the IP address of the router can only differ at the points at which there is a "0" in the subnet mask.

See also

Settings for interconnecting Ethernet devices (Page 598)

Settings for interconnecting Ethernet devices

The following paragraph describes the behavior of STEP 7 during interconnection of PROFINET devices and the effects of the port interconnection on the network view.

Relation between port interconnection and network view

Topology view:

In the topology view, you specify the physical interconnection of Ethernet ports.

You specifically determine which Ethernet port of a device is to be connected with a specific Ethernet port of another device by means of an Ethernet cable (preset topology).

Example:

You specify that port 1 of the PROFINET interface of the CPU is to be connected to port 2 of the PROFINET interface of device A by means of an Ethernet cable.

You also specify the interface for devices with several PROFINET interfaces.

Example:

You specify that port 1 of PROFINET interface X2 of the CPU is to be connected to port 2 of the PROFINET interface of device A by means of an Ethernet cable.

The Ethernet ports can be interconnected in a table or graphically.

Network view:

In the network view, you specify which devices are to be connected with each other via an Ethernet subnet. You do not specify the Ethernet ports by which the devices are interconnected with each other (that is the task of the port interconnection).

The port interconnection has effects on the network view:

When you interconnect Ethernet ports of devices with each other in the topology view, STEP 7 connects the interconnected PROFINET interfaces of the devices in the network view with an Ethernet subnet (green line).

However, the course of the green line does not reflect the actual cable routing. You specify the actual wiring in the topology view.

An Ethernet subnet always has a name and an S7 subnet ID. You can set these two values in the subnet properties.

Which Ethernet subnet is used to link interconnected devices?

STEP 7 distinguishes between the following cases:

- Ethernet subnet specified (default subnet).
- No Ethernet subnet specified (no default subnet).

Ethernet subnet specified (default subnet)

The option "Connect devices that are not linked with this subnet in case of port connection" is activated in the properties of an Ethernet subnet (default).

This option can be activated for exactly one Ethernet subnet.

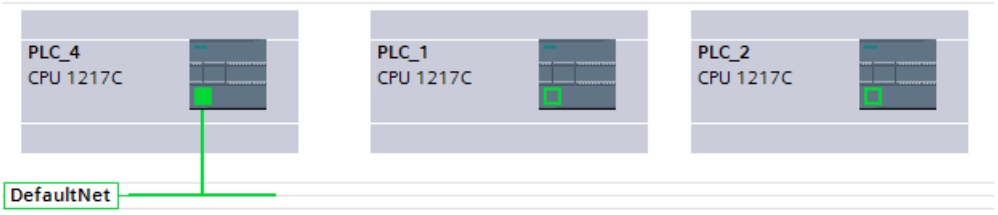
Activate the option, if necessary, for the Ethernet subnet that is to be continued when interconnecting devices that are not linked. This subnet is referred to as "default subnet" below.

Response of STEP 7:

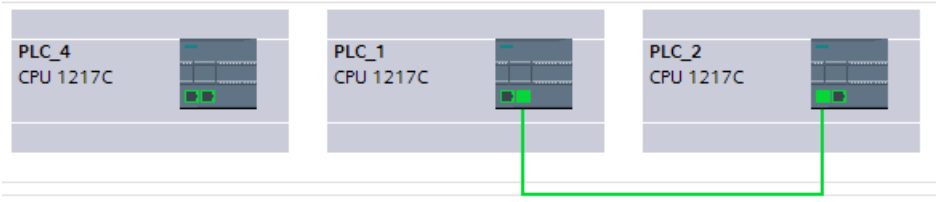
When you interconnect the ports of two devices that are not linked in the topology view, STEP 7 links these devices with the default subnet.

Example:

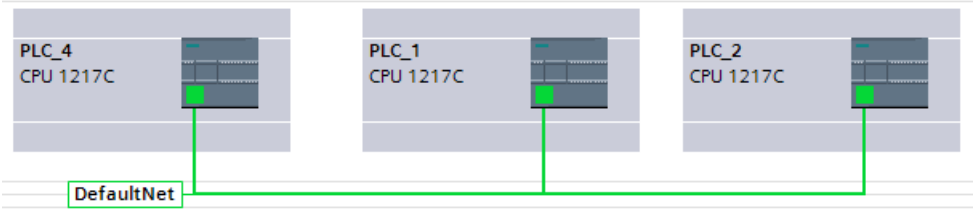
Step 1: Create a subnet in the network view at PLC_4, "Connect devices that are not linked with this subnet in case of port connection" option is **enabled**.



Step 2: Interconnect PLC_1 with PLC_2 (topology view).



Result: All PLCs are now connected to the same default subnet (network view).



No Ethernet subnet specified (no default subnet).

This is the case when the following conditions are fulfilled:

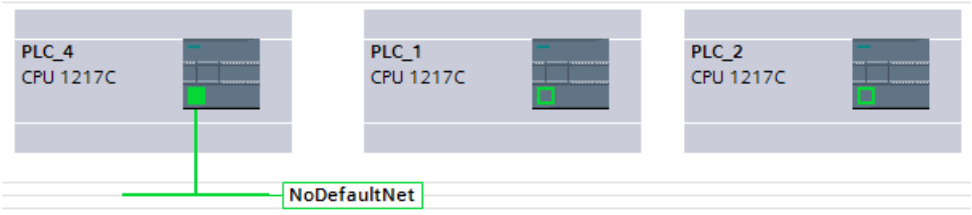
- A subnet has been added to a PROFINET interface (this interface is referred to as "Interface A" below).
- The option "Connect devices that are not linked with this subnet in case of port connection" is deactivated in the properties of this subnet (no default subnet).
- There is no other Ethernet subnet for which this option is activated.

Response of STEP 7:

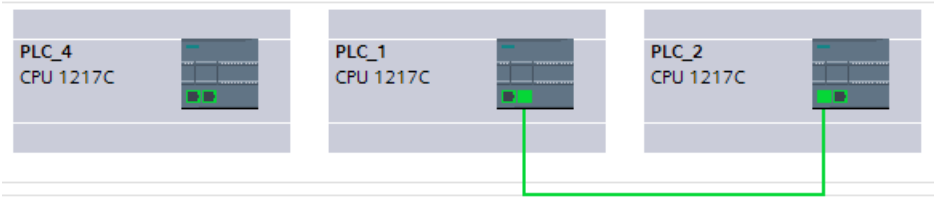
- The Ethernet subnet of interface A is only continued when you interconnect a port of interface A with a port of another device.
- When you interconnect ports of other devices that are not linked with each other, STEP 7 creates a new Ethernet subnet.

Example:

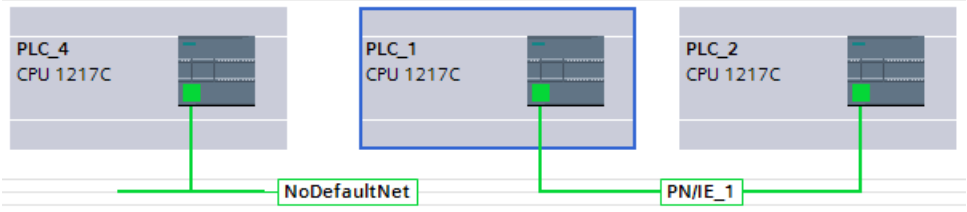
Step 1: Create a subnet in the network view at PLC_4, "Connect devices that are not linked with this subnet in case of port connection" option is **disabled**.



Step 2: Interconnect PLC_1 with PLC_2 (topology view).



Result: PLC_1 and PLC_2 are connected to a new subnet (network view).



Manual adaptation of the IP addresses

It can occur that STEP 7 does not adapt the IP addresses of the interconnected devices in such a way that the resulting network can be correctly compiled, for example, for devices with multiple PROFINET interfaces.

In these situations, you need to adapt the IP addresses of the devices manually.

The following rules apply:

- Devices that are to communicate with each other without a router cannot belong to different IP subnets.
- For devices with multiple PROFINET interfaces, the interfaces must be in different IP subnets.

To change the IP address of a PROFINET interface, perform the following steps:

1. Change to the network view (if not selected already).
2. Left-click on the icon of the PROFINET interface that is not to be part of the IP subnet.
3. Change the subnet part of the IP address in the properties of the PROFINET interface (in the "Ethernet addresses" area).

Example: Manual conversion of the IP subnet part for devices with multiple PROFINET interfaces in a subnet

The IP address is: "192.168.0.1".

The subnet mask is "255.255.255.0".

The first three numbers "192.168.0" form the IP subnet part of the IP address "192.168.0.1".

Change the IP subnet part, for example, to "192.168.1"

See also

Interconnecting ports in the graphic view (Page 670)

Interconnecting ports in the table view (Page 671)

Configuring Industrial Ethernet (Page 596)

Network configuration of AS interface

AS-Interface consists of an AS-i master and AS-i slaves connected to each other over an AS-i subnet.

Rules for AS interface network configuration

All the nodes in an AS-i subnet must have a different AS-i node address.

You should only load the settings over the network when all the modules in a subnet have different addresses and when the actual structure matches that of the network configuration you have created.

One AS-i master and up to 31 AS-i slaves can be operated on one AS-i subnet.

For more information on configuring an AS-Interface with an AS-i master and AS-i slaves, refer to the section on AS-Interface and the documentation of the AS-i master modules.

10.1.3.2 Communication via connections

Working with connections

S7 connection

Introduction to configuring connections

Definition

A connection defines a logical assignment of two communication partners in order to undertake communication services. A connection defines the following:

- Communication partner involved
- Type of connection (for example, S7 connection)
- Special properties (e.g., whether a connection is established permanently or whether it is established and terminated dynamically in the user program, and whether status messages are to be sent)
- Connection path

What you need to know about connection configuration

During connection configuration, a local connection name is assigned for an S7 connection as a unique local identification.

In the network view, a "Connections" tab is displayed in addition to the "Network overview" tab. This tab contains the connection table. A row in this connection table represents a configured connection from the viewpoint of the local communication partner with its properties, for example, between two S7-1200 CPUs.

What you need to know about using connection resources

Introduction

Each connection requires connection resources for the end point and/or transition point on the devices involved. The number of connection resources is device-specific.

If all the connection resources of a communication partner are assigned, no new connections can be established. This situation is apparent when a newly created connection in the connection table has a red background. The configuration is then inconsistent and cannot be compiled.

S7 connections

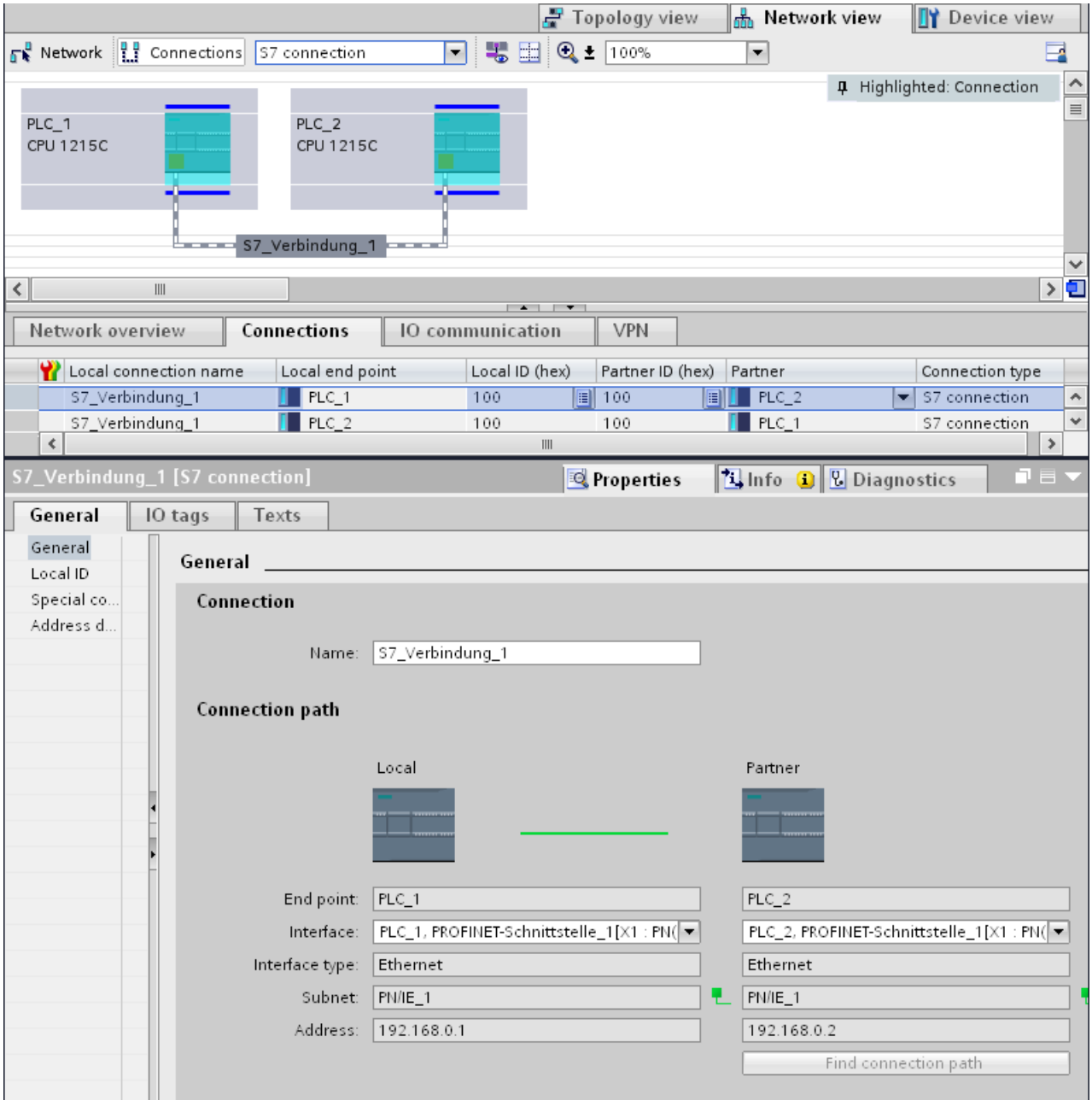
In the case of S7 connections via the PN interface, one connection resource per S7 connection is assigned for the endpoint for the S7-1200 CPU. One connection resource is also required for the connection partner.

You can find an overview of available and assigned connection resources for selected S7-1200 CPU in the Inspector window at "Properties > Connection Resources"

Views containing information about the configured connections

The views described below will provide you with comprehensive access to all the information and functions regarding configuring and checking communication connections.

- Connection display in the network view
- Connection table
- "Properties" tab for a connection in the inspector window



Benefits

The information shown in these views are always up-to-date in terms of the current user actions. This means:

- The connection table displays all connections created.
- If you have selected a connection in the connection table:
  - When connection mode is enabled, the connection path is highlighted in the network view.
  - The "Properties" tab in the Inspector window displays the parameters of this connection.

The connection table

The connection table offers the following functions:

- List of all connections in the project
- Selection of a connection and display of connections associated with it in the network view (when connection mode is enabled)
- Changing of connection partners
- Display showing status information

"Properties" tab for a connection in the inspector window

The properties dialog has the following meaning:

- Display for connection parameters
- Display of connection path
- Subsequent specification of connections using the "Find connection path" button

Creating a new connection

Creating a connection - alternatives

You have the following options for creating a connection in the network view:

- Graphic connection configuration
- Interactive connection configuration

You'll find the individual steps for this in the following chapters.

Requirement and result

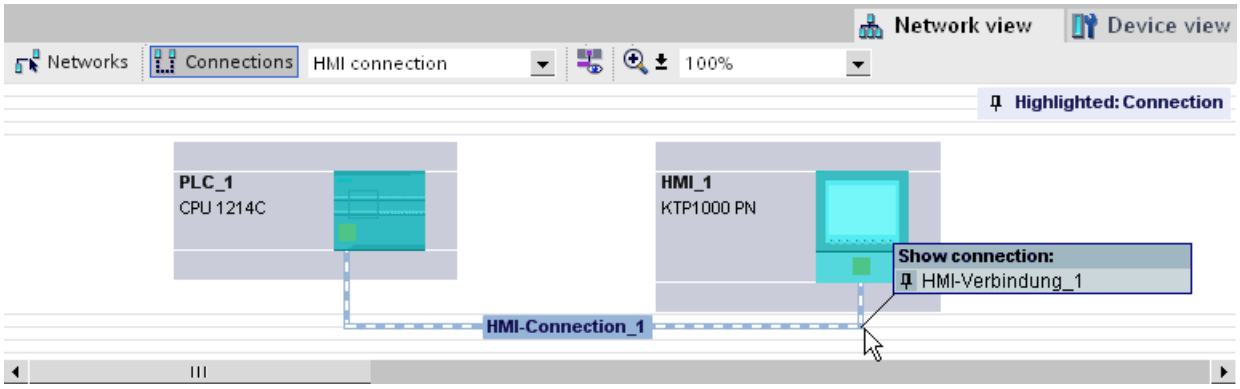
In the network view, you have add devices between which the connections should be configured.

Specifying a connection

If both partners for the connection type selected are networked on the same network, use the graphic or interactive selection of both communication partners to create a fully specified connection.

This connection is entered automatically in the connection table of the S7-1200 CPU. A local connection name is assigned for this connection.

The following schematic shows a configured connection with a networked device:



Creating a new connection graphically

Graphically configuring connections

In the case of graphic connection configuration, the connection path is automatically specified provided interfaces and resources are available. Select the devices to be connected in the current configuration.

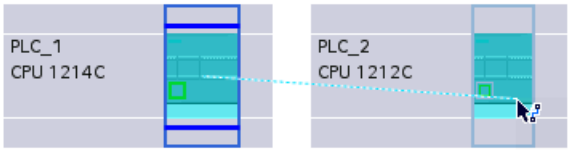
Automatically determining connection path

To create a connection graphically, follow these steps:

1. Click the "Connections" button.

   This step activates the connection mode: You can now select the connection type you want. You will see this from the following:
   The devices that can be used for the connection type selected in your project are color-highlighted in the network view.

2. Hold down the mouse button and drag the mouse pointer from the device from which the connection will originate to the device at which the connection ends.

3. Release the mouse button over the destination device to create the connection between the two devices.

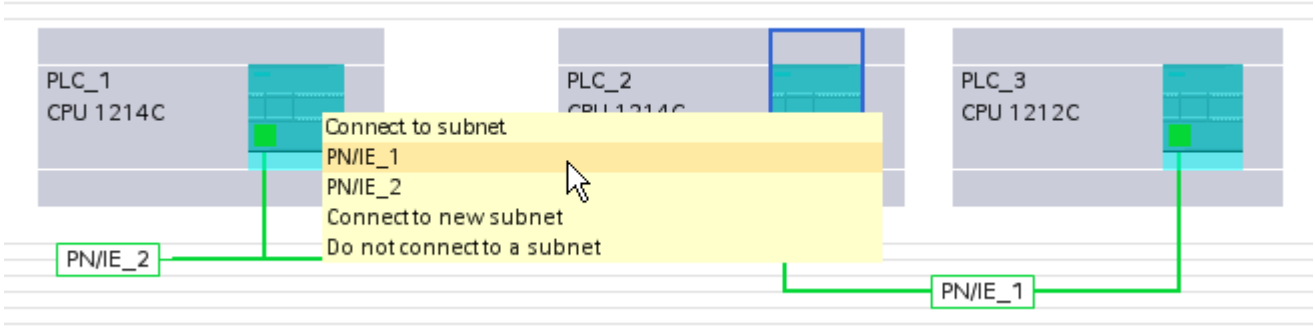
Result

- A specified connection is created.
- The connection path is highlighted.
- The connection is entered in the connection table.

Configuring a connection when there is no or no clear network assignment

Any networking not in place will if possible be created automatically when a connection is created. A query will be made upon completion of connection configuration if unique network assignment is not possible. You will be able to choose from the existing subnets.

Example below: A query is made upon creation of a connection between stations PLC_1 and PLC_2, which are not yet networked.



Interactively creating a new connection

Interactively configuring connections

Define the local device and its connection partners.

Procedure

Proceed as follows to interactively create a connection:

1. Select the "Add new connection" command in the shortcut menu of a connection partner for which you want to create a connection.
   The "Create new connection" dialog appears.
2. Select the partner endpoint.
   In the right pane of the dialog, a possible connection path fitting the selected endpoint is displayed, if available. Incomplete paths, for example, for a non-specified CPU, are marked by an exclamation mark on a red background.
3. To accept the configured connection and to configure additional connections to other endpoints, click "Add".
   To close the dialog, click "OK".

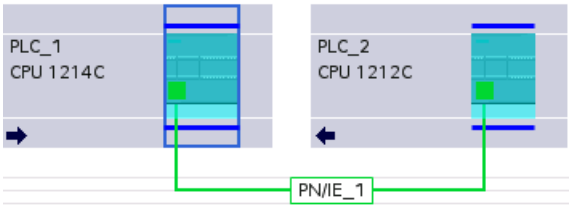
Working in the network view

Highlighting connection path and partner in the network view

To display the connection partners for all or certain connection types in the network view, proceed as follows:

1. Click the "Connections" button.

2. Select the S7-CPU for which you want to display the connection partners in the network view and then select the "Highlight connection partners" command in the shortcut menu.
3. Select "All connection partners" in the following menu.
   The local device and the CPUs of the target devices are selected. The local connection partner shows an arrow pointing right and the remote connection partners show an arrow pointing left.
4. To open a list with information on the target devices, click the arrow of the local device. This additional function is useful in complex network configurations in which some devices are not visible.



Note

You can display one of the connection partners which cannot be seen in the current display range of the network view. Click on the communication partner in the list that appears. Result: The display is moved such that the connection partner becomes visible.

Working with the connection table

Basic functions for tables

The connection table supports the following basic functions for editing a table:

- Changing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Changing column width

To adjust the width of a column to the content so that all texts in the lines are legible, follow these steps:

1. Position the cursor in the header of the connection table to the right of the column that you want to optimize until the cursor changes its shape to two parallel lines (as if you wanted to change the width of the column by dragging it with the cursor).
2. Double click on this point.

or

1. Open the shortcut menu on the header of the table.
2. Click on
   - "Optimize column width" or
   - "Optimize width of all columns".

For columns that are too narrow, the complete content of specific fields is shown when you pause with the cursor on the respective field.

Show / hide columns

You can use the shortcut menu of the header of the connection table to control the display of the various table columns. The shortcut menu entry "Show/hide columns" provides you with an overview of the available columns. Use the check box to control whether columns are shown or hidden.

If you want to store the layout, width and visibility of the table columns, click on the "Remember layout" function in the top right-hand of the network view.

Symbol	Meaning
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Using cursor keys to move within the connection table

You can use the UP and DOWN cursor keys to select a connection from the connection table; the selected connection is marked and is shown highlighted in the network view.

Changing properties of connection

You can directly edit some of the parameters displayed in the connection table. The name of the connection can, for example, only be changed in the connection table.

Changing connection partners

You can change the connection partner of a connection as follows:

1. Select the connection.
2. Select the new connection partner from the open drop-down list in the "Partner" column.

Deleting connections

You can delete configured connections using the network view or the connection table.

In the network view you can delete one highlighted connection per action. In the connection table you can delete one or several connections per action.

Procedure

To delete a connection, follow these steps:

1. Select the connection to be deleted:
   - In the network view: Select the connection to be deleted.
   - In the connection table: Select the rows of the connections to be deleted (multiple selection possible).
2. Open the shortcut menu with a right mouse click.
3. Select the "Delete" command.

Result

The selected connection is removed completely.

Copying connections

Introduction

Connections are not copied singly but always in context along with the project or the device.

You can copy:

- Entire projects
- One or more devices within a project or from one project to another

Copying a project

When you copy a project all configured connections will also be copied. No settings whatsoever are required for the copied connections because the connections remain consistent.

Copying devices

If you copy devices for which connections have been configured, the connections are copied as well. To complete the connection path, you must still finalize the networking.

An S7-1200 CPU with a V.10 firmware is merely a server for connections and has no connection configuration itself. Consequently, no connections are copied along with it when an S7-1200 CPU with a V1.0 firmware is copied.

Inconsistent connections - connections without assignment

With an inconsistent connection the structure of the connection data is destroyed or the connection is not functional in the project context.

Inconsistent connections cannot be compiled and loaded - operation is not possible with such a connection.

In the connection table inconsistent connections are marked in red.

Possible causes for inconsistent connections

- Deletion or change of the hardware configuration.
- Missing interface network links in the project, which are necessary for a connection.
- Connection resources are exceeded
- Connections to an unspecified connection partner without partner address information.

Detailed information regarding the reasons for the inconsistency can be found in the "Compile" tab following compilation (Edit > Compile).

Remedies

To assign a closed connection path to an existing open connection path, expand the device configuration in such a way that the interfaces required for the connection type are available for both partners. At "Properties > General > Interface" in the Inspector window, you can use the "Find connection path" button to create a connection to an existing partner.

S7 connection - general settings

General connection parameters

General connection parameters are displayed in the "General" parameter group under the properties of the connection; these connection parameters identify the local connection end point.

Here, you can assign the connection path and specify all aspects of the connection partner.

Local ID

The local ID of the module from which the connection is viewed is displayed here (local partner). You can change the local ID. You may need to do this if you have already programmed communication function blocks, and you want to use the local ID specified in those function blocks for the connection.

Special connection properties

Display of connection properties (can be modified depending on the components used):

- One-way
  One-way means that the connection partner functions as a server on this connection and cannot send or receive actively.
- Active connection establishment
  In the case of one-way connection, for example with a S7-1200 CPU (firmware version V1.0), a connection partner can only be provided for the active connection establishment. In the case of a two-way connection you can set which connection partner will assume the active role.
- Sending operating mode messages
  Indicates whether or not the local partner sends operating mode messages to the connection partner.

Address details

Displaying address details of the S7 connection. With an unspecified partner, the values for the rack and slot can be changed. All other values are obtained from the current configuration and cannot be changed.

S7 connection - address details

Meaning

The address details show the end points of the connection and can localize these via the specification of rack and slot.

When a connection is established, the connection-specific resources of a module are assigned permanently to this connection. This assignment requires that the connection resource can be addressed. The TSAP (Transport Service Access Point) is, as it were, the address of the resource that is formed with the help of the connection resource or, in the case of S7-1200 CPUs (firmware V2.0 or higher) with the SIMATIC-ACC (SIMATIC Application Controlled Communication).

Configuration of TSAP for S7-1200

- For S7-1200 CPU (firmare V2.0 or higher):
  "SIMATIC-ACC"<nnn><mm>
  nnn = Local ID
  mm = any value
- For S7-1200 CPU (firmare V1.0):
  <xx>.<yz>
  xx = Number of the connection resource
  y = Rack number
  z = Slot number

TSAP structure, dependent on partner

The configuration of the TSAP for S7-1200 CPUs is dependent on the respective firmware and on the remote connection partner. When a S7-1200 CPU is connected with a S7-300/400 CPU, a S7-1200 CPU also uses a TSAP configuration that includes the connection resource.

See the following examples of TSAPs of various connection configurations

- Connection between two S7-1200 CPUs (both with firmware V2.0):
  - S7-1200 CPU "A" with firmware V2.0 and local ID 100:
    TSAP: SIMATIC-ACC10001
  - S7-1200 CPU "B" with firmware V2.0 and local ID 5AE:
    TSAP: SIMATIC-ACC5AE01
- Connection between two S7-1200 CPUs (firmware V2.0 and V1.0):
  - S7-1200 CPU with firmware V2.0 and local ID 1FF:
    TSAP: SIMATIC-ACC1FF01
  - S7-1200 CPU with firmware V1.0 (rack 0, slot 1, connection resource 03):
    TSAP: 03.01
- Connection between S7-1200 CPUs (firmware V2.0) and S7-300/400 CPU:
  - S7-1200 CPU with firmware V2.0 (rack 0, slot 1, connection resource 12):
    TSAP: 12.01
  - S7-300/400 CPU (rack 0, slot 2, connection resource 11):
    TSAP: 11.02

S7 connections via CM/CP

Introduction

S7-1200 CPUs with a firmware version of V2.0 or higher support one-way and two-way S7 connections via CM/CP interfaces. This increases the number of Ethernet ports and networks that can be used for S7 connections. Even though the connection is then guided via the CM/CP, the associated S7-1200 CPU is an end point of the connection. The other end point can be any other device in the case of two-way connections. This other device must also support S7 connections.

Data volumes and configuration limit

The number of communication connections which are supported by CM/CP can be found in the manual accompanying every CM/CP. The number of connections per device can be further increased by adding other CM/CPs.

If several CM/CPs are fitted in a device, an automatic switch is made to the next CM/CP when this limit is exceeded. Specifically assign the connections using the path selection as required.

Note

Data transfer > 240 byte transfer is supported by the current CPs.

CPs with an older product version support the transfer of data with a data length of up to 240 bytes.

For more information, refer to the details provided in the Ethernet CP equipment manual.

Tasks of the Ethernet CM/CP in online mode

During data transfer via a connection, the Ethernet CM/CP takes on the following tasks:

- Receiving
  Receiving data from the Ethernet and forwarding to the user data area in the CPU
- Sending
  Transferring data from the user data area of the CPU and sending data via the Ethernet

The connection is established automatically as soon as the partner can be reached.

HMI connection

Introduction to configuring connections

Definition

A connection defines a logical assignment of two communication partners in order to undertake communication services. A connection defines the following:

- Communication partner involved
- Type of connection (e.g., HMI connection)
- Special properties (e.g., whether a connection is established permanently or whether it is established and terminated dynamically in the user program, and whether status messages are to be sent)
- Connection path

What you need to know about connection configuration

During connection configuration, a local connection name is assigned for an HMI connection as a unique local identification.

In the network view, a "Connections" tab is displayed in addition to the "Network overview" tab. This tab contains the connection table. A line in this connection table represents a configured connection, e.g., between an HMI device and PLC, along with its properties.

What you need to know about using connection resources

Introduction

Each connection requires connection resources for the end point and/or transition point on the devices involved. The number of connection resources is device-specific.

If all the connection resources of a communication partner are assigned, no new connections can be established. This situation is apparent when a newly created connection in the connection table has a red background. The configuration is then inconsistent and cannot be compiled.

HMI connections

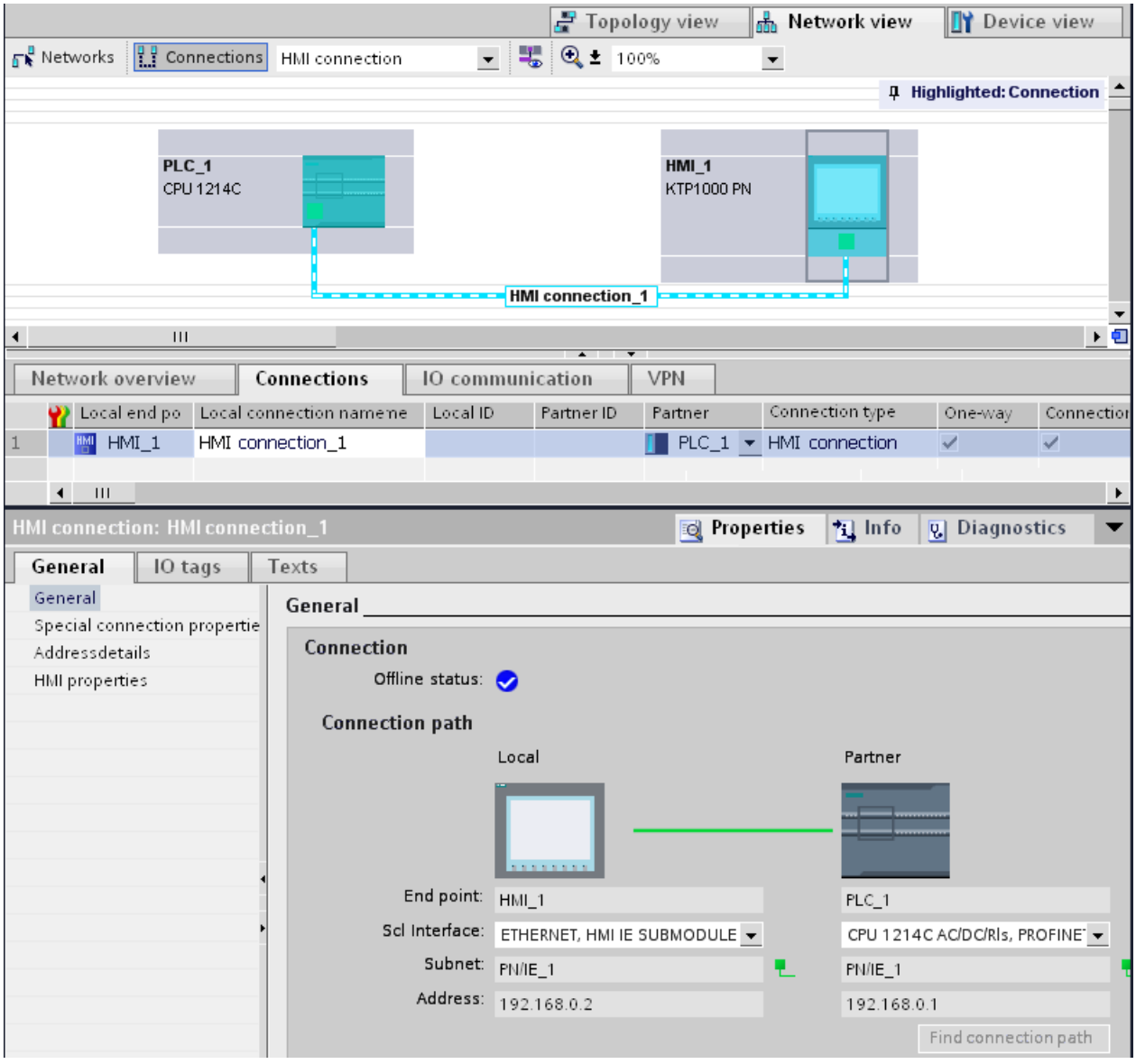
For HMI connections via the **integrated** PN interface, one connection resource for the endpoint per HMI connection is occupied for the HMI device.

One connection resource is also required for the connection partner (PLC).

Views containing information about the configured connections

The views described below will provide you with comprehensive access to all the information and functions regarding configuring and checking communication connections.

- Connection display in the network view
- Connection table
- "Properties" tab for a connection in the inspector window



Benefits

The information shown in these views are always up-to-date in terms of the current user actions. This means:

- The connection table displays all connections created.
- If you have selected a connection in the connection table:
  - You will graphically see the connection path in the network view.
  - The "Properties" tab in the Inspector window displays the parameters of this connection.

The connection table

The connection table offers the following functions:

- List of all connections in the project
- Selection of a connection and display of connections associated with it in the network view
- Changing of connection partners
- Display showing status information

"Properties" tab for a connection in the inspector window

The properties dialog has the following meaning:

- Display for connection parameters
- Display of connection path
- Subsequent specification of connections using the "Find connection path" button

Creating a new connection

Creating a connection - alternatives

You have the following options for creating a connection in the network view:

- Graphic connection configuration
- Interactive connection configuration

You'll find the individual steps for this in the following chapters.

Requirement and result

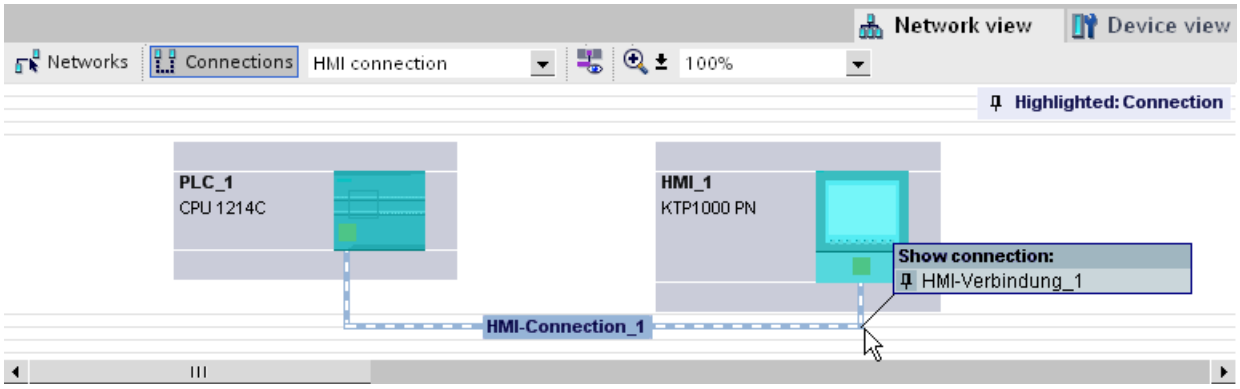
You have created the devices with CPUs and HMI devices between which you want to configure connections in the network view.

Specifying a connection

If both partners for the connection type selected are networked on the same network, use the graphic or interactive selection of both communication partners to create a fully specified connection.

This connection is entered automatically into the connection table of the HMI device. A local connection name is assigned for this connection.

The following schematic shows a configured connection with a networked device:



Creating a new connection graphically

Graphically configuring connections

When using the graphic connection configuration, if necessary the system asks you to define the connection path. Select the devices to be connected in the current configuration.

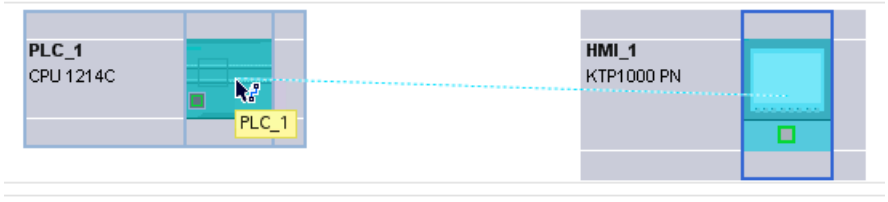
Automatically determining connection path

To create a connection graphically, follow these steps:

1. Click the "Connections" button.

   The connection mode for the connection type you have selected is then activated.
   You will see this from the following:
   The devices that can be used for the connection type selected in your project are color-highlighted in the network view.

2. Hold down the mouse button and drag the mouse pointer from the device from which the connection will originate to the device at which the connection ends.

3. Release the mouse button over the destination device to create the connection between the two devices.

Result

- A specified connection is created.
- The connection path is highlighted.
- The connection is entered in the connection table.

Interactively creating a new connection

Interactively configuring connections

Define the local device and its connection partners.

Procedure

Proceed as follows to interactively create a connection:

1. Select the "Create new connection" command in the shortcut menu of a connection partner for which you want to create a connection.
   The "Create new connection" dialog is opened.
2. Select the partner endpoint.
   In the right pane of the dialog, a possible connection path fitting the selected endpoint is displayed, if available. Incomplete paths, for example, for a non-specified CPU, are marked by an exclamation mark on a red background.
3. To close the dialog, click "OK".
   To accept the configured connection and to configure additional connections to other endpoints, click "Apply".

Working in the network view

Highlighting connection path and partner in the network view

To display the connection partners for all or certain connection types in the network view, proceed as follows:

1. Click the "Connections" button.

2. Select the "Highlight connection partners" command in the shortcut menu for the HMI device whose connection partners you want to display in the network view.
3. Select "All connection partners" in the following menu.
   The local device and the CPUs of the target devices are selected. The local connection partner shows an arrow pointing right and the remote connection partners show an arrow pointing left.
4. To open a list with information on the target devices, click the arrow of the local device. This additional function is useful in complex network configurations in which some devices are not visible.

Note

You can display one of the connection partners which cannot be seen in the current display range of the network view. Click on the communication partner in the list that appears. Result: The display is moved such that the connection partner becomes visible.

See also

Creating a new connection graphically (Page 619)

Working with the connection table

Basic functions for tables

The connection table supports the following basic functions for editing a table:

- Changing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Changing column width

To adjust the width of a column to the content so that all texts in the lines are legible, follow these steps:

1. Position the cursor in the header of the connection table to the right of the column that you want to optimize until the cursor changes its shape to two parallel lines (as if you wanted to change the width of the column by dragging it with the cursor).
2. Double click on this point.

or

1. Open the shortcut menu on the header of the table.
2. Click on
   - "Optimize column width" or
   - "Optimize width of all columns".

For columns that are too narrow, the complete content of specific fields is shown when you pause with the cursor on the respective field.

Show / hide columns

You can use the shortcut menu of the header of the connection table to control the display of the various table columns. The shortcut menu entry "Show/hide columns" provides you with an overview of the available columns. Use the check box to control whether columns are shown or hidden.

Using cursor keys to move within the connection table

You can use the UP and DOWN cursor keys to select a connection from the connection table; the selected connection is marked and is shown highlighted in the network view.

Changing properties of connection

You can directly edit the parameters displayed in the connection table in some cases. To change the name of a connection, you do not have to navigate to the Inspector window.

Changing connection partners

You can change the connection partner of a connection as follows:

1. Select the connection.
2. Select the new connection partner from the open drop-down list in the "Partner" column.

Deleting connections

You can delete configured connections using the network view or the connection table.

In the network view you can delete one highlighted connection per action. In the connection table you can delete one or several connections per action.

Procedure

To delete a connection, follow these steps:

1. Select the connection to be deleted:
   - In the network view: Select the connection to be deleted.
   - In the connection table: Select the rows of the connections to be deleted (multiple selection possible).
2. Open the shortcut menu with a right mouse click.
3. Select the "Delete" command.

Result

The selected connection is removed completely.

Copying connections

Introduction

Connections are not copied singly but always in context along with the project or the device.

You can copy:

- Entire projects
- One or more devices within a project or from one project to another

Copying a project

When you copy a project all configured connections will also be copied. No settings whatsoever are required for the copied connections because the connections remain consistent.

Copying devices

If you copy devices for which connections have been configured (HMI devices), the connections are copied as well. To complete the connection path, you must still finalize the networking.

An S7-1200 CPU with a V.10 firmware is only a server for connections and has no connection configuration itself. Consequently, no connections are copied along with it when an S7-1200 CPU with a V1.0 firmware is copied.

Inconsistent connections - connections without assignment

With an inconsistent connection the structure of the connection data is destroyed or the connection is not functional in the project context.

Inconsistent connections cannot be compiled and loaded - operation is not possible with such a connection.

In the connection table inconsistent connections are marked in red.

Possible causes for inconsistent connections

- Deletion or change of the hardware configuration.
- Missing interface network links in the project, which are necessary for a connection.
- Connection resources are exceeded
- Errors when backing up data due to insufficient memory
- Connections to an unspecified connection partner without partner address information.

Detailed information regarding the reasons for the inconsistency can be found in the "Compile" tab following compilation (Edit > Compile).

Remedies

If the connection cannot be repaired by opening the connection properties, changing them or undoing them in the configuration, then it may be necessary to delete the connection and re-create it.

HMI connection general settings

General connection parameters

General connection parameters are displayed in the "General" parameter group under the properties of the connection; these connection parameters identify the local connection end point.

Here, you can also assign the connection path and specify all aspects of the connection partner.

Special connection properties

Display of the connection properties (cannot be changed):

- Active connection establishment
  The connection establishment always starts from the HMI device. This option is selected by default if the address of the partner is specified.
- One-way
  One-way means that the connection partner functions as a server on this connection and cannot send or receive actively.
- Sending operating mode messages
  Not relevant for HMI devices.

Address details

Displaying address details of the HMI connection. With an unspecified partner, the values for the rack and slot can be changed. All other values are obtained from the current configuration and cannot be changed.

Miscellaneous

Display of the access points for the online connection between HMI device and connection partner.

Using Open User Communication

Basics of Open User Communication

Introduction

Open User Communication (OUC) is the name given to a program-controlled communication process for communicating via the integrated PN/IE interface of S7-1200/1500 and S7-300/400 CPUs. Different connection types are available for this communication process.

The main feature of Open User Communication is its high degree of flexibility in terms of the data structures transferred. This allows open data exchange with any communicating devices providing they support the connection types available here. Since this communication is controlled solely by instructions in the user program, event-driven connection establishment and termination is possible. Connections can also be modified by the user program during runtime.

For CPUs with an integrated PN/IE interface, the TCP, UDP, and ISO-on-TCP connection types are available for Open User Communication. The communication partners can be two SIMATIC PLCs or a SIMATIC PLC and a suitable third-party device.

Instructions for Open User Communication

To create the connections, you have various instructions available after opening in the program editor in the "Instructions > Communication > Open User Communication" task card:

- Compact instructions for sending or receiving data via the integrated functions for establishing and terminating the connection (S7-1200/1500 only):
  - TSEND_C (Page 3625) (connection establishment/termination, sending)
  - TRCV_C (Page 3636) (connection establishment/termination, receiving)
- Individual instructions for sending and receiving data or for establishing or terminating connections:
  - TCON (Page 3662) (connection establishment)
  - TDISCON (Page 3669) (connection termination)
  - TSEND (Page 3674) (TCP or ISO-on-TCP: Sending)
  - TRCV (Page 3677) (TCP or ISO-on-TCP: Receiving)
  - TUSEND (Page 3687) (UDP: Sending)
  - TURCV (Page 3690) (UDP: Receiving)

Connection establishment

For Open User Communication, instructions for establishing and terminating the connection must exist for both communication partners. One communication partner sends its data using TSEND, TUSEND or TSEND_C while the other communication partner receives the data using TRCV, TURCV or TRCV_C.

One of the communication partners starts the connection establishment as the active partner. The other communication partner reacts by starting its connection establishment as the passive partner. If both communication partners have initiated their connection establishment, the communication connection is fully established.

Connection configuration

You can specify establishment of the connection via a connection description DB with the TCON_Param, TCON_IP_v4, or TCON_IP_RFC structure by means of parameter assignment as follows:

- Manually create, assign parameters and write directly to the instruction.
- Supported by connection configuration.

Connection configuration supports the establishment of the connection and should, therefore, be given preference over the other methods.

You specify the following in the connection configuration:

- Connection partner
- Connection type
- Connection ID

- Connection description DB
- Address details according to selected connection type

In addition, you specify here which communication partner activates the connection establishment and which partner establishes a connection passively in response to a request from its communication partner.

See also

Principle of operation of connection-oriented protocols (Page 638)

Connection configuration

Overview of connection configuration

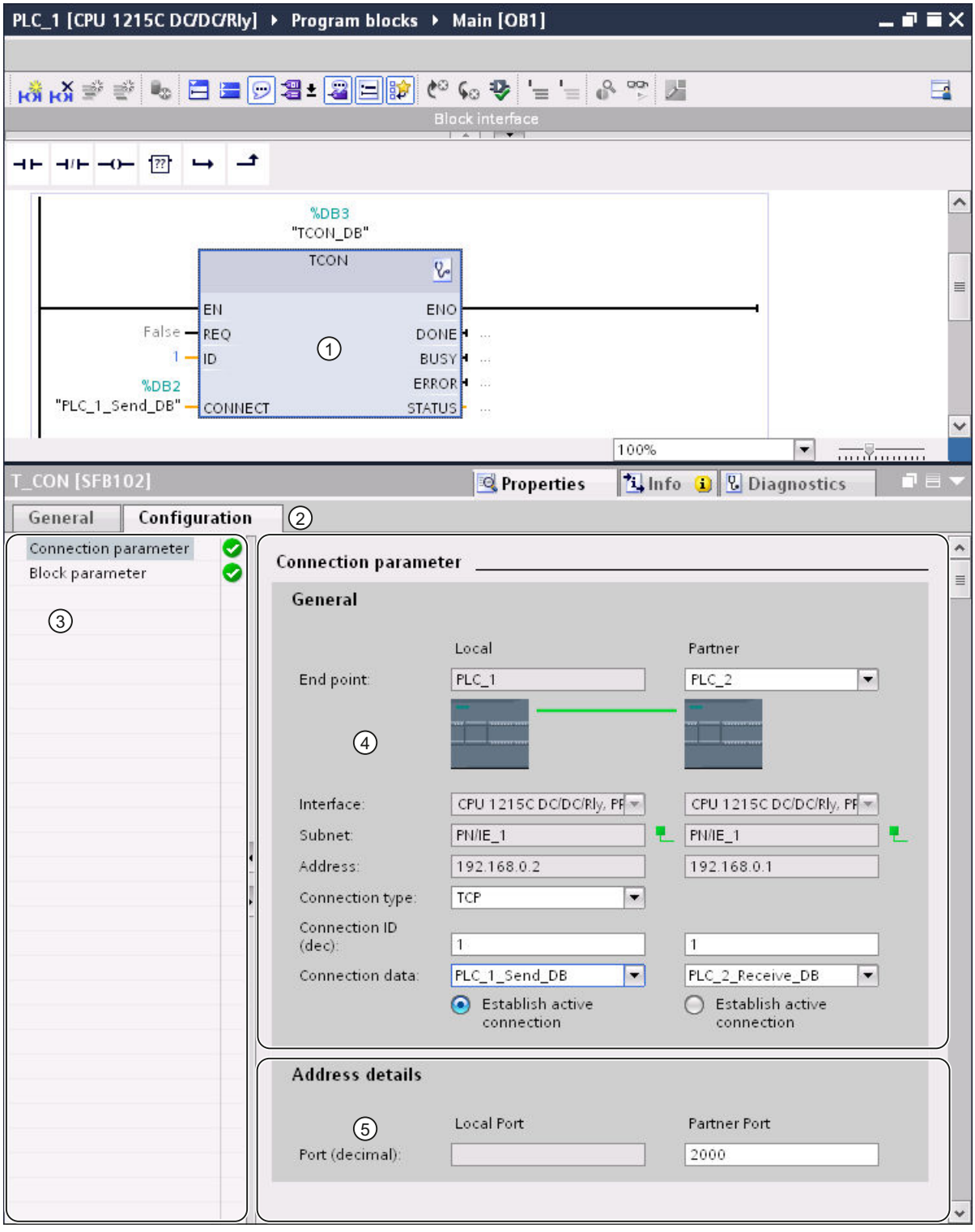
Introduction

You can find the connection configuration in the inspector window of the program editor if you want to program Open User Communication with the communication instructions TSEND_C, TRCV_C or TCON.

Connection configuration supports the flexible functionality of communication programming: The parameters entered for the connection configuration are stored in an automatically generated global DB derived from the TCON_Param, TCON_IP_v4 or TCON_IP_RFC structure. You can modify the connection parameters in this connection description DB.

Structure of the connection configuration

The connection configuration is made up of the following components:



① Communication instruction for TCON, TSEND_C or TRCV_C

② "Configuration" tab in the "Properties" tab
③ Area navigation of the "Configuration" tab
④ General properties of the connection parameters
⑤ Address details of the connection parameters (for selected connection DBs)

"Configuration" tab

Enter the desired connection parameters in the "Configuration" tab. The area navigation of the "Configuration" tab includes the "Connection parameters" group. This group contains the connection configuration. Here, you can enter the parameters for the connections and the address details with system support. Here, you also connect the CONNECT (TCON, TSEND_C, TRCV_C) or ID (TCON, TSEND, TRCV, TUSEND, TURCV) block parameters of the selected communication instructions.

When all the required parameters are assigned, a check mark is set in front of the "Connection parameters" group in the area navigation.

Note

The connection parameter assignment does not check whether the connection IDs and port numbers (TCP, UDP) or TSAPs (ISO-on-TCP, ISO) are unique. When you configure Open User Communication, you should, therefore, make sure that the parameter settings are unique within a device.

See also

Connection parameters with structure according to TCON_Param (Page 640)

Connection parameters with structure according to TCON_IP_v4 (Page 643)

Connection parameters with structure according to TCON_IP_RFC (Page 644)

Description of the connection parameters

Overview

The following table shows the general connection parameters:

Parameter	Description
End point	The names of the local end point and the partner end point are shown. The local end point is the CPU for which TCON, TSEND_C or TRCV_C is programmed. The local end point is, therefore, always known. The partner end point is selected from the drop-down list. The drop-down list shows all available possible connection partners including unspecified connection partners for devices whose data is unknown in the project. For S7-1500, broadcast can be selected as the partner end point (message to all subnet devices). For S7-1500 CPs/CMs, multicast can also be selected as the partner end point (message to a group within the subnet). The connection type is converted automatically to UDP in this case. As long as no connection partner is set, all other parameters in the mask are disabled.
Interface	The interface of the local end point is displayed. If multiple interfaces are available, e.g., by means of CPs or CMs, the interface can be selected from the drop-down list. To display or select the partner interface, a specified partner end point must first be selected.
Subnet	The subnet of the local end point is displayed, provided this exists. The partner subnet is displayed only after the partner end point has been selected. If at least one of the two connection partners is not connected with a subnet, the two connection partners are connected with each other. A connection between partners in different subnets is only possible with IP routing. The IP routing settings can be edited in the properties of the relevant Ethernet interfaces.
Address	The IP address or the PROFIBUS address of the local end point is displayed, depending on the subnet used. The corresponding address of the partner is displayed only after the partner end point has been selected. If you have selected an unspecified connection partner, the input box is empty and has a red background. In this case, you must specify a valid IP address/PROFIBUS address. The address type (IP or PROFIBUS) depends on the type of subnet that is set for the local partner. Broadcast (S7-1500 only): If "Broadcast" is set as the partner end point, a non-editable IP address with host address 255 is entered automatically for the connection partner. The network allocation corresponds to that of the sender. Example: Local IP address 192.168.0.1, partner IP address 192.168.0.255. Multicast (S7-1500 CPs/CMs only): If "Multicast" is set as the partner end point, the editable IP address 224.0.1.0 is entered automatically for the connection partner.
Connection type	Select the connection type you want to use from the "Connection type" drop-down list: • TCP • ISO-on-TCP • UDP With the S7-1500, you can also select the ISO connection type at the configuration type of the configured connections for TSEND_C and TRCV_C or TCON. The connection types can only be used for partners that support the corresponding protocol.

Parameter	Description
Connection type (for S7-1500 only)	With the S7-1500, two different configuration types can be set for TSEND_C and TRCV_C: • Programmed connections use program blocks for the connection description. • Configured connections are saved for the configuration and are only created after download to the device in runtime. You can also use the configured connection to select the connection type ISO. The specified configuration method depends on the selected connection type. If both configuration methods are possible, the programmed connection is preset. The same configuration method must be set for both connection partners.
Connection ID	Enter the connection ID in the input box. You can change the connection ID in the input boxes or enter it directly in TCON. Ensure that the connection ID assigned is unique within the device.
Connection data	The names of the connection description DBs for the connection description structured according to TCON_IP_v4, TCON_IP_RFC or TCON_Param are displayed in the drop-down lists. The drop-down list is still empty after selection of a connection partner. You can use the drop-down list to generate a new data block or to select an existing data block. This data block is filled automatically with the values from the connection configuration. The name of the selected data blocks is entered automatically in the CONNECT block parameter of the selected TSEND_C, TRCV_C or TCON instruction. From the drop-down list, you can also reference another valid data block. If a DB is referenced using the CONNECT input parameter of the TSEND_C, TRCV_C or TCON extended instructions and this does not correspond to the structure of a TCON_IP_v4, TCON_IP_RFC or TCON_Param, the drop-down list is shown with no content on a red background.
Connection name (for S7-1500 only)	If the connection type of the configured connections is set for TSEND_C and TRCV_C for the S7-1500, the "Connection data" parameter is replaced with the "Connection name" parameter. The name of the configured connection serves here as the connection data. The drop-down list is still empty after selection of a connection partner. You can use the drop-down list to generate a new connection or to select an existing connection. If needed, a data block is created and automatically filled with the values from the connection configuration. The name of the data block is entered automatically in the CONNECT block parameter of the TSEND_C or TRCV_C instruction. You can also reference an existing connection from the drop-down list.
Active connection establishment	Use the "Active connection establishment" check box to specify the active partner of the Open User Communication (only with TCP and ISO-on-TCP).
Port (only with TCP and UDP)	Address component for a TCP or UDP connection. The default after creating a new TCP connection is 2000. You can change the port numbers. The port numbers must be unique on the device!
TSAP (ISO-on-TCP only)	Address component for an ISO-on-TCP connection. The default value after creating a new ISO-on-TCP connection is E0.01.49.53.4F.6F.6E.54.43.50.2D.31 (S7-1200/1500) or E0.02.49.53.4F.6F.6E.54.43.50.2D.31 (S7-300/400). You can enter the TSAP-ID with an extension or as an ASCII TSAP. The TSAPs must be unique on the device!

Note

UDP connection for the "Broadcast" setting (S7-300/400/1200)

The parameters of the UDP connection for the "Broadcast" setting for the partner end point are stored in a connection description DB TCON_IP_v4 : With respect to UDP communication with TCON and TUSEND/TURCV , the TCON_IP_v4 is not filled with any partner parameters (value=0). However, the partner address and the partner port are necessary for sending the data and must be entered by the user in the TADDR_Param . The TADDR_Param for UDP communication is referenced by the TUSEND-/TURCV block parameter ADDR . The values for both parameters can be taken from the connection configuration.

The configuration must also be adapted for the other recipients of UDP communication. In order to receive broadcast frames, the partner port must be configured at the receiver end. For this purpose, the RemotePort parameter of the TADDR_Param must be filled at the ADDR block.

Note

Communication via TSEND_C and TRCV_C (S7-1500)

When TSEND_C and TRCV_C are used, a separate TSEND_C and TRCV_C block pair with a configured connection is required for each communication. Multiple TSEND_C and TRCV_C block pairs cannot simultaneously use the same configured connection for communication.

Additional connections for a TSEND_C or TRCV_C instruction can be created in the inspector window for the connection parameters using the appropriate button next to the connection data.

The connections configured using TSEND_C and TRCV_C are displayed in a connection table in the inspector window under "Properties > Configuration > Overview of configured connections" when the TSEND_C or TRCV_C block is selected.

See also

Assignment of port numbers (Page 645)

TSAP structure (Page 647)

Examples of TSAP assignment (Page 650)

Ability to read back connection description parameters (Page 646)

Creating and assigning parameters to connections (Page 633)

Connection parameters with structure according to TCON_Param (Page 640)

Connection parameters with structure according to TCON_IP_v4 (Page 643)

Connection parameters with structure according to TCON_IP_RFC (Page 644)

Starting connection parameter assignment

The connection configuration for Open User Communication is enabled as soon as a TCON, TSEND_C or TRCV_C instruction for communication is selected in a program block.

Requirement

- Your project must contain at least one S7-CPU.
- The program editor is open.
- A network is available.

Procedure

To insert the extended instructions for Open User Communication, proceed as follows:

1. Open the task card, pane and folder "Instructions > Communication > Open User Communication".
2. Drag one of the following instructions to a network:
   - TSEND_C
   - TRCV_C
   - TCON

   The "Call options" dialog opens.
3. Edit the properties of the instance DB in the "Call properties" dialog. You have the following options:
   - Change the default name.
   - Select the "Manual" check box to assign your own number.
   - You can also execute the DB as a multi-instance for function blocks.
4. Click "OK" to complete your entry.

Result

A corresponding instance DB is created at a single instance for the inserted instruction TSEND_C, TRCV_C or TCON. In the case of a multi-instance, the instance DB of the function block is used.

With TSEND_C, TRCV_C or TCON selected, you will see the "Configuration" tab under "Properties" in the Inspector window. The "Connection parameters" group in area navigation contains the connection parameter assignment that you can now make.

See also

Creating and assigning parameters to connections (Page 633)

Creating and assigning parameters to connections

In the connection configuration for Open User Communication, you can create and configure connections of the TCP, UDP or ISO-on-TCP type.

Requirement

A CPU exists with a TCON, TSEND_C or TRCV_C communication instruction.

Procedure

To create a connection for Open User Communication, follow these steps:

1. Select a TCON, TSEND_C or TRCV_C block of Open User Communication in the program editor.
2. Open the "Properties > Configuration" tab in the inspector window.
3. Select the "Connection parameters" group. Until you select a connection partner, only the empty drop-down list for the partner end point is enabled. All other input options are disabled.
   The connection parameters already known are displayed:
   - Name of the local end point
   - Interface of the local end point
   - IP address (for Ethernet subnet) or PROFIBUS address (for PROFIBUS subnet) of the local end point.
4. In the drop-down list box of the partner end point, select a connection partner. You can select an unspecified device or a CPU in the project as the communication partner. Certain connection parameters are then entered automatically.
   The following parameters are set:
   - Name of the partner end point
   - Interface of the partner end point
   - IP address (for Ethernet subnet) or PROFIBUS address (for PROFIBUS subnet) of the partner end point.

   If the connection partners are networked, the name of the subnet is displayed.
5. With the S7-1500, in the "Configuration type" drop-down list, you choose between using program blocks or configured connections.

6. Select an existing connection description DB in the "Connection data" drop-down list or for configured connections select an existing connection under "Connection name". You can also create a new connection description DB or a new configured connection. Later, you can still select other connection description DBs or configured connections or change the names of the connection description DBs in order to create new data blocks:
   - You can also see the selected data block at the interconnection of the CONNECT input parameter of the selected TCON, TSEND_C or TRCV_C instruction.
   - If you have already specified a connection description DB for the connection partner using the CONNECT parameter of the TCON, TSEND_C or TRCV_C instruction, you can either use this DB or create a new DB.
   - If you edit the name of the displayed data block in the drop-down list, a new data block with the changed name but with the same structure and content is generated and used for the connection.
   - Changed names of a data block must be unique in the context of the communication partner.
   - A connection description DB must have the structure TCON_Param, TCON_IP_v4 or TCON_IP_RFC, depending on CPU type and connection.
   - A data block cannot be selected for an unspecified partner.

   Additional values are determined and entered after the selection or creation of the connection description DB or configured connection.
   The following is valid for specified connection partners:
   - ISO-on-TCP connection type
   - Connection ID with default of 1
   - Active connection establishment by local partner
   - TSAP ID
     for S7-1200/1500: E0.01.49.53.4F.6F.6E.54.43.50.2D.31
     for S7-300/400: E0.02.49.53.4F.6F.6E.54.43.50.2D.31

   The following is valid for unspecified connection partners:
   - TCP connection type
   - Partner port 2000

   The following applies for a configured connection with a specified connection partner:
   - TCP connection type
   - Connection ID with default of 257
   - Active connection establishment by local partner
   - Partner port 2000

   The following applies for a configured connection with an unspecified connection partner:
   - TCP connection type
   - Local port 2000

7. Enter a connection ID as needed for the connection partner. No connection ID can be assigned to an unspecified partner.

   Note

   You must enter a unique value for the connection ID at a known connection partner. The uniqueness of the connection ID is not be checked by the connection parameter settings and there is no default value entered for the connection ID when you create a new connection.

8. Select the desired connection type in the relevant drop-down list. Default values are set for the address details depending on the connection type. You can choose between the following:
   - TCP
   - ISO-on-TCP
   - UDP

   For configured connections with S7-1500, ISO applies in addition.
9. You can edit the input boxes in the address details. Depending on the selected protocol, you can edit the ports (for TCP and UDP) or the TSAPs (for ISO-on-TCP and ISO).
10. Use the "Active connection establishment" check box to set the connection establishment characteristics for TCP, ISO and ISO-on-TCP. You can decide which communication partner establishes the connection actively.

Changed values are checked immediately for input errors by the connection configuration and entered in the data block for the connection description.

Note

Open User Communication between two communication partners can only work when the program section for the partner end point has been downloaded to the hardware. To achieve fully functional communication, make sure that you load not only the connection description of the local CPU on the device but also that of the partner CPU as well.

See also

Description of the connection parameters (Page 630)

Starting connection parameter assignment (Page 632)

TSAP structure (Page 647)

Assignment of port numbers (Page 645)

Connection parameters with structure according to TCON_Param (Page 640)

Connection parameters with structure according to TCON_IP_v4 (Page 643)

Connection parameters with structure according to TCON_IP_RFC (Page 644)

Deleting connections

Introduction

The data of a created connection for Open User Communication is stored in a connection description DB. You can delete the connection by deleting the data block containing the connection description.

Requirement

You have created an Open User Communication connection.

Procedure

To delete a connection, follow these steps:

1. Select a communication partner for Open User Communication in the project tree.
2. Open the "Program blocks > System blocks > Program resources" folder below the selected communication partner.
3. Select the "Delete" command from the shortcut menu of the data block with the connection parameter assignment.

Note

If you are not certain which block to delete, open the extended instruction TCON, TSEND_C or TRCV_C. You will find the name of the data block as the CONNECT input parameter or in the connection parameter assignment as the "Connection data" parameter.

If you only delete the instance DBs of the extended instructions TCON, TSEND_C or TRCV_C, the assigned connections are not deleted as well.

Note

If the connection DB is still being used by other blocks of the extended instructions, then the corresponding calls, their instance DBs, and, if present, the combination blocks TSEND_C and TRCV_C must also be deleted from the block folder, provided they are not used elsewhere.

This action prevents the program from being inconsistent.

Result

You have deleted the connection.

Note

Insert an extended instruction TCON, TSEND_C, or TRCV_C again in order to reference an existing connection description with the TCON_Param, TCON_IP_v4, or TCON_IP_RFC structure again via the "Connection data" parameter.

How protocols work

Principle of operation of connection-oriented protocols

Introduction

Connection-oriented protocols establish a logical connection to the communication partner before data transmission is started. After the data transmission is complete, they then terminate the connection, if necessary. Connection-oriented protocols are used especially when reliable data transmission is important. Several logical connections can exist over one physical line.

Open User Communication supports the following connection types:

- TCP
- ISO-on-TCP
- ISO (S7-1500 only)
- UDP

Both communication partners must support the same connection type for a connection. If a communication partner does not support a connection of the type ISO-on-TCP, for example, use the connection type TCP instead, if it is supported.

For communication partners that cannot be configured in the TIA Portal, such as third-party devices or PCs, enter "unspecified" for the partner end point during connection parameter assignment. The required connection type for unspecified devices is listed in the respective documentation.

Note

Connections with ISO

For S7-1500 CPUs, configured connections of the type ISO can be created using the TSEND_C and TRCV_C instructions. For additional information on these connection types, refer to the general connection descriptions.

Characteristics of TCP

TCP is a streaming protocol in which the length of the data stream is transmitted to the receiver so that it can receive the data stream as individual TCP segments. This means no information about the start and end of a message is transmitted during data transmission via a TCP connection. The receiver cannot determine by the received segments of the data stream where one message in the data stream ends and the next one begins. It is therefore recommended that the number bytes to be received (parameter LEN, instruction TRCV/TRCV_C) be assigned the same value as the number of bytes to be sent (parameter LEN, instruction TSEND/ TSEND_C).

If the length of the sent data and the length of the expected data do not match, the following occurs:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C copies the received data to the specified receive area (parameter DATA) only after the assigned length is reached. When the assigned length is reached, data of the subsequent job are already being received. As a result, the receive area contains data from two different send jobs. If you do not know the exact length of the first message, you are unable to recognize the end of the first message and the start of the second message.
- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C copies the number of bytes you specified in the LEN parameter to the receive data area (parameter DATA). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the value of LEN. With each subsequent call, you receive a further block of the sent data.

A receive area with fixed data length can be specified in the TRCV/TRCV_C instructions with the protocol version of the Ad-hoc mode.

Characteristics of ISO-on-TCP

ISO-on-TCP is a message-oriented protocol which detects the end of the message at the receiver end and indicates the data that belongs to the message to the user. This does not depend on the specified reception length of the message. This means that information regarding the length and the end of a message is included during data transmission via an ISO-on-TCP connection.

If the length of the sent data and the length of the expected data do not match, the following occurs:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C copies all the sent data to the receive data area (parameter DATA). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the length of the data sent.
- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C does not copy any data to the receive data area (parameter DATA), but instead supplies the following error information: ERROR=1, STATUS=W#16#8088 (destination buffer too small).

Characteristics of UDP

UDP is a message-oriented protocol which detects the end of the message at the receiver end and indicates the data that belongs to the message to the user. This does not depend on the specified reception length of the message. This means that information on the length and the end of a message is included during data transmission via a UDP connection.

If the length of the sent data and the length of the expected data do not match, the following occurs:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TUSEND/TSEND_C): TURCV/TRCV_C copies all the sent data to the receive data area (DATA parameter). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the length of the data sent.
- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TUSEND/TSEND_C): TRCV/TRCV_C copies as much data to the receive data area (parameter DATA) as the LEN parameter requests. No further error message is generated. In this case, the user has to call a T_URCV again in order to receive the remaining bytes.

See also

Basics of Open User Communication (Page 625)

TSEND_C: Send data via Ethernet (Page 3625)

TRCV_C: Receive data via Ethernet (Page 3636)

TSEND: Send data via communication connection (Page 3674)

TRCV: Receive data via communication connection (Page 3677)

TUSEND: Send data via Ethernet (UDP) (Page 3687)

TURCV: Receive data via Ethernet (UDP) (Page 3690)

Connection parameters with structure according to TCON_Param

Data block for connection description

A connection description DB with a structure according to TCON_Param is used for some S7-1200 CPUs when it comes to the assignment of parameters for TCP, UDP and ISO-on-TCP communication connections. The fixed data structure of the TCON_Param contains all the parameters that are needed to establish the connection. The connection description DB is automatically created for a new connection by the connection configuration for Open User Communication when the TSEND_C, TRCV_C or TCON instruction is used.

The CONNECT connection parameter of the instance DBs for TSEND_C, TRCV_C or TCON contains a reference to the data block used.

Structure of the connection description according to TCON_Param

Byte	Parameter	Data type	Start value	Description
0 … 1	block_length	UINT	64	Length: 64 bytes (fixed)
2 … 3	id	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.

Byte	Parameter	Data type	Start value	Description
4	connection_type	USINT	17	Connection type: • 17: TCP (17 dec = 0x11 hex) • 18: ISO-on-TCP (18 dec = 0x12 hex) • 19: UDP (19 dec = 0x13 hex)
5	active_est	BOOL	TRUE	Identifier for the type of connection establishment. FALSE always applies to UDP, since data can be sent and received via local ID. The following is valid for TCP and ISO-on-TCP: • FALSE: Passive connection establishment • TRUE: Active connection establishment
6	local_device_id	USINT	1	ID for the local PN/IE interface.
7	local_tsap_id_len	USINT	0	Length of parameter local_tsap_id used, in bytes; possible values: • 0 or 2, if connection type = 17 (TCP) Only the value 0 is permissible for the active side. • 2 to 16, if connection type = 18 (ISO-on-TCP) • 2, if connection type = 19 (UDP)
8	rem_subnet_id_len	USINT	0	This parameter is not used.
9	rem_staddr_len	USINT	4	Length of address of partner end point, in bytes: • 0: unspecified, in other words, parameter rem_staddr is irrelevant. • 4: valid IP address in the parameter rem_staddr (TCP and ISO-on-TCP only)
10	rem_tsap_id_len	USINT	2	Length of parameter rem_tsap_id used, in bytes; possible values: • 0 or 2, if connection type = 17 (TCP) Only the value 0 is permissible for the passive side. • 2 to 16, if connection type = 18 (ISO-on-TCP) • 0, if connection type = 19 (UDP)
11	next_staddr_len	USINT	0	This parameter is not used.
12 … 27	local_tsap_id	ARRAY [1..16] of BYTE	-	Local address component of connection: • TCP and UDP: local port no. (possible values: 1...49151; recommended values: 2000...5000); local_tsap_id[1] = high byte of port no. in hexadecimal notation; local_tsap_id[2] = low byte of port no. in hexadecimal notation; local_tsap_id[3-16] = irrelevant • ISO-on-TCP: local TSAP-ID: local_tsap_id[1] = B#16#E0; local_tsap_id[2] = rack and slot of local end points (bits 0 to 4: Slot number, bits 5 to 7: rack number); local_tsap_id[3-16] = TSAP extension, optional Note: Make sure that every value of local_tsap_id is unique within the CPU.

Byte	Parameter	Data type	Start value	Description
28 … 33	rem_subnet_id	ARRAY [1..6] of USINT	-	This parameter is not used.
34 … 39	rem_staddr	ARRAY [1..6] of USINT	-	TCP and ISO-on-TCP only: IP address of the partner end point, for example, for 192.168.002.003: • rem_staddr[1] = 192 • rem_staddr[2] = 168 • rem_staddr[3] = 002 • rem_staddr[4] = 003 • rem_staddr[5-6]= irrelevant
40 … 55	rem_tsap_id	ARRAY [1..16] of BYTE	-	Partner address component of connection • TCP: partner port no. (possible values: 1...49151; recommended values: 2000...5000); rem_tsap_id[1] = high byte of port no. in hexadecimal notation; rem_tsap_id[2] = low byte of port no. in hexadecimal notation; rem_tsap_id[3-16] = irrelevant • ISO-on-TCP: partner TSAP-ID: rem_tsap_id[1] = B#16#E0; rem_tsap_id[2] = rack and slot of partner end point (bits 0 to 4: Slot number, bits 5 to 7: rack number); rem_tsap_id[3-16] = TSAP extension, optional • UDP: This parameter is not used.
56 … 61	next_staddr	ARRAY [1..6] of BYTE	-	This parameter is not used.
62 … 63	spare	WORD	W#16#0000	Reserved.

Note

TCON_Param for S7-1500 CPU

The connection description DB with the structure according to TCON_Param is also supported by S7-1500 CPUs for migration reasons. However, we recommend that you use the new structures TCON_IP_v4 and TCON_IP_RFC.

See also

Principle of operation of connection-oriented protocols (Page 638)

Description of the connection parameters (Page 630)

Ability to read back connection description parameters (Page 646)

Overview of connection configuration (Page 627)

TSAP structure (Page 647)

Assignment of port numbers (Page 645)

Connection parameters with structure according to TCON_IP_v4

Data block for connection description

A connection description DB with a structure according to TCON_IP_v4 is used for CPUs of S7-1200 V4.0 and higher and S7-1500 to assign parameters for TCP and UDP communication connections. The fixed data structure of the TCON_IP_v4 contains all parameters that are required to establish the connection. The connection description DB is automatically created for a new connection by the connection configuration for Open User Communication when the TSEND_C, TRCV_C or TCON instruction is used.

The CONNECT connection parameter of the instance DBs for TSEND_C, TRCV_C or TCON contains a reference to the data block used.

Structure of the connection description according to TCON_IP_v4

Byte	Parameter	Data type	Start value	Description
0 … 1	interface_id	HW_ANY	64	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	id	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.
4	connection_type	BYTE	11	Connection type: • 11: TCP (11 dec = 0x0B hex) • 19: UDP (19 dec = 0x13 hex) For compatibility and migration reasons, the CPU S7-1500 also allows the values of the connection description DB with a structure according to TCON_Param. For the connection type TCP, the entry 17 is therefore also valid (17 dec = 0x11 hex).
5	active_established	BOOL	TRUE	Identifier for the type of connection establishment: • FALSE: Passive connection establishment • TRUE: Active connection establishment
6 … 9	remote_address	ARRAY [1..4] of BYTE	-	IP address of the partner end point, for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1
10 … 11	remote_port	UINT	2000	Port address of the remote connection partner (value range: 1 to 49151).
12 … 13	local_port	UINT	2000	Port address of the local connection partner (value range: 1 to 49151).

See also

Principle of operation of connection-oriented protocols (Page 638)

Description of the connection parameters (Page 630)

Ability to read back connection description parameters (Page 646)

Overview of connection configuration (Page 627)

Assignment of port numbers (Page 645)

Connection parameters with structure according to TCON_IP_RFC

Data block for connection description

A connection description DB with a structure according to TCON_IP_RFC is used for CPUs of S7-1200 V4.0 and higher and S7-1500 to assign parameters to ISO-on-TCP communication connections. The fixed data structure of the TCON_IP_RFC contains all parameters that are required to establish the connection. The connection description DB is automatically created for a new connection by the connection configuration for Open User Communication when the TSEND_C, TRCV_C or TCON instruction is used.

The CONNECT connection parameter of the instance DBs for TSEND_C, TRCV_C or TCON contains a reference to the data block used.

Structure of the connection description according to TCON_IP_RFC

Byte	Parameter	Data type	Start value	Description
0 … 1	interface_id	HW_ANY	64	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	id	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.
4	connection_type	BYTE	12	Connection type 12: ISO-on-TCP (12 dec = 0x0C hex) For compatibility and migration reasons, the CPU S7-1500 also allows the values of the connection description DB with a structure according to TCON_Param. For the connection type ISO-on-TCP, the entry 18 is therefore also valid (18 dec = 0x12 hex).
5	active_established	BOOL	TRUE	Identifier for the type of connection establishment: • FALSE: Passive connection establishment • TRUE: Active connection establishment
8 … 11	remote_address	ARRAY [1..4] of BYTE	-	IP address of the partner end point, for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1

Byte	Parameter	Data type	Start value	Description
12 … 45	remote_tselector	TSelector	-	TSelector of the remote connection partner: • TSelLength = Value range 0 to 32 as UINT • TSel[1-32] = Value range each 0 to 255 in bytes
46 … 79	local_tselector	TSelector	-	TSelector of the local connection partner: • TSelLength = Value range 0 to 32 as UINT • TSel[1-32] = Value range each 0 to 255 in bytes

See also

Principle of operation of connection-oriented protocols (Page 638)

Description of the connection parameters (Page 630)

Ability to read back connection description parameters (Page 646)

Overview of connection configuration (Page 627)

TSAP structure (Page 647)

Assignment of port numbers

Introduction

When an Open User Communication is created, the value 2000 is automatically assigned as the port number.

Permissible values for port numbers are 1 to 49151. You can assign any port number within this range. However, because some ports may already be used depending on the system, port numbers within the range from 2000 to 5000 are recommended.

Note

Port numbers must be unique. The connection configuration or a corresponding block call is rejected with an error if the port numbers are assigned twice.

Overview of port numbers

The following table summarizes the system reactions to various port numbers.

Port no.	Description	System reaction
2000 … 5000	Recommended range	No warning, no error message on entry Port number is permitted and accepted
1 … 1999, 5001 … 49151	Can be used, but is outside the recommended range	Warning message on entry Port number is permitted and accepted
0, 20, 21, 25, 80, 102, 135, 161, 34962 … 34964	Can be used conditionally*	
53, 80, 102, 135, 161, 162, 443, 520, 9001, 34962 … 34964	Can be used conditionally**	

* Ports defined for specific functions:
0: ANY - Port number is automatically assigned by S7-1500 CPU as value (>49151)
20: FTP data transmission
21: FTP control
25: TMAIL_C (Simple Mail transfer protocol)
80: Web server
102: ISO-on-TCP (RFC1006)
135: DCE Endpoint Mapper for PROFINET
161: SNMP (Simple Network Management Protocol)
34962 ... 34964: PROFINET

Note

The user usually specifies the value 0 for the local port on the active connection end point for UDP/TCP. In this case, the CPU operating system selects the next available port above 49151. The partner port usually has the default 0 with the passive connection end point. The corresponding parameter is disabled in the connection configuration.

** These ports are disabled depending on the function scope of the CPU in use. The documentation of the respective CPUs provides the assignment of these ports.

See also

Description of the connection parameters (Page 630)

Creating and assigning parameters to connections (Page 633)

Ability to read back connection description parameters

Changing parameter values in the connection description

The connection description for exactly one connection of the Open User Communication is entered from the connection configuration in the connection description DB.

You can change the parameter values of the connection description DB outside of the connection configuration in the user program. Connection description DBs containing values you changed subsequently can be read back from the connection configuration. Under "Properties > Configuration > Connection parameters", the inspector window displays only the connection parameters stored in the connection description DB.

Note

You can only change the values in the running user program if the instructions TCON, TSEND_C or TRCV_C are not being processed and the referenced connection is not established.

The connection configuration does not support nested entries of connection descriptions in DB types that can only be found via offset referencing (for example, Global-DB).

The structure of the connection description cannot be changed.

Ability to read back individual connection parameters

For the "Address" parameter of the communication partner in a TCP or ISO-on-TCP connection, its IP address is displayed from the "rem_staddr" parameter of the connection description.

The following values can also be reloaded from the connection description:

- Connection type
- Local connection ID
- Active/passive connection establishment (only with UDP)
- Local TSAP (ISO-on-TCP only)
- Partner TSAP (ISO-on-TCP only)
- Local port (only with TCP and UDP)
- Partner port (only with TCP)

The values of the connection ID parameters of the communication partner, the connection data, as well as the connection establishment, are not included in the connection description in the local connection description DB. Consequently, these parameters cannot be displayed when the connection configuration is reopened. The connection establishment of the partner results from the local connection establishment and is therefore also displayed.

A new communication partner can be selected at any time in the "Partners" drop-down list box.

When a CPU recognized in the project is selected as a specified communication partner, the entry options for the connection ID and the connection data are shown again.

See also

Connection parameters with structure according to TCON_Param (Page 640)

Description of the connection parameters (Page 630)

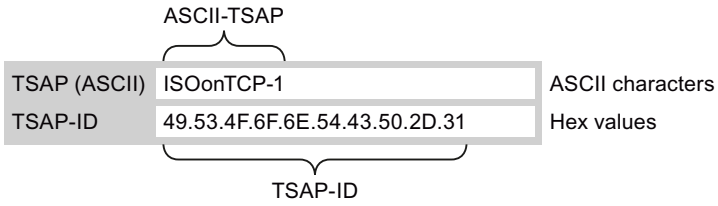
TSAP structure

Introduction

For an ISO-on-TCP connection, Transport Service Access Points (TSAPs) must be assigned for both communication partners. TSAP IDs are assigned automatically after an ISO-on-TCP connection is created. To ensure the uniqueness of TSAP IDs within a device, you can change the preassigned TSAPs in the connection parameter assignment.

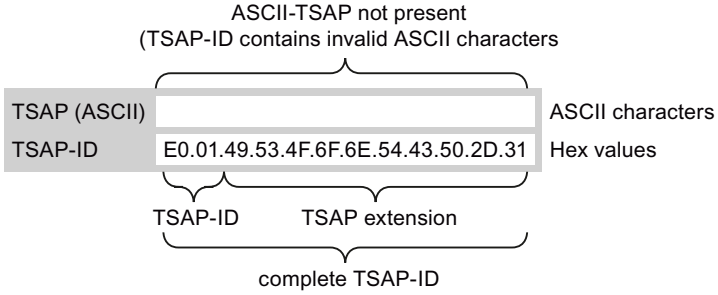
Structure of TSAPs

You must comply with certain rules when assigning TSAPs. A TSAP must contain a certain number of bytes, which are able to be displayed and entered as hexadecimal values (TSAP-ID) or as ASCII characters (ASCII-TSAP):



Entries or changes of the TSAP-ID or the ASCII-TSAP in the corresponding entry fields always take effect in the other display format as well.

If a TSAP contains no valid ASCII characters, the TSAP is displayed only as TSAP-ID and not as ASCII-TSAP. This is the case after a connection is created. The first two hex characters as TSAP-ID identify the communication type and the rack/slot. Because these characters are not valid ASCII characters for a CPU, the ASCII-TSAP is not displayed in this case.



In addition to the rules for length and structure of TSAPs, you must also ensure the uniqueness of the TSAP-ID. The assigned TSAPs are not automatically unique.

Length and content of TSAPs

Structure of the TSAP ID with TSAP extension:

- Valid for CPU S7-1200 firmware V1
  Length = 2 to 16 bytes
  x_tsap_id[0] = 0xE0 (Open User Communication)
  x_tsap_id[1] (bits 0 to 4) = slot number of CPU
  x_tsap_id[1] (bits 5 to 7) = rack number of CPU
  x_tsap_id[2...15] = any characters (TSAP extension, optional)
  (x = loc (local) or x = rem (partner))
- Valid for CPU S7-1500 and S7-1200 firmware as of V2
  Length = 2 to 16 bytes
  x_tsap_id[0] = 0xE0 (Open User Communication)
  x_tsap_id[1] = 0x00 to 0xFF
  x_tsap_id[2...15] = any characters (TSAP extension, optional)
  (x = loc (local) or x = rem (partner))

Structure of the TSAP ID as ASCII TSAP:

- Valid for CPU S7-1200 firmware V1
  Length = 3 to 16 bytes
  x_tsap_id[0...2] = 3 ASCII characters (0x20 to 0x7E)
  x_tsap_id[3...15] = any characters (optional)
  (x = loc (local) or x = rem (partner))
- Valid for CPU S7-1200 firmware as of V2
  Length = 3 to 16 bytes
  x_tsap_id[0...2] for active connection = 3 ASCII characters (0x00 to 0xFF) or any bit string*
  x_tsap_id[0...2] for passive connection = 3 ASCII characters (0x20 to 0x7E) or any bit string*
  x_tsap_id[3...15] = any characters (optional)
  (x = loc (local) or x = rem (partner))
- Valid for S7-1500 CPU
  Length = 3 to 16 bytes
  x_tsap_id[0...2] = 3 ASCII characters (0x00 to 0xFF) or any bit string*
  x_tsap_id[3...15] = any characters (optional)
  (x = loc (local) or x = rem (partner))

* ASCII character strings must not start with "SIMATIC-"

The following table shows the schematic structure of various TSAP IDs:

TSAP-ID	tsap_id_len	tsap_id[0]	tsap_id[1]	tsap_id[2]	tsap_id[3..15]
...with extension (CPU S7-1200 firmware V1)	2...16 bytes	0xE0	0x01 or 0x02 or 0x00*	Extension (optional)	Extension (optional)
...with extension (CPU S7-1500, S7-1200 firmware as of V2)	2...16 bytes	0xE0	0x00...0xFF	Extension (optional)	Extension (optional)
...as ASCII-TSAP (CPU S7-1200 firmware V1)	3...16 bytes	0x20...0x7E	0x20...0x7E	0x20...0x7	Any (optional)

*An S7-1200 CPU is normally inserted on rack 0 and slot 1, and an S7-300/400 CPU on rack 0 and slot 2. For this reason, hex value 01 or 02 is valid for the second position of the TSAP ID with extension. If the connection partner is an unspecified CPU, for example, a third-party device, the hex value 00 is also permissible for the slot address.

Note

For unspecified communication partners, the local TSAP-ID and the partner TSAP-ID can have a length of 0 to 16 bytes, in which all hex values from 00 to FF are permitted.

ASCII code table for entry of ASCII TSAPs

When entering ASCII TSAPs with the hexadecimal values from 20 to 7E, only the following characters are permitted:

Code	..0	..1	..2	..3	..4	..5	..6	..7	..8	..9	..A	..B	..C	..D	..E	..F
2..		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
3..	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4..	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5..	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6..	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7..	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

See also

Examples of TSAP assignment (Page 650)

Description of the connection parameters (Page 630)

Creating and assigning parameters to connections (Page 633)

Examples of TSAP assignment

The following examples show the processing of the TSAPs for CPUs of the S7-1200/1500 (CPU on slot 1) under various points of view:

- Example 1: Creating a new connection for PLC-PLC communication
- Example 2: Entry of a local ASCII-TSAP
- Example 3: Entry of a TSAP extension in the TSAP-ID
- Example 4: Incorrect editing of the TSAP-ID
- Example 5: Entry of an ASCII-TSAP via the "TSAP-ID" entry field

Example 1: Creating a new connection for PLC-PLC communication

Once you have created a new connection with two PLCs for the Open User Communication, the TSAP extension "ISOonTCP-1" is assigned automatically.

This TSAP extension produces the TSAP-ID E0.01.49.53.4F.6F.6E.54.43.50.2D.31, which is entered automatically in the connection description DB and in the entry fields of the local and the partner TSAP. The entry fields of the ASCII-TSAPs remain empty:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	E0.01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

You can change the values in the entry fields of the TSAP-ID and the ASCII-TSAP at any time.

The entry field of the TSAP-ID shows the complete TSAP stored in the data block of the connection description. The TSAP-ID with TSAP extension, which is limited to 16 characters,

is not displayed in the "TSAP (ASCII)" entry field because the character E0 does not represent a valid character for the ASCII-TSAP.

If the displayed TSAP-ID is a valid ASCII-TSAP, it is displayed in the "TSAP (ASCII)" entry field.

Changes in the entry fields for TSAP-ID and ASCII-TSAP affect the other field.

Example 2: Entry of a local ASCII-TSAP

If you have created a new connection and assigned an ASCII value for the local TSAP in the "TSAP (ASCII)" entry field, for example, "ISOonTCP-1", the resulting TSAP-ID is created automatically.

When you exit the "TSAP (ASCII)" entry field, the number of ASCII characters is checked automatically for compliance with the limit (3 to 16 characters) and the resulting TSAP-ID is entered into the corresponding entry field:

	Local TSAP	Partner TSAP
TSAP (ASCII)	ISOonTCP-1	
TSAP-ID	49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

Example 3: Entry of a TSAP extension in the TSAP-ID

If, following creation of a connection and entry of an ASCII-TSAP (see examples 1 and 2) in the entry field of the local TSAP-ID, you add the prefix "E0.01" to the TSAP value, the ASCII-TSAP will no longer be displayed when the entry field is exited.

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	E0.01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

Once you have exited the entry field of the TSAP-ID, a check is performed automatically to determine whether the first character of the TSAP-ID is a valid ASCII character. Since the character "E0" now present in the TSAP-ID is not a valid character for the ASCII-TSAP, the "TSAP (ASCII)" entry field no longer displays an ASCII-TSAP.

If a valid ASCII character is used, the check for compliance with the length specification of 2 to 16 characters follows.

Example 4: Incorrect editing of the TSAP-ID

If you remove the hex value "E0" from a TSAP-ID beginning with "E0.01", the TSAP-ID now begins with "01" and therefore no longer complies with the rules and is invalid:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

After the entry field is exited, a message is output because the TSAP-ID is neither a valid ASCII-TSAP (this would have to have a hex value in the range from 20 to 7E as the first value) or a valid TSAP-ID (this would have to have the identifier "E0" as the first value).

Example 5: Entry of an ASCII-TSAP via the "TSAP-ID" entry field

If you remove the value "01" in addition to the value "E0" from the incorrect TSAP-ID in example 4, the TSAP-ID begins with the hex value 49. This value is within the permissible range for ASCII-TSAPs:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

When you exit the entry field, the TSAP-ID is recognized as a valid ASCII-TSAP and the resulting ASCII-TSAP "ISOonTCP-1" is written to the "TSAP (ASCII)" entry field.

See also

TSAP structure (Page 647)

Description of the connection parameters (Page 630)

Communication via PUT and GET instructions

Basic information on communication via the PUT/GET instruction

Basic information on PUT/GET instructions

Use PUT and GET instructions to exchange data between two CPUs via an S7 connection.

The GET instruction is used to read data from a partner CPU. The PUT instruction is used to control the writing of tags by the communication partner via the user program. Apart from the PUT and GET instructions, no additional communication functions are provided for reading and writing tags.

To simplify the use of the two instructions, specify all required parameters for the connection and all block parameters in the Inspector window of the program editor.

Requirement

To be able to use the PUT and GET instructions, the following requirements must be satisfied:

- At least one S7-1200/1500 CPU or S7-300/400 CPU must be created in the project. Firmware 2.0 or higher must be installed on an S7-1200 CPU. If you have not yet created a second CPU in the project, you can initially establish the connection to an unspecified partner.
- An S7 connection must exist between the two CPUs. If you have not yet established a connection between two CPUs, a connection is automatically established during the configuration of the instructions.
- For both instructions, an instance data block is required in which all data used by the instruction is stored. The instance data block is created automatically as soon as you drag a PUT or GET instruction to a network in the program editor. For the correct execution of the program, it is essential that the instance data blocks are not changed; consequently, these data blocks are know-how protected. You only have read access to the instance data blocks.

See also

Overview of connection configuration (Page 653)

Assigning parameters to start request (Page 658)

PUT: Set parameters for write and send area (Page 659)

GET: Set parameters for read and memory area (Page 660)

Connection configuration

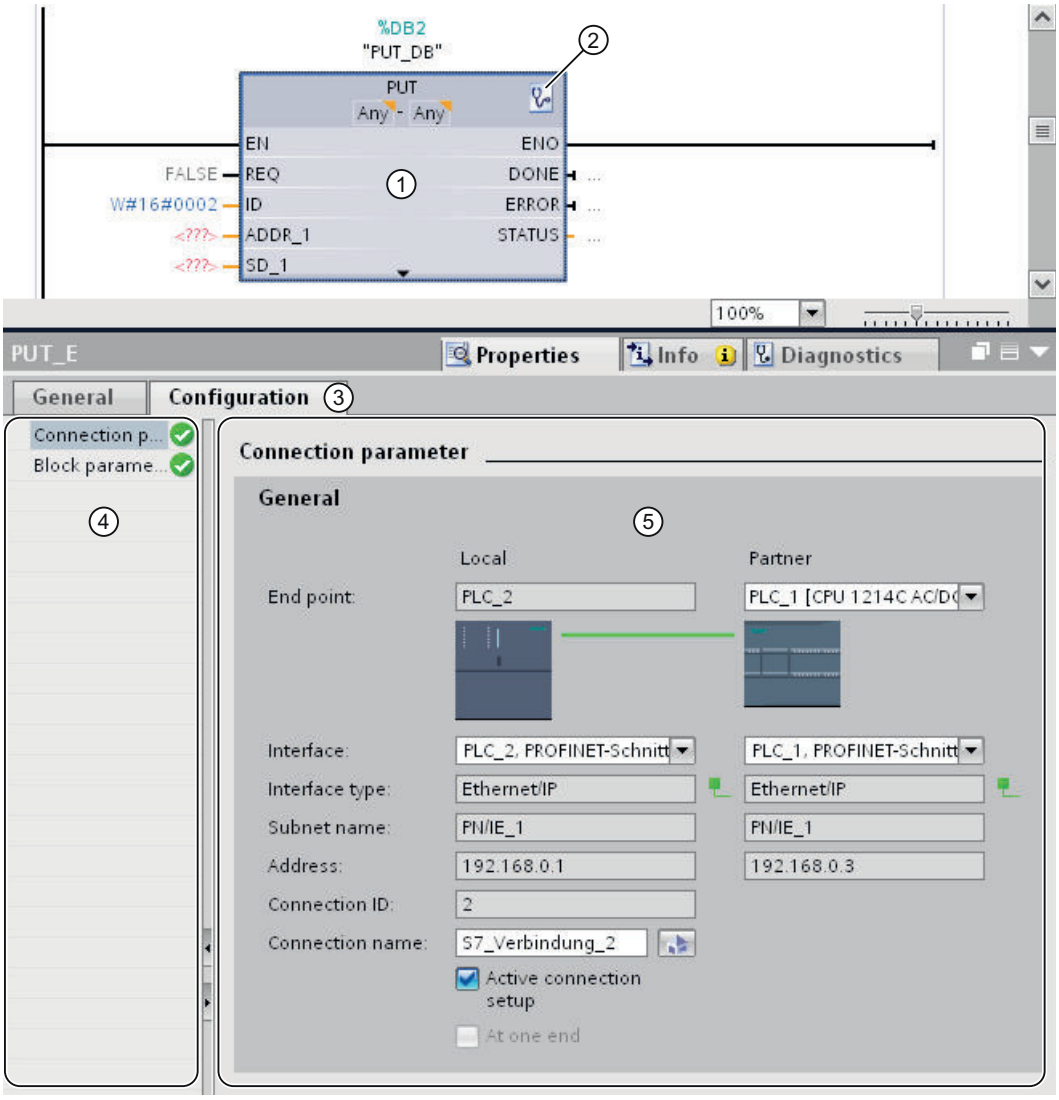
Overview of connection configuration

Introduction

The connection parameters for the PUT and GET instructions are assigned in the inspector window of the program editor. All parameters are saved in the corresponding instance data block.

Structure of the connection configuration

The connection configuration is made up of the following components:



① Communication instruction for PUT or GET
② Call of online and diagnostic functions
③ "Configuration" tab in the "Properties" tab
④ Area navigation of the "Configuration" tab
⑤ General properties of the connection parameters

Display of online and diagnostic functions

If you click the icon for starting the online and diagnostic functions, the associated CPU goes online automatically. The connection table in the network view is opened. In addition, the "Diagnostics" tab and the connection information are displayed in the inspector window.

Entering the connection parameters

Enter the desired connection parameters in the "Configuration" tab. The area navigation of the "Configuration" tab includes the "Connection parameters" group. This group contains the connection configuration. Here, you can enter the parameters for the connections using system functions. When all the required parameters are assigned, a check mark is set behind the "Connection parameters" group in the area navigation.

See also

Assigning parameters to start request (Page 658)

PUT: Set parameters for write and send area (Page 659)

GET: Set parameters for read and memory area (Page 660)

Description of the connection parameters

Overview

The following table shows the general connection parameters:

Parameter	Description
End point	The names of the local end point and the partner end point are shown. • Local end point The local end point is the CPU in which the PUT or GET instruction is programmed. • Partner end point The partner end point is selected from the drop-down list. The drop-down list shows all available possible connection partners including unspecified connection partners for devices whose data is unknown in the project. As long as no connection partner is set, all other parameters in the mask are disabled.
Interface	The interface of the partner CPU is displayed. The partner interface is not displayed until a specified partner CPU has been selected.
Interface type	The type of interface via which communication is handled is displayed.
Subnet name	The subnet of the local end point is displayed, provided this exists. The partner subnet is displayed only after the partner end point has been selected. If at least one of the two connection partners is not connected with a subnet, the two connection partners are automatically connected with each other. The partner which is not connected to a network is hereby connected to the same subnet via which the other partner is already connected to a network. A connection of connection partners to different subnets is only possible with IP or S7 routing. The IP routing settings can be edited in the properties of the relevant Ethernet interfaces.
Address	The IP address of the local end point is displayed. The IP address of the partner is displayed only after the partner end point has been selected. If you have selected an unspecified connection partner, the input box is empty and has a red background. In this case, you will need to specify a valid IP address for the connection partner.
Connection ID	The currently set connection ID is displayed. You can change the connection ID in the connection table in the network view. You can also directly access the connection table while you are setting the connection parameters. To do this, click the "Create new connection" icon.

Parameter	Description
Connection name	The name of the connection which was automatically created when the PUT/GET instruction was inserted is displayed. You can change the name of the connection by entered a different name in the field. You can also create a new connection or edit existing connections by clicking the "Create new connection" icon.
Active connection establishment	Use the "Active connection establishment" option button to specify which partner starts the communication. When the connection is created, the local partner is initially specified by default for the establishment of the connection. If a device does not support active connection establishment, you have to activate active connection establishment on the other partner.
Configured at one end	If this check box is selected, the connection partner is the server for this connection. It cannot actively send or receive. This corresponds to the behavior of the PUT/GET instructions. In this case, other instructions are not possible. If the check box is not selected, other instructions can also be used for the communication.

Starting connection parameter assignment

You can assign the connection parameters for PUT and GET in the inspector window as soon as you have inserted and selected a PUT or GET instruction in a program block.

Procedure

To insert PUT/GET instructions, follow these steps:

1. Open the "Instructions" task card in the "Communication > S7 Communication" folder.
2. Drag a PUT or GET instruction to a network.
   The "Call options" dialog opens.
3. Optional: Edit the properties of the instance DB in the "Call properties" dialog. You have the following options:
   - Change the default name.
   - Select the "Manual" check box to assign your own number.
4. Click "OK".

Result

A corresponding instance data block is created for the inserted PUT or GET instruction. For S7-300 CPUs, a function block is also created in the program resources.

When PUT or GET instruction is selected, you will see the "Configuration" tab under "Properties" in the inspector window. The "Connection parameters" group in area navigation contains the connection parameter assignment that you can now make.

See also

Creating and assigning parameters to connections (Page 657)

Deleting connections (Page 658)

Creating and assigning parameters to connections

You can create S7 connections and assign the parameters for these in the connection parameter assignment of the PUT/GET instructions. Changed values are checked immediately by the connection parameter assignment for input errors.

Requirement

A CPU exists with a PUT or GET communication instruction.

Procedure

To configure an S7 connection using PUT/GET instructions, follow these steps:

1. In the program editor, select the call of the PUT or GET instruction.
2. Open the "Properties > Configuration" tab in the inspector window.
3. Select the "Connection parameters" group. Until you select a connection partner, only the empty drop-down list for the partner end point is enabled. All other input options are disabled.
   The connection parameters already known are displayed:
   - Name of the local end point
   - Interface of the local end point
   - IP address of the local end point
4. In the drop-down list box of the partner end point, select a connection partner. You can select an unspecified device or a CPU in the project as the communications partner.
   The following parameters are automatically entered as soon as you have selected the connection partner:
   - Interface of the partner end point
   - Interface of the partner end point. If several interfaces are available, you can change the interface as required.
   - Interface type of the partner end point
   - Subnet name of both end points
   - IP address of the partner end point
   - Name of the connection which is used for the communication. If no connection exists yet, it is automatically established.
5. If required, change the connection name in the "Connection name" input box. If you want to create a new connection or edit an existing connection, click on the "Create new connection" icon.

Note

The PUT and GET instructions between two communication partners can only run if both the hardware configuration and the program part for the partner end point have been loaded into the hardware. To achieve fully functional communication, make sure that you load not only the connection description of the local CPU on the device but also that of the partner CPU as well.

Deleting connections

A connection which was automatically created during the insertion of a PUT or GET instruction appears in the connection table of the network view like every standard connection. As a result, it can be deleted in the connection table.

Procedure

To delete a connection, follow these steps:

1. Open the connection table in the network view.
2. In the connection table, select the connection that you want to delete.
3. To do this, right-click the connection and select the "Delete" command from the shortcut menu.

Result

The connection is deleted. The PUT or GET instruction and the associated instance data blocks are retained and must be manually deleted if necessary.

To continue using the PUT or GET instruction, you must configure the connection again in the inspector window of the program editor, since all connection parameters were also deleted when the connection was deleted. In this case, specify a new communication partner and a suitable connection.

Block parameter assignment

Assigning parameters to start request

To start communication via the PUT or GET instruction, you have to specify an event which activates the instruction. This event is referred to as control parameter (REQ). The communication job is activated as soon as there is a positive edge at the control parameter REQ.

Please note that the control parameter REQ is assigned the default FALSE at first call.

Requirement

- The program editor is open.
- You have already inserted a PUT or GET instruction.
- A connection has been established between two communication partners.

Procedure

To define the REQ control parameter, follow these steps:

1. Select the PUT or GET instruction in the program editor.
2. Open the "Configuration" tab in the inspector window.

3. Select the "Block parameter assignment" entry in the area navigation.
4. In the "REQ" field, select a tag of the "BOOL" data type to initialize the execution of the instruction. Alternatively, you can also interconnect a previous instruction in the program editor.

See also

Data consistency (Page 3608)

PUT: Set parameters for write and send area (Page 659)

GET: Set parameters for read and memory area (Page 660)

PUT: Set parameters for write and send area

For communication via the PUT instruction, you must specify the memory area of the partner CPU to which the data should be written. In addition, you must specify the memory area of the local CPU from which the data is to be read.

Requirement

- The program editor is open.
- You have already inserted a PUT instruction.
- A connection has been established between two communication partners.

Procedure

To specify the read and the memory area for the instruction, follow these steps:

1. Select the PUT instruction in the program editor.
2. Open the "Configuration" tab in the inspector window.
3. Select the "Block parameter assignment" entry in the area navigation.
4. In the "In/Outputs > Write area (ADDR_1) > Start" field, select a "REMOTE" data type pointer to the area of the partner CPU which is to be written.
   Only absolute addressing is permitted.
   Example: P#DB10.DBX5.0 Byte 10
5. In the "Length" field, enter the length of the write area and select the data type of the memory area from the drop-down list.
6. In the "In/Outputs > Send area (SD_1) > Start" field, select a pointer to the area in the local CPU which contains the data to be sent.
7. In the Length field, enter the length of the memory area to be read and select the data type from the drop-down list.
   Only the data types BOOL (for a bit array, "0" must be used as address and an integer multiple of byte must be used as length), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER are permitted.
   If the VARIANT pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 Byte 10).

See also

GET: Set parameters for read and memory area (Page 660)

GET: Set parameters for read and memory area

For communication via the GET instruction, you must specify the memory area of the local CPU to which the data should be written. In addition, you must specify the memory area of the partner CPU from which the data is to be read.

Requirement

- The program editor is open.
- You have already inserted a GET instruction.
- A connection has been established between two communication partners.

Procedure

To specify the read and the memory area for the instruction, follow these steps:

1. Select the GET instruction in the program editor.
2. Open the "Configuration" tab in the inspector window.
3. Select the "Block parameter assignment" entry in the area navigation.
4. In the "In/Outputs > Read area (ADDR_1) > Start" field, select a "REMOTE" data type pointer to the area of the partner CPU which is to be read.
   Only absolute addressing is permitted.
   Example: P#DB10.DBX5.0 Byte 10
5. In the "Length" field, enter the length of the read area and select the data type of the memory area from the drop-down list.
6. In the "In/Outputs > Memory area (RD_1) > Start" field, select a pointer to the area in the local CPU in which the read data is to be stored.
7. In the Length field, enter the length of the memory area and select the data type from the drop-down list.
   Only the data types BOOL (for a bit array, "0" must be used as address and an integer multiple of byte must be used as length), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER are permitted.

See also

PUT: Set parameters for write and send area (Page 659)

10.1.3.3 Displaying and configuring topology

Overview of the topology view

Functions of the topology view

The topology view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Displaying the Ethernet topology
  - Displaying all the PROFINET devices and passive Ethernet components of the project along with their ports
  - Displaying interconnections between the ports
  - Displaying corresponding logical networks
  - Display diagnostic information of all ports
- Configuring the Ethernet topology
  - Creating, modifying and deleting interconnections of the ports
  - Renaming stations, devices, interfaces or ports
  - Adding PROFINET devices and passive Ethernet components to the project from the hardware catalog
- Identifying and minimizing differences between the desired and actual topology
  - Running an offline/online comparison of Ethernet modules, ports and port interconnections
  - Adopting existing online topology information in the offline project

Note

Devices without valid IP address

Topology information (LLDP) cannot be read from a device without valid IP address.

To prevent devices from having an invalid IP address, specify in the settings of the TIA Portal for hardware configuration that devices without valid IP address are temporarily assigned an IP address.

Note

Subnet membership

Before the topology determination, you have to set for the local interface of your PG/PC the network address that the Ethernet components that are to be determined also have in the real plant. If you do not meet this requirement, no further topology information (information about ports and neighborhood relationships) can be determined.

Differences between network view and topology view

- The network view shows all the logical subnets of the project.
  The topology view shows all Ethernet components of the project. These include passive components such as switches, media converters and cables.

  Note

  Stations with non-Ethernet components are also displayed if the station has a least one Ethernet component.

- The position of a device in the network view and its position in the topology view are not dependent on each other; in other words, the same device generally appears at different locations in the two views.
- If you open the hardware catalog in the topology view, you only see devices with an Ethernet interface.

Structure of the topology view

The topology view (Page 542) essentially consists of a graphic area (called the graphic view below) and a table area (called the table view below).

Which functions are there in the graphic view and which functions are there in the table view?

- Displaying the Ethernet topology

Function	Graphic view	Table view
Displaying all the PROFINET devices and passive Ethernet components of the project along with their ports	yes	yes
Display interconnections between the ports (including type of medium)	yes	yes
Displaying corresponding logical networks	no	yes
Displaying properties of the cables between the ports	no	yes
Display diagnostic information of all ports	yes	yes

- Configuring the Ethernet topology

Function	Graphic view	Table view
Creating, modifying and deleting interconnections of the ports	• Create: yes • Modify: yes • Delete: yes	• Create: yes • Modify: yes • Delete: yes
Renaming stations, devices, interfaces or ports	no	yes
Adding PROFINET devices and passive Ethernet components to the project from the hardware catalog	yes	no

- Identifying and minimizing differences between the desired and actual topology

Function	Graphic view	Table view
Running an offline/online comparison of Ethernet modules, ports and port interconnections	no	yes
Adopting existing online topology information in the offline project	no	yes

Starting the topology view

Requirement

The device or network view is open in the hardware and network editor.

Procedure

To start the topology view of your project, follow these steps:

1. Click on the "Topology view" tab.

Or:

1. Open the network view of the hardware editor.
2. Select a PROFINET device or a PROFINET module.
3. Select the "Go to topology view" command in the shortcut menu.

Result

The graphic view of the topology view is started. If you opened the topology view using the shortcut menu, the selected component remains selected after the change of view.

Displaying topology

Displaying the graphic view of the configured topology

What is shown?

The graphic view of the configured topology shows the following:

- Configured PROFINET devices and passive Ethernet components along with their ports
- Configured stations with non-Ethernet components if there is at least one Ethernet component in the station
- Configured interconnections between the ports

Type of display

The way in which the graphic view of the topology view and the network view are displayed is very similar:

- Compared with the device view, components are shown in a simplified form.
- The interconnections between ports are shown as horizontal and vertical lines. These are dashed when an interconnection between a tool changer port and its possible partner ports is involved.

Displaying the table view of the configured topology

What is shown?

The table view of the configured topology shows exactly the same content as the graphic view except for the logical PROFINET subnets:

- All the configured PROFINET devices and passive Ethernet components along with their ports
- All the configured stations with non-Ethernet components if there is at least one Ethernet component in the station
- Configured interconnections between the ports
  For each port with the "Alternating partner port" property, there are as many completed rows as there are potential partner ports plus one empty row.

Type of display

As the name implies, the table view of the topology view consists of a table, the topology overview table. It is structured like the network overview table. It consists of the following columns:

- Device / port
  This is the most important column of the table. The entries in this column have a hierarchical structure with the PROFINET ports being the last element in the hierarchy. You can expand and collapse the hierarchical entries. For a CPU, for example, an entry consists of the following elements:
  - Station name
  - Device name
  - Name of the PROFINET interface
  - Names of the ports

  Note: All the other columns only have entries in the rows containing the port names.
- Type (as default, this column is not displayed)
  Shows what type of station, device or interface the table row relates to or whether it belongs to a port.
- Article no. (by default, this column is not displayed)
  Article no. of the device

- Subnet (as default, this column is not displayed)
  Configured subnet to which the interface belongs
- Master/IO system (as default, this column is not displayed)
  Shows whether or not the interface belongs to a PROFIBUS DP master system or a PROFINET IO system.
- Device address (as default, this column is not displayed)
  Configured address of the interface in the subnet
- Partner station
  Name of the station that contains the partner port
- Partner device
  Name of the device that contains the partner port
- Partner interface
  Interface to which the partner port belongs
- Partner port
- Cable data
  Contains the cable length and the signal delay of the cable connecting the ports

Basic functions for tables

The topology overview table supports the following basic functions for editing a table:

- Displaying and hiding table columns
  Note: The columns that define the configuration cannot be hidden.
- Optimizing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Displaying the diagnostics status of ports and cables in the graphic view

Requirements

The graphic view of the topology view is open.

Procedure

To determine the diagnostics status of the port, follow these steps:

1. Go online with the required component or components.

Result

The following icons are displayed:

- The corresponding diagnostics icon is displayed for each device.
- If there is an error in at least one lower-level component, the diagnostics icon "Error in lower-level component" is also displayed in the left-hand lower corner of the diagnostics icon.
- The corresponding diagnostics icon is displayed for each port.
- Every cable between two ports that are online has the color that matches its diagnostics status.

You will find the possible diagnostics icons for ports and the color coding of Ethernet cables in the description of hardware diagnostics. See: Displaying diagnostics status and comparison status using icons (Page 1356)

Showing the diagnostics status of hardware components in the table view

Requirement

The table view of the topology view is open.

Procedure

To obtain the diagnostics status of hardware components of the topology overview table, follow these steps:

1. Go online with the required components.

Result

The following icons are displayed at the left-hand edge of the topology overview table in each row that belongs to the component involved:

- The diagnostics icon belonging to the hardware component is displayed.
- If the hardware component has lower-level components and if there is an error in at least one of the lower-level components, the diagnostics icon "Error in lower-level component" is also displayed in the left-hand lower corner of the diagnostics icon of the hardware component.

For the possible diagnostics icons for hardware components, refer to the description of hardware diagnostics. See: Displaying diagnostics status and comparison status using icons (Page 1356)

Note

The display of the diagnostics status of hardware components in the topology overview table and the network overview table is identical.

Running an offline/online comparison and displaying the results

Requirement

The topology view is open. There may be an online connection to one or more devices, but this is not actually necessary.

Procedure

To find the differences between the configured and the actual topology, follow these steps:

1. Click the "Offline/online comparison" button in the toolbar of the topology overview.

Result

The "Partner station", "Partner interface" and "Cable data" columns in the topology overview table are removed.

Two additional groups of columns are added to the right-hand side of the table and these are initially empty:

- On the far right, columns for the topology to be identified online are added.
- Between the columns for the offline and the online topology, the "Status", "Action" and "Description" columns are added to show the result of the offline/online comparison.

Note

As default, the "Description" column is not displayed.

The following buttons are enabled in the toolbar of the table:

Button	Name	Meaning
	Update	The detection of the existing online topology is started again.
	Synchronize	• Adopting the port interconnections identified online in the project (Page 676) • Adopt the devices identified online in the project (Page 677)

After the actual topology has been identified, the added columns are filled. These steps are described in more detail in the following section.

Note

A difference between offline and online view is displayed for that port connected with the PG/PC which is only available online. This is because the PG/PC cannot be configured offline.

Columns for the topology identified online

The following columns are displayed:

- "Device / port"
- "Type" (as default, this column is not displayed)
- "Article no." (by default, this column is not displayed)
- "IP address" (as default, this column is not displayed)
- "Partner device"
- "Partner port"
- "Cable data"

Columns for the result of the offline/online comparison

The following columns are displayed:

- "Status"
  The result of the offline/online comparison is shown here in the form of diagnostics icons. The following icons are possible:

Diagnostics icon	Meaning
	Differing topology information in at least one lower-level component
	Identical topology information
	Topology information only available offline or device is disabled
	Topology information only exists online
	Differing topology information
	If a device does not support topology functions, the "Status" column remains empty.

- "Action"
  The possible actions are shown here in the form of icons. The following icons are possible:

Icon	Meaning
	No action possible
	Adopt the interconnection found online

- "Description"
  This column describes the selected action in words.

Configuring topology

Interconnecting ports

Overview

Interconnecting ports in the topology view

In the topology view, you have the following options for interconnecting ports:

- in the graphic view (Page 670)
- in the graphic view of a tool changer (Page 672)
- in the table view (Page 671)
- in the table view of a tool changer (Page 672)
- by adopting port interconnections identified online (Page 676)

Note

Interconnecting an electric with an optical port

If you want to interconnect an electric and an optical port, you have to decide between RT and IRT communication:

- With RT communication, it is not necessary to configure a media converter.
- With IRT communication, you have to make the interconnection via a media converter.

What effects does the interconnection of ports have on the network view?

Note

In the properties of a subnet in the network view, you can specify that this subnet is used when a port interconnection is created between two devices that are not networked.

When you create an interconnection between two ports, the following effects are possible in the network view:

- If the corresponding interfaces are disconnected: If you have specified a default subnet, this is used. Otherwise a new subnet is created to connect the two interfaces.
- If one (and only one) of the two interfaces involved is networked: The non-networked interface is connected to the same subnet as the already networked interface.
- In all other cases: The corresponding interfaces are not connected to a logical subnet.

See also

Interconnecting ports (Page 1129)

Settings for interconnecting Ethernet devices (Page 598)

Interconnecting ports in the graphic view

Requirement

You are in the graphic view of the topology view.

Procedure – Creating a new interconnection between two ports

To interconnect a port of a device with a port of another device, follow these steps:

1. Place the mouse cursor on the port you want to interconnect.
2. Click with the left mouse button and hold it down.
3. Move the mouse pointer.
   The pointer now shows the networking symbol to indicate "Interconnect" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear when the pointer is on a valid target.
4. Now drag the mouse cursor to the target port. You can do this while holding down or after releasing the mouse button.
5. Now release the left mouse button or press it again (depending on your previous action).

Result: A new port interconnection is created.

Note

Creating a ring for S7-300, S7-400, and S7-1500 CPUs

If you create a ring using port interconnections for S7-300, S7-400, or S7-1500 CPUs, an MRP domain is created automatically.

Procedure – Changing an existing port interconnection without deleting it first

To do this, follow these steps:

1. Place the mouse cursor on the port of an existing interconnection that is to receive a new partner port.
2. Drag the port to the new partner port.

Result: The existing port interconnection is deleted. The new port interconnection is created.

Alternative procedure:

1. Place the mouse cursor on a port without an interconnection that is to be connected with an already interconnected port.
2. Drag the port to the already interconnected port.

Result: The existing port interconnection is deleted. The new port interconnection is created.

Procedure – Interconnecting two interconnected ports with each other without first deleting the two existing port interconnections

To do this, follow these steps:

1. Place the mouse cursor on the interconnected port that is to receive a new partner port.
2. Drag the port to the new partner port that has also already been interconnected.

Result: The two existing port interconnections are deleted. The new port interconnection is created.

Interconnecting ports in the table view

Which actions are possible with port interconnections in the table view?

The following actions are possible with port interconnections in the table view:

- Creating a new port interconnection
- Changing an existing port interconnection
- Deleting an existing port interconnection

Requirement

The row with the port whose interconnection you want to create, modify or delete is visible in the topology overview.

Procedure

To create the interconnection of a port for the first time, to modify it or delete it, follow these steps:

1. Move the mouse pointer to the "Partner port" column in the row of the source port.
2. Click the drop-down list there.
3. Select the required partner port (when creating or changing a port interconnection) or the "Not interconnected" entry (when deleting a port interconnection).

Result

The required action is performed. The new partner port (after creating or modifying a port interconnection) or the "Select port" entry (after deleting a port interconnection) is displayed in the "Partner port" column.

Interconnecting a port with more than one partner port in the graphic view

Requirement

- You have configured a port of a PROFINET device with the "Alternative partners" property and have specified its possible partner ports.
- The graphic view of the topology view is open.

Procedure

1. Interconnect this port (referred to hereafter as source port) with one of the partner ports you have specified (referred to hereafter as target port).
2. Interconnect the source port with an additional target port.
   You can do this in several ways:
   - Drag the mouse pointer from a partner port that is already interconnected to a target port.
   - Drag the mouse pointer from an interconnection that has already been created to a target port.
   - Drag the mouse pointer from a target port to a partner port that is already interconnected.
   - Drag the mouse pointer from a target port to an already created interconnection.
3. If necessary, repeat the step above one or more times.

Result

An interconnection is created between the source port and the alternative partner ports. This is indicated by a dashed line.

Interconnecting a port with more than one partner port in the table view

Which actions are possible with port interconnections to several partner ports in the table view?

When working with a tool changer, the following actions can be performed with port interconnections to multiple partner ports in the table view:

- Creating a new port interconnection
- Changing an existing port interconnection
- Deleting an existing port interconnection

Requirement

- You have configured a port of a PROFINET device with the "Alternative partners" property and have specified its possible partner ports.
- The row with the port whose interconnection you want to create, modify or delete is visible in the topology overview.

Procedure

To create the interconnection of a port to one or more partner ports for the first time, to modify it, or to delete it, follow these steps:

1. Move the mouse pointer to the "Partner port" column in the row of the source port.
2. Click the drop-down list there.
3. Select the required partner port (when creating or changing a port interconnection) or the "Not interconnected" entry (when deleting a port interconnection).

Result

The required action is performed:

- If you are creating an interconnection, a new row is inserted in the topology overview table. The new partner port is displayed there in the "Partner port" column.
- If you change an interconnection, the new partner port is displayed in the "Partner port" column.
- If you delete an interconnection, the row with the previous port interconnection is deleted.

Note

With a tool changer, there are normally several rows for a port with port interconnections to more than one partner port. The last row is always an empty row. The first row can be edited, all other rows are read-only.

Renaming stations, devices, interfaces or ports

Rename a station, a device, an interface or a port

Requirement

The table view of the configured topology is open.

Procedure

To rename a station, a device, an interface or a port, proceed as follows:

1. Click twice in the relevant field of the topology overview table (the second click starts the editing mode).
2. Enter the new name and then press the ENTER key (this closes editing mode).

Result

The object is renamed.

Offline/online comparison

Automatic assignment of devices by offline/online comparison

Overview

During the offline/online comparison, the configured topology is compared with the actual existing topology. Devices identified online are automatically assigned to configured devices as far as this is possible.

Start of availability detection

You start the availability detection the first time by clicking the "Compare offline/online" button in the toolbar of the topology overview.

You restart availability detection by clicking the "Update" button.

Note

The availability detection can take several seconds. During this time, no user input is possible.

Automatic assignment

A device identified online is automatically assigned to a configured device if the following properties of the two devices match up:

- Device name
- Article number
- Number of ports

The following section describes the situations that can occur and what action you can take:

- Identical port interconnections
  This is the ideal situation. No action is necessary here.

"Action" column	Meaning
	No action

- There are interconnections for the identified and configured device, there are however differences.
  The following actions are possible:
  - If it is possible to adopt the online configuration

"Action" column	Meaning
	Adopt online interconnection (Page 676)
	No action

  - If it is not possible to adopt the configuration

"Action" column	Meaning
	No action

- The interconnection only exists online.
  The following actions are possible:
  - If it is possible to adopt the online configuration

"Action" column	Meaning
	Adopt online interconnection (Page 676)
	No action

  - If it is not possible to adopt the configuration

"Action" column	Meaning
	No action

- The interconnection only exists in the configuration.
  The following actions are possible:

"Action" column	Meaning
	Adopt the online interconnection (Page 676), in other words, the interconnection in the configuration will be deleted
	No action

No automatic assignment

In the following situations, no automatic assignment is possible:

- No device can be identified online to match a configured device. In this case the corresponding columns in the "Online topology" area of the topology overview table are empty.
  In this case, you should add the already configured device to your system or delete the configured device from the configuration.
- A device identified online cannot be assigned to any configured device. In this case the corresponding columns in the "Offline topology" area of the topology overview table are empty.
  In this case, you can adopt the device identified online in the project (Page 677).

Adopting the port interconnections identified online in the project

Requirement

You have run an offline/online comparison in the topology view. The result of this is that at least one device identified online was automatically assigned to a configured device, but that there are differences relating to the interconnection.

Procedure

To adopt one more port interconnections identified online in the project manually, follow these steps:

1. Select the value "Adopt" in the "Action" column for a port of a configured device to which a device identified online was assigned.
2. Repeat the step if necessary for other ports of the same configured device.
3. Repeat the steps up to now if necessary for other configured devices to which devices identified online were assigned and for which there are differences relating to the interconnection.
4. Click the "Synchronize" button.

Result

The port interconnections identified online and the cable information for the corresponding devices are adopted in the project. Successful adoption is indicated by the diagnostics icon "Identical topology information" for each port.

Note

If other port interconnections are recognized for a device identified online and these differ from those that exist in the project, adopting these in the project means that the port interconnections that were previously in the project are replaced by those identified online. If no port interconnections are detected for a device identified online, adopting in the project means that all the port interconnections of this device are deleted in the project.

Adopt the devices identified online in the project

Requirements

You have run an offline/online comparison in the topology view. The result of this is that at least one device identified online could not be assigned to any configured device.

Procedure

To adopt one more devices identified online in the project manually, follow these steps:

1. For a configured device without an online partner, move the mouse pointer to the "Device/ port" column of the online topology.
2. Select the device you want to assign to the configured device from the drop-down list of this box.
3. Repeat the previous steps if necessary for other configured devices without an online partner.

Result

The selected device that was identified online is moved up from the end of the table. Following this, it is in the row of the configured device to which you have just assigned it.

10.1.3.4 Industrial Ethernet Security

Configuring security

General

Supported devices

Supported devices

Security functions can be configured for the following products:

- SCALANCE S:
  - S602 V2/V3/V4
  - S612 V2/V3/V4
  - S613 V2
  - S623 V3/V4
  - S627-2M V4
- SOFTNET Security Client:
  - SOFTNET Security Client V4

- S7-CPs: CP 343-1 GX31 Advanced, CP 443-1 GX30 Advanced, CP 1543-1, CP 1243-1 BX30, CP 1242-7 KX31, CP 1243-7
- PC CP: CP 1628
- Mobile wireless router: SCALANCE M875

General terminology "security module"

In this section of the information system, the following products are grouped together under the term "security module": SCALANCE S602 / SCALANCE S612 / SCALANCE S613 / SCALANCE S623 / SCALANCE S627-2M, CP 343-1 GX31 Advanced, CP 443-1 GX30 Advanced, CP 1543-1, CP 1243-1 BX30, CP 1242-7 KX31, CP 1243-7, CP 1628.

Use of the terms "interface" and "port"

In this documentation, the ports of SCALANCE S modules are named as follows:

- "External interface": The external port of the SCALANCE S602 / S612 / S613 / S623 or an external port of the SCALANCE S627-2M (marked red)
- "Internal interface": The internal port of the SCALANCE S602 / S612 / S613 / S623 or an internal port of the SCALANCE S627-2M (marked green)
- "DMZ interface": The DMZ port of the SCALANCE S623 / S627-2M (marked yellow)

The term "port" itself is used when the focus of interest is a special port of an interface.

Naming the S7 CPs

In this documentation, the term "CP x43-1 Adv." includes the following products: CP 343-1 GX31 Advanced / CP 443-1 GX30 Advanced. The name "CP 1243-1" is used for the product CP 1243-1 BX30. The name "CP 1242-7" is used for the product CP 1242-7 KX31.

Structure of this section of the help system

Topics that are relevant to all security modules can be found in the "General" section. Information that is only relevant to certain module types can be found in the sections relating specifically to these modules.

Overview - Scope of performance and method of operation

General use of the term "STEP 7"

Configuration of security functions is supported as of STEP 7 V12. For this reason, in this section of the information system, the name "STEP 7" will be used for all versions of STEP 7 V12 or higher.

Scope of performance

You can use the following security functions in STEP 7:

- Configuration of the security modules
- Creating VPN configuration data for SOFTNET Security Client V4
- Creating VPN configuration data for SCALANCE M875
- Test and diagnostics functions, status displays

Offline configuration view and online diagnostics view

The security functions are configured in two views:

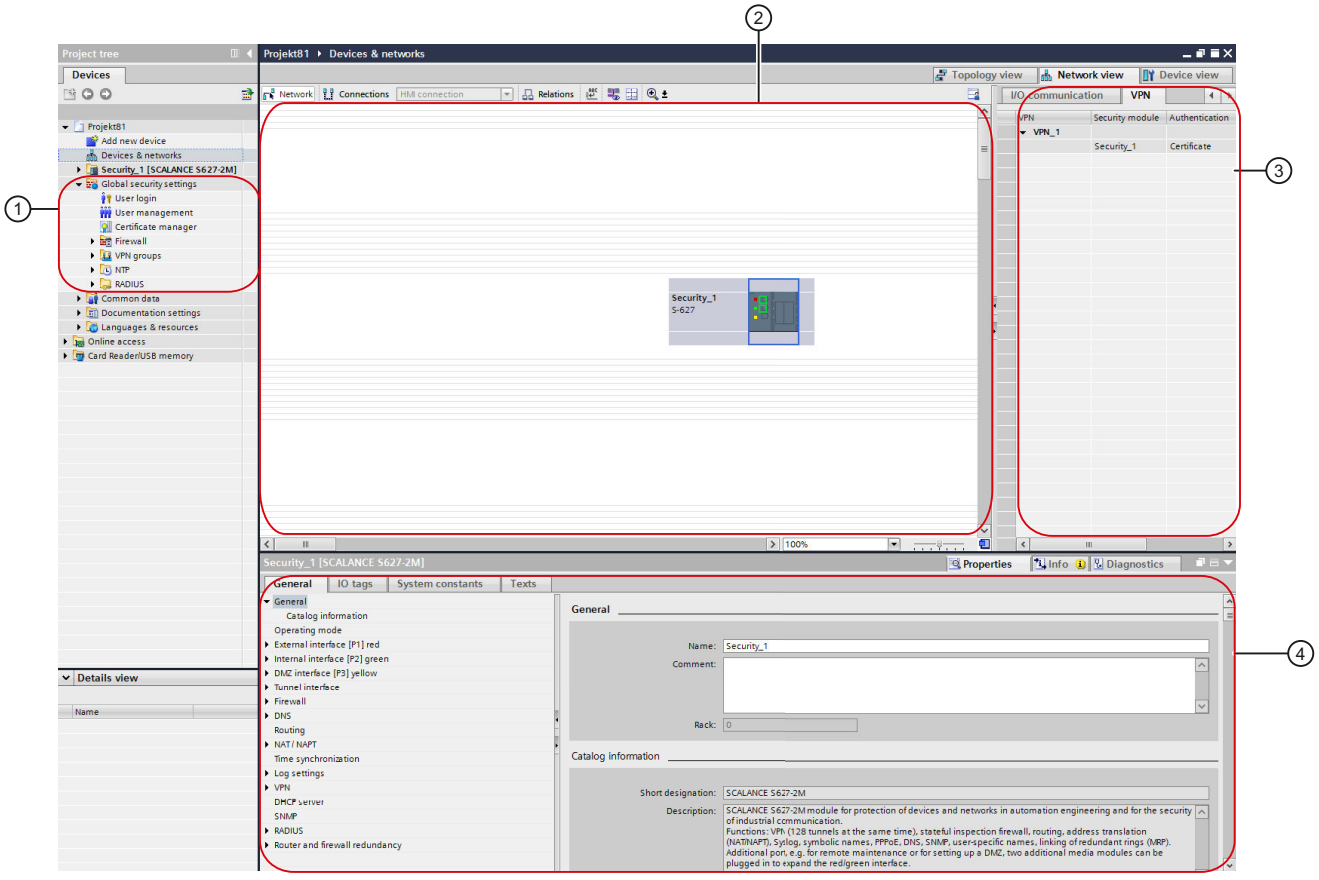
- Offline configuration view
  In the offline configuration view, you create the configuration data for the security modules and the SOFTNET Security Client. Prior to downloading, there must already be a connection to the security modules.
- Online diagnostics view
  The online diagnostics view is used for diagnostics of the security module and, among other things, allows you to run firmware updates.

How it works - security and consistency

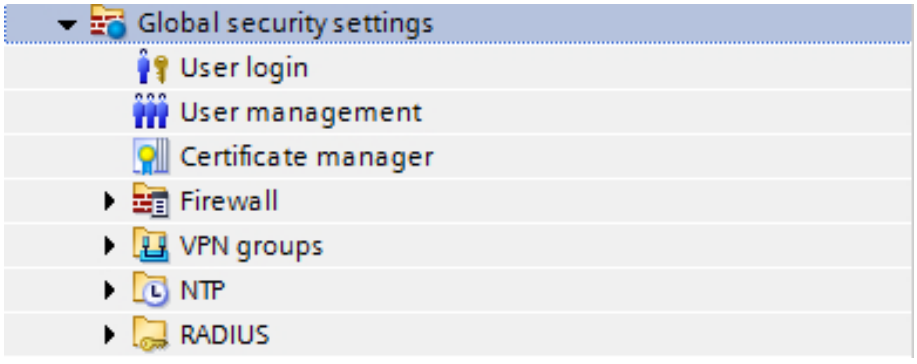
- Access only for authorized users
  The security functions of every project are protected from unauthorized access by assigning user names and passwords. With the help of password policies, project-specific rules for password assignment can be defined.
- Consistent project data
  Consistency checks are running even while you make the entries in the dialogs. In addition, cross-dialog, project-wide consistency checks are carried out.
  Only consistent project data can be downloaded to the security modules.
- Protecting project data by encryption
  The project and configuration data relevant to security are protected by encryption.
  Depending on the security module, data can be stored in the project and/or on the C-PLUG.

User interface - layout and menu commands

User interface for security functions in STEP 7



① Global security settings



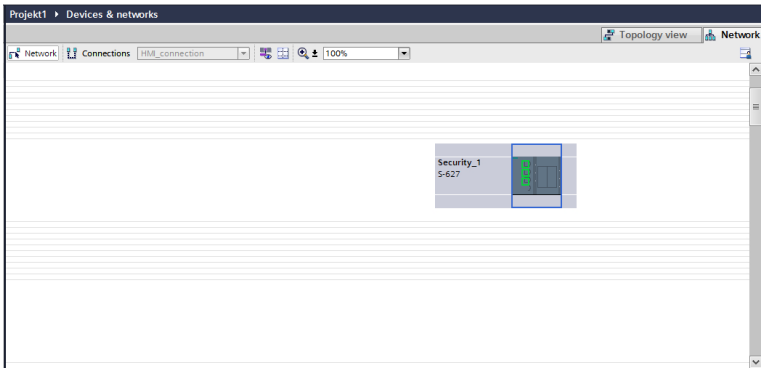
The global security settings are located in the project navigation. These security settings can be configured independently of the module and subsequently assigned to individual security modules as required. Changes to the global security settings must be loaded on all security modules involved. This also applies to the settings of redundancy relationships.

If the first security module to be configured is a CP, the global security settings are only displayed when the security functions have been enabled in the local security settings of the CP. If the first security module to be configured is

a SCALANCE S module, the global security settings are displayed after logging in to the security project. The following main folders and entries are available in the global security settings:

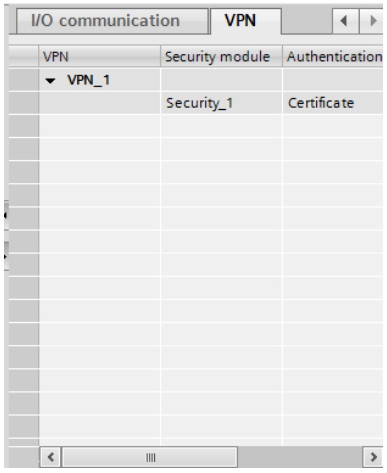
- User login
  For the security configuration within a project, there is a separate user management. Log in to the security configuration using the "User login" entry. The first time that there is a login to the security configuration, a user with the system-defined role "Administrator" is created automatically. You can create further users in the security configuration in the user management.
- User administration
  In user administration, you can create users, define rights for roles and assign these roles to users.
- Certificate manager
  In the certificate manager, you see an overview of all the certificates used in the project. You can, for example, import new certificates as well as export, renew or replace existing certificates.
- Firewall
  Under the "Firewall" entry, you can define global IP and MAC firewall rule sets and user-specific IP rule sets (SCALANCE S modules only) and assign security modules. IP and MAC service definitions are used to define the IP and MAC firewall rules compactly and clearly.
- VPN groups
  All created VPN groups are contained in this folder. You can create new VPN groups here and assign security modules to these VPN groups. You can also adapt VPN group properties of VPN groups that have already been created.
- NTP
  Here, you can create NTP servers and assign them to one or more security modules. This ensures that time synchronization is performed through the assigned NTP server. Unsecured NTP servers can only be configured in the local security settings.
- RADIUS
  Here, you can create RADIUS servers and assign them to one or more security modules. With this, you ensure that authentication queries from users who log on to the selected security module to activate user-specific IP rule sets are forwarded to the assigned RADIUS server.

② Working area with security module



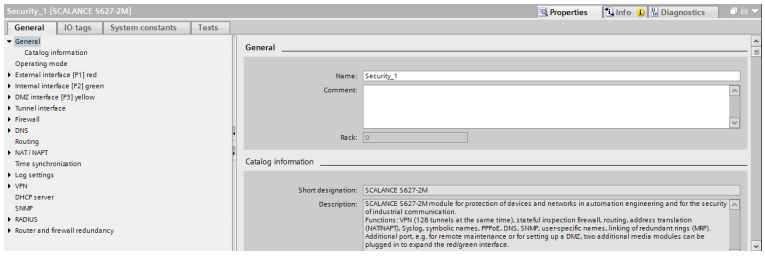
Once you have selected a security module in the work area, you can configure its local security settings in "Properties" > "General". If the selected security module is in a VPN group, related information is displayed in the VPN tab.

③ VPN tab



This tab displays information about all the VPN groups to which the security module that was selected in the working area belongs. Information about the respective participants of a VPN group can be displayed and hidden.

④ Local security settings



Local security settings are configured for a specific security module. After a security module has been selected in the working area, its local security settings are available in the Inspector window under "Properties" > "General".

Note for CPs:

Before local security settings can be configured for CPs, these must first be enabled.

To do this, log in to your security project and then in the Inspector window, select the "Activate security features" check box in the "Properties" > General" tab, "Security" entry. The local security settings are then displayed below the "Security" entry. When the check box is selected, the following settings (assuming they were enabled) are migrated automatically to the local security settings.

CP x43-1 Advanced:

- SNMP
- FTP configuration
- Time-of-day synchronization
- Web server
- Entries of IP access lists

CP 1543-1:

- SNMP
- FTP configuration
- Time-of-day synchronization

CP 1243-1:

- SNMP
- Time-of-day synchronization

CP 1242-7, CP 1243-7:

- Time-of-day synchronization

CP 1628:

- SNMP
- Time-of-day synchronization

Depending on the particular security module, additional security functions are also available such as NTP (secure), SNMPv3, FTPS.

In addition, firewall rules that enable a connection to be established are created automatically for configured connections. Log settings are available to record blocked packets.

Secure and non-secure configuration areas

The user interface can be divided into secure and non-secure configuration areas.

The secure areas are areas in which configuration is possible only after logging in to the security configuration. These areas are encrypted and therefore only available to persons authorized in the user management even if the project is accessible to a wider circle of people.

Functions from the non-secure areas, on the other hand, can be configured without logging in to the security configuration. The correctness of the settings must be checked before downloading the project to the plant components if a wider circle of people can make modifications to the project.

Below, you will find a list of the configuration areas of the user interface showing which areas are secure and which are non-secure. To some extent, this depends on the security module for which the configuration is created.

- All settings from the global security settings are secure.
- Secure and non-secure configuration areas for SCALANCE S modules:
  - All the settings for the interfaces and ports, in particular IP addresses, are non-secure.
  - The settings under the entry "General" in the local security settings are non-secure.
  - Higher-level settings (e.g. MRP settings such as MRP manager etc.) that are not configured on the security module itself but may affect the security module are non-secure. This does not relate to the global security settings.
  - The other settings are protected.
- Secure and non-secure configuration areas for CP 343-1 Advanced GX31, CP 443-1 Advanced GX30, CP 1628, CP 1543-1, CP 1243-1, CP 1242-7, CP 1243-7:
  - All settings outside the "Security" entry are non-secure.
  - Higher-level settings (e.g. MRP settings such as MRP manager, PROFINET settings, connections etc.) that are not configured on the security module itself but may affect the security module are non-secure. This does not relate to the global security settings.
  - All the settings for the interfaces and ports, in particular IP addresses, are non-secure.
  - All settings below the "Security" entry are secure.

Running a consistency check

Overview

The following consistency checks are available:

- Local consistency checks
- Project-wide consistency checks

In the individual dialog descriptions in this help system, the rules you need to take into account for your entries are listed under the keyword "Consistency check".

Local consistency checks

A consistency check is local when it can be performed directly within a dialog. With the following actions, local consistency checks are made:

- After exiting a box
- After exiting a row in a table
- When you confirm a dialog with OK.

Project-wide consistency checks

Project-wide consistency checks provide you with information indicating whether or not project data is correctly configured. With the following actions, there is a consistency check through the entire project:

- When compiling a configuration
- When downloading a configuration

Note

You can only download configured data to a security module if the project-wide consistency check for the security module was successful.

Replacing a security module

Module-specific function

Only SCALANCE S modules as of V3 can be replaced by SCALANCE S modules as of V3, refer to the section:
Replacing a security module (Page 771) in the section "SCALANCE S".

Managing certificates

Overview of certificates

How do you manage certificates?

In the certificate manager, you have an overview of all the certificates, for example, CA certificates used in the project with information about the applicant, issuer, validity, use and the existence of a private key.

The CA certificate is issued by a certificate authority from which the device certificates are derived. These include the SSL certificates and if the security module is a member of a VPN group the VPN group certificates. The SSL certificates are required for authentication during

secure communication between a network node and a security module. Certificate authorities can be:

- STEP 7 itself. If the "applicant" and "issuer" are the same, this is a self-signed certificate; in other words, issued by STEP 7.
- A higher ranking (commercial) certificate authority. These third-party certificates are external to the project and are imported and stored in the certificate store of STEP 7.

Certificates created by one of the two certificate authorities always have a private key so that the device certificates can be derived from them.

The following functions are also available in the certificate manager:

- Import of new certificates and certificate authorities.
- Import of SSL certificates (CP x43-1 Adv. only), e.g. for FTP communication.
- Export of the certificates and certificate authorities used in the project.
- Renewal of expired certificates and certificate authorities.
- Replacing existing certificate authorities.
- Addition of trusted certificates and certificate authorities.
- Deleting manually imported certificates.

Note

Downloading the configuration

After replacing or renewing certificates, the configuration must be downloaded to the relevant security modules.

After replacing or renewing CA certificates, the configuration must be downloaded to all security modules.

Note

Current date and current time of day on the security modules

When using secure communication (for example, HTTPS, VPN...), make sure that the security modules involved have the current time of day and the current date. Otherwise the certificates used are not evaluated as valid and the secure communication does not work.

How to access this function

Double-click on the "Certificate manager" entry in the global security settings.

In the individual tabs, you have the following commands available in the shortcut menu:

Command	Meaning
Import / Export	Import / export of device certificates or CA certificates. The certificates are transferred to the security module. The following formats are permitted: *.cer (certificate only) *.crt (certificate only) *.pem (certificate only) *.p12 (certificate and corresponding private key)
Displays	Opens the certificate dialog of Windows where you see an overview of all certificate data.
Renew (only in the "CA" and "Device certificates" tabs)	Opens the "Create new certificate" dialog in which you can import a certificate or have a new certificate created by STEP 7 when necessary, for example with compromised certificates.
Replace (only in the "CA" tab)	Opens the "Change certificate authority (CA)" dialog in which you can replace an existing certificate authority with a new one.
Delete	Deletes a certificate in the "Trusted certificates and root certification authorities" tab.

Certificate authorities

"CA" tab

The certificates displayed here are created by a certificate authority.

- CA certificates of a project: When you create a new project, a CA certificate is generated for the project. The SSL certificates for the individual security modules are derived from this certificate.
- CA group certificates: When you create a new VPN group, a CA certificate is generated for the VPN group. The VPN group certificates of security modules located in the VPN group are derived from this certificate.

Device certificates

"Device certificates" tab

Display of the device-specific certificates generated by STEP 7 for a security module. These include:

- SSL certificate of a security module: An SSL certificate that is derived from a CA certificate of the project is generated for each security module that you create. SSL certificates are used for authentication during secure communication between PG/PC and security module when downloading the configuration.
- VPN group certificate of a security module: A VPN group certificate is also generated for each security module for each VPN group in which it is located.

Trusted certificates and root certification authorities

"Trusted root certification authorities" tab

Display of the third-party certificates imported into STEP 7. For example, server certificates can be imported from external FTPS servers or project certificates from other projects that were created with STEP 7.

With CPs, the imported third-party certificate is transferred to all the CPs managed in the project and these check the certificate. If you classify the certificate sent to the security modules as trusted, a connection can be established, for example, to an FTPS server. The imported certificate is not used at any other point in STEP 7.

With SCALANCE S modules, the certificate authorities required for verifying the security modules using external services such as dyn. DNS are displayed in this tab.

Renewing certificates

Meaning

In this dialog, you renew CA certificates and device certificates. If necessary, for example with compromised certificates, you can import a certificate or have a new certificate generated by STEP 7.

How to access this function

1. Right-click on a list entry in the certificate manager.
2. Select the "Renew" entry in the shortcut menu.
3. Decide whether or not the new certificate will be self-signed or signed by a certificate authority.
4. If the certificate is to be signed by a certificate authority, select the certificate authority to be used with the "Select" button. Only certificate authorities stored in the certificate store of the current project can be selected.

5. Depending on the certificate, enter the following values in the "Applicant" or "Alternative applicant name" input box:

Certificate to be renewed	Parameter	
	Applicant	Alternative applicant name
CA certificates of the project	Name of the CA certificate	-
CA group certificate	Name of the CA group certificate	-
SSL certificate for S7 CP	Name of the security module	Comma-separated IP addresses of the Gigabit and PROFINET interface
SSL certificate for PC CP	Name of the security module	IP address of the security module
SSL certificate for SCALANCE S, SCALANCE M, SOFTNET Security Client	Name of the security module	For SCALANCE S: External IP address, internal IP address and, if applicable, IP address of the DMZ interface For SCALANCE M: External IP address, internal IP address For SOFTNET Security Client: DNS name
VPN group certificate of the security module	Name of the VPN group certificate	Derived from the CA group certificate.

6. Select a period during which the certificate is valid. As default the current time is entered in the "Valid from:" box and the value of the current certificate is entered in the "Valid to:".

Replacing certificates

Meaning

Open the "Change certification authority (CA)" to replace the existing CA certificate of the project or the CA group certificate with a new one.

How to access this function

1. Right-click a list entry in the "CA" tab.
2. Select the "Replace" entry in the shortcut menu.
3. The "Change certification authority (CA)" dialog opens.

All certificates listed in the "Certificates involved" table are derived once again. This means that the CA group certificate of an already configured VPN group can be replaced in the project by the CA group certificate of a different project. The VPN group certificates for the VPN group members are therefore derived from the same CA group certificate in both projects.

After making changes in the certificate manager, download the configuration to all security modules affected.

Which format can the certificate have?

Other certificates are derived from the imported certificate authority in STEP 7. For this reason, you can only select certificates with a private key.

- *.p12

Managing users and roles

Rules for user names, roles and passwords

Which rules apply to user names, role names and passwords?

When creating or modifying a user, a role or a password, remember the following rules:

Table 10-1 Rules for user management

Permitted characters	The following characters from the ANSI X 3.4-1986 character set are permitted: 0123456789 A...Z a...z !#$%&()*+,-./:;<=>?@ [\]_{|}~^
Characters not allowed	" ' `
Length of the user name (authentication method "password")	1 ... 32 characters
Length of the user name (authentication method "RADIUS")	1 ... 255 characters
Length of the password	8 ... 32 characters
Length of the role name	1 ... 32 characters
Maximum number of users per project	128
Maximum number of users on one security module	32 + 1 administrator when creating the project
Maximum number of roles per project	125
Maximum number of roles on one security module	37

Note

User names and passwords

As an important measure for increasing security, always make sure that user names and passwords are as long as possible and include special characters, upper and lowercase letters and numerals.

Using password policies, you can tighten the restrictions listed above for passwords even further. How to define password policies is described in the section:

Configuring password policies (Page 697)

Password strength

When a new password is entered, its password strength is checked. The following levels are distinguished for the password strength:

- Very weak
- Weak
- Medium
- Good
- Strong
- Very strong

Create users

Meaning

The security functions configured in STEP 7 are protected from unauthorized access by a separate user management. Before you can access the global and local security settings of security modules, you need to log in to the security configuration as a user.

Creating the first user in the project

After creating the first security module in the project, you first need to create a user. To do this, in the local security settings of the created security module, click the "User login" button under the "Security properties" entry and enter the login data of the user you want to create. You will then be logged in as the created user and the user is assigned the system-defined "Administrator" role. This role includes all configuration and module rights.

Creating users in the user management

When you are logged in as a user in the security configuration, you can create further users or delete users with the "User management" entry in the global security settings.

Note

Users with the "Administrator" role

There must always be at least one user with full configuration rights within a project. The administrator that is created automatically when you first enable the security functions in the project can only be deleted if at least one other user exists with the system-defined role "Administrator".

The following parameters are available in user management in the "User" tab:

Table 10-2 Information in the "User" tab

Parameter	Meaning
User name	Name of the user to be created. Click on the "Add new user" entry in the "User name" column to create a new user.
Password (only with the "Password" authentication method)	Entry of the password for the user. When it is entered, the password strength is checked. For more detailed information on password strength, refer to the following section: Rules for user names, roles and passwords (Page 690)
Authentication method	• **Password**: Use this authentication method for users that edit and download the STEP 7 Security project and that need to run diagnostics on the security module. The authentication of the user is performed by the security module when user-specific IP rule sets are activated. • **RADIUS**: (only for SCALANCE S as of V4): The authentication of the user is performed by a RADIUS server when user-specific IP rule sets are activated. The password of the user is not configured in STEP 7 when using this authentication method but must be stored on the RADIUS server. Only use this authentication method for users that only need to log on to the Web page of a security module. A user with the "RADIUS" authentication method cannot log on to STEP 7 Security projects.
Role	Selecting a system-defined or user-defined role.
Maximum time of the session (only for SCALANCE S V3 or higher)	Entry of the time after which a user logged on to the Web page for user-specific IP rule sets of SCALANCE S modules is automatically logged off. The time entered here starts after the logon and after renewing the session on the Web page of the security module. • Default setting: 30 minutes • Minimum value: 5 minutes • Maximum value: 480 minutes
Comment	Entry of optional comments.

Creating roles

Overview

You can assign a system-defined or a user-defined role to a user. Specify the module rights of a user-defined role for each security module.

System-defined roles

The system-defined roles listed below are predefined. Certain rights are assigned to the roles that are the same on all modules and that the administrator can neither change nor delete.

- Administrator
  Default role when creating a security configuration.
  Unlimited access rights to all configuration data and all security modules.
- Standard
  Role with restricted access rights.
- Diagnose
  - Read access to configurations.
  - Read access to the security module in the "Online" mode for testing and diagnostics.
- Remote-Access
  No rights except for logging on to the Web page for user-specific firewall rule sets.
- administrator (radius)
  Role that can be used to activate user-specific IP rule sets with authentication using a RADIUS server.
  Access rights to all configuration data except SNMP MIBs.
- radius
  Role that can be used to activate user-specific IP rule sets with authentication using a RADIUS server.
  Read-only access.

You will find a detailed list of the configuration and module rights assigned to the system-defined roles "Administrator", "Standard" and "Diagnostics" in tables 1-3 to 1-7 of the section Managing rights (Page 694).

For more detailed information on user-specific IP rule sets, refer to the following section: Auto-Hotspot

For more detailed information on authentication using a RADIUS server, refer to the following section: Auto-Hotspot

User-defined role

In addition to the system-defined roles, you can create user-defined roles. For a user-defined role, select the configuration or module rights and specify the appropriate module rights for every security module used in the project. You manually assign the user-defined roles to the relevant user.

How to access this function

1. Double-click on the "User management" entry in the global security settings.
2. Select the "Roles" tab in User management.

Table 10-3 Information in the "Roles" tab

Parameter	Meaning
Role	Freely selectable role name. Double-click on the "Add new role" entry to create a new user-defined role. You can then set the rights for the created role.
Description	Specifying the system-defined role With user-defined roles, the "User-defined role" character string is displayed.
Maximum time of the session (only for SCALANCE S V3 or higher)	Entry of the time after which a user with the assigned role is automatically logged off from the Web page for user-specific IP rule sets of SCALANCE S modules. The time entered here starts after the logon and after renewing the session on the Web page of the security module. • Default setting: 30 minutes • Minimum value: 5 minutes • Maximum value: 480 minutes
Comment	Entry of additional, optional comments.

Note

Deleting roles

A user-defined role can only be deleted when it is no longer assigned to any user. If necessary, assign the user a different role.

System-defined roles cannot be deleted.

Managing rights

How to access this function

1. Double-click on the "User management" entry in the global security settings.
2. Select the "Roles" tab in User management.

Creating and assigning a user-defined role

1. Double-click on the "Add new role" entry.
2. Enter a name for the role and, if applicable, specify the maximum session time after which users with the assigned role are automatically logged off from the Web page for user-specific IP rule sets.
3. If necessary, select the system-defined role whose rights will be used as the template for the user-defined role from the drop-down list labeled "<Copy rights from>". User-defined roles cannot be selected from the drop-down list.
   Result: In the list of rights of the user roles, the rights are selected that are assigned to the selected system-defined role.
4. For each security module, enable or disable the rights to be assigned to the user-defined role.
5. Assign the role to a user in the "User" tab.

Configuration rights

Configuration rights are not module dependent and control the rights for configuration in STEP 7.

Depending on the user type, the following configuration rights are available for selection:

Table 10-4 Configuration rights

Configuration right	Administrator	Standard	Diagnose
Diagnose security	x	x	x
Configure security	x	x	-
Managing users and roles	x	-	-

Module rights

Module rights are configured per module. The "Service" column shows the service to which the particular right relates. With the "Copy rights" and "Paste rights" commands in the shortcut menu, you can transfer the rights from one module to another.

Depending on the user type, the following module rights are available for selection:

Table 10-5 Module rights CP x43-1 Advanced

Right within the service	Administrator	Standard	Diagnose	Service
Web: Format CP file system *	x	-	-	File system
FTP: Read files from the CP file system	x	x	x	
FTP: Write files to the CP file system	x	x	-	
FTP: Read files (DBs) from the S7 CPU **	x	x	x	PLC
FTP: Write files (DBs) to the S7 CPU ***	x	x	-	
Applet: Read tags using configured symbols *	x	x	x	
Applet: Write tags using configured symbols *	x	x	-	
Applet: Read tags using absolute addresses *	x	x	x	
Applet: Write tags using absolute addresses *	x	x	-	
Applet: Read status of the modules in the rack *	x	x	x	
Applet: Query order numbers of the modules in the rack *	x	x	x	
SNMP: Read MIB-II	x	x	x	SNMP
SNMP: Write MIB-II	x	x	-	
SNMP: Read automation MIB	x	x	x	
SNMP: Read LLDP-MIB	x	x	x	
SNMP: Read SNMPv2-MIB	x	x	x	
SNMP: Read MRP MIB	x	x	x	
SNMP: Write MRP MIB	x	x	-	
TIA Portal: Run diagnostics of the security module ****	x	x	x	Security

Right within the service	Administrator	Standard	Diagnose	Service
Web: Expand IP access control list *	x	-	-	Web
Web: Access Web diagnostics and CP file system	x	x	x	
Web: Send test e-mails *	x	x	x	
Web: Update firmware *	x	x	-	Maintenance
Web: Load diagnostics texts later *	x	x	-	

Table 10-6 Module rights CP 1628

Right within the service	Administrator	Standard	Diagnose	Service
SNMP: Read MIB-II	x	x	x	SNMP
SNMP: Write MIB-II	x	x	-	
SNMP: Read automation MIB	x	x	x	
SNMP: Read SNMPv2-MIB	x	x	x	
TIA Portal: Run diagnostics of the security module ****	x	x	x	Security

Table 10-7 Module rights SCALANCE S

Right within the service	Administrator	Standard	Diagnose	Service
Download the configuration files	x	x	-	Security
TIA Portal: Run diagnostics of the security module ****	x	x	x	
SNMP: Read automation MIB	x	x	x	SNMP
SNMP: Read MIB-II	x	x	x	
SNMP: Write MIB-II	x	x	-	
SNMP: Read MRP-MIB	x	x	x	
SNMP: Write MRP-MIB	x	x	-	
SNMP: Read SNMPv2-MIB	x	x	x	
Web: Update firmware	x	x	-	Maintenance

Table 10-8 Module rights CP 1543-1

Right within the service	Administrator	Standard	Diagnose	Service
FTP: Read files from the CP file system	x	x	x	File system
FTP: Write files to the CP file system	x	x	-	
TIA Portal: Run diagnostics of the security module ****	x	x	x	Security
SNMP: Read automation MIB	x	x	x	SNMP
SNMP: Read IPv6 MIB	x	x	x	
SNMP: Read LLDP-MIB	x	x	x	
SNMP: Read MIB-II	x	x	x	
SNMP: Write MIB-II	x	x	-	
SNMP: Read SNMPv2-MIB	x	x	x	
FTP: Read files (DBs) from the S7 CPU **	x	x	x	PLC
FTP: Write files (DBs) to the S7 CPU ***	x	x	-	

Table 10-9 Module rights CP 1243-1

Right within the service	Administrator	Standard	Diagnose	Service
TIA Portal: Run diagnostics of the security module ****	x	x	x	Security
SNMP: Read automation MIB	x	x	x	SNMP
SNMP: Read IPv6 MIB	x	x	x	
SNMP: Read LLDP-MIB	x	x	x	
SNMP: Read MIB-II	x	x	x	
SNMP: Write MIB-II	x	x	-	
SNMP: Read SNMPv2-MIB	x	x	x	

* To be able to use the function, the module right "Web: Access Web diagnostics and CP file system" must be enabled as well.

** To be able to use the function, the module right "FTP: Read files from CP file system" must be enabled as well.

*** To be able to use the function, the module right "FTP: Write files to CP file system" must be enabled as well.

**** To use the function, the configuration right "Security diagnostics" must also be enabled.

Setting module rights before and after creating the security modules

Within a user-defined role, the module rights for each security module are defined separately. If a security module was created before this role was added and module rights within a role need to be set, STEP 7 presets the module rights for the security module according to the selected rights template. The preset module rights can then be adapted when adding the role. If a security module was created after adding a role, no rights are preset for this security module. In this case, you will need to edit an existing role and set all the module rights yourself for the security module.

You can also transfer existing module rights to another module by copying and, if necessary, adapting them there. To do this, select a module in the shortcut menu in the module rights and select the "Copy rights" or "Paste rights" menu command.

Configuring password policies

Meaning

Using the password policies, specifications can be defined that need to be taken into account when assigning passwords to new users.

How to access this function

1. Double-click on the "User management" entry in the global security settings.
2. Select the "Password policies" tab in User management.

After selecting a check box, the corresponding policy is active and can, if necessary, be adapted using the relevant input box.

Parameter	Meaning
Minimum password length	Minimum number of characters that passwords are required to contain. The corresponding check box is enabled as default and cannot be disabled. • Minimum value: 8 characters • Maximum value: 32 characters
Minimum number of digits	Minimum number of digits that passwords are required to contain. • Minimum value: 1 digit • Maximum value: 32 digits
Minimum number of special characters	Minimum number of special characters that passwords are required to contain. A special character is any character that is neither a letter nor digit. • Minimum value: 1 special character • Maximum value: 32 special characters
Number of user passwords blocked for re-use	Number of the most recently used passwords that are not available for use as a new password if the password is changed. • Minimum value: 1 password • Maximum value: 10 passwords
At least one uppercase and lowercase character	If you select this check box, passwords must contain at least one uppercase and one lowercase letter.

Authentication using a RADIUS server

Module-specific function

This function is available only for SCALANCE S V4 modules or higher, refer to:

Auto-Hotspot in the section "SCALANCE S".

Generating configuration data for SCALANCE M modules

Context

You can generate the VPN information for the assignment of parameters to a SCALANCE M using STEP 7. For this to be possible, the module must be in at least one VPN group with a security module or a SOFTNET Security Client. With the generated files, you can then configure the SCALANCE M using the Web Based Management of the device.

Generated files

The following file types are generated:

- Export file with the configuration data
  - File type: *.txt file in ASCII format
  - Contains the exported configuration information for the SCALANCE M including information on the additionally generated certificates.
- VPN group certificates of the module
  - File type of the private key: *.p12 file
  - The file contains the module certificate and the key material.
  - Access is password protected.
- CA certificates of VPN groups
  - File type: *.cer file

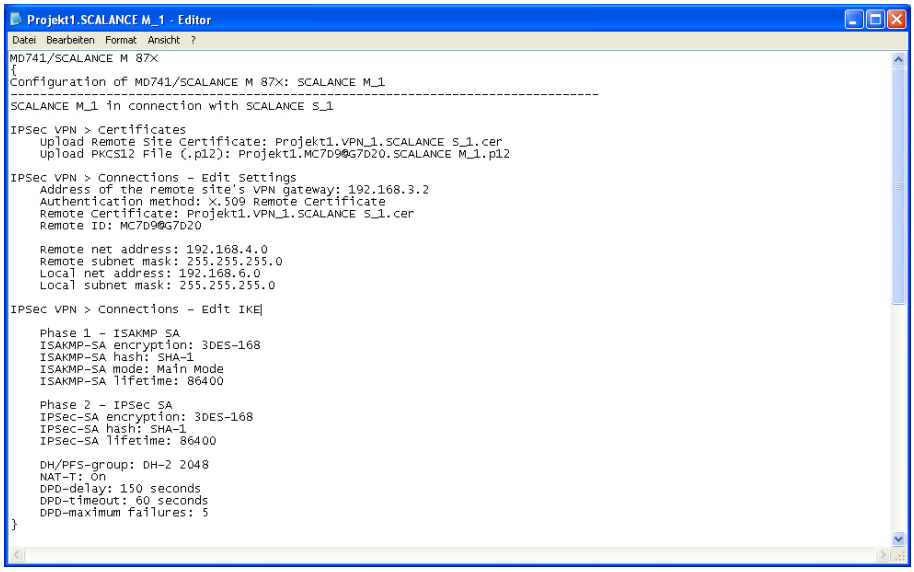


Figure 10-1 SCALANCE M configuration file

Note

No transfer to the security module

Configuration files are not transferred to the security module. An ASCII file is generated with which you can configure the VPN-relevant properties of the SCALANCE M. To allow this, the SCALANCE M must be located in at least one VPN group with another security module.

Note

Protecting exported configuration files from unauthorized access

Configuration files for SCALANCE M exported from STEP 7 can contain security related information. You should therefore make sure that these files are protected from unauthorized access. This is particularly important when passing on the files.

Follow the steps below

1. Select the module of the type "SCALANCE M".
2. In the local security settings, select the entry "SCALANCE M configuration".
3. Select the "Generate SCALANCE M files" check box and select a storage location for the configuration files.
4. Specify a password for the encryption of the VPN group certificate by using the project name as the password or assigning a password of your own.
5. Compile the configuration of the SCALANCE M module.

Result: The files (.txt file and certificates) are stored in the folder you specify.

Configuring interfaces for SCALANCE S modules

Overview

You will find information on configuration interfaces of SCALANCE S modules in Auto-Hotspot of the section "SCALANCE S". The information contained here deals with the following configuration options:

- Mode (bridge mode / routing mode / ghost mode (only for SCALANCE S602 as of V3.1)): Setting the operating mode (Page 773)
- IP address parameters: Configuring IP address parameters (Page 774)
- Port settings (only for SCALANCE S as of V3): Configuring port mode (Page 775)
- Settings for the Internet Service Provider (ISP) if one of the interfaces is operated using PPPoE (only for SCALANCE S as of V3): Configuring an Internet connection (Page 776)
- Dynamic DNS (only for SCALANCE S as of V3): Configuring dynamic DNS (Page 778)
- LLDP (only for SCALANCE S as of V4 in routing mode): Configuring LLDP (Page 781)
- Media redundancy in ring topologies (MRP client), (only for SCALANCE S627-2M as of V4 in routing mode): Auto-Hotspot

For more information on special features of the Ghost mode, refer to the following section: Special features of the ghost mode (Page 784)

The configuration of the interfaces of CPs is described in the sections dealing with the CPs.

Setting up a firewall

Overview of the firewall

Module-specific function

Configuration of the firewall is not possible for the CP 1242-7.

Meaning

The firewall functionality of the security modules is intended to protect networks and stations from third-party influence and interference. This means that only certain, previously specified communications relations are permitted. The firewall discards invalid frames without sending a response.

To filter the data traffic, IP addresses, IP subnets, services or MAC addresses can be used. You can also set a bandwidth limitation.

The firewall functionality can be configured for the following protocol levels:

- IP firewall with stateful packet inspection (layer 3 and 4)
- Firewall also for Ethernet "non-IP" frames according to IEEE 802.3 (layer 2)

With security modules capable of VPN, the firewall can also be used for the encrypted data traffic (IPsec tunnel). With the SCALANCE S602 Security module, the firewall can only be used for unencrypted data traffic.

Firewall rules

Firewall rules describe which packets in which direction are permitted or forbidden. IP rules affect all IP packets of layer 3 or higher. MAC rules only affect frames lower than layer 3.

Types of firewall rules

- Global firewall rule sets: Global firewall rule sets can be assigned to several security modules at the same time. Global firewall rule sets are configured in the global security settings.
- Local firewall rules: Local firewall rules are configured in the local security settings of a security module.
- User-specific IP rule sets (only for SCALANCE S as of V3): User-specific IP rule sets can be assigned to individual or several security modules at the same time. User-specific IP rule sets are configured in the global security settings and assigned there to one or more users.
  SCALANCE S as of V4 (RADIUS): User-specific IP rule sets can be assigned individual or multiple users as well as individual or multiple roles.

Service definitions

With the aid of service definitions, you can also define firewall rules clearly in a compact form. Service definitions are configured in the global security settings and can be used in the global, local and user-specific firewall rules.

Adapting standard rules for IP services

For SCALANCE S modules V3 or higher, you can adapt service-specific firewall rules that are set as default for the interfaces of the security modules. You will find information on configuring these firewall rules in Adapting standard rules for IP services (Page 800) of the section "SCALANCE S".

Automatically generated firewall rules for CP connections

For connections that were configured using CPs, STEP 7 automatically creates firewall rules that allow communication with the partner of the CP in the specified direction (CP active/ passive). The connection establishment directions are taken into account. To display these firewall rules, if the advanced firewall mode is enabled, the "Update connection rules" button needs to be clicked. The firewall rules are then displayed in advanced firewall mode.

Which firewall rules are generated automatically is described in the following sections.

- For S7-300 CPs/S7-400 CPs/PC CPs: Connection-related automatic firewall rules (Page 833) in the section "Security for S7-300 / S7-400 / PC CPs".
- For S7-1200-/S7-1500-CPs: Connection-related automatic firewall rules (Page 845) in the section "Security for S7-1200 / S7-1500 CPs".

Enabling the firewall

Firewall functionality for a specific security module is controlled in the local security settings using the "Enable firewall" check box. If the check box is enabled, the firewall can be configured and is effective following the download. For a security module that is a member of a VPN group, the "Enable firewall" check box is activated by default and cannot be deactivated. After switching to advanced firewall mode, it is not possible to switch back to standard mode. For more information about the standard and advanced firewall modes, refer to the section:

Overview of local firewall rules (Page 711).

Global firewall rule sets

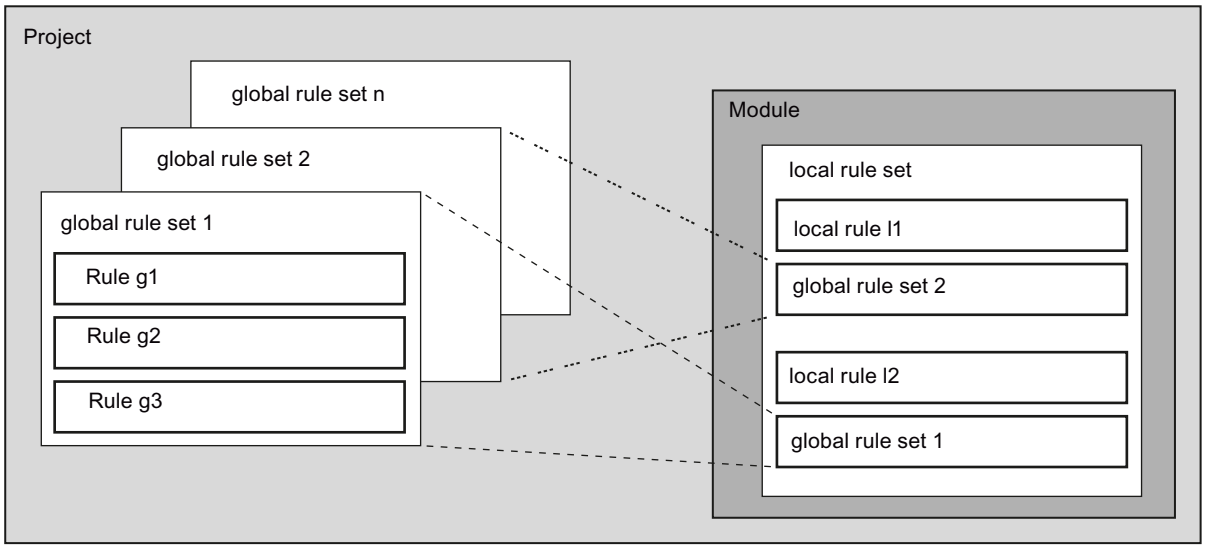
Application

Regardless of the module, global firewall rule sets are configured in the global security settings. A firewall rule set consists of one or more firewall rules and is assigned to the individual security modules.

In the global firewall rule sets, a distinction is made between the following:

- IP rule sets
- MAC rule sets

The following schematic illustrates the relationship between globally defined rule sets and locally used rule sets.



Configuring

When configuring global firewall rules, you can make detailed firewall settings. You can allow individual services for a single node or all services for the node for access to the station or network.

When are global IP and MAC firewall rules useful?

Global firewall rules are useful if you want to define identical filter criteria for communication with several security modules.

Note

Assigning firewall rule sets with incompatible firewall rules

For a security module, only the firewall rules from firewall rule sets are adopted correctly if the security module supports them. A firewall rule contained in a global firewall rule set with the direction "From: External" or "To: Any" is, for example not assigned to a CP 1628. The other firewall rules of the global firewall rule set are adopted if the CP 1628 supports them.

Global firewall rule sets - conventions

Global firewall rule sets are used locally

The following conventions apply when creating a global set of firewall rules and when assigning it to a module:

- Configuration view
  Global firewall rule sets are configured in the global security settings.
- Priority
  As default, locally defined rules have a higher priority than the global IP and MAC firewall rule sets. Newly assigned global IP and MAC firewall rule sets are therefore initially entered at the bottom of the local rule list.
  The priority can be changed by changing the position in the rule list.
- Entering, changing or deleting rule sets
  Global firewall rule sets cannot be edited in the local rule list of the firewall rules in the module properties. They can only be displayed there and positioned according to the required priority.
  It is not possible to delete a single rule from an assigned rule set in the local security settings. Only the entire rule set can be removed from the local list of rules. The firewall rule sets in the global security settings remain unaffected.

Creating global firewall rule sets

How to access this function

1. In the global security settings, select the entry "Firewall" > "Global firewall rule sets" > "IP rule sets" or "MAC rule sets".
   Result: The previously created IP rule sets or MAC rule sets are displayed under the selected entry.
2. Double-click on the entry "Add new IP rule set" or "Add new MAC rule set".
3. Enter the following data:
   – Name: Project-wide, unique name of the rule set. The name appears in the local rule list of the security module after the rule set is assigned.
   – Description (optional): Enter a description of the global rule set.
4. Enter the firewall rules one by one in the list.
   Note the parameter description in the following sections:
   For IP rule sets: Defining IP packet filter rules (Page 712)
   For MAC rule sets: Defining MAC packet filter rules (Page 715)

Result

You have created the global firewall rule set and can now assign this to the required security modules.
Note the descriptions in the following section:

Assigning global firewall rule sets (Page 705)

Assigning global firewall rule sets

Requirement

You have enabled the advanced firewall mode for the security modules you want to assign to a firewall rule set.

Procedure

1. In the global security settings, select the entry "Firewall" > "Global firewall rule sets" > "Assign module to a firewall rule set".
2. From the "Rule set" drop-down list, select the rule set to which you want to assign the security module.
   In the right-hand table, you will see the security modules that you can assign to the selected firewall rule set. In the left-hand table, you will see the security modules already assigned to the selected firewall rule set.
3. In the "Available modules" area, select the security modules you want to assign to the selected rule set.
4. Click the "<<" button to assign the selected modules to the selected rule set.

Result

The global rule set is used by the assigned security modules as a local rule set and appears automatically at the end of the list of firewall rules in the local security settings.

IP services

Defining IP services

How to access this function

In the global security settings, select the entry "Firewall" > "Services" > "Define services for IP rules".

Procedure

Using the definition of IP services, you can define succinct and clear firewall rules for specific services. You select a name and assign the service parameters to it.

These services defined in this way can also be grouped together under a group name.

When you configure the packet filter rules, you then use this name.

Parameters for IP services

You define the IP services using the following parameters:

Table 10-10 IP services: Parameter

Parameter	Meaning/comment	Available options / ranges of values
Name	Name for the service that is used as identification in the rule definition or in the group. The names of pre-defined services cannot be changed.	• Name must start with a letter. • Name must not contain any special characters. • Name must not be redundant.
Protocol	Selection of the protocol type	• TCP • UDP • TCP+UDP • All
Source port	Filtering is based on the port number specified here; this defines the service access at the frame sender.	With the protocol selection "TCP+UDP", it is not possible to specify a port. Examples: *: Port is not checked 20 or 21: FTP service
Destination port	Filtering is based on the port number specified here; this defines the service access at the frame recipient.	With the protocol selection "TCP+UDP", it is not possible to specify a port. Examples: *: Port is not checked TCP 80: Web HTTP service TCP 102: S7 protocol

Defining ICMP services

How to access this function

1. In the global security settings, select the entry "Firewall" > "Services" > "Define services for IP rules".
2. Select the "ICMP" tab.

Procedure

Using the definition of ICMP services, you can define succinct and clear firewall rules for specific services. You select a name and assign the service parameters to it.

These services defined in this way can be grouped together under a group name.

When you configure the packet filter rules, you then use this name.

Parameters for ICMP services

Parameter	Meaning/comment	Available options / ranges of values
Name	User-definable name for the service that is used as identification in the rule definition or in the group. The names of predefined ICMPv6 services cannot be modified.	• Name must start with a letter. • Name must not contain any special characters. • Name must not contain more than 20 characters. • Name must not occur more than once
ICMPv6	If you enable this check box, the ICMP service is declared as an ICMPv6 service and you can select an ICMPv6-specific type and code for the service. An ICMPv6 service can only be used in the firewall rule of a security module that supports IPv6.	• Enabled • Disabled (default)
Type	Type of the ICMPv4 or ICMPv6 message.	If the "ICMPv6" check box is deselected, ICMPv4-specific types can be selected. If the check box is selected, ICMPv6-specific types can be selected.
Code	Code of the ICMP type.	Values depend on the selected type.

Creating service groups

How to access this function

1. In the global security settings, select the entry "Firewall" > "Services" > "Define services for IP rules".
2. Select the "Service groups" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name. IPv4 and IPv6 services can be collected in the same service group.

Create groups in the open "Service groups" tab. You can then assign services to a group in the "Group management" tab.

Follow the steps below

1. First, create groups in this tab with names to suit your requirements and add a description if required.
2. Select the "Group management" tab. You can assign the previously specified IP services to the groups defined in this tab.

Managing service groups

How to access this function

1. In the global security settings, select the entry "Firewall" > "Services" > "Define services for IP rules".
2. Select the "Group management" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name. IPv4 and IPv6 services can be collected in the same service group.

Use the "Group management" tab to assign services to a selected group you created previously in the "Service groups" tab.

Follow the steps below

1. Select a group created previously in the "Service groups" tab from the "Service groups" drop-down list in this tab.
2. Then assign the required services from the right-hand "Available services" list box to the group.

MAC services

Define MAC services

How to access this function

In the global security settings, select the entry "Firewall" > "Services" > "Define services for MAC rules".

Meaning

Using the definition of MAC services, you can define succinct and clear firewall rules for specific services. You select a name and assign the service parameters to it.

These services defined in this way can be grouped together under a group name.

When you configure the global or local packet filter rules, you use this name.

Parameters for MAC services

A MAC service definition is formed using protocol-specific MAC parameters:

Table 10-11 MAC services - parameters

Parameter	Meaning/comment	Available options / ranges of values
Name	User-definable name for the service that is used as identification in the rule definition or in the group.	• Name must start with a letter. • Name must not contain any special characters. • Name must not be redundant.
Protocol	Name of the protocol type: • ISO ISO identifies frames with the following properties: Lengthfield <= 05DC (hex), DSAP= userdefined SSAP= userdefined CTRL= userdefined • SNAP SNAP identifies frames with the following properties: Lengthfield <= 05DC (hex), DSAP=AA (hex), SSAP=AA (hex), CTRL=03 (hex), OUI=userdefined, OUI-Type=userdefined • PROFINET IO • As an alternative, a protocol number can be entered. The protocol entries 0800 (hex) and 0806 (hex) are not accepted since these values apply to IP or ARP frames.	• ISO • SNAP • PROFINET IO • 0x (entry of the protocol number)
DSAP	Destination Service Access Point: LLC recipient address	
SSAP	Source Service Access Point: LLC sender address	
CTRL	LLC control field	
OUI	Organizationally Unique Identifier (the first 3 bytes of the MAC address = vendor ID)	
OUI type	Protocol type/identification	

Note

Processing for S7-CPs

Only settings for ISO frames with DSAP=SSAP=FE (hex) are processed. Other frame types are not relevant for S7 CPs and are therefore discarded even before processing by the firewall.

Special settings for SIMATIC NET services

To filter special SIMATIC NET services, please use the following protocol settings:

- DCP:
  PROFINET IO
- SiClock :
  OUI= 08 00 06 (hex) , OUI-Type= 01 00 (hex)

Creating service groups

How to access this function

1. In the global security settings, select the entry "Firewall" > "Services" > "Define services for MAC rules".
2. Select the "Service groups" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name.

Create groups in the open "Service groups" tab. You can then assign services to a group in the "Group management" tab.

Follow the steps below

1. First, create groups in this tab with names to suit your requirements and add a description if required.
2. Select the "Group management" tab. You can assign the previously specified MAC services to the groups defined in this tab.

Managing service groups

How to access this function

1. In the global security settings, select the entry "Firewall" > "Services" > "Define services for MAC rules".
2. Select the "Group management" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name.

Use the "Group management" tab to assign services to a selected group you created previously in the "Service groups" tab.

Follow the steps below

1. Select a group created previously in the "Service groups" tab from the "Service groups" drop-down list in this tab.
2. Then assign the required services from the right-hand "Available services" list box to the group.

Overview of local firewall rules

Meaning

Local firewall rules are configured in the local security settings of a security module and apply only to this security module. After enabling the firewall functionality you can either use predefined firewall rules or define firewall rules in the advanced firewall mode.

Using predefined firewall rules

Here, you use simple, predefined rules. You can only enable service-specific rules. The enabled services are allowed for all nodes in the specified direction. You can find detailed information on the definition of firewall rules in this dialog in the following module-specific sections:

- For SCALANCE S: Auto-Hotspot
- For S7-300 CPs/S7-400 CPs/PC CPs: Auto-Hotspot
- For S7-1200-/S7-1500-CPs: Auto-Hotspot

Defining firewall rules in advanced firewall mode

In advanced firewall mode, you can make detailed firewall settings. You can allow individual services for a single node or all services for the node for access to the station or network. You enable the advanced firewall mode using the "Activate firewall in advanced mode" check box. In the local security settings, the firewall rules can then be configured in "Firewall" > "IP rules" or "MAC rules". The configuration options available here are described individually in the following section:

For IP packet filter rules: Defining IP packet filter rules (Page 712)

For MAC packet filter rules: Defining MAC packet filter rules (Page 715)

Note

Deactivating the advanced firewall mode not possible

Once you have activated the advanced firewall mode, you can no longer deactivate it.

Configuration limits

Number of firewall rules (advanced firewall mode)	
SCALANCE S as of V3	Maximum of 256
SCALANCE S below V3	Maximum of 226
CP 1543-1	Maximum of 256
CP 1243-1	Maximum of 256
CP 1243-7	Maximum of 256
CP x43-1 Adv.	Maximum of 226
CP 1628	Maximum of 226

Defining IP packet filter rules

Meaning

With IP packet filter rules, you filter according to IP frames, such as TCP, UDP and ICMP.

Within a packet filter rule, you can also fall back on the definitions of the IP services.

Entering IP packet filter rules

Enter the firewall rules one by one in the list. Note the following parameter description.

Table 10-12 IP rules: Parameter

Parameter	Meaning/comment	Available options / ranges of values
Action	Allow/disallow (enable/block)	• Allow Allow frames according to definition. • Drop Block frames according to definition. For firewall rules generated automatically as result of configuring a connection and subsequently adapted manually: • Allow* • Drop* If you change automatically created connection rules, when the "*" option is selected they are not recreated and overwritten by STEP 7.
From / To	Selection of the communications directions for which the rule will apply.	Described in separate sections. • For SCALANCE S modules: IP packet filter directions SCALANCE S (Page 798) • For S7-300 CPs/S7-400 CPs/PC CPs: IP packet filter directions S7-300-/S7-400-/PC-CPs (Page 831) • For S7-1200-/S7-1500-CPs: IP packet filter directions S7-1200 / S7-1500 CPs (Page 844)

Parameter	Meaning/comment	Available options / ranges of values
IPv6 (for the CP 1243-1 and CP 1543-1 only)	If this check box is selected, you can use a previously defined ICMPv6 service in the firewall rule. For the CP 1243-1 and CP 1543-1 V1.1, after selecting the check box you can enter additional IPv6 addresses in the "Source IP address" and "Destination IP address" input boxes. For the CP 1543-1 V1.0, after enabling the check box you cannot enter any source IP address or any destination IP address for the firewall rule.	The check box can only be selected and deselected if there are no entries in the "Source IP address" and "Destination IP address" input boxes. If IPv6 is disabled in the local settings of the CP 1243-1 and CP 1543-1, you cannot select the "IPv6" check box in the local security settings of the CP and therefore cannot use ICMPv6 services or IPv6 addresses in firewall rules. Existing firewall rules that use IPv6 are shown grayed out if IPv6 is disabled.
Source IP address	The firewall rule is applied to the frames whose sender has the IP address specified here. If you do not specify an IP address, the firewall rule applies to all nodes in the communications direction you selected in the "From" column.	You can find additional information about IP addresses in the IP addresses in IP packet filter rules (Page 717) section. **Configuration options in ghost mode (only for SCALANCE S S602 as of V3.1):** If ghost mode is activated, the IP address of the internal node is dynamically determined by the security module at runtime. Depending on the selected direction, you can select one of the following options in the column "Source IP address" (for direction "From internal to external) or in the column "Destination IP address" (for direction "From external to internal"): • IP address of the internal node: The IP address of the internal node is inserted in the firewall rule by the SCALANCE S. • Limited broadcast: The broadcast IP address 255.255.255.255 is inserted in the firewall rule by the SCALANCE S. • Directed broadcast: The broadcast IP address of the SCALANCE S network is inserted in the firewall rule by the SCALANCE S. A directed broadcast can also be forwarded into the destination network via routers. • Multicast: The multicast address band 224.0.0.0 /24 is inserted in the firewall rule by the SCALANCE S. After selecting this option, as an alternative a specific multicast IP address from the multicast address band can be specified.
Destination IP address	The firewall rule is applied to the frames whose recipient has the IP address specified here. If you do not specify an IP address, the firewall rules applies to all nodes in the communications direction you selected in the "To" column.	
Service	Name of the IP/ICMP service or service group used. Here, you select one of the services you defined in the "IP services" dialog: • IP services or • ICMP services Before you select an ICMPv6 service, you need to select the "IPv6" check box.	The drop-down list box displays the services and service groups configured in the global security settings and you can select them.

Parameter	Meaning/comment	Available options / ranges of values
Bandwidth (Mbps)	Option for setting a bandwidth limitation Can only be entered if the "Allow" action is selected. A packet passes through the firewall if the allow rule matches and the permitted bandwidth for this rule has not yet been exceeded.	CP x43-1, CP 1243-1, CP 1243-7, CP 1543-1 and SCALANCE S below V3.0: 0.001 ... 100 Mbps CP 1628 and SCALANCE S as of V3.0: 0.001 ... 1000 Mbps For global and user-specific rules: 0.001 ... 100 Mbps Note: If the direction "From tunnel to station" is configured in a firewall rule for the CP 1543-1 or CP 1243-1, no bandwidth limitation can be specified.
Logging	Enable and disable logging for this rule. If logging is enabled, the settings for the packet filter logging configured in the local security settings apply.	• Enabled • Disabled (default)
Number	Automatically assigned number for the rule. The numbers are recalculated when rules are moved.	
Comment	Space for your own explanation of the rule	If a comment is marked with "AUTO", it was created for an automatic connection rule. For rules you have created, entry of a comment is optional.

Table 10-13 Meaning of the entries in the shortcut menu

Entry in the shortcut menu	Meaning
Delete	This deletes the selected rule or the selected rule set. Notes on removing a globally defined and locally assigned rule set: if you delete a rule set here, this only cancels the assignment to the security module.
Save as global rule set (only for local firewall rules)	Copies the selected firewall rule(s) and inserts it as a global firewall rule set in the global security settings. The firewall configuration currently configured for the security module remains unaffected by this procedure.
Move up	Use this button to move the selected rule or selected rule set up one position in the list. As an alternative, you can move the selected rule or the selected rule set by dragging it with the mouse. You can select multiple entries. The rule / rule set you moved is therefore handled with higher priority.
Move down	Use this button to move the selected rule or selected rule set down one position in the list. As an alternative, you can move the selected rule or the selected rule set by dragging it with the mouse. You can select multiple entries. The rule / rule set you moved is therefore handled with lower priority.
Define service for IP rules	This opens the dialog in which you can manage the IP services and service groups.

Defining MAC packet filter rules

Meaning

With the MAC packet filter rules, you filter according to MAC frames.

Within a packet filter rule, you can also fall back on the definitions of the MAC services.

Entering MAC packet filter rules

Enter the firewall rules one by one in the list. Note the following parameter description.

Table 10-14 MAC rules: Parameter

Parameter	Meaning/comment	Available options / ranges of values
Action	Allow/disallow (enable/block)	• Allow Allow frames according to definition. • Drop Block frames according to definition. For firewall rules generated automatically as result of configuring a connection and subsequently adapted manually: • Allow* • Drop* If you change automatically created connection rules, when the "*" option is selected they are not recreated and overwritten by STEP 7.
From / To	Selection of the communications directions for which the rule will apply.	Described in separate sections. • For SCALANCE S modules: MAC packet filter directions SCALANCE S (Page 799) • For S7-300 CPs/S7-400 CPs/PC CPs: MAC packet filter directions S7-300-/S7-400-/PC-CPs (Page 831) • For S7-1200-/S7-1500-CPs: MAC packet filter directions S7-1200 / S7-1500 CPs (Page 844)
Source MAC address	The firewall rule is applied to the frames whose sender has the MAC address specified here. If you do not specify a MAC address, the firewall rule applies to all nodes in the communications direction you selected in the "From" column.	MAC address in the correct format
Destination MAC address	The firewall rule is applied to the frames whose recipient has the MAC address specified here. If you do not specify a MAC address, the firewall rule applies to all nodes in the communications direction you selected in the "To" column.	
Service	Name of the MAC service or service group used	The drop-down list box displays the configured services and service groups you can select.

Parameter	Meaning/comment	Available options / ranges of values
Bandwidth (Mbps)	Option for setting a bandwidth limitation. Can only be entered if the "Allow" action is selected. A packet passes through the firewall if the allow rule matches and the permitted bandwidth for this rule has not yet been exceeded.	CP x43-1, CP 1243-1, CP 1243-7, CP 1543-1 and SCALANCE S below V3.0: 0.001 ... 100 Mbps CP 1628 and SCALANCE S as of V3.0: 0.001 ... 1000 Mbps For global rules: 0.001 ... 100 Mbps Note: If the direction "From tunnel to station" is configured in a firewall rule for the CP 1543-1 or CP 1243-1, no bandwidth limitation can be specified.
Logging	Enable and disable logging for this rule. If logging is enabled, the settings for the packet filter logging configured in the local security settings apply.	• Enabled • Disabled (default)
Number	Automatically assigned number for the rule. The numbers are recalculated when rules are moved.	
Comment	Space for your own explanation of the rule	If a comment is marked with "AUTO", it was created for an automatic connection rule. For rules you have created, entry of a comment is optional.

Table 10-15 Meaning of the menu commands

Button	Meaning
Delete	This deletes the selected rule or the selected global rule set. Notes on removing a globally defined and locally assigned rule set: if you delete a rule set here, this only cancels the assignment to the security module.
Save as global rule set (only for local firewall rules)	Copies the selected firewall rule(s) and inserts it as a global firewall rule set in the global security settings. The firewall configuration currently configured for the security module remains unaffected by this procedure.
Move up	Use this button to move the selected rule or selected global rule set up one position in the list. As an alternative, you can move the selected rule or the selected rule set by dragging it with the mouse. You can select multiple entries. The rule / rule set you moved is therefore handled with higher priority.
Move down	Use this button to move the selected rule or selected global rule set down one position in the list. As an alternative, you can move the selected rule or the selected rule set by dragging it with the mouse. You can select multiple entries. The rule / rule set you moved is therefore handled with lower priority.
Define service for MAC rules	This opens the dialog in which you can manage the MAC services and service groups.

IP addresses in IP packet filter rules

Entering IP addresses in IP packet filter rules

In IP packet filter rules, you have the following options for entering IP addresses:

- Nothing specified
  The rule applies to all IP addresses.
- An IP address
  The rule applies specifically to the specified address.
- Address range
  The rule applies to all the IP addresses covered by the address range.
  An address range is defined by specifying the number of valid bit places in the IP address in the format: [IP address]/[number of bits to be included]
  - [IP address]/24 therefore means that only the most significant 24 bits of the IP address are included in the filter rule. These are the first three octets of the IP address.
  - [IP address]/25 means that only the first three octets and the highest bit of the fourth octet of the IP address are included in the filter rule.
- Address area
  For the source IP address, an address range can be entered in the following format:
  [Start IP address]-[End IP address]

IPv4 addresses

An IPv4 address consists of 4 decimal numbers in the range from 0 to 255 separated from each other by a dot.

Table 10-16 Examples of address ranges in IPv4 addresses

Source IP address or destination IP address	Address range		Number of addresses*)
	from	to	
192.168.0.0**/16**	192.168.0.0	192.168.255.255	65.536
192.168.10.0**/24**	192.168.10.0	192.168.10.255	256
192.168.10.0**/25**	192.168.10.0	192.168.10.127	128
192.168.10.0**/26**	192.168.10.0	192.168.10.63	64
192.168.10.0**/27**	192.168.10.0	192.168.10.31	32
192.168.10.0**/28**	192.168.10.0	192.168.10.15	16
192.168.10.0**/29**	192.168.10.0	192.168.10.7	8
192.168.10.0**/30**	192.168.10.0	192.168.10.3	4
*) Note: Note that the network address and the broadcast address are not available as IP addresses of network nodes in an address range.			

IPv6 addresses

IPv6 addresses consist of 8 fields each with four hexadecimal numbers (128 bits in total). The fields are separated by a colon. IPv6 addresses can only be entered in IP packet filter rules for the CP 1243-1 and CP 1543-1 V1.1.

Example: fd00:ffff:ffff:ffff:ffff:ffff:2f33:8f21

Rules / simplifications:

- Leading zeros within a field can be omitted.
  Example: Instead of 2001:0db8:2426:08d3:1457:8a2e:0070:7344 it is also possible to use the notation 2001:db8:2426:8d3:1457:8a2e:70:7344.
- If one or more fields have the value 0 (or 0000), a shortened notation is possible.
  Example: Instead of 2001:0db8:0:0:0:0:1428:57ab it is also possible to use the notation 2001:db8::1428:57ab.
  To ensure uniqueness, this shortened form can only be used once within the entire address.
- Decimal notation with periods
  The last 2 fields or 4 bytes can be written in the normal decimal notation with periods.
  Example: The IPv6 address fd00::ffff.125.1.0.1 is equivalent to fd00::ffff:7d01:1.
- Address range notation in IP packet filter rules: In the same way as IPv4 addresses, IPv6 addresses can also be noted in the form of address ranges.
  Example: The entry "2001:0db8:85a3:08d3:1319:8a2e:0:0/96" covers all IPv6 addresses from 2001:0db8:85a3:08d3:1319:8a2e:0:0 to 2001:0db8:85a3:08d3:1319:8a2e:ffff:ffff.

Carrying out module-specific log settings

Log settings - overview

Module-specific function

The recording of packet filter events is not available for the CP 1242-7 and CP 1243-7. The recording of audit events and system events is not available for the CP 1242-7.

Log settings in the configuration

The log settings made here are loaded on the module with the configuration and take effect when the security module starts up.

You can restrict the configured packet filter log settings in the online functions if necessary. For example, you can use the online functions to specify that merely IP logging is displayed if you have configured IP and MAC logging.

Logging procedures and event classes

Here, you can specify which data should be logged. As a result, you enable logging as soon as you download the configuration to the security module.

During configuration, you also select one or both of the possible logging procedures:

- Local logging
- Network Syslog

In both logging procedures, the security module recognizes the three following types of events:

- Packet filter events
- Audit events
- System events

Configuring local logging

How to access this function

1. Select the module to be edited.
2. Select the "Log settings" > "Local log memory" entry in the local security settings.

Configuring local logging

Table 10-17 Local logging - settings for log events

Log event	Meaning	Comments
Packet filter log (firewall)	The packet filter log records certain packets of the data traffic. Only those data packets for which a configured packet filter rule (firewall) applies or to which the basic protection reacts (corrupt or invalid packets) are logged. This is only possible when logging is enabled for the packet filter rule. In the "Packets to be logged" drop-down list, you can specify the data packets to be logged: • "All packets": The data packets that are logged are those to which a configured firewall rule applies. In addition to this, all the response packets to such packets are recorded that have passed the firewall according to a configured allow rule. • "Status generating packets": The only data packets that are logged are those to which a configured firewall rule (standard mode or advanced mode) applies and that initially generate a status in the firewall. Data packets that pass through the firewall by using this firewall status are not logged.	Packet filter log data is not retentive The data is stored in volatile memory on the security module and is therefore no longer available after the power supply has been turned off. For retentive storage, you can also save the log data displayed in the "Online & diagnostics" dialog in a file.
Audit log	The logging of audit events is always enabled. The logged information is always stored in the ring buffer. The audit log automatically records security-relevant events, for example user actions such as activating or deactivating packet logging or downloading configurations to the security module.	Audit log data is retentive The data is stored in a retentive memory of the security module and is therefore still available after turning off the power supply. **Note for CPs:** The audit log files are not retentive on CPs. You should therefore use a Syslog server to backup this data.
System log	The system log automatically logs successive system events, for example the start of a process or the failed login attempt of a user. Select the "Log settings" > "Configure system events" entry to configure the event filter and cable diagnostics functions.	System log data is not retentive The data is stored in volatile memory on the security module and is therefore no longer available after the power supply has been turned off. For retentive storage, you can also save the log data displayed in the "Online & diagnostics" dialog in a file.

Table 10-18 Local logging - storage of recorded data

Storage	Meaning
Ring buffer	At the end of the buffer, the recording continues at the start of the buffer and overwrites the oldest entries.
One-shot buffer	Recording stops when the buffer is full.

Configuring system events

How to access this function

1. Select the module to be edited.
2. Select the "Log settings" > "Configure system events" entry in the local security settings.

Filtering of the system events

In this dialog, you set a filter level for the system events. As default, the following values are set:

- SCALANCE S: Level 3 (Error)
- CP: Level 3 (Error)

The priority of the selected filter level must be the same or lower than the severity set for the cable diagnostics; see the "Setting parameters for line diagnostics" table (not for CPs).

Recommendation: Select "Error" as the filter level or a higher value to exclude logging of general, uncritical events.

Note for CP

For CPs, select only level 3 or level 6 since only events of these levels are generated for CPs.

- Level 0 to level 3 error messages are output if level 3 is selected.
- If you select level 6, the error messages of levels 0 to 6 are output.

Properties of the system events - line diagnostics (only for SCALANCE S)

Line diagnostics generates a special system event. A system event is generated when a selectable percentage of bad frames is reached. This system event is assigned the severity and facility set in this dialog.

Table 10-19 Setting parameters for line diagnostics

Function / option / parameter	Meaning
Enable	Enabling and disabling logging
Limit	Selectable percentage of faulty frames representing the limit at which a system event is triggered.

Function / option / parameter	Meaning
Facility	Select a facility from the drop-down list that identifies the system event to be logged.
Severity	Using the severity, you weight the system events of line diagnostics relative to the severity of the other system events.

Note

Severity of the system events of line diagnostics

Den System-Ereignissen der Leitungsdiagnose darf keine geringere Severity zugewiesen werden, als Sie für den Filter eingestellt haben. Bei einer geringeren Severity können diese Ereignisse den Filter nicht passieren und werden nicht aufgezeichnet.

Configuring the network Syslog

How to access this function

1. Select the module to be edited.
2. Select the "Log settings" > "Network Syslog" entry in the local security settings.

Configuring the network Syslog

Table 10-20 Network Syslog - basic settings

Option / parameter	
Enable network Syslog	Enables and disables the transfer of logging events to the Syslog server.
Syslog server	Here, enter the IP address of the Syslog server. As an alternative, an FQDN can be entered for SCALANCE S modules as of V4. The Syslog server must be accessible from the security module at the specified address and, if applicable, using the router configuration under the "Routing" entry of the local security settings. If the Syslog server cannot be accessed no Syslog messages are sent. You can recognize this operating state on the basis of corresponding system alarms. To activate the sending of the Syslog information again, you may have to update the routing information and initiate a restart of the security module.
Enabling event classes	Enable the classes of events to be transferred to the Syslog server. You can classify packet filter and audit events according to the Severity and using the Facility according to their origin.
Module name	The module name is displayed and cannot be changed at this point.

Note

Non-secure transfer of logging events

Logging events are transferred to Syslog servers in plain language. This should be taken into account when using Syslog servers.

Table 10-21 Network Syslog - settings for log events

Log event	Configuring	Comments
Packet filter events (firewall)	The packet filter log records certain packets of the data traffic. Only those data packets for which a configured packet filter rule (firewall) applies or to which the basic protection reacts (corrupt or invalid packets) are logged. This is only possible if logging is activated for the packet filter rule. Syslog alarms can be classified according to their origin and their severity level by setting Facility and Severity. This assignment is carried out with drop-down lists. The severity and facility that you set here is assigned to every event.	The value you select here depends on the evaluation in the Syslog server. If you retain the "default" value setting, the security module specifies the combination of facility and severity with which the event is displayed.
Audit events	The audit log automatically records security-relevant events, for example user actions such as activating or deactivating packet logging or downloading configurations to the security module. Assignment of the severity and facility is carried out with drop-down lists. The severity and facility that you set here is assigned to every event.	The value you select here for the severity and facility depends on the evaluation in the Syslog server. If you retain the "default" value setting, the security module specifies the combination of facility and severity with which the event is displayed.
System events	The system log automatically logs successive system events, for example the start of a process or the failed login attempt of a user.	Select the "Log settings" > "Configure system events" entry in the local security settings to configure the event filter and cable diagnostics functions.

Security module as router

Overview of the routing settings

Meaning

If you operate the security module in routing mode, the networks connected to the internal and external interface are transformed into separate subnets. The DMZ interface (SCALANCE S623/S627-2M only) is connected in routing mode regardless of the mode. In routing mode, the frames intended for an existing IP address in the subnet are forwarded. The firewall rules for the direction of transmission also apply.

You have the following additional options:

- Setting specific routes - can be set in the local security settings under "Routing" (SCALANCE S only), see Specifying routes (Page 801) in the section "SCALANCE S".
- Use standard router - can be set in the local security settings with "External interface [P1] red", "Internal interface [P2] green" or "DMZ interface [P3] yellow" (SCALANCE S623/ S627-2M only) see Configuring IP address parameters (Page 774) in the section "SCALANCE S".
  A maximum of one standard router can be used per security module.
- NAT/NAPT routing - can be set with "NAT / NAPT" (only for SCALANCE S and CP x43-1 Adv.) in the local security settings. To be able to use NAT/NAPT, the security module must be in routing mode.

Enabling routing mode (required for SCALANCE S modules only)

For this mode, you configure an internal IP address and an internal subnet mask for addressing the router in the internal subnet in the local security settings. All network requests that do not belong to a subnet are forwarded by the security module to a different subnet.

Note: In contrast to the bridge mode of the security module, VLAN tags are lost in routing mode.

1. Select the "Routing mode" option under "Mode" in the local security settings.
2. In the local security settings, enter an internal IP address and an internal subnet mask for addressing the router on the internal subnet in the input boxes under "Internal interface [P2] green" > Ethernet addresses".

Overview of NAT/NAPT

Module-specific function

This function is only available for SCALANCE S and CP x43-1 Adv.

Requirements

- The security module is in routing mode or the DMZ interface is activated (SCALANCE S623 / S627-2M only).
- Since firewall rules that enable communication in the configured address translation direction are generated automatically for NAT / NAPT rules, the advanced firewall mode must be enabled for the security module. For more detailed information, refer to section Relationship between NAT/NAPT router and firewall (Page 732)

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "NAT / NAPT".
3. When required, enable address translation according to NAT(Network Address Translation) or NAPT (Network Address Port Translation).

Address translation with NAT (Network Address Translation)

NAT is a procedure for address translation between two address spaces.
The main task is to translate private addresses into public addresses; in other words into IP addresses that are used and even routed on the Internet. As a result, the IP addresses of the internal network are not known to the outside in the external network. The internal nodes are only visible to the external network by means of external IP address that is specified in the address translation list (NAT table). If the external IP address is not the address of the security module and if the internal IP address is unique, this is known as 1:1 NAT. With 1:1 NAT, the internal address is translated to this external address without port translation. Otherwise, n:1 NAT is being used.

Address translation with NAPT (Network Address Port Translation)

The address translation with NAPT changes the target address and the target port to a communication relation (port forwarding).

Frames coming from the external network or DMZ network and intended for the IP address of the security module are translated. If the destination port of the frame is identical to one of the values specified in the "Source port" column, the security module replaces the destination IP address and the destination port as specified in the corresponding row of the NAPT table. With the reply, the security module uses the values for the destination IP address and destination port as contained in the initial frame as the source IP address and the source port.

The difference to NAT is that with this protocol ports can also be translated. There is no 1:1 translation of the IP address. There is now only a public IP address that is translated to a series of private IP addresses with the addition of port numbers.

Address translation in VPN tunnels

Address translations with NAT/NAPT can also be performed for communications relations established via a VPN tunnel. This is supported for connection partners of the type SCALANCE S612 / S623 / S627-2M as of V4.

You will find further information on address translations in VPN tunnels in the following sections:

- NAT/NAPT routing (Page 726)
- Address translation with NAT/NAPT in VPN tunnels (Page 732)

Consistency check - these rules must be adhered to

Among other things, remember the following rules to obtain consistent entries:

- The IP address of the internal interface must not be used in the NAT / NAPT table.
- An IP address used in the NAT/NAPT address conversion list must not be a multicast or broadcast address.
- The external ports assigned for the NAPT translation are in the range > 0 and ≤ 65535. Port 123 (NTP), 443 (HTTPS), 514 (Syslog), 161 (SNMP), 67+68 (DHCP) and 500+4500 (IPsec) are excluded if the relevant services are activated on the security module.
- The external IP address of the security module or the IP address of the DMZ interface may only be used in the NAT table for the action "Source NAT".

- Checking for duplicates in the NAT table
  An external IP address or an IP address in the DMZ network used in the direction "Destination NAT", "Source NAT +Destination NAT" or "Double NAT" may only be used once in each specified direction.
- Checking for duplicates in the NAPT table: A source port number may only be entered once for each interface.
- Internal NAPT ports can be in the range > 0 and ≤ 65535.

See also

Overview of the routing settings (Page 723)

NAT/NAPT routing

Enabling NAT

The input boxes for NAT are enabled. NAT address translations only take effect with the entries in the address translation list described below. After creating NAT rules, the corresponding firewall rules are generated and displayed in advanced firewall mode, see section:

Relationship between NAT/NAPT router and firewall (Page 732)

If PPPoE is activated for the external interface or the DMZ interface, the action "Destination NAT" cannot be configured. When configuring the action "Source NAT", the IP address cannot be entered in the "Source translation" input box because this is obtained dynamically during runtime.

Possible address translations for NAT

The following tables show the input options for address translation with NAT.

Action "Destination NAT" - "Redirect"

The action "Destination NAT" can be performed in the following direction:

- External to internal

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Destination NAT" can also be performed in the following directions:

- External to DMZ
- DMZ to internal
- DMZ to external

If the SCALANCE S module (only SCALANCE S612/S623/S627-2M as of V4) is in a VPN group and the tunnel interface is enabled, the "Destination NAT" action can also be performed in the following directions:

- Tunnel to internal
- Tunnel to external
- Tunnel to DMZ (only if the DMZ interface is activated)

The following applies, for example for the direction "external to internal": The destination IP address of a frame coming from the external network is checked to see whether it matches the IP address specified in the "Destination IP address" input box. If it matches, the frame is forwarded into the internal network by replacing the destination IP address of the frame with the IP address specified in the "Destination translation" input box. Access from external to internal using the external address is possible.

The following table shows the input required for the action "Destination NAT".

Box	Possible entries	Meaning
Source IP address	Not relevant for this action.	-
Source translation	Not relevant for this action.	-
Destination IP address	IP address in the source network	Destination IP address in the source network with which an IP address in the destination network will be accessed. The destination IP address must not match the IP address of the security module in the source network. If the destination IP address in a frame matches the address entered, the address is replaced by the corresponding IP address in the destination network. The specified destination IP address becomes the alias address. This means that the specified IP address is also registered as an IP address on the selected interface. Make sure that the alias address does not cause an IP address conflict in the network. The alias IP addresses of a security module are displayed under the entry "Alias IP addresses" of the relevant interface.
Destination translation	IP address in the destination network	The destination IP address is replaced by the IP address specified here.
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAT rule.

Action "Source NAT" - "Masquerading"

The action "Source NAT" can be performed in the following direction:

- Internal to external

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Source NAT" can also be performed in the following directions:

- Internal to DMZ
- External to DMZ
- DMZ to external

If the SCALANCE S module (only SCALANCE S612/S623/S627-2M as of V4) is in a VPN group and the tunnel interface is enabled, the "Source NAT" action can also be performed in the following directions:

- Internal to tunnel
- External to tunnel
- DMZ to tunnel (only if the DMZ interface is activated)

The following applies, for example for the direction "internal to external": The source IP address of a frame coming from the internal network is checked to see whether it matches the IP address specified in the "Source IP address" input box. If it matches, the frame with the external IP address specified in the "Source translation" input box is forwarded to the external network as a new source IP address. In the external network, the external IP address is effective.

The following table shows the input required for the action "Source NAT".

Box	Possible entries	Meaning
Source IP address	IP address in the source network	The source IP address of the specified node is replaced by the IP address specified in the "Source translation" input box.
	IP address range / IP address band in the source network	The IP addresses of the IP address range / IP address band are replaced by the IP address specified in the "Source translation" input box.
	*	The IP addresses of all nodes in the source network are replaced by the IP address specified in the "Source translation" input box.
Source translation	IP address in the destination network	Entry of the IP address that will be used as the new source IP address. If the IP address entered here is not the IP address of the security module, this becomes an alias address. This means that the specified address is also registered as an IP address on the selected interface. Make sure that the alias address does not cause an IP address conflict in the network. The alias IP addresses of a security module are displayed under the entry "Alias IP addresses" of the relevant interface.
Destination IP address	Not relevant for this action.	-
Destination translation	Not relevant for this action.	-
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAT rule.

Note

You can configure an address translation to the module IP address in the destination network for all frames going from a source network to a destination network. The security module also assigns a port number for each frame. This is an n:1 NAT address translation in which multiple IP addresses of the source network are translated to one IP address of the destination network.

Enter, for example, the following parameters for the direction "internal to external":

- Action: "Source NAT"
- From: "Internal"
- To "External"
- Source IP address: "*"
- Source translation: External IP address of the security module

Action "Source NAT + Destination NAT" - "1:1-NAT"

The action "Source NAT + Destination NAT" can be performed in the following direction:

- Internal to external

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Source NAT + Destination" can also be performed in the following directions:

- Internal to DMZ
- External to DMZ
- DMZ to external

If the SCALANCE S module (only SCALANCE S612/S623/S627-2M as of V4) is in a VPN group and the tunnel interface is enabled, the "Source-NAT + Destination-NAT" action can also be performed in the following directions:

- External to tunnel
- Internal to tunnel
- DMZ to tunnel (only if the DMZ interface is activated)

The following applies, for example for the direction "internal to external": When accessing from internal to external, the action "Source NAT" is performed. When accessing from external to internal, the action "Destination NAT" is performed.

The following table shows the input required for the action "Source NAT + Destination NAT":

Box	Possible entries	Meaning
Source IP address	IP address in the source network	The configuration is always specified in the source NAT direction. The IP addresses of the destination NAT direction are then inserted automatically by STEP 7.
Source translation	IP address in the destination network	
Destination IP address	Not relevant for this action.	
Destination translation	Not relevant for this action.	
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rules generated by STEP 7 for the NAT rule.

Action "Double NAT"

The action "Double NAT" can be performed for SCALANCE S modules in the following directions:

- Internal to external
- External to internal

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Double NAT" can also be performed in the following directions:

- Internal to DMZ
- External to DMZ
- DMZ to internal
- DMZ to external

In every direction, Source and Destination NAT always take place at the same time.
The following applies, for example for the direction "external to internal": When accessing from external to internal, the source IP address of the external node is replaced (Source NAT). Access to the internal network also uses the external IP address specified in the "Destination IP address" input box (Destination NAT).

You can, for example, use this action if a standard router other than the security module is entered for a device to be accessed using Destination NAT. Response frames from this device are then not sent to the entered standard router but to the corresponding interface of the security module.

The following table shows the input required for the action "Double NAT":

Box	Possible entries	Meaning
Source IP address	IP address in the source network	IP address of the node in the source network
Source translation	-	The Source NAT address translation is always to the IP address of the security module in the destination network. For this reason, the "Source translation" input box cannot be configured.

Box	Possible entries	Meaning
Destination IP address	IP address in the source network	Destination IP address in the source network with which an IP address in the destination network will be accessed. If the destination IP address in a frame matches the IP address entered, the IP address is replaced by the IP address specified in the "Destination translation" input box. If the IP address entered here is not the IP address of the security module, this becomes an alias address. This means that the specified address is also registered as an IP address on the selected interface. Make sure that the alias address does not cause an IP address conflict in the network. The alias IP addresses of a security module are displayed under the entry "Alias IP addresses" of the relevant interface.
Destination translation	IP address in the destination network	The destination IP address is replaced by the IP address specified here.
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAT rule.

Enabling NAPT

The input boxes for NAPT are enabled. NAPT translations only take effect with the entries in the list described below. After creating NAPT rules, the corresponding firewall rules are generated and displayed in advanced firewall mode, see section:
Relationship between NAT/NAPT router and firewall (Page 732)

The IP address translation with NAPT can be performed in the following direction:

- External to internal

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the IP address translation with NAPT can also be performed in the following directions:

- External to DMZ
- DMZ to internal
- DMZ to external

If the SCALANCE S module (only SCALANCE S612/S623/S627-2M as of V4) is in a VPN group and the tunnel interface is enabled, the IP address translation with NAPT can also be performed in the following directions:

- Tunnel to internal
- Tunnel to external
- Tunnel to DMZ (only if the DMZ interface is activated)

The following applies, for example for the direction "external to internal": Frames intended for the external IP address of the security module and for the port entered in the "Source port" column are forwarded to the specified destination IP address in the internal network and to the specified destination port.

The following table shows the input required for address translation with NAPT:

Box	Possible entries	Meaning
Source port	TCP/UDP port or port range Example of entering a port range: 78:99	A node in the source network can send a frame to a node in the destination network by using this port number.
Destination IP address	IP address in the destination network	Frames intended for the IP address of the security module in the source network and the TCP/UDP port specified in the "Source port" box are forwarded to the IP address specified here.
Destination port	TCP/UDP port	Port number to which the frames coming from the source network are forwarded.
Protocol	• TCP+UDP • TCP • UDP	Selection of the protocol family for the specified port numbers
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAPT rule.

Address translation with NAT/NAPT in VPN tunnels

Module-specific function

Address translation with NAT/NAPT in VPN tunnels is only available for SCALANCE S612/ S623/S627-2M modules as of V4, refer to the section:
Address translation with NAT/NAPT in VPN tunnels (Page 802)

Relationship between NAT/NAPT router and firewall

Meaning

After creating NAT/NAPT rules, STEP 7 automatically generates firewall rules that enable communication in the configured address translation direction. The generated firewall rules are visible in advanced firewall mode and can, if necessary, be moved and expanded (additional IP address / IP address range / IP address band, services, bandwidth). In addition to this, the automatically generated firewall rules should be checked in terms of their priority and their position. If there are also manually configured firewall rules in the rule list that have higher priority than the automatically generated firewall rules, under certain circumstances no NAT / NAPT will be performed.

If there are several identical NAT / firewall pairs of rules, the priority in the firewall rule list decides which rule is used.

Firewall parameters generated by STEP 7 cannot be adapted. After deactivating NAT/NAPT, the firewall rules generated by STEP 7 are disabled.

To clarify the relationship between the NAT/NAPT rules and the corresponding firewall rules, they are identified by corresponding, consecutive numbers.

The following table shows the system behind the firewall rules generated for NAT rules for SCALANCE S modules.

Table 10-22 NAT address translation and corresponding firewall rules for SCALANCE S modules

NAT action	Created firewall rule				
	Action	From	To	Source IP address	Destination IP address
Destination NAT	Allow	Source network	Destination network	-	IP address specified in the "Destination IP address" input box
Source NAT	Allow	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	-
Source NAT + Destination NAT	Allow	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	-
	Allow	Destination network	Source network	-	IP address that was inserted in the "Destination IP address" input box by STEP 7
Double NAT	Allow	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	IP address specified in the "Destination IP address" input box
	Allow	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	IP address of the node specified in the "Destination translation" input box

The following table shows the system behind the firewall rules generated for NAT rules for CP x43-1 Adv.

Table 10-23 NAT address translation and corresponding firewall rules for CP x43-1 Adv.

NAT action	Created firewall rule				
	Action	From	To	Source IP address	Destination IP address
Destination NAT	Drop	External	Station	-	IP address of the security module in the external network
	Allow	External	Any	-	IP address of the node specified in the "Destination translation" input box

NAT action	Created firewall rule				
	Action	From	To	Source IP address	Destination IP address
Source NAT	Allow	Any	External	IP address specified in the "Source translation" input box	-
Source NAT + Destination NAT	Allow	Any	External	IP address specified in the "Source translation" input box	-
	Drop	External	Station	-	IP address of the security module in the external network
	Allow	External	Any	-	IP address of the node that was inserted in the "Destination translation" input box by STEP 7

The following table shows the system behind the firewall rules generated for NAT rules for SCALANCE S modules.

Table 10-24 NAPT translation and firewall rules created for SCALANCE S modules

Created firewall rule					
Action	From	To	Source IP address	Destination IP address	Service
Allow	Source network	Destination network	-	IP address of the security module in the source network	[Service_NAPT_rule]

The following table shows the system behind the firewall rules generated for NAPT rules for CP x43-1 Adv.

Table 10-25 NAPT translations and created firewall rules for CP x43-1 Adv.

Created firewall rules					
Action	**From**	**To**	**Source IP address**	**Destination IP address**	**Service**
Drop	External	Station	-	IP address of the security module in the external network	[Service_NAPT_rule]
Allow	External	Any	-	IP address of the security module in the external network	[Service_NAPT_rule]

Stateful packet inspection

The firewall and NAT/NAPT router supports the "Stateful Packet Inspection" mechanism. As a result, reply frames can pass through the NAT/NAPT router and firewall without it being necessary for their addresses to be included extra in the firewall rule and the NAT/NAPT address translation.

Relationship between NAT/NAPT router and user-specific firewall

Module-specific function

The configuration of NAT/NAPT rules in the user-specific firewall is only available for SCALANCE S modules as of V3, refer to the section:

Relationship between NAT/NAPT router and user-specific firewall (Page 804)

Configuring time-of-day synchronization

Overview of time-of-day synchronization

Meaning

The date and time are maintained on the security module to verify the validity (time) of a certificate and for time stamping log entries.

The following alternatives can be configured:

- Set time with each download: The module time is set automatically to the PC time when a configuration is downloaded.
- SIMATIC: If the security module receives MMS time-of-day messages, its local time is synchronized provided the NTP procedure was not configured (MMS = Manufacturing Message Specification).
- Time from partner: The time is obtained from a telecontrol server.
- NTP: Automatic setting and cyclic synchronization of the time-of-day by means of Network Time Protocol server.

Note

Time-of-day synchronization relates solely to the security module and cannot be used to synchronize devices in the internal network of the security module. S7 CPs can forward the time of day to other modules of the station.

Note

Configuring the firewall for communication with NTP servers

If the NTP server cannot be reached by the security module, you will need to allow the frames from the NTP server explicitly in the firewall (UDP, port 123).

Note

Before the security functions of a CP (time-of-day slave) are used, this must receive a valid time-of-day synchronization frame from the time master.

Defining an NTP server

Creating NTP servers in the global security settings

In the global security settings only NTP servers of the type "NTP (secure)" can be created and assigned CPs or SCALANCE S modules as of V4. Non-secure NTP servers for SCALANCE S modules below V4 and CPs therefore need to be created in the local security settings.

1. Double-click the "NTP" entry in the global security settings.
2. Double-click on the "Add new NTP server" entry.
3. Enter a name for the NTP server (secure).
4. Enter the IP address of the NTP server (secure). If the NTP server will only be assigned SCALANCE S modules as of V4, an FQDN can be specified as an alternative.

5. Specify the encryption parameters for the NTP server (secure).

Property	Meaning
Key ID	Numeric value between 1 and 65534.
Authentication	Select the authentication algorithm.
Hex/ASCII	Select the format for the NTP key.
Key	Enter the NTP key with the following lengths: Hex: 22 ... 40 characters ASCII: 11 ... 20 characters

6. Assign a the required security module to the NTP server (secure) you have created, see section:
Assigning the security module to an NTP server (secure) (Page 739).

Creating NTP servers in the local security settings

1. Select the module to be edited.
2. Select the "Time-of-day synchronization" entry in the local security settings.
3. Select the required synchronization mode.
4. Enter a name and the IP address of the NTP server. If you have selected the "NTP (secure)" synchronization mode, you can select an NTP server (secure) that you created in the global security settings in the "Name" column.

Configuration limits for NTP servers

You can assign a maximum of 4 NTP servers to one security module.

Importing/exporting NTP servers (secure)

Using the "Import" or "Export" commands in the shortcut menu, you can export the key list of the currently selected NTP server (secure) and import the file into an NTP server (secure) or vice versa.

Configuring time-of-day synchronization for a security module

How to access this function

1. Select the module to be edited.
2. Select the "Time-of-day synchronization" entry in the local security settings.
3. Select the "Activate time-of-day synchronization" check box.

Alternatives for time-of-day synchronization

The following alternatives can be configured:

Table 10-26 Time synchronization for CP and SCALANCE S

Possible selection	Meaning / effect
Set time with each download (only for SCALANCE S)	The module time is set automatically to the PC time when a configuration is downloaded.
SIMATIC (for CP x43-1 Adv. and CP 1628 only)	If the security module receives MMS time-of-day messages, its local time is synchronized provided the NTP procedure was not configured (MMS = Manufacturing Message Specification).
Time from partner (only for CP 1243-1, CP 1242-7 and CP 1243-7 with communication type "Telecontrol communication")	The time is obtained from the partner (telecontrol server).
NTP (not for CP 1243-1, CP 1242-7 and CP 1243-7 with communication type "Telecontrol communication")	Automatic setting and periodic synchronization of the time using an NTP server.
NTP (secure) (not for SCALANCE S below V4 and CP 1243-1, CP 1242-7 and CP 1243-7 with communication type "Telecontrol communication")	Automatic setting and periodic synchronization of the time using an NTP server (secure).

Selecting the mode of time-of-day synchronization

Follow these steps:

1. Select the synchronization mode.
2. The following configuration options are available to you depending on the selected mode:

- **Set time with each download:** When a configuration is downloaded to a SCALANCE S module, the module time of day is set with the PC time of day.
- **SIMATIC:** For the CP x43-1 Adv. select whether the CP is to adopt or forward the time of day. The CP 1628 always forwards the time of day. Also the set direction for forwarding the time.
  Available directions:
  - Automatic (only for CP x43-1 Adv.): The CP receives the time from the station or from the LAN and forwards it to the station or to the LAN. If multiple CPs are being operated in the station, this automatic setting can cause collisions. To avoid this, you can specifically define the forwarding direction.
  - From station (for CP x43-1 Adv. only).
  - From LAN.

  If forwarding of the time of day is enabled, you can use the "Use corrected time" check box to specify whether or not a correction factor included in the time-of-day frame is used. For the CP 1628, this option is enabled as default and cannot be disabled.

- **Time from partner:** For the CP 1243-1, CP 1242-7 and CP 1243-7 in the communication type "Telecontrol communication", the time of day is obtained from the partner (telecontrol server).
  - Synchronization cycle: Specifies the cycle for time-of-day synchronization. For the synchronization cycle of the CP, and individual hour or minute interval can be specified.
- **NTP:**
  - Time zone (for CP x43-1 Adv. / CP 1628 only): In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred. This corresponds to GMT (Greenwich Mean Time). The time offset from UTC can be set by configuring the local time zone.
  - Update interval in seconds: Defines the interval between the time queries in seconds. For SCALANCE S as of V3, the time interval for querying the NTP server is specified automatically.

    Note

    Setting the update interval for CPs

    If the "Enable security functions" check box is enabled in the local security settings of a CP, the setting for the update interval is transferred from the CP's local settings into the CP's local security settings.

  - Time-of-day synchronization on the full minute (for CP x43-1 Adv./CP 1243-1/CP 1628 only): With this option, you can decide whether or not the time of day is forwarded to the communications bus on the full minute. This option is required for certain special applications.
  - Accept time of non-synchronized NTP servers (only for CPs): Here you can specify whether the security module also accepts the time-of-day from non-synchronized NTP servers.
  - Forward time of day to station (only for CP x43-1 Adv./CP 1628): Disable this option if the CPU requests the time separately from an NTP server. This prevents the time on the CPU obtained directly from the NTP server from being overwritten by the time detected in the CP. The accuracy may be reduced slightly due to forwarding via the CP.
  - NTP server: Creating NTP servers in the local security settings is described in the section Defining an NTP server (Page 736).

Assigning the security module to an NTP server (secure)

Requirement

- You have defined an NTP server (secure) in the global security settings.
- "NTP" or "NTP (secure)" is selected as the time-of-day synchronization mode in the local security settings of the security module that you want to assign to an NTP server (secure).

Procedure

1. Double-click the "NTP" entry in the global security settings.
2. Double-click the entry "Assign module to an NTP server".

3. From the "NTP Server" drop-down list, select the NTP server (secure) to which you want to assign a security module.
4. In the "Available modules" section, select the security module that you want to assign to the selected NTP server (secure).
5. Click the "<<" button to assign the selected security module to the selected NTP server (secure).

Result

You have assigned the security module to the NTP server (secure). The NTP server (secure) is displayed automatically in the local security settings in the list of NTP servers.

Security module as DHCP server

Module-specific function

The use of the security module as a DHCP server is only possible with SCALANCE S modules, see subsection:
Auto-Hotspot in the section "SCALANCE S".

Configuring SNMP

Overview of SNMP

What is SNMP?

The security module supports the transfer of management information using the Simple Network Management Protocol (SNMP). For this purpose, an SNMP agent that receives and responds to SNMP requests is installed on the security module. The information on the properties of SNMPcompliant devices is entered in MIB files (MIB = Management Information Base) for which the user must have the required rights.

In SNMPv1, the "community string" is also sent. The "community string" is like a password that is transmitted along with the SNMP request. If the community string is correct, the security module replies with the required information. If the string is incorrect, the security module discards the query and does not reply. The community string is transmitted via SNMPv1 without encryption.

SNMPv3 lets you transmit encrypted data.

Configuring SNMP - "SNMP" entry

Module-specific function

The configuration of SNMP is possible only for SCALANCE S V3 or higher, CP x43-1 Adv., CP 1543-1, CP 1243-1 and CP 1628.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "SNMP".
3. Activate the "Activate SNMP" check box.
4. Select one of the following SNMP protocol versions:

   Note

   Encrypted data transmission with SNMPv3

   You should use SNMPv3 to transmit data in encrypted form in order to enhance security.

   – SNMPv1
   The security module uses the following default values for the community strings to control the access rights in the SNMP agent. These default values should be adapted to increase security.
   For read access: public
   For read and write access: private
   To enable write access using SNMP, select the "Allow write access" option.

   – SNMPv3
   Select either only an authentication algorithm or an authentication algorithm and an encryption algorithm.
   Authentication algorithm: none, MD5, SHA-1
   Encryption algorithm: none, AES-128, DES

   Note

   Preventing the use of DES

   DES is an insecure encryption algorithm. Therefore, it should only be used for reasons of down compatibility.

   Note

   When using SNMPv3 no RADIUS authentication is possible.

5. If SNMPv3 is to be used, assign a user a role with corresponding activated SNMP rights to enable access to the module via SNMP. An overview of SNMP rights is available in the section:
   Managing rights (Page 694).
6. In the "Advanced settings" area, for SCALANCE S modules configure module-specific information about the author, location and e-mail address overwriting the information from the project properties.
   The following applies to values written to the security module by an SNMP tool using an SNMP-SET command:
   If you select the "Keep values written by SNMP set" check box, the values are not overwritten when there is another download of a STEP 7 configuration to the security module.

Configuring proxy ARP

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Configuring proxy ARP (Page 808)

Activate Web server on security module

Module-specific function

This function is only available for CP x43-1 Advanced, see subsection:
Activating the Web server on CP x43-1 Advanced (Page 835) in the section "Security for S7-300 / S7-400 / PC CPs".

IPsec tunnel: Creating and assigning VPN groups

How to create an IPsec tunnel with VPN groups

Module-specific function

This function is available only for SCALANCE S612/S613/S623/S627-2M, CP x43-1 Adv., CP 1243-1, CP 1543-1 V1.1, 1243-7 and CP 1628.

Requirement

Note

Current date and current time of day on the security modules

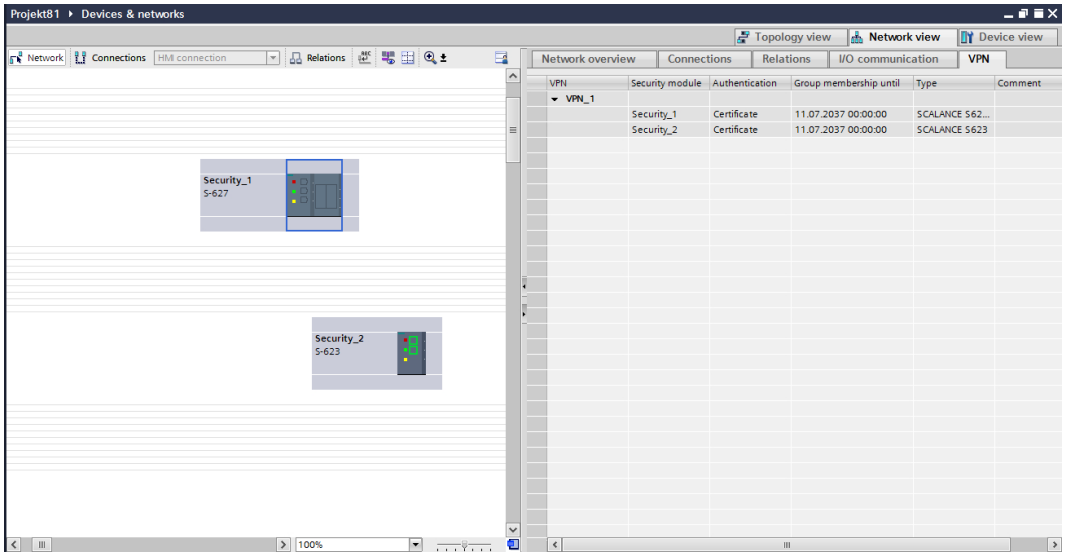
When using secure communication (for example, HTTPS, VPN...), make sure that the security modules involved have the current time of day and the current date. Otherwise the certificates used are not evaluated as valid and the secure communication does not work.

How to access this function

1. Double-click on the entry "VPN groups" > "Add new VPN group" in the global security settings to create a VPN group. As an alternative, in the network view you can right click on an interface of a module with VPN capability and create a VPN group with the shortcut menu command "Add new VPN group" (not possible for SOFTNET Security Client).
2. Double-click on the entry "VPN groups" > "Assign module to a VPN group" in the global security settings and assign the security modules and SOFTNET security client modules to the VPN group between which VPN tunnels will be established. As an alternative, in the network view you can right click on an interface of a module with VPN capability and assign this to the VPN group using the shortcut menu "Assign module to a VPN group" (not possible for SOFTNET Security Client).
   When doing this, remember the rules for forming VPN groups. You will also find these rules in the section:
   Modes of VPN groups (Page 747).

Display of the VPN groups with their properties

If you select a security module that is in one or more VPN group(s), the properties of the VPN group(s) of which the security module is a member are shown in the "Network data" area.



The following properties of the VPN groups are displayed in columns in the "VPN" tab of the "Network data" area:

Property/column	Meaning
VPN	Names of the VPN groups of which the selected security module is member
Security module	Names of the assigned security modules
Authentication	Type of authentication: Preshared key or certificate
Group membership until	Date and time up to which the VPN group certificate of the security module is valid
Type	Model numbers of the assigned security modules
Comment	Comment

Setting the life of certificates

Open the dialog in which you can set the expiry date of the certificate as follows:

1. Select the VPN group you want to edit in the "VPN" tab.
2. In the "Properties" > "General" tab of the Inspector window, select the entry "Authentication".

Note

Expiry of a certificate

Communication through the VPN tunnel continues after the certificate has expired until the tunnel is terminated or the SA lifetime expires. For more information on certificates, refer to section:
Auto-Hotspot.

Configuration limits

Number of IPSec tunnels	
SCALANCE S612 V2	Maximum of 64
SCALANCE S612 V3/V4	Maximum of 128
SCALANCE S613	Maximum of 128
SCALANCE S623 V3/V4	Maximum of 128
SCALANCE S627-2M V4	Maximum of 128
CP x43-1 Advanced	Maximum of 32
CP 1628	Maximum of 64
CP 1543-1 V1.1	Maximum of 16
CP 1243-1	Maximum of 16
CP 1243-7	Maximum of 16

Authentication methods

The following methods are available:

The authentication method is specified per VPN group and decides the type of authentication used.

The following key-based or certificate-based authentication methods are supported:

- Preshared key
  Authentication is achieved using a previously specified character string that is distributed to all modules in the group.
  Enter a preshared key in the "Key" field under "Authentication" > "General" in the VPN group properties dialog.
- Certificate
  Certificate-based authentication "Certificate" is the default setting. The procedure is as follows:
  - When a group is created, a CA certificate is generated as a root certificate.
  - Each security module that is a member of the VPN group also receives a group certificate signed with the key of the CA certificate.

  All certificates are based on the ITU standard X.509v3 (ITU, International Telecommunications Union).
  The certificates are generated by a certificate authority contained in STEP 7.

Note

Restriction in VLAN operation

No VLAN tagging is transmitted in IP frames via the VPN tunnel of the security module. The VLAN tags included in IP frames are lost when they pass through the security modules because IPsec is used to transfer the IP frames.

As default, no IP broadcast or IP multicast frames can be transferred with IPsec through a layer 3 VPN tunnel. Through a layer 2 VPN tunnel of the security module, IP broadcast or IP multicast frames are packaged just like MAC packets including the Ethernet header in UDP and transferred. With these packets, the VLAN tagging is therefore retained.

Group properties for selected VPN group

VPN group properties

Note

Knowledge of IPsec necessary

To be able to set these parameters, you require IPsec experience. If you do not make or modify any settings, the defaults apply.

The following settings can be configured in the properties of a VPN group:

- Authentication method (entry: "General")
- IKE settings (entry: "Advanced settings phase 1")
- IPsec settings (entry: "Advanced settings phase 2")

How to access this function

1. In the global security settings select the VPN group whose properties you want to configure under the entry "VPN groups".
2. Select "Open" in the shortcut menu of this entry.
   Result: The VPN group properties appear in the area of the local security settings.
3. In the "Authentication" entry, select whether to use a preshared key or certificate for authentication. For more detailed information, refer to section:
   Authentication methods (Page 744).

Parameters for advanced settings phase 1

Phase 1: IKE negotiation of the Security Association (SA) for phase 2:

Here you set the parameters for negotiating the security parameters to be used in phase 2:

Parameter	Description
IKE mode	• Main mode • Aggressive mode The difference between the main and aggressive mode is the "identity protection" used in the main mode. The identity is transferred encrypted in main mode but not in aggressive mode.
DH group phase 1	Groups selectable for the Diffie-Hellman Key exchange: • Group 1 • Group 2 • Group 5 • Group 14
SA lifetime type	Phase 1 Security Association (SA): • Time: Time limit in minutes The lifetime of the current key material is limited in time. When the time expires, the key material is renegotiated.
SA lifetime	Numeric value: Range of values for time: 1440 ... 2500000 minutes (default: 2500000)
Phase 1 encryption	Encryption algorithm: • DES*: Data Encryption Standard (56 bit key length, mode CBC) • 3DES-168: Triple DES (168-bit key length, mode CBC) • AES-128, 192, 256: Advanced Encryption Standard (128 bit, 192 bit or 256 bit key length, mode CBC)
Phase 1 authentication	Authentication algorithm: • MD5: Message Digest Algorithm 5 • SHA1: Secure Hash Algorithm 1

*DES is an insecure encryption algorithm. It should only be used for reasons of down compatibility. DES is not supported by CP 1543-1 V1.1, CP 1243-1, CP 1242-7 and CP 1243-7.

Parameters for advanced settings phase 2

Phase 2: IKE negotiation of the Security Association (SA) for IPsec data exchange:

Here, you set the parameters for negotiating the security parameters used for the IPsec data exchange with ESP (Encapsulating Security Payload) and AH (Authentication Header). Communication in phase 2 is already encrypted.

Parameter	Description
SA lifetime type	Phase 2 Security Association (SA): • Time: Time limit in minutes. The lifetime of the current key material is limited in time. When the time expires, the key material is renegotiated. • Limit: Limitation of the data volume in MB
SA lifetime	Numeric value: • Range of values for time: 60 ... 16666666 minutes (default: 2880) • Range of values for limit: 2000 ... 500000 MB (default: 4000)
Phase 2 encryption	Encryption algorithm: • DES*: Data Encryption Standard (56 bit key length, mode CBC) • 3DES-168: Triple DES (168-bit key length, mode CBC) • AES-128: Advanced Encryption Standard (128-bit key length, mode CBC)
Phase 2 authentication	Authentication algorithm: • MD5: Message Digest Algorithm 5 • SHA1: Secure Hash Algorithm 1
Perfect Forward Secrecy	If you enable this check box, new Diffie-Hellmann public key values are exchanged for recalculation of the keys. If the check box is disabled, the values already exchanged in phase 1 are used for recalculation of the keys.

*DES is an insecure encryption algorithm. It should only be used for reasons of down compatibility. DES is not supported by CP 1543-1 V1.1, CP 1243-1, CP 1242-7 and CP 1243-7.

Modes of VPN groups

VPN modes

Depending on the mode of the security modules that were added to a VPN group, different modes of VPN groups are distinguished. The mode of a VPN group provides information about the security modules and their modes that can be added to the VPN group.

Rules for forming groups

Remember the following rules if you want to create VPN groups:

- For SCALANCE S612 / S613 / S623 / S627-2M / SCALANCE M
  The first module assigned in a VPN group decides which other modules can be added to it.
  If the first SCALANCE S module to be added is in routing mode or if the first security module is a SCALANCE M module, only additional SCALANCE S modules with routing activated or SCALANCE M modules can be added because SCALANCE M modules always operate in routing mode. If the first SCALANCE S module added is in bridge mode, you can only add other SCALANCE S modules in bridge mode. If you want to change the mode of a VPN group, you have to remove all the modules contained in the group and add them again. A CP and an SSC can be added to a group with a SCALANCE S in bridge or routing mode.
- For CP / SSC
  If a CP / SSC is the first module in a VPN group, security modules can be added in any mode. The next security module that decides the mode then also decides the mode of the VPN group. A CP / SSC can be assigned to several VPN groups at the same time and use different modes. The CP / SSC is then operated in mixed mode.
- It is not possible to add a SCALANCE M module to a VPN group that contains a module in bridge mode.

Refer to the following table to see which modules can grouped together in a VPN group:

Table 10-27 Security modules and VPN modes

Module	Can be included in a VPN group in...	
	Bridge mode	Routing mode
SCALANCE S612 / S613 / S623 / S627-2M in bridge mode	x	-*
SCALANCE S612 / S613 / S623 / S627-2M in routing mode	-	x
CP x43-1 Adv.	x	x
CP 1543-1 V1.1	x	x
CP 1243-1	x	x
CP 1243-7	x	x
CP 1628	x	x
SOFTNET Security Client V4.0	x	x
SCALANCE M875	-	x

* SCALANCE S623/S627-2M modules in bridge mode can be inserted in a VPN group in routing mode if their DMZ interface is activated (not at the same time).

Including security module in configured VPN group

The configured VPN group properties are adopted for security modules added to in an existing VPN group.

Procedure after including a security module in a configured VPN group

Depending on whether or not you have changed the VPN group properties since the last download, you must make a distinction between the following:

- **Case a:** If you have not changed the VPN group properties and the module to be added actively establishes the connection to modules of the type SCALANCE S, CP x43-1 Adv. or CP 1628:

1. Add the new security module to the VPN group.
2. Download the configuration to the new module.

- **Case b:**If you have changed the VPN group properties or the module being added does not actively establish the connection to the already configured modules:

1. Add the new security module to the VPN group.
2. Download the configuration to all modules that belong to the VPN group.

In case a, it is not necessary to reconfigure and load the already commissioned security modules. Active communication is not influenced or interrupted.

Settings for nodes with an unknown IP address

Nodes whose IP address is unknown at the time of configuration (Unknown Peers) can be inserted in an existing VPN group. Since the nodes are usually mobile and obtain their IP addresses dynamically (for example a SOFTNET security client or SCALANCE M), the VPN tunnel from an Unknown Peer to SCALANCE S, CP x43-1 Adv. and CP 1628 can only be established if you set the parameters for Phase 1 according to one of the following tables ("Encryption parameter 1" to "Encryption parameter 4"). If you use other settings, you cannot establish a VPN tunnel to the modules listed. For establishing a VPN tunnel to CP 1543-1 V1.1, CP 1243-1 and CP 1243-7 there are no restrictions relating to this.

Table 10-28 Encryption parameter 1

Parameter	Setting
Authentication method	Certificate
DH group phase 1	Group 2
SA lifetime	1440 … 2500000 minutes
Encryption phase 1	AES-256
Authentication phase 1	SHA-1

Table 10-29 Encryption parameter 2

Parameter	Setting
Authentication method	Certificate
DH group phase 1	Group 2
SA lifetime	1440 … 2500000 minutes
Encryption phase 1	3DES-168
Authentication phase 1	SHA-1

Table 10-30 Encryption parameter 3

Parameter	Setting
Authentication method	Certificate
DH group phase 1	Group 2
SA lifetime	1440 … 2500000 minutes
Encryption phase 1	DES
Authentication phase 1	MD5

Table 10-31 Encryption parameter 4

Parameter	Setting
Authentication method	Preshared key
DH group phase 1	Group 2
SA lifetime	1440 … 2500000 minutes
Encryption phase 1	3DES-168
Authentication phase 1	SHA1

Additional restrictions for the SOFTNET Security Client

For the SOFTNET security client, the following restrictions also apply:

Table 10-32 Encryption parameters for the SOFTNET Security Client

Parameter	Setting / special feature
Encryption phase 1	AES-256 possible only with Windows 7
SA lifetime phase 1	1440 to 2879 minutes
SA lifetime type	Must be selected identical for both phases
Encryption phase 2	No AES 128 possible
SA lifetime phase 2	60 to 2879 minutes
Authentication phase 2	No MD5 possible

Procedure after removing an active member from a VPN group

If you remove an active node from an existing VPN group, this can still establish a connection to the group members even if you have downloaded the project to all members of the VPN group again.

If you do not want the removed active node to be able to establish the connection any longer, renew the CA group certificate and download the project again to the members of the VPN group. The certificate can be renewed in the group properties of the VPN group or in the "CA" tab of Certificate Manager.

Configuring internal network nodes

Overview of configuring internal network nodes

Module-specific function

This function is available only for SCALANCE S612/S613/S623/S627-2M, CP x43-1 Adv. and CP 1628.

Configuring internal network nodes

Each security module must know the network nodes in the entire internal network to be able to recognize the authenticity of a frame.

The security module must know both its own internal nodes as well as the internal nodes of the security modules in the same VPN group. This information is used on a security module to decide which data packet will be transferred in which tunnel.

By adding a security module to a VPN group, the local internal nodes/subnets of the security module are automatically made known to the security module. To allow communication via the VPN tunnel with other subnets or nodes of another subnet (routed internal network; DMZ network when the VPN tunnel is on the external interface and vice versa), these subnets or nodes must be enabled for VPN tunnel communication in the configuration.

SCALANCE S modules allow network nodes to be learned automatically or configured statically. The mode of the security module also decides the options available for learning internal network nodes.

SCALANCE S in bridge mode

In bridge mode, you can configure the internal IP/MAC nodes and the internal subnets or alternatively can allow automatic learning of internal nodes by the SCALANCE S.

SCALANCE S in routing mode

In routing mode there is no automatic learning mode available. Instead, enter entire subnets here that need to be released for tunnel communication.

CP x43-1 Advanced and CP 1628

- CP x43-1 Adv.
  Decide whether or not tunnel communication to the CP (Gbit interface) and/or to the internal subnet (PROFINET subnet) is permitted for VPN connection partners in routing mode (SCALANCE S / M).
- CP 1628
  Enter the NDIS nodes you want to be reachable through the tunnel of VPN connection partners in routing mode (SCALANCE S / M).

Automatic learning of internal network nodes

Module-specific function

SCALANCE S modules in bridge mode provide a learning mode with which the internal network nodes can be learned automatically during operation. For more detailed information, refer to subsection:
Using the learning mode to learn internal nodes (Page 809) in the section "SCALANCE S".

Configuring IP network nodes manually for SCALANCE S

Module-specific function

How to configure IP network nodes for SCALANCE S modules manually is explained in:
Configuring IP network nodes manually (Page 810) in the section "SCALANCE S".

Configuring MAC network nodes manually for SCALANCE S

Module-specific function

How to configure MAC network nodes for SCALANCE S modules manually is explained in Configuring MAC network nodes manually (Page 811) in the section "SCALANCE S".

Configuring internal subnets for SCALANCE S manually

Module-specific function

How to configure the internal subnets for SCALANCE S modules is explained in Configuring internal subnets manually (Page 811) in the section "SCALANCE S".

Allow access to S7-300 / S7-400 CPs for VPN connection partners

Module-specific function

How to allow access to S7-300 / S7-400 CPs for VPN connection partners is explained in Allow access to S7-300 / S7-400 CPs for VPN connection partners (Page 836) in the section "Security for S7-300 /S7-400 /PC CPs".

Configuring NDIS nodes for PC CPs that can be reached through the tunnel

Module-specific function

How you configure NDIS nodes for PC CPs that can be reached through the tunnel is explained in
Configuring NDIS nodes manually for PC CPs that can be reached through the tunnel (Page 836) in the section "Security for S7-300 / S7-400 / PC CPs".

Configuring module- and connection-specific VPN settings

Requirement

The module is a member of a VPN group.

Module- and connection-specific settings

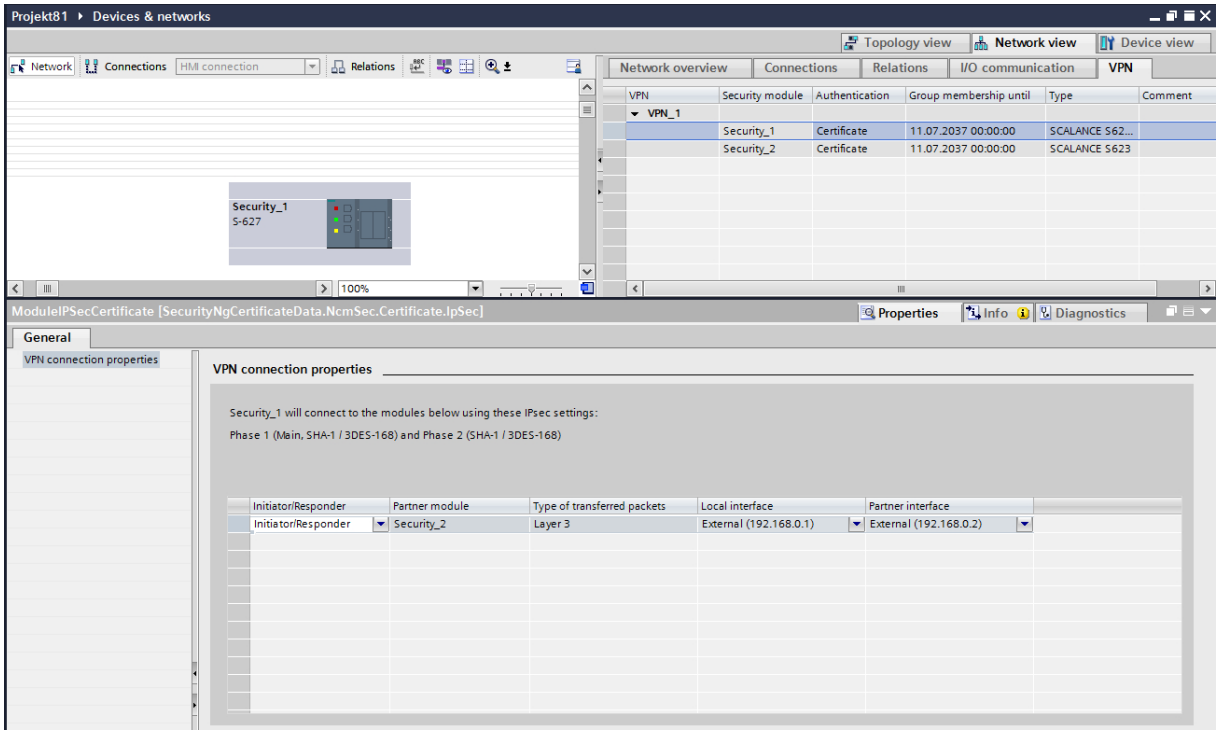
With module- and connection-specific settings, you can configure specific VPN settings. Module-specific settings are configured specifically for a security module while connection-specific settings are configured specifically for a security module in a certain VPN group.

You can configure the following **module-specific** properties in the local security settings with the "VPN" entry:

- Dead peer detection
- Permission to initiate connection establishment
- Public address (IP address / FQDN) for communication via Internet gateways
- Nodes that need to be enabled for tunnel communication

If you select a security module in the list of VPN groups in the "Network data" area, the following **connection-specific** VPN settings can be seen and can be configured:

- Permission to initiate connection establishment
- Partner modules to which tunnel connections exist
- Type of transferred packets
- Selection of the local interface of the selected security module that will serve as the tunnel endpoint
- Selection of the partner interface that will serve as the tunnel endpoint



Dead peer detection (DPD)

As default, DPD is enabled.

When DPD is enabled, the modules exchange additional messages at selectable intervals if no communication occurs at these points in time. This means that it is possible to recognize whether the IPsec connection is still valid or possibly needs to be re-established. If there is no longer a connection, the security associations (SA) of phase 2 are terminated prematurely. If DPD is disabled, the SA is ended only after the SA lifetime has expired. For information on setting the SA lifetime, see section
Group properties for selected VPN group (Page 745).

Permission to initiate connection establishment

You can restrict the permission for initiating the VPN connection establishment to certain modules in the VPN.

The decisive factor for the setting of the parameter described is the assignment of the address for the gateway of the module you are configuring. If a static IP address is assigned, the module

can be found by the partner. If the IP address is assigned dynamically and therefore changes constantly, the partner cannot establish a connection as things stand.

Mode	Meaning
Start connection to partner (initiator/responder) (default)	If this option is selected, the module is "active", in other words, it attempts to establish a connection to the partner with a fixed IP address. The reception of requests for VPN connection establishment is also possible. This option is recommended when you obtain a dynamic IP address from the provider for the gateway of the security module you are configuring. The partner is addressed using its configured WAN IP address, its configured external module IP address or the configured DNS name.
Wait for partner (responder)	If this option is selected, the module is "passive", in other words, it waits for the partner to initiate the connection. This option is recommended when you have been assigned a static IP address by the provider for the gateway of the security module you are configuring. It means attempts to establish a connection can only be initiated by the partner. This partner can, for example, have a dynamic WAN IP address.

Note

Make sure that you do not set all the modules in a VPN group to "Wait for connection from remote VPN gateway" otherwise no connection is established.

WAN IP address / FQDN - addresses of the modules and gateways in a VPN over Internet

When operating a VPN with IPsec tunnel over the Internet, additional IP addresses are generally required for the Internet gateways such as DSL routers. The individual security or SCALANCE M modules must know the external IP addresses of the partner modules in the VPN.

Note

To use a WAN as an external public network, enter the IP address that you received from the provider as the external IP address, through which the security module is then reachable in the WAN (Internet). To allow the security module to send packets via the WAN, you need to enter your DSL router as "Standard router".

If you use a DSL router as Internet gateway, the following ports (at least) must be opened on it as described in the relevant documentation:

- Port 500 (ISAKMP)
- Port 4500 (NAT-T)

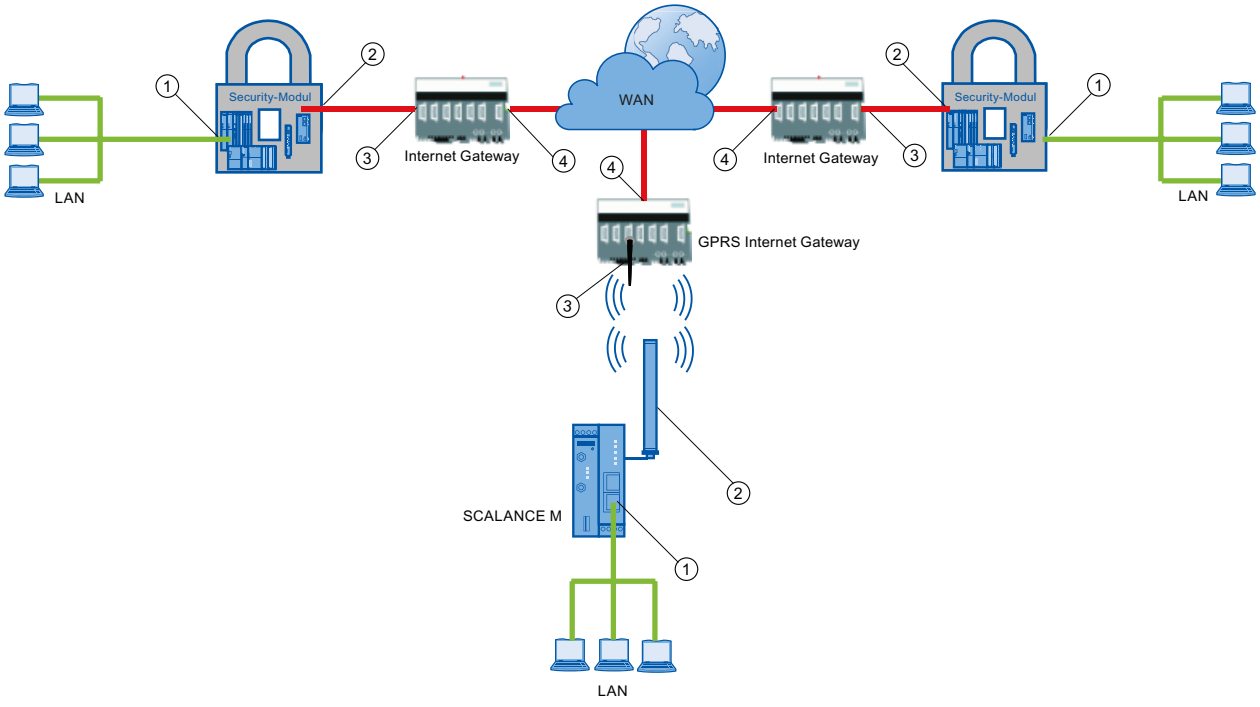
To allow this, in the module-specific VPN settings, you have the option of assigning an IP address as a "WAN IP address". When you download the module configuration, the group members are then informed of the WAN IP addresses of the partner modules. As an alternative

to a WAN IP address, you can also enter an FQDN. Depending on the existing addresses, VPN endpoints are used as default according to the following priorities:

1. WAN address
2. FQDN of the primary dynamic DNS service
3. FQDN of the secondary dynamic DNS service
4. External IP address / DMZ IP address of the security module

Note: After removing an existing WAN address, the external IP address / DMZ IP address is always used.

In the interface selection of the connection-specific VPN settings, you yourself can decide which address the partner will be informed of. In these settings, you can also specify the interface to be used for communication by the nodes of a VPN group and the security module that is authorized to set up connections.



① Internal IP address - of a module
② External IP address - of a module
③ IP address of an Internet gateway (for example, GPRS gateway)
④ IP address (WAN IP address) of an Internet gateway (for example, DSL router)

Configuring internal network nodes

The configuration of internal network nodes is described in the following section: Auto-Hotspot

See also

How to create an IPsec tunnel with VPN groups (Page 742)

Configuring router and firewall redundancy

Module-specific function

This function is available only for SCALANCE S623/S627-2M as of V4, see section Auto-Hotspot.

Online functions - diagnostics and logging

Overview of diagnostics and logging in

For test and monitoring purposes, the security module has diagnostic and logging functions.

- Diagnostic functions
  These include various system and status functions that you can use for an existing network connection to the security module.
- Logging functions
  This involves the logging of system and security events and data packets.

Recording events with logging functions

You select the events to be logged in the log settings for the relevant security module.

You can configure the following variants for logging:

- Local logging
  In this variant, you log events in the local buffer of the security module. You can then access these logs, display them and archive them on the service station in the "Online & diagnostics" dialog. The evaluation of the buffer areas of the security module is only possible if there is a network connection to the selected security module.
- Network Syslog
  With Network Syslog, you use a Syslog server that exists in the network. This logs the events according to the configuration in the log settings for the relevant security module.

Archiving log data and reading in from a file

You can save the logged events for archiving in a log file and open this later even without a network connection to the security module. For more detailed information, refer to section Auto-Hotspot.

Diagnostics in ghost mode (only for SCALANCE S S602 as of V3.1)

When you operate the security module in ghost mode, the external interface of the security module takes over the IP address of the internal node at runtime. Before you can run diagnostics of a security module in ghost mode, you need to establish the connection to the security module via the IP address that the security module obtained from the internal node at runtime.

To find out which IP address the security module currently has, you can search for accessible nodes in STEP 7 using the "Online" > "Accessible devices".

Protecting exported log files from unauthorized access

Log files exported from STEP 7 can contain security related information. You should therefore make sure that these files are protected from unauthorized access. This is particularly important when passing on the files.

Function overview online dialog

Functions of the "Online & diagnostics" dialog

In STEP 7, the security module provides the following functions in the "Online & diagnostics" dialog:

Entry in the online dialog	Function
Status	Display of status information about the selected security module, such as the current IP addresses of the interfaces and the current time and date.
Interface settings (only for SCALANCE S V3 or higher)	Overview of the settings of the individual interfaces.
Dynamic DNS (only for SCALANCE S as of V3)	Overview of the settings for dynamic DNS.
System log (not for CP 1242-7)	Display of logged system events, starting and stopping logging (only if there is an online connection to SCALANCE S modules) and starting and stopping the reading out of log data from the local buffer of the security module.
Audit log (not for CP 1242-7)	Display of logged security events, starting and stopping reading out of log data from the local buffer of the security module.
Packet filter log (not for CP 1242-7 and CP 1243-7)	Display of logged data packets, starting and stopping logging (only if there is an online connection to SCALANCE S modules) and starting and stopping the reading out of log data from the local buffer of the security module.
ARP table (only for SCALANCE S V3 or higher)	Display of the ARP table of the security module.
Logged in users (only for SCALANCE S V3 or higher)	Shows the users logged in to the Internet page for user-specific IP rule sets.
Communication status (not for SCALANCE S602, CP 1242-7 and CP 1543-1 V1.0)	Display of status information about VPN tunnel connections and members of VPN groups to which the selected security module belongs.
Internal nodes (not for SCALANCE S602, CP 1243-1, CP 1242-7, CP 1243-7, CP 1543-1 and PC-CPs)	Display of the learned or configured internal network nodes of the security module.
Dynamically updated firewall rules (only for CP x43-1 Adv.)	Display of the IP addresses that were released dynamically over HTTP or HTTPS, or loaded by a user.

Entry in the online dialog	Function
Firewall blacklist (only for SCALANCE S as of V4)	Display of the IP addresses that were entered in the blacklist of the firewall.
Ghost mode (only for SCALANCE S S602 as of V3.1)	Dialog for the ghost mode of the SCALANCE S602 with information on the address parameters of the internal node (identical to the external IP address of the security module) and on IP address changes of the internal node.
Log files (offline view)	
• System log (not for CP 1242-7)	Display of logged system events as well as starting and stopping the display.
• Audit log (not for CP 1242-7)	Display of logged security events as well as starting and stopping the display.
• Packet filter log (not for CP 1242-7 and CP 1243-7)	Display of logged data packets as well as starting and stopping the display.
Date and time (only for SCALANCE S)	Date and time setting.
Firmware update	Updating the firmware.

Requirements for access

The following requirements must be met to use the online functions of a security module:

- There is a network connection to the selected module
- The project with which the module was configured is open
- a user with the required rights must be logged in to the project
- For CPs the diagnostics access must be opened in the firewall (CP x43-1 and CP 1628: TCP 443; CP 1243-1, CP 1242-7, CP 1243-7 and CP 1543-1: TCP 8448)

Note

Requirement for online diagnostics in ghost mode (only for SCALANCE S S602 as of V3.1)

Online diagnostics is only available in ghost mode if the security module has learned the IP address of the internal node and has adopted this for its external interface. After this, the security module can be reached via the IP address of the external interface.

How to access this function

1. Right-click on the module to process.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. If there is no online connection established to the security module, click the "Connect online" button in the "Diagnostics" entry.

Online settings are not saved in the configuration

Settings that you make in online mode (for example settings for the logging memory) are not stored in the configuration on the security module. This is why the configuration settings are always applied at the restart of the module.

Functions of the online diagnostics

Status information of the security module - "Status" entry

Meaning

Display of the status of the security module selected in the project.

Table 10-33 Online & diagnostics: - "Status" entry

System and status functions	Meaning
Overview	
Hardware type	The type of the security module.
External IP address	The external IP address of the security module. For CP 1243-1, CP 1242-7, CP 1243-7, CP 1543-1, CP 1628: The IP address of the Industrial Ethernet interface. For CP x43-1 Adv.: The IP address of the Gbit interface.
Internal IP address	The internal IP address of the security module. For CP 1628: The IP address of the NDIS interface. If there is more than one NDIS address, only one is displayed. For CP x43-1 Adv.: The IP address of the PROFINET interface.
DMZ IP address (SCALANCE S623 / S627-2M only)	The DMZ IP address of the security module.
Tunnel IP address (only SCALANCE S612/S623/ S627-2M as of V4)	The first alias internal IP address of the security module in the VPN tunnel.
Serial number	The serial number of the security module.
Order number	The MLFB identifier of the security module that is used when ordering.
Firmware version	The firmware version of the security module.
Operating mode	Mode of the security module (bridge mode / routing mode)
External MAC address	The external MAC address of the security module. For CP 1243-1, CP 1242-7, CP 1243-7, CP 1543-1, CP 1628: The MAC address of the Industrial Ethernet interface. For CP x43-1 Adv.: The MAC address of the gigabit interface.
Internal MAC address	The internal MAC address of the security module. For CP 1628: The MAC address of the NDIS interface. For CP x43-1 Adv.: The MAC address of the PROFINET interface.
DMZ MAC address (SCALANCE S623 / S627-2M only)	The DMZ MAC address of the security module

System and status functions	Meaning
Hardware release	The hardware product version of the security module.
C-PLUG	Shows whether or not a C-PLUG is inserted.
Alias IP addresses (only for SCALANCE S as of V4)	
IP address	Alias IP address that was registered on an interface of the security module by a NAT rule.
Corresponding Interface	Interface of the security module on which the alias IP address was registered.
Local time	
Current time	Date and time that is displayed on the security module. Format for "German" user interface language: dd.mm.yyyy (date) hh:mm:ss (time) Format for "English" user interface language: mm/dd/yyyy (date) hh:mm:ss AM/PM (time) Format for "French", "Italian" and "Spanish" user interface language: dd/mm/yyyy (date) hh:mm:ss (time) Format for "Chinese" user interface language: yyyy/mm/dd (date) hh:mm:ss **Note (not for CPs)** You set the local time on the SCALANCE S module in "Functions" > "Date and time".
Operating time	Time since the last restart of the security module. Format with the user interface language "German", "English", "French", "Italian", "Spanish" and "Chinese": dddd.hh:mm:ss
Time-of-day source	The source from which the date and time are obtained.
Configuration	
Created	Date and time when the project was first created. Format with the user interface language "German", "English", "French", "Italian", "Spanish" and "Chinese": dd.mm.yyyy (date) hh:mm:ss (time)
Name	File name of the project last loaded.
Author	Name of the user who created the project. Is adopted from the project properties.
Loaded	Date and time when the project was last loaded on the security module. Format with the user interface language "German", "English", "French", "Italian", "Spanish" and "Chinese": dd.mm.yyyy (date) hh:mm:ss (time)
Storage location	Specifies the location (e.g. town) that was entered in the properties of an SCT project.
File system (not for CPs)	
RAM	Indicates how much RAM and flash is occupied in the file system.
Flash	

Overview of the individual interfaces - "Interface settings" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Overview of the individual interfaces - "Interface settings" entry (Page 816)

Overview of the Dyn. DNS settings - "Dynamic DNS" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Overview of the Dyn. DNS settings - "Dynamic DNS" entry (Page 818)

Display of the ARP table - "ARP table" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Display of the ARP table - "ARP table" entry (Page 820)

Users logged in to the Web page - "Logged in users" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Users logged in to the Web page - "Logged in users" entry (Page 820)

VPN connections of the security module - "Communication status" entry

Module-specific function

This function is available only for SCALANCE S612/S613/S623/S627-2M, CP x43-1 Adv., CP 1243-1, CP 1243-7, CP 1543-1 V1.1 and CP 1628.

Meaning

Display of the communication status of the following network components:

- Other security modules of the VPN group to which the selected security module belongs
- Internal network nodes of these security modules

Table 10-34 Online & diagnostics: "Communication Status" entry

System and status functions	Meaning
Known security devices or modules	Display of the nodes with which the selected security module is in a VPN group. This also shows whether the tunnel status is active or passive. To obtain additional information on one of the nodes, select this in the list. **Note:** Configured, inactive tunnels are indicated for CPs only.
Endpoints	Display of information on the internal network nodes of the security module that you selected in the "Known security devices or modules" table. For each internal network node, this shows whether it was learned or configured. It also shows the subnet in which the internal network node is located. With SCALANCE S modules, the subnet of network nodes is only displayed in bridge mode.
Tunnel properties	Display of properties of the VPN tunnel established to the security module that you selected in the "Known security devices or modules" table.

Found internal network nodes - "Internal nodes" entry

Module-specific function

This function is available only for SCALANCE S612/S613/S623/S627-2M and CP x43-1 Adv.

Meaning

Display of all learned and configured network nodes. This also displays whether or not the learning mode of the security module is enabled.

Updated firewall rules - "Dynamically updated firewall rules" entry

Module-specific function

This function is only available for CP x43-1, see subsection:
Updated firewall rules - "Dynamically updated firewall rules" entry (Page 836) in the section "Security for S7-300 / S7-400 / PC CPs".

Display of the firewall blacklist - "Firewall blacklist" entry

Module-specific function

This function is available only for SCALANCE S V4 modules or higher, see section
Display of the firewall blacklist - "Firewall blacklist" entry (Page 821).

Setting the date and time - "Date and Time" entry

Module-specific function

This function is available only for SCALANCE S, see section Setting the date and time - "Date and time" entry (Page 821).

Diagnostics in ghost mode - "Ghost mode" entry

Module-specific function

This function is available only for SCALANCE S602 as of V3.1, see section Diagnostics in ghost mode - "Ghost mode" entry (Page 821).

Logging functions

Logging system events - "System log" entry

Module-specific function

This function is available only for SCALANCE S, CP x43-1 Adv., CP 1243-1, CP 1243-7, CP 1543-1 and CP 1628.

Meaning

Display of logged system events and starting and stopping reading system events from the local memory of the security module.

The system log automatically logs successive system events, for example the start of a process. The logging can be scaled based on event classes.

System and status functions	Meaning
Start/stop logging (not for CPs)	Starts/stops recording of system events. The method and the event classes that are logged are configured in the local security settings.
Start/stop reading	Starts/stops reading of system events from the local memory of the security module. If you select the "Save log file" check box, the recorded log data is also saved as file. Select the storage location and enter a file name. **Note** If you select the "Save log file" check box after starting reading, the data read out up to then can no longer be saved in a log file.
Clear display	Deletes the log data shown in the table.

For more information on opening saved system events in log files, refer to section Evaluating system events in offline mode - "System log" entry (offline view) (Page 766).

Logging security events - "Audit log" entry

Module-specific function

This function is available only for SCALANCE S, CP x43-1 Adv., CP 1243-1, CP 1243-7, CP 1543-1 and CP 1628.

Meaning

Display of logged security events and starting and stopping reading of security events from the local memory of the security module.

The audit log automatically logs successive security-relevant events. This includes, for example, user actions such as enabling and disabling packet logging.

System and status functions	Meaning
Start/stop reading	Starts/stops reading of security events from the local memory of the security module. If you select the "Save log file" check box, the recorded log data is also saved as file. Select the storage location and enter a file name. **Note** If you select the "Save log file" check box after starting reading, the data read out up to then can no longer be saved in a log file.
Clear display	Deletes the log data shown in the table.

For more information on opening security events stored in log files, refer to section Evaluating security events in offline mode - "Audit log" entry (offline view) (Page 767).

Logging data packets - "Packet filter log" entry

Module-specific function

This function is available only for SCALANCE S, CP x43-1 Adv., CP 1243-1, CP 1543-1 and CP 1628.

Meaning

Display of logged data packets and starting and stopping reading of packet filter events.

The packet filter log records certain packets of the data traffic. Data packets are only logged if they match a configured packet filter rule (firewall) or to which the basic protection reacts (corrupt or invalid packets). This is only possible when logging is enabled for the packet filter rule.

For information about activation of the logging, refer to section Auto-Hotspot.

As well as reading the log data from the buffer and transferring it to the display, it can also be saved in a file for archiving.

System and status functions	Meaning
Start/stop logging (not for CPs)	Starts/stops logging of data packets. The method with which the data is logged is configured in the local security settings.
Start/stop reading	Starts/stops reading out of logged data packets from the local memory of the security module. If you select the "Save log file" check box, the recorded log data is also saved as file. Select the storage location and enter a file name. **Note** If you select the "Save log file" check box after starting reading, the data read out up to then can no longer be saved in a log file.
Clear display	Deletes the log data shown in the table.
Log category	Select the data packets for which the logging will be displayed. The selection depends on the settings you configured offline in the local security settings. Only the data packets for which logging was enabled are logged. If you select a category for which logging was not enabled, no data will be logged for this category.

For information on opening the stored packet filter log data, refer to section Evaluating packet filter events in offline mode - "Packet filter log" entry (offline view) (Page 767).

Evaluating log files in offline mode

Evaluating system events in offline mode - "System log" entry (offline view)

Module-specific function

This function is available only for SCALANCE S, CP x43-1 Adv., CP 1243-1, CP 1243-7, CP 1543-1 and CP 1628.

How to access this function

1. Right-click on the module to process.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. Select "Diagnostics" > "Log files (offline view)" > "System log".

Meaning

Opens logged system events that you saved as a file in the online view.

For more information, refer to chapter Logging system events - "System log" entry (Page 764).

Evaluating security events in offline mode - "Audit log" entry (offline view)

Module-specific function

This function is available only for SCALANCE S, CP x43-1 Adv., CP 1243-1, CP 1243-7, CP 1543-1 and CP 1628.

How to access this function

1. Right-click on the module to process.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. Select "Diagnostics" > "Log files (offline view)" > "Audit log".

Meaning

Opens logged security events that you saved as a file in the online view.

For more detailed information, refer to section Logging security events - "Audit log" entry (Page 765).

Evaluating packet filter events in offline mode - "Packet filter log" entry (offline view)

Module-specific function

This function is available only for SCALANCE S, CP x43-1 Adv., CP 1243-1, CP 1543-1 and CP 1628.

How to access this function

1. Right-click on the module to process.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. Select "Diagnostics" > "Log files (offline view)" > "Packet filter log".

Meaning

Opens logged data packets that you saved as a file in the online view.

For more detailed information, refer to section Logging data packets - "Packet filter log" entry (Page 765).

Download functions

Special features when downloading security configurations

Security configurations can influence the reachability of the security module for the configuration PC. This, for example, is the case if there is a tunnel connection configured for one security module to another security module in a configuration and this configuration is downloaded from the configuration PC to the security module. After the download by the configuration PC, the security module can no longer be reached and the reachability test performed as default by STEP 7 following the download of the configuration fails. The error message output by STEP 7 relates solely to the reachability test; the actual download of the configuration is ensured if the project data is consistent and the IP address relationship between the security module and the configuration PC is correct.

Special features when downloading configurations and firmware to SCALANCE S modules are described in the following subsection of the section "SCALANCE S":
Auto-Hotspot

Uploading configurations to the engineering station

Uploading a configuration from a SCALANCE S module or from the CP 1628 to an engineering station is not possible.

Uploading a configuration from an S7 CP that supports security to an engineering station is also possible if security features were included in the configuration. Configured security functions are not, however, transferred to the engineering station. In the transferred configuration on the engineering station, the "Activate security features" check box is also deselected.

SOFTNET Security Client

Using the SOFTNET Security Client

Area of application - access over VPN

With the SOFTNET Security Client (SSC) PC software, secure remote access is possible from PGs/PCs to automation systems protected by a security module via public networks. You require SOFTNET Security Client V4.0 HF1 for S7-300/S7-400 CPs and for the PC CP 1628. These CPs are not approved for operation with SOFTNET Security Client ≤ V4.0. The SOFTNET Security Client has not been released for other CPs.

With the SOFTNET Security Client, a PG/PC is configured automatically so that it can establish a secure IPsec tunnel communication in the VPN (Virtual Private Network) with one or more security modules.

This IPsec tunnel communication makes it possible for PG/PC applications such as NCM diagnostics to securely access devices or networks that are located in an internal network protected by the security module.

How does the SOFTNET Security Client work?

The SOFTNET Security Client reads the configuration that was created by STEP 7 and, if necessary, determines from the file which certificates are to be imported.

The CA certificate and the private key are imported when applicable and stored on the local PG/PC.

Following this, security settings are made based on the data from the configuration so that applications can access IP addresses downstream from the security modules.

If the learning mode is enabled for the internal nodes or programmable controllers, a security policy is first set for secure access to security modules. Then, the SOFTNET Security Client addresses the security modules in order to obtain the IP addresses of the relevant internal nodes.

The SOFTNET Security Client enters these IP addresses in special filter lists belonging to this security policy. Applications can then communicate with the programmable controllers via VPN.

Note

Additional security measures when using the SOFTNET Security Client

The SOFTNET Security Client provides a solution for secure communication with automation cells via VPN. For self-protection of the PC/PG and the corresponding automation cell, it is advisable to use additional measures such as a virus scanner and the Windows firewall.

In Windows 7, the firewall of the operating system must be enabled so that VPN tunnel establishment works.

Creating a configuration file in STEP 7

Configuring a SOFTNET Security Client module in the project

The SOFTNET security client is created as a module in the project. In contrast to the other security modules, you do not need to configure any further properties.

Assign the SSC module to the VPN group or groups at which an IPsec tunnel to the PG/PC is to be set up.

Note

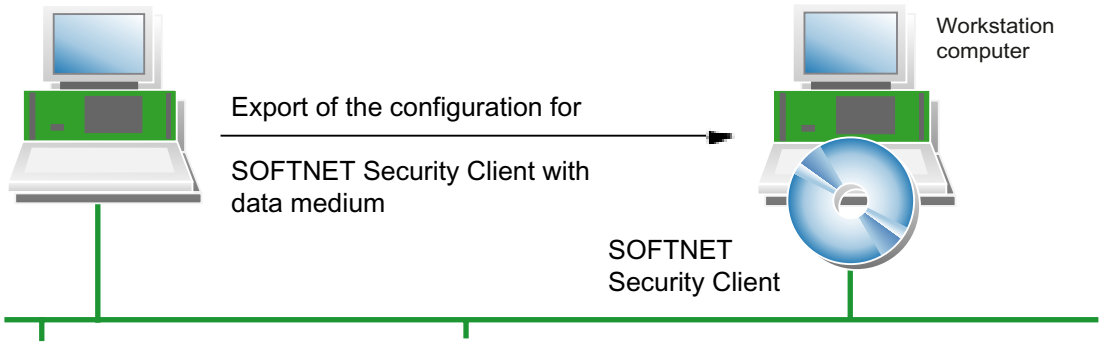
Refer to the information on parameters in section
Including security module in configured VPN group (Page 748).

Note

If you create several SOFTNET Security Clients within a group, no tunnels are set up between these clients but only from the relevant client to the security modules.

Configuration files for the SOFTNET Security Client

The interface between STEP 7 as configuration tool and the SOFTNET Security Client is controlled by configuration files.



The configuration is stored in the file types "*.dat", "*.p12" and "*.cer".

Procedure

To generate the configuration files, perform the following steps in STEP 7:

1. In the "Devices & networks" view, select the "Topology view" or the "Network view" tab.
2. Insert a PC system of the type "SOFTNET Security Client" in the selected tab from the hardware catalog.
3. Assign the SOFTNET Security Client to the VPN groups in which the PG/PC will communicate over IPsec tunnels.
4. Make sure that the "Generate SSC files" check box is selected under the "Configuration of the SOFTNET Security Client" entry in the local security settings of the SOFTNET Security Client.
5. Select the storage location for the configuration files.
6. Compile the configuration of the SOFTNET Security client to export the configuration file.
7. If you selected certificate as the authentication method, specify a password for the certificate of the VPN configuration. If you do not assign a password, the project name (not the project password) is used as the password.
   Result: Export of the configuration files is completed.
8. Adopt the files of the type *.dat, *.p12, *.cer on the PG/PC on which you want to operate the SOFTNET Security Client.

Note

Protecting exported configuration files from unauthorized access

Configuration files for SOFTNET Security Client exported from STEP 7 can contain security related information. You should therefore make sure that these files are protected from unauthorized access. This is particularly important when passing on the files.

SCALANCE S

Replacing a security module

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Requirement

To be able to replace security modules, their module descriptions must be up to date. To update the module description of security modules, follow the steps below:

1. Select the security module to be edited.
2. In the local security settings, click on the entry "General" > "Catalog information".
3. Click the "Update module description" button.

How to access this function

1. Select the security module to be edited in the topology or network view.
2. Right-click on the security module and select the shortcut menu command "Change device...".
   Based on the following table, you can see which security modules you can replace without data loss and which could involve a possible data loss.

Initial module	Possible module replacement									
	S602 V3	S602 V3.1	S602 V4	S612 V3	S612 V4	S623 V3	S623 V4	S623 V4.0.1	S627-2 M V4	S627-2 M V4.0.1
S602 V3	-	x	x	x	x	x	x	x	x	x
S602 V3.1	!	-	x	!	!	!	!	!	!	!
S602 V4	!	!	-	!	!	!	!	!	!	!
S612 V3	!	!	!	-	x	x	x	x	x	x
S612 V4	!	!	!	!	-	!	x	x	x	x
S623 V3	!	!	!	!	!	-	x	x	x	x
S623 V4	!	!	!	!	!	!	-	x	x	x
S623 V4.0.1	!	!	!	!	!	!	!	-	!	x
S627-2M V4	!	!	!	!	!	!	!	!	-	x
S627-2M V4.0.1	!	!	!	!	!	!	!	!	!	-

x Without losses

! Possibly with losses

- The module type and the firmware version are not changed.

Configuring interfaces for SCALANCE S modules

Overview

Configuring the mode

With the mode you specify how interface routing is handled (external/internal). The DMZ interface of the security module (SCALANCE S623/S627-2M only) is always connected in routing mode. For more detailed information, refer to the section
Configuring IP address parameters (Page 774)

Configuring the interfaces

If the external interface, the DMZ interface (SCALANCE S623/S627-2M only) or the tunnel interface (only SCALANCE S612/S623/S627-2M as of V4 in VPN group(s)) of a security module needs to be configured, it must be activated using the "Activate interface" check box. Set the IP address information for each interface and settings for the individual ports. The following options are available with which you can assign an IP address for the external interface and for the DMZ interface in the "General" entry (SCALANCE S623/S627-2M only):

- Static IP address with subnet mask. For more detailed information, refer to the section Configuring IP address parameters (Page 774)
- Address assignment using PPPoE. For more detailed information, refer to the section Configuring an Internet connection (Page 776)
  The internal interface and the tunnel interface can only be configured using a static IP address.

If alias IP addresses were registered on the interfaces of the security module due to configuring NAT rules, these are displayed in the "Alias IP addresses" entry.

Note

External interface and DMZ interface as Internet access

The simultaneous operation of PPPoE on the external interface and on the DMZ interface (dual ISP) is not possible.

Point to Point Protocol over Ethernet (PPPoE)

To allow Internet/WAN access directly via a DSL modem, the IP address on the external interface or on the DMZ interface is assigned using PPPoE. PPPoE is a dial-in protocol for obtaining IP addresses from an Internet service provider (ISP). SCALANCE S is operated here in routing mode.

To use this IP address assignment mode, specify the ISP in the "PPPoE" entry. The IP address, the subnet mask, the standard router and the DNS server of the interface are specified by the ISP.

Note

A configured standard router is not taken into account when using PPPoE. This is assigned dynamically to the module by the ISP.

Note

No network components between SCALANCE S and DSL modem

If the interface of a SCALANCE S module is operated using PPPoE, there must be no other network components between this interface and the connected DSL modem, otherwise the dial-in data of the Internet Service Provider may be transferred unencrypted over this link. When using the "CHAP" authentication protocol, the data is transferred encrypted.

Configuring media modules

In addition to the functions of the SCALANCE S623, the S627-2M has two media module slots in which an electrical or optical media module with two ports can be inserted. This expands both the external and internal interface by up to two ports. In routing mode, the additional ports of the security module can be used to link the external and internal interface to ring topologies.

To integrate media modules in the SCALANCE S627-2M, select the security module and change to the device view. Then select the required media modules from the hardware catalog.

For ports with the port type "Copper", you can set the transmission speed and the duplex method manually using the port mode. For ports with the port type "Optical", the port mode is fixed by the media module used or by the SFP transceiver used and cannot be adapted.

You will find information on connecting the media module ports to MRP rings in the following section:
Auto-Hotspot

Setting the operating mode

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Mode".

Operating mode - possible selections

You can change the operating mode in this dialog if the security module is not included in a VPN group. If the security module is in a VPN group, the mode cannot be changed.

The selection applies to interface routing between the external and internal interface. The DMZ interface (SCALANCE S623 and S627-2M only) is always connected in routing mode.

Bridge mode	For operation in flat networks. External and internal interface are in the same IP subnet. For S623 / S627-2M: External and internal interface are in the same IP subnet, the DMZ interface is in a different IP subnet or is deactivated.
Routing mode	All interfaces are in different IP subnets. If you have activated the routing mode, you must configure an internal IP address and subnet mask for the internal interface of the security module. **Note** If you have enabled the routing mode for the SCALANCE S module, no MAC firewall rules can be defined.
Ghost mode (only for SCALANCE S S602 as of V3.1)	In operation, the security module adopts the IP address of the node connected to the internal interface of the security module for the external interface. The IP address data specified for the external interface is only used for downloading the configuration prior to operation in ghost mode.

Configuring IP address parameters

Meaning

Specify network parameters such as the IP address and subnet mask for the interface(s) of the security module.

How to access this function

1. Select the module to be edited.
2. Select "External interface [P1] red" or "DMZ interface [P3] yellow" in the local security settings. The virtual tunnel interface can only be configured when the security module (only SCALANCE S612/S623/S627-2M as of V4) is in at least one VPN group, refer to the section "Meaning of the tunnel IP address".

   Note

   Configuration of the internal interface in routing mode

   If you have selected the "Routing" mode for the security module, you must also configure an internal IP address and subnet mask for the internal interface of the security module. You can access this function in the local security settings under "Internal interface [P2] green" > Ethernet addresses".

3. If applicable, enable the interface using the "Activate interface" check box.
4. Select the "Ethernet addresses" entry.
5. Complete the settings specified in the following table.

Parameter	Meaning
IP address	IP address for the external interface. The IP address consists of four decimal numbers from 0 to 255, with each number being separated by a period, for example, 141.80.0.16.
Subnet mask	The subnet mask consists of four decimal numbers that are separated by period, for example, 255.255.0.0
Use router (not possible for the tunnel interface)	Select this check box if you want to use a standard router and enter its IP address in the "Router address" input box.

Note

Networking the physical interfaces

Network the physical interfaces of the security module with suitable subnets to avoid IP address conflicts.

Meaning of the tunnel IP address

If you use the function "NAT/NAPT in the VPN tunnel" for a SCALANCE S612/S623/S627-2M module as of V4, you need to assign a tunnel IP address for the security module. This ensures the reachability of the security module via the VPN tunnel and provides a configuration and diagnostics option. The configured tunnel IP address can be expanded with alias tunnel IP addresses using suitable NAT / NAPT rules. The subnet mask is fixed at 32 bits for the tunnel IP address and cannot be changed by the configuration. The tunnel IP address can only be configured if the security module is in at least one VPN group.

You will find further information on address translation with NAT/NAPT in VPN tunnels in the following section:
Address translation with NAT/NAPT in VPN tunnels (Page 802)

Special features with a standard router

- If the IP assignment configured is via "PPPoE", a configured standard router is ignored because the standard route is always automatically via the PPPoE interface.
- If the address assignment configured is with "Static address" and if the security module is connected to the Internet via a DSL (NAPT) router, the DSL router must be entered as the standard router.
- For security modules in ghost mode (SCALANCE S602 as of V3.1 only), no standard routers can be configured since these are identified during runtime. Specific routes cannot be configured for security modules in ghost mode.

Configuring port mode

Meaning

The port mode specifies the transmission speed and the duplex method. The same parameters should be set for the ports involved in communication.

For SCALANCE S V2 modules, the port mode is set to "Autonegotiation" as default. This means the transmission speed and the duplex setting are selected automatically. Moreover, the auto-crossing function is supported.

Configurable port modes

For SCALANCE S V3 and higher, the following port modes can be configured for the permanently installed ports:

Port mode	Meaning
Autonegotiation	The transmission speed and the duplex setting are selected automatically. **Note** A transmission speed of 1000 Mbps and the autocrossing function are supported only if autonegotiation is selected.
10 Mbps, half and full duplex	Transmission speed of 10 Mbps
100 Mbps, half and full duplex	Transmission speed of 100 Mbps

Deactivating a port is possible only for external ports and for the DMZ port of SCALANCE S623/ S627-2M. The port modes for media module ports are configured in the device view and are based on the range of functions of the media module in question.

Configuring an Internet connection

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Requirement

The "PPPoE" entry is only displayed in the local security settings if the IP assignment method "PPPoE" is configured for one of the interfaces.

How to access this function

1. Select the security module to be edited
2. In the local security settings, select the entry "PPPoE".

Meaning

In this entry, you make settings related to the Internet Service Provider (ISP) if a connection using PPPoE is set for one of the interfaces of the security module.

Table 10-35 Settings for the ISP account

Function	Description
Authentication protocol	Select none or one of the following authentication protocols: • PAP (Password Authentication Protocol) • CHAP (Challenge Handshake Authentication Protocol) **Note** Both communications partners have to use the same authentication method otherwise no connection can be established.
User name	Enter the name for logging in with the ISP account.
Password	Enter the password for logging in with the ISP account.
Repeat password	Enter the password for logging in with the ISP account again.

Table 10-36 Rules for user names and passwords

Permitted characters	The following characters from the ANSI X 3.4-1986 character set are permitted: 0123456789 A...Z a...z !#$%&()"*'+`,-./:;<=>?@ [\]_{|}~^
Length of the user name	1 ... 255 characters
Length of the password	1 ... 31 characters

Table 10-37 Settings for the connection

Function	Description
Permanent connection	Permanent Internet connection. After the connection has been terminated by the provider, the connection is automatically restored even if there are currently no packets to be sent.
On-demand connection	The Internet connection is established automatically if packets need to be sent to the Internet. In this setting, delays in the sending of packets are possible.

Maximum idle time (only with the setting "on-demand connection")	If no packets are sent during a certain time, the Internet connection is automatically terminated. In the "Maximum idle time" box, enter the time in seconds after which the connection will be terminated. • Default setting: 300 • Minimum value: 10 • Maximum value: 3600
Forced disconnection (only with the "Permanent connection" setting)	Select the check box to be able to adapt the time of the forced disconnection by the security module.
Forced disconnection time (only with the "Permanent connection" setting)	The provider terminates the Internet connection automatically after a certain period. If you enter a time of day in this box, the security module terminates the Internet connection itself at this time. This allows disconnection of the Internet connection by the provider to be delayed under certain circumstances. A self-initiated forced disconnection is only possible with an existing permanent connection. • Default setting: 00.00 • Permitted entries: 00:00 ... 23:59

Configuring dynamic DNS

Module-specific function

FQDNs can be configured for SCALANCE S V3 or higher. The resolution of FQDNs by SCALANCE S modules is possible for SCALANCE S as of V4.

Meaning

With dynamic DNS, you can access a constantly changing IP address with a permanently defined name (FQDN). This is necessary, for example, if you want to access a server that can be reached via a public IP address that changes.

How it works

The security module signals the current WAN IP address via which the security module can be reached to a provider for dynamic DNS (for example DynDNS.org, no-ip.com). The provider makes sure that DNS queries sent to the FQDN of the security module are replied to with the current WAN IP address of the security module.

Dynamic DNS is permitted on the following interfaces:

- External interface
- DMZ interface

Setting up dynamic DNS - Requirements

Requirement:

- An account has been created with a provider of dynamic DNS and an FQDN has been registered.

Setting up dynamic DNS - Follow the steps below:

1. Select the security module to be edited.
2. In the local security settings, select the entry "DNS".
3. If the security module is downstream from a DSL router or DSL modem, you specify a valid DNS server address. To do this, two options are available:

Option	Meaning
Obtain DNS server address automatically	The address of the DNS server can be obtained automatically using PPPoE if the security module is connected to the Internet via a DSL modem. Can only be set for the external interface and the DMZ interface.
Use the following DNS server address:	Enter the address of the preferred and of the alternative DNS server manually.

4. Activate the "Activate service" check box in the "Primary dynamic DNS service" area and make the following settings:

Setting	Meaning
Provider	Choose the provider with which you have set up an account for dynamic DNS. With the predefined providers (DynDNS.org and No-IP.com), the provider update URL and the check IP service URL are already completed. To use a different provider and/or to use an HTTP URL as the provider update URL, you need to set up a user-defined provider.
User account with the provider	Enter the user name that you specified when you created the account.
Password with the provider	Enter the password that you specified when you created the account.
FQDN	Enter the host name (e.g. mysecuritymodule) and the domain name (e.g. dyndns.org) that is registered with the provider separated by a period. The FQDN can function as a VPN endpoint and differ from the FQDN in the "VPN" entry. You can configure which VPN endpoint the VPN partner is informed of in the connection-specific VPN settings.
Monitor IP address change on DSL router	If the security module is connected to the Internet via a DSL router, enabling this function activates the function of the Check IP service. The security module periodically sends queries to determine the current IP address of the DSL router and to detect an IP address change on the DSL router. The IP address specified in this way is sent to the provider with each change ID.
Period	Specify the interval at which the Check IP service is called. • Default setting: 20 minutes • Minimum value: 10 minutes • Maximum value: 1440 minutes

5. In case the primary provider fails, create a second provider in the "Secondary dynamic DNS service" area (optional setting).

Setting up a user-defined provider - follow the steps below:

Set up a user-defined provider if you are not registered with DynDNS.org or No-IP.com and/or want to use an HTTP URL as the provider update URL. To do this, select the "User-defined" entry in the "Provider" drop-down list and make the following entries:

Setting	Meaning
Ignore errors when checking the server certificate	To ensure that the authentication data is protected, the certificate of the update server is checked as default. If the certificate check fails, the HTTPS connection is terminated and the account data is not transferred. If you select the check box, the function is disabled, for example if the server certificate of the dynamic DNS service is invalid (for example expired). It is advisable not to ignore the check and not to select the check box.
Provider update URL	Enter the URL you received from your provider. The placeholder texts <FQDN> and <CurrentWanIP> need to be placed at the correct positions in the URL.
Check IP service URL	Enter the URL you received from your provider.

Configuring LLDP

Module-specific function

This function is available only for SCALANCE S V4 modules or higher.

Requirement

The security module is in routing mode.

Meaning

LLDP (Link Layer Discovery Protocol) is a protocol used to discover network topologies. A device capable of LLDP can send information about itself to neighboring devices at regular intervals and at the same time receive information from neighboring devices. The received information is stored on every device with LLDP capability in an LLDP MIB file. Network management systems can access these LLDP MIB files using SNMP and therefore recreate the existing network topology.

Configurable parameters

The degree of activity of the security module in terms of LLDP can be configured under the "LLDP mode" entry of the relevant interface.

Parameter	Description
Name	Name of the port for which the setting is configured.
LLDP mode	Configured LLDP mode: • RxTx: LLDP frames can be sent and received • Off: No LLDP frames can be sent or received

Media redundancy in ring topologies

Media redundancy with MRP

Module-specific function

This function is available only for SCALANCE S627-2M modules.

Meaning

The term "media redundancy" groups together various methods for increasing availability in Industrial Ethernet networks in which devices can be reached over different paths. This might be achieved by meshing networks, arranging parallel transmission paths or by closing a linear bus topology to form a ring.

Media redundancy method MRP

Media redundancy within a ring topology is available with SIMATIC NET products among other things with the MRP method (Media Redundancy Protocol).

With this method, one of the nodes is configured as the redundancy manager. The other nodes are redundancy clients. SCALANCE S627-2M modules can only adopt the role of an MRP client. Using test frames, the redundancy manager checks the ring to make sure it is not interrupted. The redundancy clients forward the test frames. If the test frames of the redundancy manager no longer arrive at the other ring port of the redundancy manager due to an interruption, the redundancy manager switches through its two ring ports and informs the redundancy clients of the change immediately.

The time the SCALANCE X switches need to switch through their ring ports as redundancy manager is 200 ms.

Note on the use of MRP

- MRP is supported in ring topologies with up to 100 devices. Exceeding this number of devices can lead to a loss of data traffic.
- It is recommended that you set the ring ports involved to full duplex and 100 Mbps. Otherwise there may be a loss of data traffic.

Possible uses of MRP on media module ports

MRP is supported only on the media module ports of the SCALANCE S627-2M. The following table shows the possible uses of MRP on the media module ports of a SCALANCE S627-2M:

Ring ports	Media module 1		Media module 2	
	P4	P5	P6	P7
MRP Client	-	-	-	-
	Ring 1	Ring 1	-	-
	-	-	Ring 2	Ring 2
	Ring 1	Ring 1	Ring 2	Ring 2

With two lower-layer rings per SCALANCE S module, layer 3 communication is possible between the rings.

Configuring MRP for the security module

Requirement

- The security module is in routing mode.
- Media modules are configured for the interfaces to be connected to MRP rings.
- The interfaces of the security module to be connected to rings are networked with the relevant ring managers.

How to access this function

1. Select the security module to be edited.
2. In the settings of the required interface, select the "Media Redundancy" entry.

Configurable parameters

Parameter	Meaning	Possible selections
MRP domain (only if the "MRP client" media redundancy role is selected)	The members of an MRP ring are specified with the help of MRP domains. The same MRP domain must be selected for the interfaces of all modules to be connected to the same MRP ring.	Display of the MRP domain used for the interface.
Media redundancy role	Selection of the media redundancy protocol or disabling of media redundancy for the interface.	• Not a node in the ring • MRP Client
Ring port 1 (only when the media redundancy role "MRP client" is selected)	Name of the first ring port of the selected interface if the media redundancy role "MRP client" was selected for it.	-

Parameter	Meaning	Possible selections
Ring port 2 (only when the media redundancy role "MRP client" is selected)	Name of the second ring port of the selected interface if the media redundancy role "MRP client" was selected for it.	-
Domain settings	Using the domain settings, you can add new MRP domains, edit the names of existing MRP domains and delete existing MRP domains.	-
Alternative media redundancy protocol	Select this check box to enable the interface of the security module for other media redundancy protocols.	• Enable interface for other media redundancy protocols • Disable interface for other media redundancy protocols (default setting)
Passive listening	Enable this check box if the selected interface will be connected to third-party networks in which STP/RSTP (Spanning Tree Protocol/ Rapid Spanning Tree Protocol) is used.	• Activate passive listening (default) • Deactivate passive listening

Special features of the ghost mode

Module-specific function

This function is available only for SCALANCE S602 as of V3.1.

Meaning

In ghost mode, the security module has no IP address of its own, neither on the internal nor on the external interface. Instead, the security module obtains the IP address for its external interface during runtime from a node connected to the internal interface of the security module whose IP address parameters may be unknown at the time of configuration. It is possible to change an IP address of the internal node and a corresponding IP address at the external interface. Since the internal node is identified based on its MAC address, IP address changes are made only for the learned MAC address. No IP address is configured or obtained at the internal interface of the security module.

As regards the MAC addresses, the security module replaces the MAC address of the internal node with the MAC address of the security module in all outgoing packets on the external interface (responses from the internal node).

Activating ghost mode

1. Select the module to be edited.
2. In the local security settings, select the entry "Mode".
3. Select the "Ghost mode" option.

Configurable module properties

In ghost mode, the following module properties can be configured in the local security settings:

- External interface [P1] red
- Firewall
- Time-of-day synchronization
- Log settings
- SNMP

Since no DNS servers can be configured in ghost mode, no FQDN resolution is possible.

Requirement for identifying an internal node

The security module can only obtain the IP address of the internal node if the internal node initiates data communication with a communications partner of the external network.
In addition to this, the security module does not provide any server services while obtaining the IP address. The security module can only reply to queries from external after data packets have been sent to the security module by the internal node.

Port assignment for incoming and outgoing data connections

Since the external interface of the security module and the internal node have the same IP address, the network components must be addressed explicitly via the TCP/UDP ports. For this reason, the ports are either assigned to the security module or the internal node. The assignments of the ports to the relevant devices are shown in the following tables for incoming and outgoing data connections:

Table 10-38 Port assignment for incoming connections (from external to security module)

Service	Port	Protocol	Comment
Web services, configuration and diagnostics access	443	TCP	The HTTPS port is permanently activated for configuration and diagnostics access using STEP 7 and cannot be changed.
SNMP	161	TCP UDP	Once SNMP is activated in STEP 7, incoming SNMP queries are transmitted via UDP port 161. Transfer via TCP port 161 is also possible, for example, to be able to reach the internal node. **Note** After activating SNMP, the SNMP port is permanently assigned to the security module. If SNMP is not activated, the internal node can be accessed using SNMP with the aid of a firewall rule.

Table 10-39 Port assignment for outgoing connections (from security module to external)

Service	Port	Protocol	Comment
Syslog	514	UDP	If the Syslog service is activated in STEP 7, Syslog messages are transferred via UDP port 514 by the security module. This port assignment cannot be changed.
NTP	123	UDP	If NTP servers are used for time-of-day synchronization, NTP queries are transferred via UDP port 123. This port assignment cannot be changed.

Recognizable IP addresses and subnet masks

The security module only recognizes internal nodes with IP addresses in the network class ranges A, B or C. The subnet mask is identified by the security module based on the network class (refer to the table "Network classes and corresponding subnet masks"). To allow the subnet mask to be determined correctly, a standard router must be entered for the internal node.

Nodes with IP addresses in the network classes D and E are rejected by the security module.

Network class	IP addresses		Subnet mask
	Low limit	High limit	
A	0.0.0.0	127.255.255.255	255.0.0.0
B	128.0.0.0	191.255.255.255	255.255.0.0
C	192.0.0.0	223.255.255.255	255.255.255.0
D	224.0.0.0	239.255.255.255	Rejected by the security module
E	240.0.0.0	255.255.255.255	Rejected by the security module

Configuration limits

A maximum of one internal node is recognized by the security module. If several internal nodes exist, the security module reacts as follows:

- The first device the security module recognizes in the internal network obtains access to the external network segment if the firewall is suitably configured.
- The data traffic of any additional nodes in the internal network area is blocked in the outgoing direction at level 2 (MAC layer) based on the sender address.

Loading configurations and diagnostics after commissioning

After obtaining an IP address from the internal node, the security module has an IP address on the external interface that can differ from the IP address with which the security module was initially configured. To load a configuration or to run diagnostics, in STEP 7 you need to specify the IP address for the connection to the external interface that the security module obtained from the internal node during runtime. This is possible in the local security settings or directly in the "Advanced download" or "Connect online" dialogs. For more detailed information on establishing online connections, refer to the section: Downloading a configuration (Page 823)

Routing information for hierarchical networks on the external interface

If there are hierarchical networks with subnet transitions on the external interface of the security module, the security module needs to obtain the relevant routing information from the internal node. To achieve this, the internal node must respond to ICMP queries sent to it. Responding to ICMP broadcasts is not necessary.

Authentication using a RADIUS server

Overview

Module-specific function

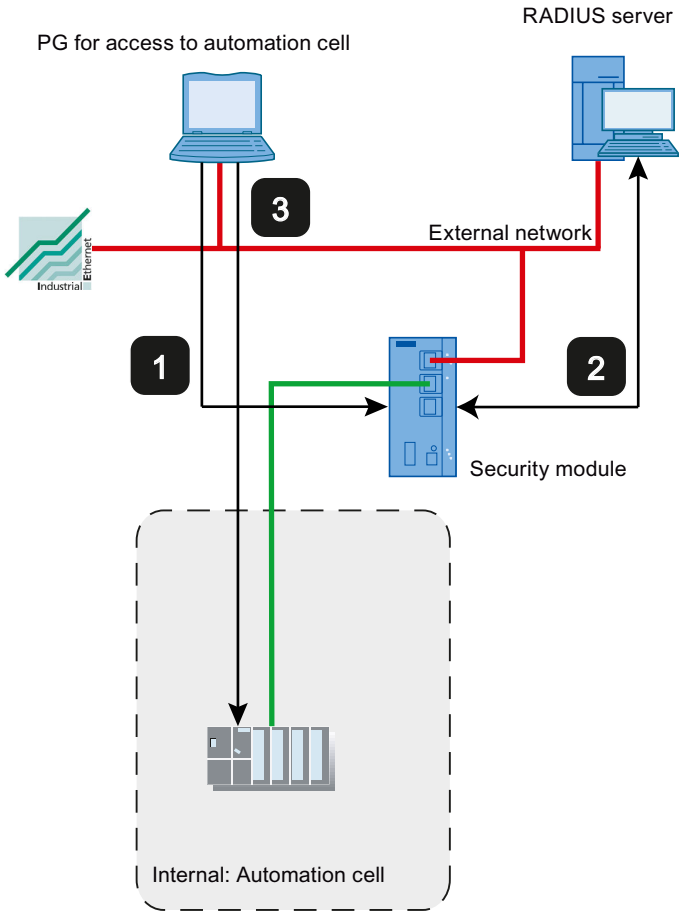
This function is available only for SCALANCE S V4 modules or higher.

Meaning

RADIUS (Remote Authentication Dial-In User Service) is a protocol for authenticating users by servers on which user data can be stored centrally. The use of RADIUS servers can increase the protection of user names, assigned roles and passwords.

Scenario for the use of RADIUS servers

Authentication by RADIUS servers can be performed when activating user-specific IP rule sets.



1 Entry of the user data on the Web page of the security module

2 Authentication by RADIUS server and activation of the user-specific IP rule set

3 Access to an automation cell

The network setup shown above is simply an example. The RADIUS server can also be located in the internal network or in the DMZ network of the security module.

For the configuration options described below, it is assumed that a RADIUS server is configured in STEP 7 and was assigned to the relevant security module. In addition to this, one user or role must be configured with the "RADIUS" authentication method. For more detailed information, refer to the following sections:

- Defining a RADIUS server (Page 791)
- Assigning a RADIUS server to a security module (Page 791)
- Create users (Page 691)
- Creating roles (Page 692)

For general information on user-specific IP rule sets, refer to the following section:

- Auto-Hotspot

Configuration options

To authenticate the user using a RADIUS server, there are two configuration options available:

- The user and the user's role are known on the security module, only the password management for the user is performed on the RADIUS server. The user and the password are configured on the RADIUS server.
  - A user with the "RADIUS" authentication method is configured.
  - The user is assigned to the user-specific IP rule set.

  Result:
  - When a user logs on to the Web page of the security module, the authentication query is forwarded to the RADIUS server.
  - The RADIUS server runs a password check and signals the result back to the security module.
  - If the password check is passed successfully, the user-specific IP rule set is activated.
- The role is known on the security module, user management is via the RADIUS server. The user and the password are configured on the RADIUS server.
  - A user-defined role or a system-defined role is assigned to the user-specific IP rule set.
  - Under the entry "RADIUS" > "RADIUS settings" in the local security settings of the security module, the check box "Allow RADIUS authentication of unconfigured users" and the check box "Filter ID is required for authentication" are enabled.

  Result:
  - When a user logs on to the Web page of the security module, the authentication and authorization query is forwarded to the RADIUS server.
  - The RADIUS server runs a password check and signals the result back to the security module.
  - Case a: If, in addition to this, the role name is configured on the RADIUS server:
    The RADIUS server returns the role name assigned to the user to the security module.
  - Case b: If the role name is not configured on the RADIUS server:
    The security module assigns the user the system-defined role "radius".
  - If the password check is passed successfully, the user-specific IP rule set is activated.

Conventions for RADIUS servers

- The RADIUS servers can be in any network connected to the security module.
- A maximum of two RADIUS servers can be configured per security module. During operation only one of the RADIUS servers is active.
- When defining a RADIUS server, an FQDN can also be used instead of IP an address.

Defining a RADIUS server

Meaning

Before authentication by a RADIUS server is possible, this first needs to be stored in the STEP 7 project. Following this, you assign the defined RADIUS server to the security module for which the RADIUS server will handle user authentication.

Procedure

1. Double-click on the "RADIUS" entry in the global security settings.
2. Double-click on the "Add new RADIUS server" entry.
3. Enter the required parameters according to the following table.

Parameter	Meaning
Name	Freely selectable name for the RADIUS server.
IP address / FQDN	IP address or FQDN of the RADIUS server.
Port	UDP port via which the RADIUS server can be reached. As default, authentication data is received at port 1812.
Shared secret	Entry of the password that will be used when transferring the logon data between the RADIUS server and security modules for encryption. The following characters from the ANSI X 3.4-1986 character set are permitted: 0123456789 A...Z a...z !#$%&()"*'+`,-./:;<=>?@ [\]_{|}~^ Length of the shared secret: 1 ... 31 characters
Repeat shared secret	Confirmation of the password
Authentication method	Display of the method used to check the user data. Only the "PAP" method (Password Authentication Protocol) is supported.
Comment	Entry of freely selectable, optional comments.

Result

You have defined a RADIUS server and can now assign this to the required security modules.

Assigning a RADIUS server to a security module

Requirement

You have defined a RADIUS server.

Procedure

1. Select the module to be edited.
2. Select the "RADIUS" entry in the local security settings.
3. Select the "Enable RADIUS authentication" check box.

   Note

   Changing the method of authentication with the Web server on the security module

   If RADIUS authentication is enabled on the security module, the method for authentication with the Web server is changed from "Digest Access Authentication" to "Basic Access Authentication".

4. In the "RADIUS timeout" input box, enter the maximum time in seconds that the security module will wait for a response from the RADIUS server.
5. In the "RADIUS repetitions" input box, enter the number of connection establishment attempts with the RADIUS server.
6. Select the "Allow RADIUS authentication of unconfigured users" check box if the user-specific IP rule to be activated was assigned a role instead of a user.
7. Select the "Filter ID is required for authentication" check box if the assigned role is a user-defined role.
8. Under the entry "RADIUS server", select the RADIUS server you want to assign to the security module from the "Name" drop-down list.
   As an alternative, you can assign security modules for which RADIUS authentication is enabled to RADIUS servers in the global security settings. You will find general information on authentication by the RADIUS server in the following section:
   Auto-Hotspot

Setting up a firewall

Local firewall rules for SCALANCE S modules

Configuring a firewall with predefined firewall rules

Configuring a firewall using predefined firewall rules

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" entry in the local security settings.

Firewall enabled as default

The "Enable firewall" check box is enabled by default. The firewall is therefore activated automatically and all access from external to the security module is blocked. By clicking the relevant check box, enable the firewall rules for the specific direction.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Enable firewall in advanced mode" check box. For more detailed information about the advanced firewall mode, refer to the section
Overview of local firewall rules (Page 711)

Firewall configuration with VPN

If the security module is in a VPN group, the "Tunnel communication only" check box is enabled as default. This means that only encrypted IPsec data transfer is permitted via the external interface or via the DMZ interface. Only HTTPS access to the module (TCP port 443) remains allowed untunneled.

If you deselect the check box, tunneled communication and also the types of communication selected in the other boxes are permitted.

Table 10-40 Available firewall rules and directions

Service	From internal to external	From external to internal	From internal to DMZ	From DMZ to internal	Enabled ports	Meaning
Allow IP communication	x	x	x	x	-	IP communication for the selected communication directions is allowed.
Allow S7 protocol	x	x	x	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/ FTPS (explicit mode)	x	x	x	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	x	x	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	x	x	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow DNS	x	x	x	x	TCP port 53 UDP port 53	Communication connection to a DNS server is allowed.
Allow SNMP	x	x	x	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	x	x	x	TCP port 25	For sending e-mails via an SMTP server.

Service	From internal to external	From external to internal	From internal to DMZ	From DMZ to internal	Enabled ports	Meaning
Allow NTP	x	x	x	x	UDP port 123	For synchronization of the time of day.
Allow DHCP	x	x	x	x	UDP port 67 UDP port 68	Only in bridge mode Communication connection with a DHCP server is permitted.

Table 10-41 Logging

Option	Action when activated
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.
Log blocked incoming packets	All incoming IP packets that are discarded are logged.
Log blocked outgoing packets	All outgoing IP packets that are discarded are logged.

You can view the logged packets at the "Packet filter log" entry in the "Online & Diagnostics" dialog. For more detailed information, refer to the section Logging data packets - "Packet filter log" entry (Page 765).

Configuring a firewall with predefined MAC rules

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" entry in the local security settings.

Firewall enabled as default

The "Enable firewall" check box is enabled by default. The firewall is therefore activated automatically and all access from external to the security module is blocked. By clicking the relevant check box, enable the firewall rules for the specific direction.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Enable firewall in advanced mode" check box. For more detailed information about the advanced firewall mode, refer to the section Overview of local firewall rules (Page 711).

Firewall configuration with VPN

If the security module is in a VPN group, the "Tunnel communication only" check box is enabled as default.

If you deselect the check box, tunneled communication and also the types of communication selected in the other boxes are permitted.

Available MAC rules and directions

Service	From internal to external	From external to internal	Meaning
Allow MAC communication	x	x	MAC traffic for the selected communication directions is allowed.
Allow ISO protocol	x	x	ISO traffic for the selected communication directions is allowed.
Allow SiClock	x	x	SiClock time-of-day frames for the selected communication directions are allowed.
Allow DCP	x	x	DCP traffic for assignment of IP addresses is allowed for the selected communications directions.

Table 10-42 Logging

Option	Action when activated
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.

User-specific IP rule sets

Overview

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Meaning

Initially, individual or multiple users are assigned to user-specific IP rule sets. The user-specific IP rule sets are then assigned to individual or multiple security modules. This makes it possible, to allow user-specific access. If, for example all access to the networks downstream from a security module is blocked, certain nodes can be allowed for a user based on their IP addresses. This means that access is allowed for this user but access remains blocked for other users.

User logon via the Internet

The user can log in to the external interface or the DMZ interface via the Web page of the security module. If authentication is successful, the IP rule set defined for the user for the IP address of the device from which the login is made is enabled.

The connection to the Web page of the security module is via HTTPS using the IP address of the connected port and taking into account the valid routing rules:

Example:

External interface: 192.168.10.1

Call up of the login page with: https://192.168.10.1/

Users can log on with every role as long as the user or the role is assigned to a user-specific firewall rule set.

Options for authenticating the user

Depending on the authentication method selected when the user who will log in to the security module was created, the authentication is handled by different instances:

- Authentication method "Password": Authentication is handled by the security module.
- Authentication method "RADIUS": Authentication is handled by a RADIUS server (SCALANCE S as of V4 only).

Assignment of roles to user-specific IP rule sets

On SCALANCE S modules as of V4, it is also possible to assign user-specific IP rule sets that are assigned to roles. This makes it possible to enable a group of users for access to certain IP addresses.

If a RADIUS server is used for user authentication and a role is assigned to the user-specific IP rule set, users can also be authenticated by the RADIUS server although they are not configured on the security module. These users must be stored on the RADIUS server or on a separate database where they need to be assigned to the role assigned to the user-specific IP rule set in STEP 7. This procedure has the advantage that all user data is stored exclusively on the RADIUS server.

You will find more information on authentication by the RADIUS server in the following section:

Auto-Hotspot

User-specific IP rule sets are used locally - conventions

The same conventions as described in the following section apply:
Global firewall rule sets - conventions (Page 704)

Creating and assigning user-specific IP rule sets

Creating user-specific IP rule sets

1. In the global security settings, select the entry "Firewall" > "User-specific IP rule sets" > "IP rule sets".
2. Double-click on the "Add new IP rule set" entry to create a user-specific IP rule set.
   Result: The created user-specific IP rule set is shown in the entry.
3. Double-click on the created user-specific IP rule set.
   Result: In the "Properties" > "General" tab of the Inspector window, the configurable properties of the user-specific IP rules set are displayed.
4. In the Inspector window, click on the "General" entry and enter the following data:
   - Name: Project-wide, unique name of the rule set. The name appears in the local rule list of the security module after the rule set is assigned.
   - Description (optional): Enter a description of the user-specific IP rule set.
5. Click on the "IP rules" entry and enter the firewall rules one after the other in the list. No IP address can be entered in the "Source IP address" box. This is entered automatically when the node logs on to the security module.
   Note the parameter description in the following sections:
   Defining IP packet filter rules (Page 712)
   Note the special features of firewall rules generated automatically by STEP 7 for NAT/NAPT rules:
   Relationship between NAT/NAPT router and user-specific firewall (Page 804)

Assigning user-specific IP rule sets

1. Click on the "Users and roles" entry in the Inspector window.
2. In the "Available users and roles" area, select the users you want to assign to the user-specific rule set.
3. Click the "<<" button to assign the selected users or roles to the user-specific rule set. The assignment of roles to user-specific IP rule sets is possible only for SCALANCE S modules as of V4.
4. Assign the created user-specific IP rule set to the required security modules using the entry "Assign user-specific IP rule set" in the global security settings. For this, advanced firewall mode must be enabled for the security modules.

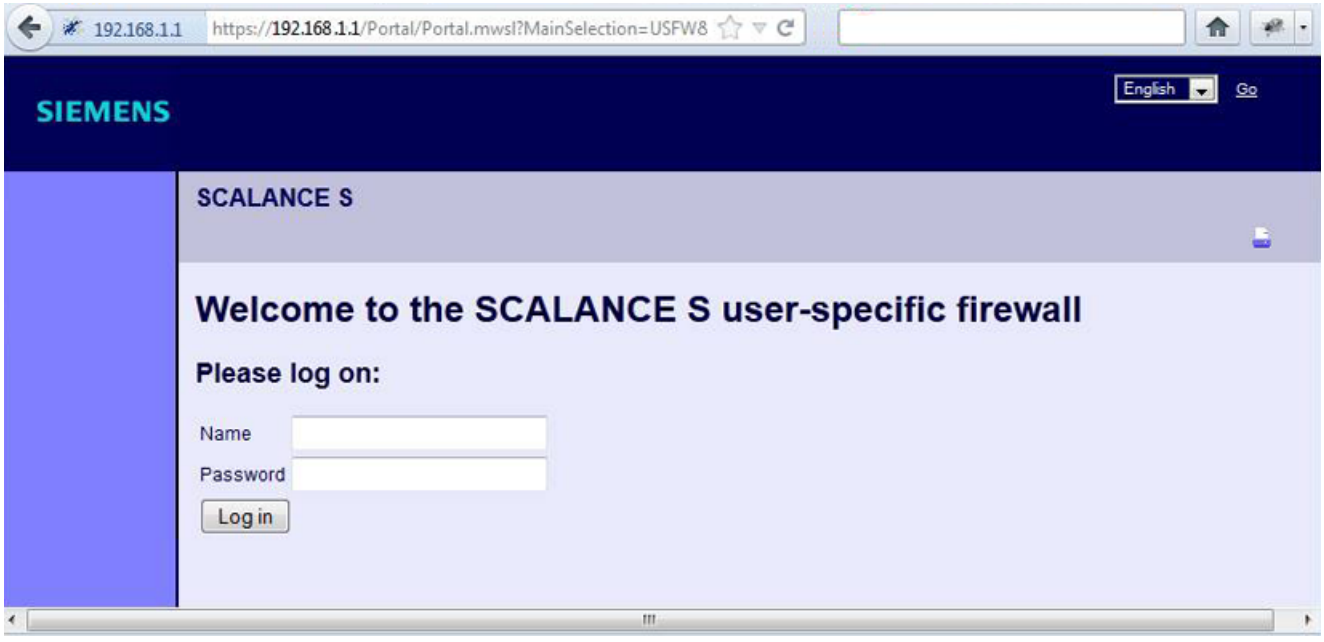
Note

Assignment of user-specific IP rule sets

- A security module can only be assigned one user-specific rule set per user.
- Due to the assignment, the right to log in to the security module is activated implicitly for all users and roles assigned to the IP rule set.

Result

- The user-specific rule set is used by the assigned security modules as a local rule set and automatically appears in the local list of firewall rules.
- The user can log on with the security module. Authentication of the user is performed depending on the selected authentication method either by the security module or a RADIUS server.



Range of values for the maximum session time

The time after which the user is automatically logged out can be specified when creating or editing a user; the default is 30 minutes. On the Web page of the security module, the session time can be extended to the value assigned to the user.

You will find more information creating users in the following section:

Create users (Page 691)

IP packet filter directions SCALANCE S

Meaning

Possible selections for the communication directions "From" and "To" in the IP rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		
From	**To**	**S602**	**S61x**	**S623 / S627-2M**
Internal	External	**x**	**x**	**x**
	Tunnel	-	**x**	**x**
	Any	-	**x**	**x**
	DMZ	-	-	**x**
	Internal	**x**	**x**	**x**
External	Internal	**x**	**x**	**x**
	Any	-	-	**x**
	Tunnel	-	-	**x**
	DMZ	-	-	**x**
Tunnel	Internal	-	**x**	**x**
	External	-	**x**	**x**
	DMZ	-	-	**x**
Any	Internal	-	**x**	**x**
	External	-	-	**x**
	DMZ	-	-	**x**
DMZ	Internal	-	-	**x**
	External	-	-	**x**
	Any	-	-	**x**
	Tunnel	-	-	**x**

x = communication direction is configurable

- = communication direction is not configurable

MAC packet filter directions SCALANCE S

Meaning

Possible selections for the communication directions "From" and "To" in the MAC rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		
From	**To**	**S602**	**S61x**	**S623 / S627-2M**
Internal	External	**x**	**x**	**x**
	Tunnel	-	**x**	**x**
	Any	-	**x**	**x**
External	Internal	**x**	**x**	**x**
	Any	-	-	**x**
	Tunnel	-	-	**x**

Available options / ranges of values		Security module		
Tunnel	Internal	-	x	x
	External	-	x	x
Any	Internal	-	x	x
	External	-	-	x

x = communication direction is configurable

- = communication direction is not configurable

Adapting standard rules for IP services

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Requirement

This function is available only in advanced firewall mode.

How to access this function

1. Select the security module to be edited.
2. In the local security settings, select the entry "Firewall" > "Standard rules for IP services".

Meaning of the advanced settings

Parameter	Meaning when activated
Use advanced status options	The permitted number of connections and firewall statuses per time unit is limited. If a network node exceeds this limit, its IP address is entered in the IP blacklist of the security module. The node can then no longer communicate via the security module. The IP blacklist of the security module can be viewed in online mode.
Log all activated rules	Packets that are allowed according to the default rules for IP services are logged.
Enable ICMP test for interfaces	Ping queries coming in on an interface of the security module can be forwarded to other interfaces. This means that, for example, ping queries can be made from the external network to the internal interface of the security module.

Default firewall rules for SCALANCE S

The following table lists the default firewall rules for SCALANCE S modules. In some cases, the firewall rules are only set if the relevant service is used by the security module (e.g. SNMP).

Service	Direction	X1 interface (red)	Interface X2 (green)	Interface X3 (yellow) (only for S623, S627-2M)	Tunnel interface* (not for S602)
Interface rerouting	outgoing	-	x	-	-
HTTPS		x	x*	x	x
ICMP	incoming	-	x	-	x
ICMP pathfinder (only for SCALANCE S S602 as of V3.1 in ghost mode)	outgoing	-	x	-	-
SNMP	incoming	x	x	x	x
Syslog	outgoing	x	x	x	x
NTP	outgoing	x	x	x	x
DNS	outgoing	x	x	x	x
HTTP	outgoing	x	-	x	-
VPN (IKE)		x	-	x	-
VPN (NAT Traversal)		x	-	x	-
BootP Server	incoming	-	x	x	-
BootP Client	outgoing	-	x	x	-
RADIUS	outgoing	x	x	x	x
CARP (only for SCALANCE S62x as of V4)	outgoing	x*	x*	-	-
Pfsync (only for SCALANCE S62x as of V4)	outgoing	-	-	x*	-

x enabled as default

- disabled as default

* cannot be adapted

SCALANCE S module as router

Specifying routes

Meaning

Specifies routes for addressing subnets that cannot be reached directly via the security module.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the "Routing" entry.
3. Double-click "Add new" in the table to add a route.
4. Enter the following values:

Parameter	Function	Example of a value
Network ID	Requests to nodes of the subnet with the network ID and the subnet mask specified here are forwarded to the subnet via the specified router IP address. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet. The specified network ID must not be located in the same subnet as the IP address of the security module.	192.168.11.0
Subnet mask	The subnet mask determines the network structure. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	255.255.255.0
Router IP address	IP address of the router that connects to the subnet. As an alternative, an FQDN can be entered for SCALANCE S modules as of V4. The specified router IP address must be in the same subnet as the IP address of the security module.	192.168.10.2
Activate rerouting (only for SCALANCE S as of V3)	Select this check box if the frames of the entered route will enter and leave on the same interface of the security module (rerouting). Rerouting is only supported on the internal interface of the security module.	

Address translation with NAT/NAPT in VPN tunnels

Module-specific function

Address translation with NAT/NAPT in VPN tunnels is only available for SCALANCE S612/ S623/S627-2M modules as of V4.

Meaning

Address translations with NAT/NAPT can also be performed for communications relations established via a VPN tunnel.

Requirements

The following requirements apply generally to a SCALANCE S module that will perform an address translation with NAT/NAPT in a VPN tunnel:

- The SCALANCE S module is in a VPN group.
- The SCALANCE S module is in routing mode and/or the DMZ interface of the SCALANCE S module is activated.

- The tunnel interface is enabled.
- The advanced firewall mode is enabled.

Supported address translation directions

The address translation directions described in the following section are supported:
NAT/NAPT routing (Page 726)

Supported address translation actions

With tunneled communications relations, the following address translation actions are supported:

- Destination NAT ("Redirect")
- Source NAT ("Masquerading")
- Source and Destination NAT ("1:1-NAT")
- NAPT ("Port forwarding")

You will find basic information on these address translation actions in the following section:
NAT/NAPT routing (Page 726)

Supported VPN links

In conjunction with NAT/NAPT, the following VPN links are supported:

VPN link		VPN link is initiated by	Address translation is performed by
SCALANCE S (a)	SCALANCE S (b)	SCALANCE S (a) or SCALANCE S (b)	SCALANCE S (a) and/or SCALANCE S (b)
SCALANCE S	CP x43-1 Adv. / PC-CP	SCALANCE S or CP x43-1 Adv. / PC-CP	SCALANCE S
SCALANCE S	SCALANCE M	SCALANCE M	SCALANCE S and/or SCALANCE M*
SOFTNET Security Client	SCALANCE S	SOFTNET Security Client	SCALANCE S

* Only 1:1 NAT is supported.

SCALANCE S modules of the type SCALANCE S623 V4 and SCALANCE S627-2M V4 that have a VPN endpoint on the external interface and on the DMZ interface can perform address translations on both interfaces at the same time.

Address translation characteristics when involved in several VPN groups

If a SCALANCE S module is a member of several VPN groups, the address translation rules configured for the tunnel interface of the SCALANCE S module apply for all VPN connections of this SCALANCE S module.

Please note: Once you have configured a NAT address translation to or from the direction of the tunnel, only the IP addresses involved in the NAT address translation rules can be reached via the VPN tunnel.

Relationship between NAT/NAPT router and user-specific firewall

Module-specific function

The configuration of NAT/NAPT rules in the user-specific firewall is only available for SCALANCE S modules as of V3.

Meaning

After creating NAT/NAPT rules, STEP 7 automatically generates a user-specific IP rule set in the user-specific firewall that enables communication in the configured address translation direction. You can then assign this user-specific IP rule set to individual or multiple users and/or individual or multiple roles (only for SCALANCE S modules as of V4).

The generated firewall rules can, if necessary, be moved and expanded (additional IP address, services, bandwidth). Firewall parameters generated by STEP 7 cannot be adapted. If the user-specific IP rule set is assigned to a security module with NAT/NAPT deactivated, the NAT/NAPT rules from the user-specific firewall are also applied to this security module.

Note

The address translation actions "Source-NAT + Destination-NAT" and "Double-NAT" are not supported in conjunction with the user-specific firewall.

How to access this function

"NAT" or "NAPT" entry in the editor for user-specific IP rules sets, refer to the following section: Creating and assigning user-specific IP rule sets (Page 797)

Supported address translation directions for the action "Source NAT"

The action "Source NAT" can be performed in the following directions:

- External to DMZ
- DMZ to external

No IP address can be entered in the "Source IP address" box. This is entered automatically when the node logs on to the security module.

Supported address translation directions for the action "Destination NAT"

The action "Destination NAT" can be performed in the following directions:

- External to internal
- External to DMZ
- DMZ to internal
- DMZ to external
- Tunnel to internal (only SCALANCE S612/S623/S627-2M as of V4)

- Tunnel to external (only SCALANCE S612/S623/S627-2M as of V4)
- Tunnel to DMZ (only SCALANCE S612/S623/S627-2M as of V4)

Supported address translation directions for NAPT

The address translation with NAPT can be performed in the following directions:

- External to internal
- External to DMZ
- DMZ to internal
- DMZ to external
- Tunnel to internal (only SCALANCE S612/S623/S627-2M as of V4)
- Tunnel to external (only SCALANCE S612/S623/S627-2M as of V4)
- Tunnel to DMZ (only SCALANCE S612/S623/S627-2M as of V4)

NAT/NAPT address translation and corresponding user-specific IP rule sets

In the firewall rules for user-specific IP rule sets generated based on NAT/NAPT rules, no IP address can be entered in the "Source IP address" box. This is entered automatically when the node logs on to the security module. The remaining properties generated locally for individual security modules are identical. Refer to the section
Relationship between NAT/NAPT router and firewall (Page 732)

Security module as DHCP server

Overview DHCP server

Overview

You can operate the SCALANCE S module on the internal network and on the DMZ network as DHCP server (DHCP = Dynamic Host Configuration Protocol). This allows IP addresses to be assigned automatically to the devices connected to the internal network or the DMZ network.

Simultaneous DHCP server operation on both interfaces is possible (SCALANCE S623/ S627-2M only).

The IP addresses are either distributed dynamically from an address range you have specified or you can select a specific IP address and assign it to a particular device. If devices on the internal interface or on the DMZ interface should always be assigned the same IP address for firewall configuration, the address assignment must only be static based on the MAC address or based on the client ID.

Requirements

You configure the devices on the internal network or on the DMZ network so that they obtain the IP address from a DHCP server.

Depending on the mode, the security module sends an IP address of the standard router to the nodes in the relevant subnet or you need to inform the nodes in the subnet of a router IP address.

- Router IP address will be transferred
  In the following situations, the DHCP protocol of the security module will inform the nodes of the router IP address:
  - The node is connected to the DMZ interface (SCALANCE S623/S627-2M only)
    In this case, the security module sends its own IP address as the router IP address.
  - The node is connected to the internal interface and the security module is configured for router mode
    In this case, the security module sends its own IP address as the router IP address.
  - The node is connected to the internal interface and the security module is not configured for router mode, there is, however, a standard router specified in the configuration of the security module.
    In this case, the security module transfers the IP address of the standard router as the router IP address.
- Router IP address will not be transferred
  In the following situations, enter the router IP address manually on the node:
  - The node is connected to the internal interface and the security module is not configured for router mode. There is also no standard router specified in the configuration of the security module.

See also

Configuring a DHCP server (Page 806)

Configuring a DHCP server

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "DHCP server".

3. Select the interface for which you want to make the DHCP settings.
4. Make the address assignment. You have the following configuration options:
   - Static address assignment
     Devices with a specific MAC address or client ID are assigned the specified IP addresses. You specify these addresses by entering the devices in the address list in the "Static address assignment" group box.
   - Dynamic address assignment
     Devices whose MAC address or whose client ID was not specified specifically, are assigned a random IP address from the specified address range. For this purpose, activate the "Dynamic IP address pool" check box. Set the address range in the "Dynamic IP address pool" input area.

Note

Dynamic address assignment - reaction after interrupting the power supply

Please note that dynamically assigned IP addresses are not saved if the power supply is interrupted. On return of the power, you must therefore make sure that the nodes request an IP address again.

You should therefore only use dynamic address assignment for the following nodes:

- Nodes that are used temporarily in the subnet (such as service devices).
- Nodes that have been assigned an IP address and send this as the "preferred address" the next time they request an address from the DHCP server (for example PC stations).

For permanent nodes you should preferably use static address assignment by specifying a client ID or the MAC address.

Consistency check - these rules must be adhered to

Remember the following rules when making the entries.

- The IP addresses assigned in the address list in the "Static address assignments" group box must not be in the range of the dynamic IP addresses.
- IP addresses, MAC addresses and client IDs may only occur once in the "Static address assignment" table (related to the security module).
- For the statically assigned IP addresses, you must specify either the MAC address or the client ID (computer name).
- The client ID is a string with a maximum of 63 characters. Only the following characters may be used: a-z, A-Z, 0-9 and - (dash).
  Note
  In SIMATIC S7, a client ID can be assigned to the devices on the Ethernet interface to allow them to obtain an IP address using DHCP.
  With PCs, the procedure depends on the operating system being used; it is advisable to use the MAC address here for the assignment.
- For the statically assigned IP addresses, you must specify the IP address.

- The following IP addresses must not be in the range of the free IP address range (dynamic IP addresses):
  - All router IP addresses in the "Routing" entry
  - Syslog server
  - Standard router
  - Security module address(es)
- DHCP is supported by the security module on the interface to the internal subnet and on the interface to the DMZ network. The following additional requirements for IP addresses in the range of the dynamic address assignments result from this operational behavior of the security module:
  - Bridge mode
    The range must be within the network defined by the security module.
  - Routing mode
    The range must be within the internal subnet defined by the security module.
    Note
    The DMZ network always represents a separate subnet. When using DHCP on the DMZ interface, make sure that the free IP address range (dynamic IP addresses) is within the DMZ subnet.
- The free IP address range must be fully specified by entering the start address and the end address. This end address must be higher than the start address.
- The IP addresses you enter in the address list in the "Static address assignments" input area must be in the address range of the internal subnet or in the DMZ subnet of the security module.

See also

Running a consistency check (Page 684)

Configuring proxy ARP

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Overview

Proxy ARP allows routers to respond to ARP queries for hosts. The hosts are in networks separated by routers but use the same IP address range.

If PC1 sends an ARP request to PC2, it receives an ARP response and the hardware address of the interface (MAC address of the port of the security module) on which the query was received from the security module located in between and not from PC2. The querying PC1 then sends its data to the security module that then forwards it to PC2.

How to access this function

This function is only available for the internal interface of a security module that is a member of a VPN group and is in bridge mode.

1. Select the security module to be edited.
2. In the local security settings, select the entry "Proxy ARP".
3. If the security module is to respond to an ARP query from its own LAN as a substitute for the specific connection partner, enter the corresponding IP address.

IPsec tunnel: Creating and assigning groups

Configuring internal network nodes

Using the learning mode to learn internal nodes

Finding nodes for tunnel communication automatically

One great advantage when configuring and operating tunnel communication is that SCALANCE S modules in bridge mode can find the nodes on the internal interface automatically. New nodes are detected by the security module during operation. The detected nodes are signaled to the security modules belonging to the same VPN group. This allows data exchange within the tunnels of a group in both directions at any time.

Detectable nodes

The following nodes are detected:

- Network nodes with IP capability
  Network nodes with IP capability are found when they transmit an ICMP response to the ICMP subnet broadcast.
  IP nodes downstream from routers can be found if the routers pass on ICMP broadcasts.
- ISO network nodes
  You can also teach-in network nodes without IP capability that can be addressed by means of ISO protocols.
  This is only possible if they reply to XID or TEST packets. TEST and XID (Exchange Identification) are auxiliary protocols for exchanging information on layer 2. By sending these packets with a broadcast address, these network nodes can be located.
- PROFINET nodes
  Using DCP (Discovery and basic Configuration Protocol), it is possible to find PROFINET nodes.

Network nodes that do not meet these conditions must be configured manually.

Subnets located on the other side of internal routers also need to be configured manually.

How to access the function

1. Select the module.
2. In the local security settings, select the entry "VPN" > "Nodes".

Enabling/disabling the learning mode

The learning function is enabled in the configuration as default for every security module.

Learning can also be disabled completely for SCALANCE S. In this case, you will need to configure all internal network nodes participating in the tunnel communication manually.

When is it useful to disable the automatic learning mode?

The default settings for the security module assume that internal networks are always secure; in other words, in a normal situation, no network node is connected to the internal network if it is not trustworthy.

Disabling the learning mode can be useful if the internal network is static; in other words, when the number of internal nodes and their addresses do not change.

If the learning mode is disabled, this reduces the load on the medium and the nodes in the internal network resulting from the learning packets. The performance of the security module is also slightly improved since it does not need to process the learning packets.

Note: In the learning mode, all nodes in the internal network are detected. The information relating to VPN configuration limits relates only to nodes that communicate over VPN in the internal network.

Note

If more than 128 internal nodes are being operated, the permitted configuration limits are exceeded and an illegal operating status results. Due to the dynamics in the network traffic, this causes internal nodes that have already been learned to be replaced by new previously unknown internal nodes.

See also

Configuring internal subnets manually (Page 811)

Configuring IP network nodes manually

Meaning

As an alternative to the learning mode that you enable using the "Enable learning of internal nodes" check box and that allows the security module to learn the internal network nodes dynamically, you can enter the network nodes to be learned manually in the "Internal IP nodes" entry and in doing so enable them for VPN tunnel communication. The MAC address of a network node can be specified as an option.

Requirement

- The security module is in bridge mode.
- The security module is a member of a VPN group.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN" > "Nodes" > "Internal IP nodes".

Configuring MAC network nodes manually

Meaning

As an alternative to the learning mode that you enable using the "Enable learning of internal nodes" check box and that allows the security module to learn the internal network nodes dynamically, you can enter the network nodes to be learned manually in the "Internal MAC nodes" entry and in doing so enable them for VPN tunnel communication.

Requirement

- The security module is in bridge mode.
- The security module is a member of a VPN group.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN" > "Nodes" > "Internal MAC nodes".

Configuring internal subnets manually

Requirement

- The security module is a member of a VPN group.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN" > "Nodes" > "Internal subnets".

Security module in bridge mode - "Internal subnets" entry

To be able to enable internal subnets for VPN tunnel communication manually, you need to enter the following address parameters:

Parameter	Function	Example of a value
Network ID	Network ID of the subnet to be enabled for VPN tunnel communication. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet. Must not be located in the same subnet as the IP address of the security module.	192.168.11.0
Subnet mask	The subnet mask determines the network structure. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	255.255.255.0
Router IP address	IP address of the router via which the subnet you are allowing is reached. Must be located in the same subnet as the IP address of the security module.	192.168.10.2

Security module in routing mode - "Subnets reachable through tunnel" entry

In routing mode, entire subnets are always tunneled. To be able to enable internal subnets reachable via routers, the external subnet or the DMZ subnet for VPN tunnel communication manually, you need to enter the following address parameters:

Parameter	Function	Example of a value
Network ID	Network ID of the subnet to be enabled for VPN tunnel communication. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	192.168.11.0
Subnet mask	The subnet mask determines the network structure. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	255.255.255.0
Comment	Entry of additional, optional comments.	

Router and firewall redundancy

Overview

Meaning

Failures of the security modules SCALANCE S623 as of V4 and SCALANCE S627-2M as of V4 can be automatically compensated by routers and firewall redundancy during operation. To do this, group two security modules of the type SCALANCE S623 or SCALANCE S627-2M in a redundancy relationship by activating routers and firewall redundancy for both security modules. Following this, you decide which security module of the redundancy relationship is passive in normal mode (secondary module). You make this setting for the security module of the redundancy relationship that is active in normal operation (primary module). If the primary module fails during operation, the secondary module automatically adopts the function as firewall and (NAT/NAPT) router. To ensure the identical configuration of both security modules, these are connected together via their DMZ interfaces and their configurations are synchronized during operation. In this case, the DMZ interfaces of the security modules involved cannot be used for other purposes.

Address redundancy

In addition to their module IP addresses, the two security modules share a common IP address on the external and on the internal interface so that if one of the security modules fails, the IP addresses do not need to be changed. To do this, you need to configure an IP address for the external and for the internal interface of the redundancy relationship.

Effects of redundancy relationships on security modules

When you create redundancy relationships between security modules, some properties of these security modules are automatically adapted to establish compatibility with the redundancy relationship. The following properties are affected by this adaptation:

Module property	Effect on the module property
Operating mode	Where necessary, the mode is changed to the "Routing mode" option.
Membership of VPN groups	Where necessary, the security module is removed from VPN groups.
Interface configuration	Where necessary, the external interface and the DMZ interface of the security module are enabled. Where necessary, the IP assignment method "Static address" is configured for all interfaces.

Configuration of security modules in redundancy relationships

After router and firewall redundancy have been enabled and the primary module of the redundancy relationship selected, some of the module properties are configured only using the primary module. The properties configured for the primary module then apply to the redundancy relationship and cannot be configured for the secondary module. The following properties can be configured for the redundancy relationship:

- Basic settings of the redundancy relationship (secondary module, network parameters)
- Firewall (standard rules for IP services are configured separately for the individual security modules).
- Routing
- NAT/NAPT routing (not 1:1 NAT)

The values of the properties listed above are initially adopted from the primary module for the redundancy relationship.

The settings listed below remain active for the individual security modules even after including them in a redundancy relationship. Configuring these properties for the primary module therefore has no effect on the secondary module.

- Interface configuration (disabling interfaces and changing the VIP assignment method "Static address" is not possible).
- Standard rules for IP services (firewall)
- DDNS
- Time-of-day synchronization
- Log settings
- SNMP
- RADIUS

Note

Loading a configuration on security modules of a redundancy relationship (only SCALANCE S623/S627-2M as of V4)

The properties of a redundancy relationship configured for the primary module must be loaded both on the primary module and on the secondary module. To load the configuration, the physical IP address via which your engineering station can reach the security module must be used. The virtual IP addresses of the redundancy relationship cannot be used for loading.

Creating redundancy relationships between security modules

Requirement

The security modules SCALANCE S623/S627-2M as of V4 are not assigned to any other redundancy relationship.

Procedure

1. Select the security module that will be the active security module in normal operation (primary module).
2. In the local security settings, select the entry "Router and firewall redundancy".
3. Select the "Router and firewall redundancy" check box.
4. In the "Secondary module" drop-down list, select the security module that will be the passive security module in normal operation.

Result: You have created a redundancy relationship between the security modules.

Configuring redundancy relationships

How to access this function

1. Select the primary module of the redundancy relationship.
2. In the local security settings, select the entry "Router and firewall redundancy".

Configuring network parameters of the redundancy relationship

Configurable parameter	Meaning
IP address	IP address of the virtual external or internal interface of the redundancy relationship. The IP address must be located in the external or internal subnet of the primary module.
Subnet mask	Subnet mask of the virtual external or internal interface of the redundancy relationship
MAC address (adaptable only for SCALANCE S623/ S627-2M as of V4.0.1)	MAC address of the virtual external or internal interface of the redundancy relationship

For general information on configuring network parameters, refer to the following section:
Configuring IP address parameters (Page 774)

Configuring the firewall

IP packet filter rules for redundancy relationships are configured on the primary module. The communications directions "From external to internal" and "From internal to external" are available.

For general information on configuring IP packet filter rules in advanced firewall mode, refer to the following section:
Defining IP packet filter rules (Page 712)

Configuring address translation with NAT/NAPT

Address translation with NAT/NAPT for the redundancy relationship is configured on the primary module. For redundancy relationships, only Source NAT and NAPT can be configured. With Source NAT, source IP addresses in the internal subnet can only be replaced with the virtual external IP address of the redundancy relationship. No alias IP addresses can be registered on the external interface of the redundancy relationship. With NAPT, only the "External to internal" address translation direction can be configured.

For general information on configuring address translations with NAT/NAPT, refer to the following section:
Overview of NAT/NAPT (Page 724)

Configuring routing

Routes for the redundancy relationship are configured on the primary module. Standard routers must be specified in the "External interface [P1] red" or "Internal interface [P2] green" entry and must be identical per interface.

For general information on configuring routing, refer to the following section:
Specifying routes (Page 801)

Online functions - diagnostics and logging

Overview of the individual interfaces - "Interface settings" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Meaning

Table 10-43 Online diagnostics: "Interface settings" entry

System and status functions	Meaning
Interfaces	Table above: General overview of the interfaces of the security module. Table below: Information on the interface operated via PPPoE. • Status: Indicates whether or not a connection was established to the Internet Service Provider (ISP). • Current IP address: Current IP address of the interface • Gateway: IP address of the gateway • Primary dynamic DNS service: IP address of the primary dynamic DNS service • Secondary dynamic DNS service: IP address of the secondary dynamic DNS service • Error code (numeric): Error information if no connection could be established to the ISP.
CARP (only SCALANCE S623/S627-2M as of V4)	• CARP interface: Display of the virtual CARP interface • Physical interface: Physical interface on which the virtual CARP interface is operated (external / internal). • Status: Displays which of the modules of the redundancy relationship is active. • MAC address: MAC address of the virtual CARP interface • Preferred: Displays which of the modules of the redundancy relationship is configured as the primary module.

System and status functions	Meaning
Media redundancy (SCALANCE S627-2M only)	• Interface: Interface connected to the MRP ring. • Protocol: Protocol used (MRP) • Ring port 1: Name of the first media module port of the interface connected to the MRP ring. • Ring port 2: Name of the second media module port of the interface connected to the MRP ring. • Domain name: Name of the MRP domain. • Discrepancy: Display whether the domain of the client differs from that of the redundancy manager. • Domain UUID: UUID of the MRP domain.
Media modules (SCALANCE S627-2M only)	• Port: Port ID(s) of the media module port(s) • Name: Name of the media module • MLFB: MLFB of the media module • Revision: Version of the media module • Discrepancy: Displays whether there are differences between the configuration data in STEP 7 and the media modules.

Overview of the Dyn. DNS settings - "Dynamic DNS" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Meaning

Table 10-44 Online diagnostics: "Dynamic DNS"

System and status functions	Meaning
Client status	Indicates whether or not a connection was established to a dyn. DNS server.
Current IP address	WAN IP address via which the security module can currently be reached.
Defined IP address	IP address currently assigned to the FQDN.
Current time	The current time-of-day.
Force update	The security module obtains the current IP address of its Internet access and sends an update query to the configured update server(s). This publishes the current IP address on the Internet. The status is displayed in the primary or secondary dynamic DNS service fields. This means, for example that it is possible to check whether configured data such as the user name and password of the dynamic DNS account are correct.
Cancel suspension	Cancels the suspension (blocking of IP address updates from the dynamic DNS provider on the security module), for example after the dynamic DNS provider password has been corrected or an error has been eliminated.
Primary and secondary dynamic DNS service	
FQDN	Fully Qualified Domain Name registered with the provider.

System and status functions	Meaning
Server IP Address	IP address of the update server used.
Successful update	Successful update in the dynamic DNS update service.
Last update attempt	Last update attempt for DynDNS update service.
Last failed update	Last update error in the dynamic DNS update service.
Error code	Error status of the last unsuccessful update attempt during a dynamic DNS update.

What is the meaning of the messages?

The messages of the last DDNS update attempt have the following meaning:

Message	Meaning
Success	
DDNS_OK	The update query was successful.
Connection-related status messages	
DDNS_E_CON_UDP_SRV_RESOLV_ERR	DNS name of the update server unknown, FQDN cannot be resolved using the known DNS server.
DDNS_E_CON_UDP_SRV_UNREACHABLE	Update server unreachable ("Timeout").
Security-related status messages (errors)	
DDNS_E_CERT_SUBJECT_INVALID	The common name of the subject in the certificate does not match the domain name of the update server or its IP address.
DDNS_E_CERT_UNABLE_TO_GET_ISSUER_CERT	Issuer certificate not found. The certificate chain could not be followed back to the root CA because an issuer certificate was not found. The trust chain is incomplete.
DDNS_E_CERT_SIGNATURE_INVALID	The signature of a certificate could not be read or is invalid.
DDNS_E_CERT_NO_TRUST	A certificate in the trust chain is invalid, in other words: • Not yet valid or ready expired • V3 extensions invalid • Critical V3 extension is not supported
DDNS_E_CERT_DEPTH_ZERO_SELFSIGNED_CERT	The update server has supplied a self-signed certificate and the certificate is not in the certificate store for trustworthy root CA certificates.
DDNS_E_CERT_SELF_SIGNED_CERT_IN_CHAIN	The certificate chain could be established using non trustworthy certificates but no suitable root CA certificate was found in the certificate store for trustworthy certificates.
DDNS_E_CERT_CHAIN_TOO_LONG	The certificate chain exceeds the maximum supported verification depth.
DDNS_E_CERT_INVALID_CA	A CA certificate is invalid, in other words expired, not yet valid or the V3 extensions are not suitable for the intended purpose (for example CA not set to TRUE for CA certificates).
DDNS_E_CERT_KEYUSAGE_UNSUITED	The V3 extensions key usage or extended key usage set in a certificate of the trust chain are not suitable for the use of the certificate.

Message	Meaning
DDNS_E_CERT_EXTENSION_UNSUPPORTED	A certificate in the trust chain used an extension marked as critical that is not supported.
Agent-related status messages (errors)	
DDNS_E_AGT_BAD_AGENT	• The update request does not correspond to the structure required by cRSP, for example URL parameters missing. • The update request was sent to an illegal URL on the update server. • The configured update string contains errors.

Display of the ARP table - "ARP table" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Meaning

Display of the ARP table of the security module.

Table 10-45 Online diagnostics: "ARP table" tab

System and status functions	Meaning
ARP table	Display of the static (proxy ARP) and dynamic entries of the ARP table on the security module. The "Publication type" tab specifies whether the entry is configured statically or learnt.

Users logged in to the Web page - "Logged in users" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Meaning

Shows the users logged in to the Internet page for user-specific IP rule sets.

System and status functions	Meaning
User name	Name of the logged on user.
Source IP address	IP address with which the user logged on.
Time remaining	Remaining time before the user is automatically logged off.

System and status functions	Meaning
Maximum time of the session	Configured total time of the session.
Log off	The selected user is logged off.

Display of the firewall blacklist - "Firewall blacklist" entry

Meaning

Displays the IP addresses of nodes that have exceeded the permitted number of connections and firewall statuses per time unit. These nodes are entered in the IP blacklist of the firewall.

The number of connections and firewall statuses per time unit is only limited if the "Use extended status options" check box is selected in the "Standard rules for IP services" entry in the local security settings.

If you click the "Delete all" button, the displayed IP addresses are removed from the firewall blacklist of the security module. The IP addresses are also no longer displayed.

Setting the date and time - "Date and time" entry

How to access this function

1. Select the security module whose time and date you want to check or set.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. In online diagnostics, select the "Functions" > "Date and time" entry.

Setting the local time on the security module

In this area, you can read out and set the time and date of the security module. When you click the "Apply" button, the security module is assigned the time and date currently entered in the "Date" and "Time" input boxes.

Setting the local time on PC

This area shows the current time and current date of the PC on which STEP 7 is installed. If you click the "Adopt for module" button, the security module is assigned the current time and current date of the PC.

Diagnostics in ghost mode - "Ghost mode" entry

Module-specific function

This function is available only for SCALANCE S602 as of V3.1.

Meaning

Display of address information as well as information on IP address changes of the internal node.

System and status functions	Meaning
Status of SCALANCE S602	Display of the status of the security module in relation to operation in ghost mode.
IP address	IP address of the internal node (identical to the external IP address of the security module).
Subnet mask	Subnet mask of the security module.
MAC address	MAC address of the internal node.
Node found on	Displays when the internal node was recognized by the security module or when an IP address change was made on the internal node.
Number of IP address changes	Number of IP address changes detected by the security module.
IP address	IP address in the external network for which the security module requires route information.
Standard router	Standard router for the IP address in the external network.

Download functions

Downloading a configuration

Downloading the configuration / establishing an online connection

1. From the "PG/PC interface" drop-down list of the "Advanced download" dialog or "Connect online", select the network adapter via which you can reach the module.
2. If the module is in the factory settings status, follow the steps below:
   - From the "Connection to interface/subnet" drop-down list, with which your engineering station is connected and for which the IP address to be assigned is configured in the local security settings.
   - Select the "Show all compatible nodes" check box.
   - Click the "Start search" button.
   - Result: The module is displayed in the "Compatible devices in target subnet" table with its detected MAC address.
   - Select the module entry in the table and click the "Assign IP address" button.
   - Result: The module is assigned the IP address configured in the local security settings for the selected interface.
3. If the module is not in the factory settings status, follow the steps below:
   - From the "Connection with interface/subnet", select the interface / the FQDN address / WAN address of the module via which your engineering station can reach the module. STEP 7 then uses the address configured in the local security settings for the selected component for access to the module.
   - Deselect the "Show all compatible devices" check box.
   - Click the "Start search" button.
   - Result: The "Address" column of the "Compatible devices in the target subnet" displays the detected IP address / FQDN address of the module.
   - Select the address entry in the table and click the "Load" or "Go online" button.

Ideally, you should configure the modules of a group via the common external network of these modules (interface X1). If the engineering station is located in an internal network, you will

need to enable the IP addresses of the other modules of the group explicitly in the firewall of this SCALANCE S and configure this module first.

Note

Downloading the configuration when operating in ghost mode (only SCALANCE S602 as of V3.1)

When you operate the security module in ghost mode, the external interface of the security module takes over the IP address of the internal node at runtime. Before you can download a new configuration via the external interface to the security module, you need to specify the IP address for downloading a configuration that the security module obtained from the internal node during runtime.
To find out the current IP address of the security module, you can search for reachable nodes in STEP 7 with the menu command "Online" > "Accessible devices".

Note

Loading a configuration on security modules of a redundancy relationship (only SCALANCE S623/S627-2M as of V4)

The properties of a redundancy relationship configured for the primary module must be loaded both on the primary module and on the secondary module. To load the configuration, the physical IP address via which your engineering station can reach the security module must be used. The virtual IP addresses of the redundancy relationship cannot be used for loading.

Specifying a different address

In the "Compatible devices in target subnet" dialog area, you have the option of specifying an IP address / an FQDN address that differs from the IP address / FQDN address in the local security settings. To do this, enter the IP address / FQDN address of the module in the editable cell in the "Address" table column.

Firmware version

The configuration of a SCALANCE S as of V3 can also be downloaded to a SCALANCE S module whose firmware version is higher than the firmware version of the SCALANCE S module in STEP 7.

Operating mode

Configurations can be downloaded while the SCALANCE S modules are operating. Restart the SCALANCE S module to activate your configuration changes.

Note

Special characteristics

- As long as a module has not yet set IP parameters (in other words, prior to the first configuration), there must be no router between the module and the configuration computer.
- If you swap a PC from the internal to the external interface of the SCALANCE S, access from this PC to the SCALANCE S is blocked for approximately 20 minutes.

Configuration status

Prior to each download, the existing configuration on the security module is checked and compared with the configuration to be downloaded from the STEP 7 project. If the configuration of the module originates from the STEP 7 project currently to be downloaded and there are differences between these configurations, it is possible to download only files with differences between the module and project configuration to the security module. In some situations, this can speed up the download.

Transferring firmware

This needs to be taken into consideration before transferring new firmware

To transfer new firmware to a security module, the following conditions must be met:

- You have the rights required to transfer firmware; refer to section Auto-Hotspot.
- The security module is configured with an IP address.

The transfer is secure

The firmware is transferred over a secure connection and can therefore also be transferred from the unprotected network.

The firmware itself is signed and encrypted. This ensures that only authentic firmware can be downloaded to the SCALANCE S module.

Restart necessary following transfer

Newly downloaded firmware only becomes active after the SCALANCE S module has been restarted. If the transfer is disturbed and aborted, the module starts up again with the old firmware version.

Security for S7-300 /S7-400 / PC CPs

Setting up a firewall

Local firewall rules for S7-300 /S7-400 / PC CPs

Overview S7-300 CPs / S7-400 CPs /PC CPs

Enabling packet filter rules

If you enable the security function for the CPs in the local security settings, initially all access to and via the CP is permitted. To enable individual packet filter rules, select the "Activate firewall" check box. Then enable the required services. Firewall rules created automatically due to a connection configuration have priority over rules set manually.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Activate firewall in advanced mode" check box.

Firewall configuration with VPN

If the security module is added to a VPN group, the firewall is enabled by default. In addition, the "Tunnel communication only" check box is enabled. This means that only encrypted IPsec data transfer is permitted via the external interface. External data traffic is blocked.

If you deselect the check box, tunneled communication and also the types of communication selected in the other boxes are permitted.

Updating connection rules

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button. The modified firewall rules are then displayed in advanced firewall mode.

Configuring a firewall with predefined firewall rules - CP x43-1 Advanced

Configuring a firewall with predefined IP rules - CP x43-1 Advanced

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Firewall" > "Predefined IP rules".

Table 10-46 Available services and directions

Service	From station/ internal to external	External to internal	From external to station	Enabled ports	Meaning
Allow IP communication	x	x	x	All	IP traffic for the selected communication directions is allowed.
Allow S7 protocol	x	x	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/ FTPS (explicit mode)	x	x	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	x	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	x	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow DNS	x	x	-	TCP port 53 UDP port 53	Communication connection to a DNS server is allowed.
Allow SNMP	x	x	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	x	-	TCP port 25	For sending e-mails via an SMTP server.
Allow NTP	x	x	-	UDP port 123	For synchronization of the time of day.

Table 10-47 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules - CP x43-1 Advanced

How to access this function

1. Select the module to be edited.
2. Select the entry "Firewall" > "Predefined MAC rules".

Table 10-48 Available services and directions

Service	From station to external	From external to station	Meaning
Allow MAC communication	x	x	The MAC traffic from station to external and vice versa is allowed.
Allow ISO protocol	x	x	The ISO traffic from station to external and vice versa is allowed.

Table 10-49 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined firewall rules - CP 1628

Configuring a firewall with predefined IP rules - CP 1628

How to access this function

1. Select the module to be edited.
2. Select the "Security" > "Firewall" > "Predefined IP rules" entry.

Table 10-50 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/FTPS (explicit mode)	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow DNS	x	TCP port 53 UDP port 53	Communication connection to a DNS server is allowed.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	TCP port 25	For sending e-mails via an SMTP server.
Allow NTP	x	UDP port 123	For synchronization of the time of day.

Table 10-51 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules - CP 1628

How to access this function

1. Select the module to be edited.
2. Select the entry "Security" > "Firewall" > "MAC rules".

Table 10-52 Available services and directions

Service	From station to external	From external to station	Meaning
Allow MAC level communication	x	x	The MAC traffic from external to the station and vice versa is allowed.
Allow ISO communication	x	x	ISO traffic from external to the station and vice versa is allowed.
Allow SiClock	x	x	SiClock time-of-day frames from external to the station and vice versa are allowed.

Table 10-53 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		**Action**	**From**	**To**
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

IP packet filter directions S7-300-/S7-400-/PC-CPs

Meaning

Possible selections for the communication directions "From" and "To" in the IP rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		Meaning
From	To	CP x43-1 Adv.	CP 1628	
Internal	Station	x	-	Access from the internal network to the station.
	Any	x	-	Access from internal to the external network, VPN tunnel partner and the station.
External	Station	x	x	Access from the external network to the station.
	Any	x	-	Access from external to the internal network and the station.
Station	Internal	x	-	Access from the station to the internal network.
	External	x	x	Access from the station to the external network.
	Tunnel	x	x	Access from the station to the VPN tunnel partner.
Tunnel	Station	x	x	Access via the VPN tunnel partner to the station.
	Any	x	-	Access from VPN tunnel partners to the internal network and the station.
Any	External	x	-	Access from the internal network and the station to the external network.

MAC packet filter directions S7-300-/S7-400-/PC-CPs

Context

Possible selections for the communication directions "From" and "To" in the MAC rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		Meaning
From	To	CP x43-1 Adv.	CP 1628	
External	Station	x	x	Access from the external network to the station.
Station	External	x	x	Access from the station to the external network.
	Tunnel	x	x	Access from the station to the VPN tunnel partner.
Tunnel	Station	x	x	Access via the VPN tunnel partner to the station.

Configuring the access list

Module-specific function

This function is not available for the CP 1628.

Meaning

You set access protection for certain IP addresses using the IP access lists. List entries that have already been created and the corresponding rights are displayed in the local settings of the CP in the entry "Firewall" > "IP rules" (advanced firewall mode).

Note

Changed behavior after activation of security

- Once you have activated the security function for a CP, access protection will only apply to the external interface. You can apply access protection to the internal interface as well, by configuring suitable firewall rules in advanced firewall mode.
- The CP also responds to ARP requests from IP addresses that have not been released (layer 2).
- If the IP access list of a CP contains no entries and you activate security for the CP, the firewall will be activated and prevent access to the CP from external locations. Configure the corresponding firewall rules in advanced firewall mode so that the CP can be reached.

Effect of IP access list entries at activation of security

If security is enabled in the local settings of a CP, the corresponding rules are created in the advanced firewall mode. A firewall rule "Allow" > "External" > "Station" is created for an IP address you specified in the address list. The IP address from the IP access list is used accordingly as source IP address. IP addresses from a defined IP address range are also integrated into corresponding firewall rules.

Requirements for editing

Before you can edit created firewall rules, the following condition must be met:

- for editing using STEP 7: "Configure security" configuration right
- for editing using a Web server: Module right "Web: Expand IP access control list"

The requirements for editing the IP access lists outside the local security settings are described in the sections on the specific CPs.

Connection-related automatic firewall rules

Meaning

For connections that were configured using CPs, STEP 7 automatically creates firewall rules that allow communication with the partner of the CP in the specified direction (CP active/ passive). The connection establishment directions are taken into account. To display these firewall rules, if the advanced firewall mode is enabled, the "Update connection rules" button needs to be clicked. The firewall rules are then displayed in advanced firewall mode.

Note

Enabling UDP multicast and UDP broadcast connections manually

No automatic firewall rules are created for UDP multicast and UDP broadcast connections. To enable the connections, add the relevant firewall rules manually in advanced firewall mode.

Depending on how the connection establishment is configured, the following level 3 firewall rules are created. If the security module is in a VPN group, the direction "External" changes to "Tunnel". This applies only to CPs that support VPN.

The IP address of the connection partner is entered in the "Source IP address" or "Destination IP address" column of these firewall rules.

CP->external	Action	From	To
active	Allow	Station	External
	Drop	External	Station
passive	Drop	Station	External
	Allow	External	Station
active and passive	Allow	External	Station
	Allow	Station	External

CP->internal	Action	From	To
active	Allow	Station	Internal
	Drop	Internal	Station
passive	Drop	Station	Internal
	Allow	Internal	Station
active and passive	Allow	Internal	Station
	Allow	Station	Internal

For level 2 connections, "Allow" rules are created for both directions. If the security module is in a VPN group, the direction "External" changes to "Tunnel".

The MAC address of the connection partner is entered in the "Source MAC address" or "Destination MAC address" column of these firewall rules.

CP->external	Action	From	To
active, passive, active and passive	Allow	Station	External
	Allow	External	Station

Note

Changing the connection configuration

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button.

Conventions for automatically created firewall rules

- Priority
  The rules have highest priority and are therefore inserted at the top in the local rule list.
- Deleting rules
  The rules cannot be deleted. Logging can be enabled and services can be assigned. Moreover, you may insert a bandwidth and a comment.
- Changing the action
  If you set the action from "Allow" to "Drop" or vice versa, this is overwritten again during renewed system synchronization. Select "Allow*" or "Drop*" to retain your changes. In this case, only the IP address is synchronized and the action and direction remain as set. Settings for logging, service, bandwidth and comment are also retained after a renewed system synchronization even without changing the action to "Allow*" or "Drop*". If the configured connection is deleted, the corresponding rules are removed from the list.

Security module in VPN group

As default, the "Tunnel communication only" check box is enabled. If you deselect the check box, in addition to tunnel communication between tunnel partners, communication is also possible with network nodes to which there is no tunnel.

- Communication is untunneled if the partner address belongs to a station known in STEP 7 for which no VPN tunnel is configured.
- Communication is through the tunnel if the partner address is a VPN endpoint.
- If it is not clear whether connection should bypass or run through the VPN tunnel, the connection is assigned to the VPN tunnel and a message to this effect is displayed. The assignment can be adapted in advanced firewall mode, for example, by changing the "From" direction "Tunnel" to "External". To avoid this adaptation being overwritten by the next system synchronization, the "Allow*" or "Drop*" action must be selected.

Note

If you want to ensure that only communication through the tunnel is possible, you will need to create suitable firewall rules in advanced firewall mode, for example, for internal nodes or NDIS addresses.

To allow only tunneled communication for a CP, add a rule with the following settings:

- "Action": "Drop"
- "From": "Any"
- "To": "External"

For the CP 1628, add a rule with the following settings:

- "Action": "Drop"
- "From": "Station"
- "To": "External"

In addition to this, you need to remove existing firewall rules that allow untunneled communication.

Activating the Web server on CP x43-1 Advanced

Module-specific function

This function is only available for CP x43-1.

Meaning

After activating the Web server, you have access to the Web pages of the module. In the local security settings, you can restrict access to these Web pages using the HTTPS protocol. This access is controlled using the "Allow access only using HTTPS" check box. In addition, you must configure the firewall accordingly.

IPsec tunnel: Creating and assigning groups

Configuring internal network nodes - "Nodes" entry

Allow access to S7-300 / S7-400 CPs for VPN connection partners

Possible selections

Decide whether or not the VPN connection partner can have access to the CP and/or the internal subnet of the CP in routing mode (SCALANCE S / M).

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Node".
3. Configure access for the VPN connection partner of the CP in routing mode (SCALANCE S / M):
   - Allow connection to the CP (Gbit interface)
   - Allow connection to the internal subnet (PROFINET subnet)

Configuring NDIS nodes manually for PC CPs that can be reached through the tunnel

Configuring NDIS nodes that can be reached through the tunnel

The internal nodes are learned and assigned to the routes dynamically. This concerns the NDIS IP addresses of the Windows PC.

Follow the steps below

1. Select the module to be edited.
2. In the local security settings, select the entry "Nodes" > "NDIS nodes reachable via tunnel".
3. Enter the NDIS IP addresses.

Online functions - Debug / Diagnostics and Logging

Updated firewall rules - "Dynamically updated firewall rules" entry

Module-specific function

This function is only available for CP x43-1 Adv.

Meaning

Display of the IP addresses or IP address ranges that were enabled dynamically using HTTP or HTTPS, or loaded by a user. The rights assigned for accessing the S7 CP are displayed for the enabled IP addresses. An update of the IP addresses in this tab can only be triggered by the following events:

- Extension/modification of the IP access control list
- Update of firewall rules
- Dynamic extensions transmitted to the CP at runtime, for example, PROFINET IO devices

Since only the dynamically updated firewall rules are displayed here, you also need to take into account the firewall rules that were configured offline and downloaded to the station for a full picture of the current firewall status of the module.

Security for S7-1200-/S7-1500-CPs

Setting up a firewall

Local firewall rules for S7-1200 / S7-1500 CPs

Overview of the local firewall rules for S7-1200 / S7-1500 CPs

Enabling packet filter rules

If you enable the security function for the CPs in the local security settings, initially all access to and via the CP is permitted. To enable individual packet filter rules, select the "Activate firewall" check box. Then enable the required services. Firewall rules created automatically due to a connection configuration have priority over rules set manually.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Activate firewall in advanced mode" check box.

Updating connection rules

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button. The modified firewall rules are then displayed in advanced firewall mode.

Configuring a firewall with predefined firewall rules - CP 1543-1

Configuring a firewall with predefined IP rules - CP 1543-1

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Firewall" > "Predefined IP rules".

Table 10-54 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/FTPS (explicit mode)	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow security diagnostics	x	TCP port 8448	Allow security diagnostics.

Table 10-55 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		**Action**	**From**	**To**
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined IPv6 rules - CP 1543-1

Meaning

With the predefined IPv6 rules, you have the option of configuring the firewall for services in which IPv6 is used. By enabling a predefined IPv6 rule in the local security settings of the CP 1543-1 V1.1, the system-defined ICMPv6 services that can be seen in the global security settings in the "ICMP" tab in "Firewall" > "Services" > "Define services for IP rules"are also enabled in the firewall. The firewall of the CP 1543-1 V1.0 allows ICMPv6 packets through even without enabling a predefined IPv6 rule.

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" > "Pre-defined IPv6 rules" item in the local security settings.

Table 10-56 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/FTPS (explicit mode)	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.

Table 10-57 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings made in "Predefined IPv6 rules" have no effect on firewall rules that were automatically generated as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules - CP 1543-1

How to access this function

1. Select the module to be edited.
2. Select the entry "Firewall" > "Predefined MAC rules".

Table 10-58 Available services and directions

Service	From station to external	From external to station	Enabled ports	Meaning
Allow MAC communication	x	x	-	The MAC traffic from external to the station and vice versa is allowed.
Allow ISO protocol	x	x	-	ISO traffic from external to the station and vice versa is allowed.
Allow DCP	x	x	-	DCP traffic from external to the station and vice versa is allowed.
Allow LLDP	x	x	-	LLDP traffic from external to the station and vice versa is allowed.

Table 10-59 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined firewall rules - CP 1243-1 and CP 1243-7

Configuring a firewall with predefined IP rules - CP 1243-1 and CP 1243-7

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Firewall" > "Predefined IP rules".

Table 10-60 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow HTTP	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow security diagnostics	x	TCP port 8448	Allow security diagnostics.

Table 10-61 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined IPv6 rules - CP 1243-1

Meaning

With the predefined IPv6 rules, you have the option of configuring the firewall for services in which IPv6 is used. By enabling a predefined IPv6 rule in the local security settings of the CP 1243-1, the system-defined ICMPv6 services that can be seen in the global security settings in the "ICMP" tab in "Firewall" > "Services" > "Define services for IP rules" are also enabled in the firewall.

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" > "Predefined IPv6 rules" item in the local security settings.

Table 10-62 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.

Table 10-63 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings made in "Predefined IPv6 rules" have no effect on firewall rules that were automatically generated as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules - CP 1243-1 and CP 1243-7

How to access this function

1. Select the module to be edited.
2. Select the entry "Firewall" > "Predefined MAC rules".

Table 10-64 Available services and directions

Service	From station to external	From external to station	Enabled ports	Meaning
Allow MAC communication	x	x	-	The MAC traffic from external to the station and vice versa is allowed.
Allow DCP	x	x	-	DCP traffic from external to the station and vice versa is allowed.

Table 10-65 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

IP packet filter directions S7-1200 / S7-1500 CPs

Meaning

Possible selections for the communication directions "From" and "To" in the IP rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Meaning
From	**To**	
External	Station	Access from the external network to the station.
Station	External	Access from the station to the external network.
	Tunnel	Access from the station to VPN tunnel partners.*
Tunnel	Station	Access by VPN tunnel partners to the station.*

* Not for the CP 1543-1 V1.0.

MAC packet filter directions S7-1200 / S7-1500 CPs

Meaning

Possible selections for the communication directions "From" and "To" in the MAC rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Meaning
From	**To**	
External	Station	Access from the external network to the station.
Station	External	Access from the station to the external network.
	Tunnel	Access from the station to VPN tunnel partners.*
Tunnel	Station	Access by VPN tunnel partners to the station.*

* Not for the CP 1543-1 V1.0.

Connection-related automatic firewall rules

Meaning

For connections that were configured using CPs, STEP 7 automatically creates firewall rules that allow communication with the partner of the CP in the specified direction (CP active/ passive). The connection establishment directions are taken into account. To display these firewall rules, if the advanced firewall mode is enabled, the "Update connection rules" button needs to be clicked. The firewall rules are then displayed in advanced firewall mode.

Note

Enabling UDP multicast and UDP broadcast connections manually

No automatic firewall rules are created for UDP multicast and UDP broadcast connections. To enable the connections, add the relevant firewall rules manually in advanced firewall mode.

Depending on how the connection establishment is configured, the following level 3 firewall rules are created. If the CP (not CP 1543-1 V1.0) is in a VPN group, the direction "External" changes to "Tunnel".

The IP address of the connection partner is entered in the "Source IP address" or "Destination IP address" column of these firewall rules.

CP->external	Action	From	To
active	Drop	External	Station
	Allow	Station	External
passive	Drop	Station	External
	Allow	External	Station
active and passive	Allow	External	Station
	Allow	Station	External

For level 2 connections, "Allow" rules are created for both directions. If the CP (not CP 1543-1 V1.0) is in a VPN group, the direction "External" changes to "Tunnel".

The MAC address of the connection partner is entered in the "Source MAC address" or "Destination MAC address" column of these firewall rules.

CP->external	Action	From	To
active, passive, active and passive	Allow	Station	External
	Allow	External	Station

Note

Changing the connection configuration

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button.

Conventions for automatically created firewall rules

- Priority
  The rules have highest priority and are therefore inserted at the top in the local rule list.
- Deleting rules
  The rules cannot be deleted. Logging can be enabled and services can be assigned. Moreover, you may insert a bandwidth and a comment.
- Changing the action
  If you set the action from "Allow" to "Drop" or vice versa, this is overwritten again during renewed system synchronization. Select "Allow*" or "Drop*" to retain your changes. In this case, only the IP address is synchronized and the action and direction remain as set. Settings for logging, service, bandwidth and comment are also retained after a renewed system synchronization even without changing the action to "Allow*" or "Drop*". If the configured connection is deleted, the corresponding rules are removed from the list.

Security module in VPN group

As default, the "Tunnel communication only" check box is enabled. If you deselect the check box, in addition to tunnel communication between tunnel partners, communication is also possible with devices to which there is no tunnel.

- Communication is untunneled if the partner address belongs to a station known in STEP 7 for which no VPN tunnel is configured.
- Communication is through the tunnel if the partner address is a VPN endpoint.

Note

If you want to ensure that only communication through the tunnel is possible, you will need to create suitable firewall rules in advanced firewall mode.

To allow only tunneled communication for a CP, add a rule with the following settings:

- "Action": "Drop"
- "From": "Station"
- "To": "External"

In addition to this, you need to remove existing firewall rules that allow untunneled communication.

10.1.4 Creating configurations

10.1.4.1 Information about the web server

Introduction

The Web server lets you monitor and administer the CPU through authorized users by means of a network. This permits evaluation and diagnostics over long distances. All you need is a web browser.

Alarms and status information are visualized on HTML pages.

Web browser

You need a web browser to access the HTML pages of the CPU.

The following web browsers have been tested for communication with the CPU:

- Internet Explorer (Version 8)
- Mozilla Firefox (Version 21)
- mobileSafari (iOS5)

Web access to the CPU via PG/PC

Proceed as follows to access the Web server:

1. Connect the client (PG/PC) to the CPU via the PROFINET interface.
2. Open the web browser.
   Enter the IP address of the CPU in the "Address" field of the web browser in the format http://ww.xx.yy.zz (example: http://192.168.3.141).
   The start page of the CPU opens. From the start page, you can navigate to further information.

Additional information

Additional information about the Web server of the various CPU families is available under the key word "Web server" in the information system.

Information on creating your own web pages for access to the CPU is available under the keyword "User-defined web pages" in the information system.

You can find links to additional manuals about the subject "Web server" under "See also".

See also

S7-1500 web server (http://support.automation.siemens.com/WW/view/en/59193560)

Documentation S7-300 (http://support.automation.siemens.com/WW/view/en/12996906)

Documentation S7-400 (http://support.automation.siemens.com/WW/view/en/44444467)

S7-1200 web server (http://support.automation.siemens.com/WW/view/en/36932465)

10.1.4.2 Things you should know about PROFIBUS DP operating modes

Introduction

DP master systems that consist of a DP master and DP slaves which are connected via a bus and communicate with one another via the PROFIBUS DP protocol are referred to as distributed I/O.

Below, we refer to communication-capable modules with DP interface that can take on the role of DP master or DP slave.

"DP master" and "DP slave" option

Communication-capable modules, such as CPUs with DP interface and CPs or CMs with DP interface, have the area "Mode" in their module properties.

For S7-300 CPUs with integrated DP interface, for example, you can set the mode "DP master" and "DP slave". A CPU or a CP that is configured as DP slave is also referred to as intelligent DP slave (I-slave).

For S7-1500 CPUs with integrated DP interface, only the "DP master" mode is possible. To operate S7-1500 CPUs as I-slave, you have to insert the communication module CM 1542-5 and configure it as DP slave.

S7-1200 CPUs do not have integrated DP interfaces. To operate an S7-1200 as DP master or DP slave, you must insert a communication module CM 1243-5 (DP master only) or a communication module CM 1242-5 (DP slave only; I-slave).

Additional information

Additional information on distributed I/O is available under the key word "Distributed I/O" and "I-slave" in the information system.

10.1.4.3 Configuring automation systems

Addressing modules

Addressing modules

Introduction

In the device overview, you see the addresses or address ranges of the modules in the I address and Q address columns. There are other addresses as well, which are explained below.

I/O address

I/O addresses (input/output addresses) are required to read inputs and set outputs in the user program.

Input and output addresses are assigned automatically when modules are inserted in the rack. The address of the first channel is the start address of a module. The addresses of the other channels are derived from this start address. The address end is obtained from the module-specific address length.

Device address (e.g., Ethernet address)

Device addresses are addresses of programmable modules (Industrial Ethernet addresses). They are required to address the different stations of a subnet, for example, to download a user program to a CPU.

Hardware identifier for identifying modules and submodules

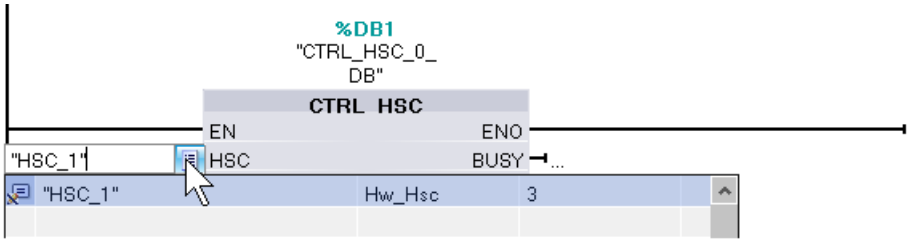
In addition to the I addresses and Q addresses, a hardware identifier (HW ID) is assigned automatically and is used to address and identify the module. Submodules (units of a module), such as an integrated counter, also receive such a hardware identifier.

The hardware identifier consists of an integer and is output by the system with diagnostics alarms to allow the faulty module or the faulty submodule to be localized.

In addition, the hardware identifier is used for a number of instructions to address the corresponding module.

The hardware identifier cannot be changed.

Example: Identifying high-speed counters of the S7-1200 CPU



The hardware identifier is assigned automatically when components are inserted in the device or network view and in the PLC tags. A name is also assigned automatically for the hardware identifier. The system constants of the PLC tags cannot be changed either.

See also

Specifying input and output addresses (Page 849)

Assigning addresses to a location in the program (Page 850)

Introduction to loading a configuration (Page 1146)

Inspector window (Page 547)

Specifying input and output addresses

Default input and output addresses are set automatically. You can, however, change the address assignment later.

All addresses of modules are located in the process image area. The process image is automatically updated cyclically.

Requirement

You are in the device view.

Procedure

To change the preset address range proceed as follows:

1. In the device view, click on the module for which you want to set the start address.
2. Go to "I/O addresses" in "Properties" in the inspector window.
3. Under "Start address" enter the required start address.
4. Press <Return> or click on any object to accept a modified value.

If you have entered an invalid address, a message indicating the next available address is displayed.

Note

You can also change the addresses directly in the device overview.

See also

Editing properties and parameters (Page 574)

Input and output addresses in the address overview (Page 576)

Assigning addresses to a location in the program

You can assign addresses of the I/O channels of modules directly to the points of use in the program or to a tag table.

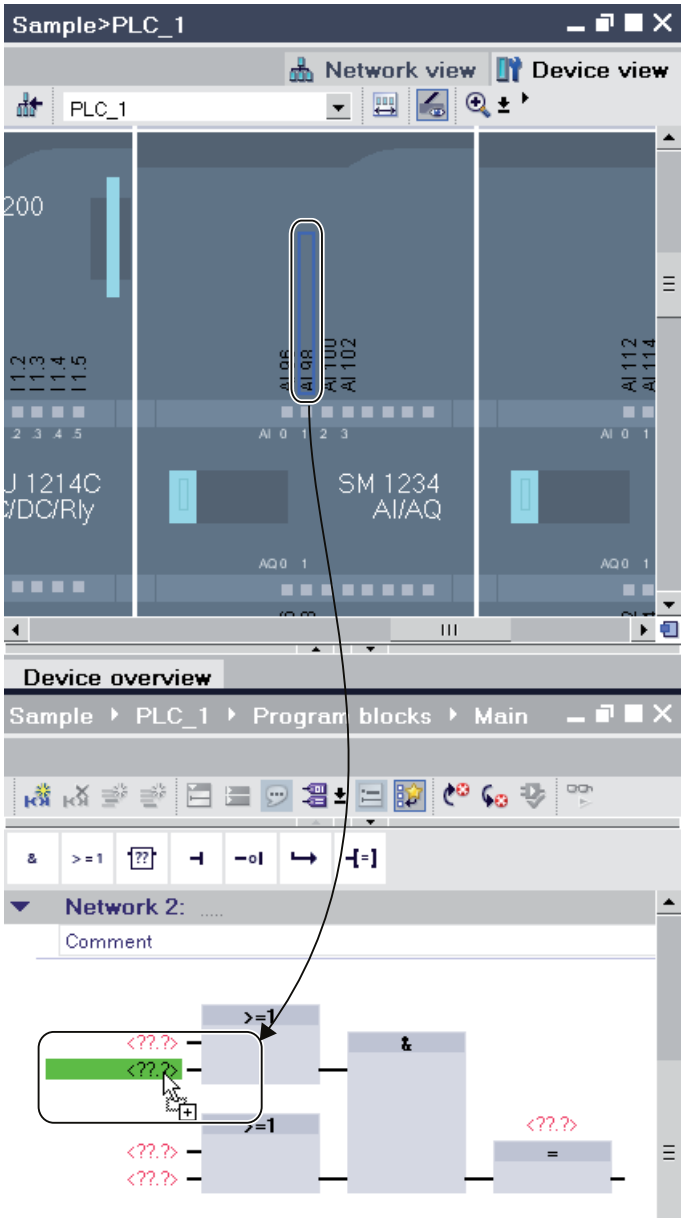
Requirement

- The device view of the hardware and network editor is open.
- The zoom level in the device view must be set to at least 200% to allow you to see the individual I/O channels.
- The instruction window of the programming editor or a tag table is open.

Procedure

To assign I/O channels of modules to the points of use in the program or to a tag table, follow these steps:

1. In the device view, navigate to the module with the desired I/O channel.
2. Click and hold down the mouse button to drag the desired I/O address to the corresponding point of use of the block or to the tag table.



The address of the module is assigned to the point of use in the program or entered as a tag in the tag table.

Note

The tag for an input or output of a block can also be dragged to the input or output of a module in order to link the tag to the I/O channel of the module.

Signal board

Inserting a signal board in a CPU

Introduction

Signal boards allows you to increase the number of the S7-1200 CPU's own inputs and outputs. Just like all other hardware components, you will find signal boards in the hardware catalog. However, you do not insert signal boards in the rack like other modules but directly in a slot of the CPU itself.

Note the following points when using a signal board:

- Each CPU can have only one signal board inserted in it.
- A signal board can only be inserted when the slot in the CPU is free.

There are various ways of inserting a signal board in a CPU:

- Double click on a signal board in the hardware catalog when there is a free slot in the CPU
- Drag from the hardware catalog to a free slot in the CPU
- Shortcut menu of a signal board in the hardware catalog for copying and pasting

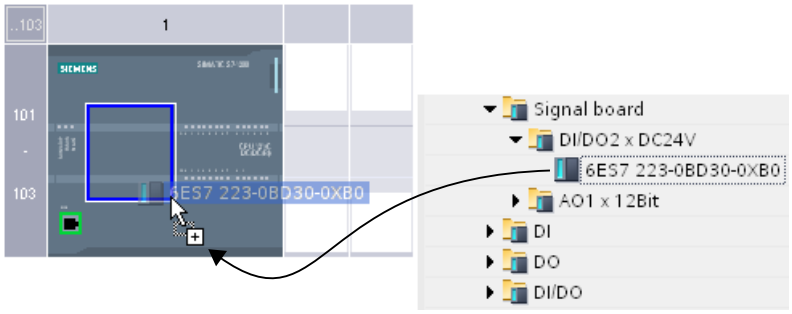
Requirement

- The hardware catalog is open.
- The S7-1200 CPU has a free slot for the signal board.

Inserting a signal board in a CPU

To insert a signal board in a CPU, proceed as follows:

1. Go to the required signal board in the hardware catalog.
2. Select the signal board you want to use.
3. Drag the signal board to the free slot in the CPU.



You have now inserted a signal board in the slot of the CPU.

If you are in the network view, you can also drag a signal board to a device. If the CPU has a an empty slot for a signal board, the signal board is inserted automatically into this slot.

Configurations for Web server

Information about the web server

Introduction

The Web server allows you to monitor the CPU via the Internet or the intranet of your company. This permits evaluation and diagnostics over long distances.

Alarms and status information are visualized on HTML pages.

Web browser

You need a web browser to access the HTML pages of the CPU.

The following PC web browsers, for example, are suitable for communication with the CPU:

- Internet Explorer (Version 8.0, 9.0)
- Mozilla Firefox (Version 17.0.1 and higher)
- Google Chrome 23.0
- Apple Safari 5.1.7 (Windows)
- Apple Safari 6.0.2 (Mac)

The following Web browsers of mobile devices are also suitable:

- Internet Explorer 6.0 for HMI Panels
- Mobile Safari (iOS 5.0.1)

- Mobile Android Browser 2.3.4
- Mobile Google Chrome 23.0

Reading information via the Web server

The following information can be read from the CPU. The availability of the respective web pages depends on the CPU and its firmware version.

As of firmware version 4, the web pages are also available in several languages.

Page/information	Description
Intro/Introduction	Entry page for the standard web pages
Start Page Start page with general CPU information	The start page provides an overview of general information on the CPU, the name of the CPU, the type of CPU and basic information on the current operating state.
Identification Identification information	Displays the static identification information such as serial number, article number and version numbers.
Diagnostic Buffer Diagnostic information	Displays the content of the diagnostics buffer with the most recent entries first.
Module Information Module information	Displays whether the centrally inserted components of a station are OK, whether there are maintenance requirements or components cannot be reached, for example. As of firmware version 4 a firmware update is possible via this Web page.
Communication Communication	Displays the communication connections during open communication (OUC); displays resources and address parameters.
Varable Status Tags	Displays the status of operands of the user program to monitor and change the values.
Data Logs (File Browser as of firmware version 4)	Data logs in CSV format to transfer to the hard disk of the programming device. The data logs are created with data log instructions in the user program and filled with data. As of firmware version 4 you have access to files of the internal load memory and of the external load memory (Memory Card), for example to the content of the directory "DataLogs" and "Recipes" via the Web page "File Browser".
User Pages User pages (if user-defined web pages have been configured and loaded)	The user web pages deliver a list of web pages with customer-specific web applications.

Web access to the CPU via PG/PC

Proceed as follows to access the Web server:

1. Connect the client (PG/PC) to the CPU via the PROFINET interface.
2. Open the web browser.
   Enter the IP address of the CPU in the "Address" field of the web browser in the format http://ww.xx.yy.zz (example: http://192.168.3.141).
   The start page of the CPU opens. From the start page, you can navigate to further information.

See also

S7-1200 web server (http://support.automation.siemens.com/WW/view/en/36932465)

Standard web pages

Requirements for web access

The requirements for access to standard CPU web pages are explained in the following, as well as the effects of missing or existing configuration information.

Requirement

The web server must be started.

The web server only starts when it has been activated in the properties of the CPU in the "Web server" section.

Note the following:

The web pages are normally transmitted via an non-secure connection and are not secured against hacker attacks. If you want to transfer the web pages in encrypted form to the browser, use the URL https://, followed by the IP address of the CPU.

Logon

No logon is required to access the standard web pages read-only. A user must be logged on to execute certain actions like changing the operating mode of the CPU or for write access.

For S7-1200 CPUs up to FW version V3:

You must be logged on as "admin" for the actions listed above. The logon input boxes are on the top left of each standard web page.

If you log on as "admin", you must enter the user name and password there.

Name: admin.

Password: configured CPU password (for password-protected CPU).

For S7-1200 CPUs as of FW version 4:

You can freely select the names of users and the passwords (CPU parameter "Web server", area "User management").

You assign rights to the users, for example the right to query diagnostics or to update the firmware.

JavaScript and cookies

The standard web pages use JavaScript and cookies. You must enable both in your web browser.

If JavaScript is not enabled, the following limitations apply:

- Data from standard web pages is not automatically updated.
- You cannot log on as user.
- Fields cannot be sorted (module information)

If cookies are not enabled, you cannot log on.

See also

Access for HTTPS (Page 857)

Settings for operation

Settings for operation

To be able to use the web server of an S7-1200-CPU, you must select the CPU in the network view or the device view and make the following settings in the inspector window under "Properties > General > Web server":

- Enable the web server
- Restricting access to the CPU to HTTPS transmission protocol (encrypted transmission) Access via port 80 is then blocked. Communication is only possible via port 443.

- Enabling automatic update of web pages
  The update interval is set by default and cannot be changed. The CPU updates web pages with changing content ((for example, status information or diagnostics information) at regular intervals.
- Creating and managing users
  Users are exclusively entitled to options that are assigned to the access rights. Depending on the CPU and firmware used, you can assign different user rights. Rights that your CPU does not support cannot be activated. A user called "Everybody" with minimal access rights, that you can, however, extend, is set by default in the user list. A user who uses the Web server without entering a password has "Everybody" user rights.
  You have the possibility of configuring further users with different access rights. These users have to log on with the configured user name and password.

Warning

Unauthorized access to the CPU by means of the Web server

Unauthorized access to the CPU or the changing of PLC variables to invalid values can result in interruptions of the processes controlled by the CPU and cause death, serious bodily harm or material damage!

Since the activation of the Web server allows authorized persons for example to change operating modes, to write-access CPU data or to update the firmware, we recommend that the following security measures be taken:

- If possible limit access to the HTTPS protocol.
- Create users with secure passwords. An example of a secure password is one which is only used for a single application, is more than 8 characters long, and consists of lower- and upper-case letters as well as symbols and numbers (?!+%$1234...). In addition, passwords based on common keyboard sequences or words from the dictionary should be avoided.
  Change the password at regular intervals.
- Do not extend the rights of the "Everybody" user.
- Check the PLC variables in the user program and limit the range of values to permissible ranges since users can set invalid values via the Web server.

Access for HTTPS

Access via HTTPS

HTTPS is used for encrypting and authentication of communication between the browser and web server.

To transfer data between the browser and the CPU using the HTTPS protocol, enter the URL as https://ww.xx.yy.zz in the address line of your browser, whereby ww.xx.yy.zz stands for the IP address of the CPU.

You require a valid, installed certificate for error-free HTTPS access to the CPU.

If no certificate is installed a warning is displayed with a recommendation not to use this page. To view the page you must explicitly "Add exception".

You can receive a valid certificate (Certification Authority) "SIMATIC CONTROLLER" as a download from the "Intro" web page under "Download certificate". The help function for your respective web browser provides information on how to install a certificate.

Accessing data of the CPU memory

You can access data in the internal or external CPU load memory by means of a standard web page.

- Use the web page "Data logs" for S7-1200 CPUs up to and including FW version 3. From this web page, you transfer the data logs from the CPU to a drive on your PC.
- Use the web page "File Browser" for S7-1200 CPUs as of FW version 4. From this web page, you transfer data from the folders "Data logs" or "Recipes", for example, to a drive on your PC.

Depending on the file type and the access permissions you have configured for the web server user, you can download, delete, rename or upload the files. The actual directories can only be created, deleted or renamed.

Example: Data logs

To open a data log, click on the link of the desired data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

Special note: Data logs are saved in U.S. American CSV format. You can only open the file directly using the U.S. version of Microsoft Excel. If you are using another national version of Microsoft Excel, you must import the file, selecting "comma" in the import assistant as the delimiter.

Downloading a data log

To download a data log, click on the download icon of the desired data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

Downloading and clearing or deleting a data log

For a CPU with FW version up to V3.0:

To download and delete the current entries of the data log, you must be logged on. To do this, click on the "Download and delete" icon of the required data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

For a CPU as of FW version V4.0:

To reset the data log, follow these steps:

1. Open the CSV file, for example with Excel.
2. Delete the rows between the headline and the row with the entry "//END", if this row exists.

3. Save the file to a drive on your PC.
4. On the web page "File Browser", delete the data log (which means the CSV file) and load the prepared CSV file to the CPU with the "Upload file" button of the "File Browser" web page.

Additional information is available in the S7-1500 CPU system manual.

See also

Data logging - Overview (Page 3318)

Create and download user-defined websites

What you need to know about user-defined web pages

Concept

The concept of user-defined web pages allows you to access freely-designed web pages of the CPU from a web browser. The Web server of the CPU provides this function.

You are not dependent on special tools for the design and functionality of the user-defined web pages. You can adapt the pages in the layout with CSS, provide dynamic content with JavaScript or use any framework to produce web pages.

The totality of files processed by the Web server is also referred to as the "web application".

Web application and user program

Using HTML code in user-defined web pages, you can also transmit data via a web browser to the user program of the CPU for further processing and can display data from the operand area of the CPU in the web browser.

You can use script instructions (such as Javascript) to optimize your web pages, for example to dynamically change contents or validate user entries.

To synchronize between the user program and the Web server, but also to initialize, you must call the WWW (SFC 99) instruction in the user program.

- If no interaction is required between the web application and the user program, for example, if a web page only provides static information, only initialization in the user program is required.
- If a simple data exchange is necessary between PLC tags and tags in web applications, to display the contents of PLC tags or write a value in a PLC tag for example, the syntax for reading and writing tags has to be observed. In this case only an initialization is required in the user program, for example in the startup OB.
- If a further interaction is required between the web application and the user program, you must handle status and control information from the Web Control DB in addition to the synchronization between Web server and user program. This is the case, for example, when user entries are transmitted via the web browser to the Web server for evaluation by the CPU. Unlike simple data exchange, the user program directly influences the time at which the requested web page is relayed back to the web browser. In this case, you must be acquainted with the concept of manual fragments and the structure of the Web Control DB.

Initialization

User-defined web pages are "packaged" in data blocks for processing by the CPU. You must generate appropriate data blocks from the source data (HTML files, images, JavaScript files, etc.) during configuration to be able to download the web application into the CPU. The Web Control DB takes on a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. Data blocks that contain coded web pages are termed "Fragment DBs".

When the data block is downloaded into the CPU, the CPU does not "know" that user-defined web pages are coded inside it. The "WWW" (SFC 99) instruction, for example, in the Startup OB informs the CPU which DB is the Web Control DB. The user-defined web pages can be accessed via a web browser after this initialization.

Synchronization

If the user program is to exchange data with the user-defined web pages, the WWW (SFC 99) instruction must be used in the cyclic program section.

Examples of interaction between user program and web page:

- Check received data
- Assemble and send back data to the web browser making the request

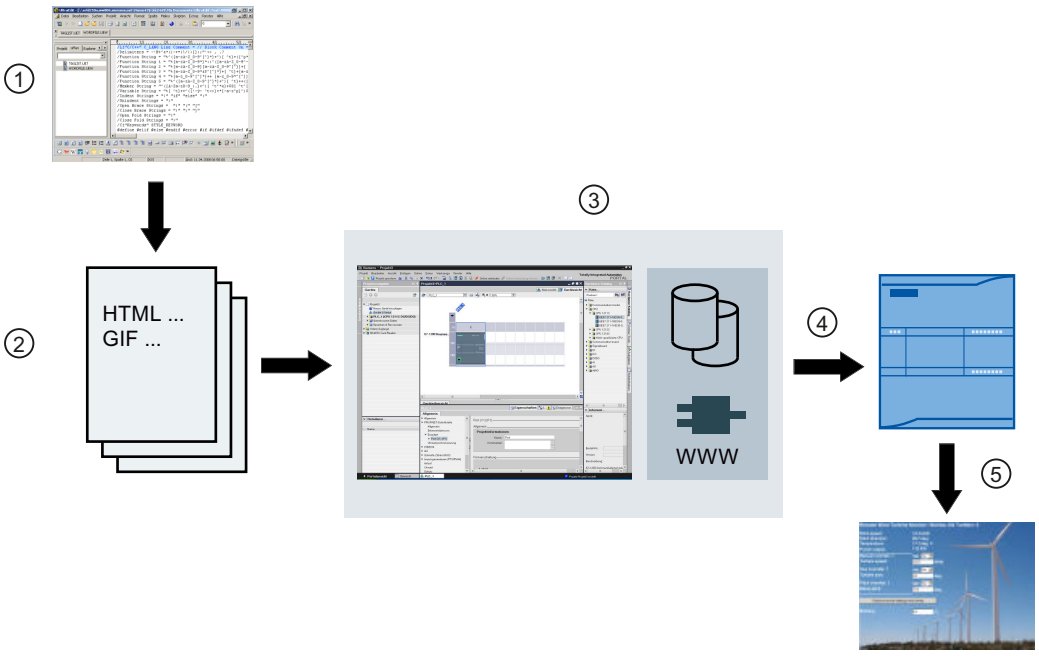
In this case, the status information must be able to be evaluated and control information must be transmitted to the Web server, for example, to release a requested web page.

Procedural overview

Basic information

This section provides a step-by-step explanation of the basic procedure used to create and download custom web pages and to use them in the operating phase.

The following graphic provides a simplified representation of the process used in creating and displaying custom web pages:



① Programming a web application (using suitable tools when required and AWP commands for dynamic pages when applicable).

② The web application is comprised of single source files, for example, *.html, *.gif, *.js, etc.

③ Using STEP 7:

- Generate the data blocks (Web Control DB and fragment DBs) from source files. The DBs contain meta information and the complete web application, including the images and the dynamic and static parts of the web application. The DBs are stored under "System blocks" in the project tree.
- Call the "WWW" instruction in the user program. This instruction initializes the web server of the CPU for a web application.
- If required, complete final programming for interaction between the web server and user program

④ Downloading the blocks to the CPU.

⑤ Call the web page in the browser. The web pages of the CPU are called by entering the IP address of the CPU.

Additional information

You can find additional information and examples relating to the S7-1200 web server on the Internet (http://support.automation.siemens.com/WW/view/en/36932465).

Creating web pages

Web design tools from various companies can be used to create user-defined web pages. As a rule, the web pages should be programmed and designed compliant to the conventions of the W3C (World Wide Web Consortium). No check is made for compliance to W3C criteria in the web server of the CPU.

Rules

- The tool must be able to directly edit the HTML code so that the AWP command can be inserted into the HTML page.
  Only the AWP commands are parsed in the CPU and, for example, replaced by values from the user program/process image of the CPU.
- Files containing AWP commands must be coded in UTF-8. In the metadata of the HTML page, therefore, set the attribute charset to UTF-8 and save the file UTF-8 coded.
- Files containing AWP commands must not contain the following sequence:]]
- Files containing AWP commands must not contain the following sequence outside of the "Tag read ranges" (:=<Tag name>:): :=
  Tip: Replace the first character of a prohibited sequence with its character coding; for the colon, for example, :.

A small example for a custom web page should make clear the basic design.

Requirement

- The CPU must have a web server and the web server of the CPU must be activated.
- To be able to access PLC tags with write access as a user, you must be logged on as "admin".
- For the example below, PLC tags must be defined for those PLC tags that are to be shown on the web page. This is shown here for the first tab used, "Tank_below_max".

		Name	Data type	Address
1		Tank_below_max	Bool	%I0.0

Creating user-defined web pages

The following code for an example web page reads values from the process image and provides them in a table.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/
TR/html4/strict.dtd">
<html>
  <head>
   <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
```

```
   <title>Mix</title>
  </head>
   <body>
    <h1>Mix</h1>
    <h2> Actual State </h2>
    <table border="1">
     <tr>
      <th>Variable</th>
      <th>State</th>
     </tr>
     <tr>
      <td>Tank below max</td>
      <td>:="Tank_below_max":</td>
     </tr>
     <tr>
      <td>Tank above min</td>
      <td>:="Tank_above_min":</td>
     </tr>
    </table>
   </body>
</html>
```

AWP commands

The interface between a freely-programmable web application for a CPU that has a Web server and the CPU data is declared by the AWP command (Automation Web Programming).

To develop web applications you are only subject to the restrictions of the web browser. In one of the programming languages of STEP 7, control with the user program which CPU data is displayed at what time in the web browser of the viewer. Use AWP commands, which you comment within the HTML files, to declare data to be used for intentional interaction between the web application and the user program.

AWP commands are inserted as HTML comments with a special syntax into HTML files; they declare the following features:

- Read PLC tags
- Write PLC tag
- Read special tags
- Write special tags
- Define enum types
- Assign tags to enum types
- Defining fragments
- Import fragments

Syntax of AWP commands

An AWP command begins with "`<!-- AWP_`" and ends with "` -->`". In JavaScript files, the commands should also be enclosed by JavaScript comments ("`/*...*/`").

Notation rules for PLC tag names within an AWP command

The AWP commands "AWP_In_Variable" and "AWP_Out_Variable" contain a name attribute and optionally a use attribute. A PLC tag name is assigned to these attributes, by means of which the PLC tags in the browser are written or read. The following rules apply to handling PLC tag names in HTML code:

- PLC tags must be enclosed in quotation marks (" ... ").
- PLC tags used in AWP commands must also be enclosed by single quotation marks ('" ... "') or with quotation marks masked by a backslash ("\" ... \"").
- If the PLC tag name contains the character \ (backslash) or * (star), this character must be designated with the escape sequence \\ as standard character of the PLC tag name. See below for examples.
- If the PLC tag name in the AWP command is also enclosed by single quotation marks and the single quotation mark (') occurs within the name, it must also be designed as normal character by the escape sequence \'.
- If an absolute address (input, output, bit memory) is used in AWP command, it is enclosed by single quotation marks.

PLC tag	PLC tag in HTML code
"Velocity"	<!-- AWP_In_Variable Name='"Velocity"' -->
	<!-- AWP_In_Variable Name="\"Velocity\"" -->
"abc\de"	<!-- AWP_In_Variable Name='"abc\\de"' -->
"abc'de"	<!-- AWP_In_Variable Name='"abc\'de"' -->
"abc'de"	<!-- AWP_In_Variable Name="abcde" Use'"abc\'de"' -->
"DB name".tag	<!-- AWP_In_Variable Name='"DB name".tag' -->
"DB name"."ta.g"	<!-- AWP_In_Variable Name='"DB name"."ta.g"' -->
-	<!-- AWP_Out_Variable Name=’flag1’ Use='M0.0' -->

See also

Reading tags (Page 864)

Writing tags (Page 867)

Special tags (Page 868)

Reading tags

User-defined web pages can read PLC tags.

The PLC tag must be specified by a PLC tag name.

These OUT variables (direction of output as viewed from the controller) are inserted at any location within the HTML text with the syntax described in the following.

Syntax

```
:=<varname>:
```

These references are replaced when the Web server is in operation by the current values of the PLC tag in each case.

<varname> can be a simple, global PLC tag but also a complete tag path to a structural element.

Notation rules for PLC tag names

- PLC tags in HTML code are enclosed by quotation marks ("), if they are defined in the tag table. In the case of data block tags, the name of the data block is enclosed by quotation marks. If special characters are used in the structure elements of the data block, for example the dot (.) or blank, this part must also be enclosed by quotation marks.
- Quotation marks are not used for absolute addresses of inputs, outputs or bit memories.

PLC tag	PLC tag in HTML code
"DB_name".var_name	:="DB_name".var_name:
"DB_name"."var.name"	:="DB_name"."var.name":
"memory"	:="memory":
-	:=I0.0:
	:=Q0.0:
	:=MW100:
	:=%MW100:
"My_Data_Block".flag1	<!-- AWP_Out_Variable Name='flag1' Use='"My_Data_Block".flag1' -->
	...
	:=flag1:

- If the PLC tag name contains the character : (colon) or \ (backslash), this character must be designated with the escape sequence \: or \\ as standard character of the PLC tag name.

PLC tag	PLC tag in HTML code
"abc:de"	:="abc\:de":
"abc\de"	:="abc\\de":

- Special characters "<, &, >"
  Display problems can occur if these characters are contained in the tag name (for example, "a<b").
  Avoid expressions such as :="a<b": in the HTML page.
  To prevent display problems from occurring, use e.g. an AWP command with a use expression according to the pattern depicted below. The use attribute defines the PLC tag with the problematic character, the name attribute defines the name without problematic character, as it is used in the HTML page.

PLC tag	PLC tag in HTML code
"a<b"	<!-- AWP_Out_Variable Name='simplename' Use='"a<b"' -->
	...
	:=simplename:

Reading tags of the type String and Character

Below, these types of quotation marks are used in the explanation: single quotes ('), double quotes (").

As of firmware V1.6, with the "Read PLC tags" function, an S7-1500 CPU outputs tags of the type String or Character enclosed in single quotes to the browser.
Example:

- String tag "Varname".MyString with the content ABC
- You read the tag in HTML using the function :="Varname".MyString:
- The Web server outputs the character string 'ABC' to the browser

Using String or Character tags in expressions

On your HTML page, you use an expression in which the character string for reading a tag is enclosed in quotes, for example in forms.

Possible HTML code used:

```
<input type="text" name="appfield" value="myvalue">
```

If you read the displayed value for the "value" attribute from a PLC tag in this expression, the HTML code appears as follows:

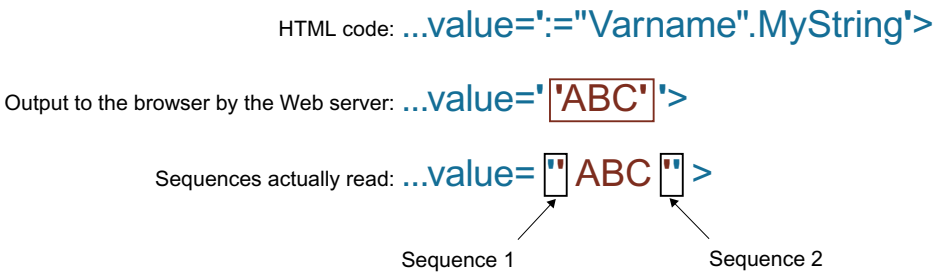
```
<input type="text" name="appfield" value=":="Varname".MyString:">
```

By reading the PLC tag, the Web server outputs the value 'ABC'. In HTML, the code is then represented as follows:

```
<input type="text" name="appfield" value=" 'ABC' ">
```

If you have used single quotes instead of double quotes in your HTML code to enclose the attributes, the Web server passes on the content of the tags enclosed in two single quotes to the browser. As a result of this, the browser does not output the content of the String or Character tag, since two consecutive single quotes each form a closed sequence. The values to be read are located between these sequences and are not output by the browser.

In this context, note in particular that the character string with double quotes is not identical to two single quotes even if they appear to be identical, as shown in the following figure.



See also

AWP commands (Page 863)

Writing tags

Custom web pages can write data into the CPU.

This requires an AWP command that identifies the PLC tag to be written.

The PLC tag must also be specified by a PLC tag name.

The IN tags (direction of input as viewed from the controller) are placed on the browser page. This can be done, for example, in a form.

The tags are either set in the HTTP header (by cookie or POST method) or in the URL (GET method) by the browser and are then written by the Web server into the respective PLC tag.

Syntax

To allow the IN tags to be written to the CPU, the tags must first be defined by an explicit AWP instruction:

```
<!-- AWP_In_Variable Name='<PLC_Varname1>' Name='<PLC_Varname2>'
Name='<PLC_Varname3>' -->
```

Several tags can be defined in an instruction - such as that shown above.

The specific PLC tag name is hereby written in double quotation marks; for example <PLC_Varname1> = "myVar".

In cases where the name of the tag that you use for the web application is not identical to the name of the PLC tag, the "Use" parameter can be used to assign to a PLC tag:

```
<!-- AWP_In_Variable Name='<Webapp_Varname>' Use='<PLC_Varname>'
```

Example

The "AWP_In_Variable" AWP command is indispensable when handling forms.

```
<form method='post' action='/awp/appl/x.html'>
 <p>
  <input name='"var1"' type='text'>
  <input value='set' name='Button1' type='submit'>
 </p>
</form>
```

In the form defined above, the HTTP request method "post" is used to transfer the tag "var1" to the Web server. The user places the "var1" tag in the form field. The tag 'Button1' has the value 'set', but is not required for the CPU. To allow the "var1" tag to be written to the CPU, the following instruction must be included in the same fragment:

```
<!-- AWP_In_Variable Name='"var1"' -->
```

Since PLC tags are enclosed in double quotation marks ("), the name in the AWP command must be enclosed in single quotation marks (') or in masked quotation marks (\"). To avoid the numerous escape sequences, we recommend the use of single quotation marks.

```
<!-- AWP_In_Variable Name='"Info".par1' -->
<!-- AWP_In_Variable Name="\"Info".par1\"" -->
```

Conditions for write access during operation

The following requirement has to be met in order for a user to be able to write to PLC tags from a user-defined web page:

The user must have rights to change tags. The Web server ignores the commands if the user has no change rights.

This rules applies to all writing access to web pages on a CPU.

See also

Requirements for web access (Page 855)

AWP commands (Page 863)

Special tags

Special tags are mainly HTTP tags set in the definition of the World Wide Web Consortium (W3C) . Special tags are also used for cookies and server tags.

The AWP command to read and write special tags differ only in that they have additional parameters than the AWP command used to read and write normal tags.

Reading a special tag

The Web server can read PLC tags and transfer these to special tags in the HTTP Response Header. You can, for example, read a URL for a diversion to another web page and transfer to the special tag HEADER:Location using the special tag HEADER:Location.

The following special tags can be read:

Name	Description
COOKIE_VALUE:name	Value of cookie with name: "name"
COOKIE_EXPIRES:name	Execution time of cookie with name: "name" in seconds (must be set beforehand).
HEADER:Status	HTTP status code (if no other value has been set, status code 302 is returned).
HEADER:Location	Path for forwarding to another page. Status code 302 must be set.
HEADER:Retry-After	Anticipated time in which the service is not available. Status code 503 must be set.
HEADER: …	All other header tags can also be forwarded in this way.

Use the AWP command "AWP_Out_Variable" to specify which PLC tags are to be transferred in the HTTP header to the web browser.

Basic structure:

```
<!-- AWP_Out_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description

- Name: Type and name of special tag
- Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, parameter "Use" can be used to assign to a PLC tag.

Example:

```
<!-- AWP_Out_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

Writing a special tag

In principle, all HTTP tags written in the HTTP header by the web browser can be evaluated by the user program of the CPU. Examples of tag types:

Name	Description
HEADER:Accept-Language	Accepted or preferred language
HEADER:Authorization	Proof of authorization for a requested resource
HEADER:Host	Host and port of the requested resource
HEADER:User-Agent	Information on the browser
HEADER: …	All other header tags can also be forwarded in this way
SERVER:current_user_id	Indicates whether a user is logged on (current_user_id=0: no user logged on)
SERVER:current_user_name	User name of the user logged on
SERVER:GET	Request method is GET
SERVER:POST	Request method is POST
COOKIE_VALUE:name	Value of cookie with name: "name"

The AWP command "AWP_In_Variable" is used to define which special tags are to be evaluated in the user program of the CPU.

Basic structure:

```
<!-- AWP_In_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description:

Name: Type and name of special tag

Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, the parameter Use can be used to assign to a PLC tag.

Examples:

```
<!-- AWP_In_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use . The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='COOKIE_VALUE:siemens' Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use. The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='"COOKIE_VALUE:siemens"' -->
```

The HTTP-header variable is written in the same-name PLC variable.

See also

AWP commands (Page 863)

Enumeration types

Enumeration types (enums)

Numerical values from the PLC program can be converted into text and vice versa using enums. The numerical values can also be assigned for several languages.

Creating enums

Enter an AWP command using the following syntax at the start of the HTML file:

```
<!-- AWP_Enum_Def Name="<Name of the enum type>"
Values='0:"<Text_1>", 1:"<Text_2>", ... , x:"<Text_x>"' -->
```

For example, for German values to be saved as an HTML file in the "de" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"an", 1:"aus", 2:"Störung"' -->
```

For example, for English values, to be saved as an HTML file in the "en" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"on", 1:"off", 2:"error"' -->
```

Assigning enums

Tags are assigned from the user program to the individual enum texts using a special AWP command:

```
<!-- AWP_Enum_Ref Name="<VarName>" Enum="<EnumTypeName>" -->
```

<VarName> is thereby the symbolic name from the user program and <EnumTypeName> is the previously set name of the enum type.

Note

In each fragment in which enum texts are referenced by a PLC tag, this PLC tag must be assigned by the appropriate AWP command of the enum type name.

Ensure that no AWP command for importing fragments is positioned between an enum assignment and enum usage because this import can result in the enum assignment lying in a different fragment than the enum usage.

Example

Enum type "state" is defined with values "0" and "1". "0" means "off", "1" means "on":

```
<!-- AWP_Enum_Def Name="state" Values='0:"off", 1:"on"' -->
```

The following code is contained in the HTML code of the web page to be output:

```
<!-- AWP_Enum_Ref Name="operating state" Enum="state" -->
:=operating state:
```

Depending on the value of the "operating state" tag, the ´result displayed is no longer "0" or "1", but "off" or "on".

Simplified use of enumeration types

At S7-1200 CPUs as of firmware version 4 it is possible to use enumerations directly in AWP commands to read and write PLC variables.

You create enums as described in the previous section, and you can then utilize the values with user program read and write commands.

Creating enums

```
<!-- AWP_Enum_Def Name="<Name des Enum Typs>" Values='0:"<Text_1>",
1:"<Text_2>", ... , x:"<Text_x>"' -->
```

Utilizing enums in the user program read and write commands

```
<!-- AWP_In_Variable Name='<Varname>' Enum="<EnumType>" -->
<!-- AWP_Out_Variable Name='<Varname>' Enum="<EnumType>" -->
```

Example of reading PLC tags

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' --><!-- AWP_Out_Variable Name='"Alarm"'
Enum="AlarmEnum" -->...<p>The current value of "Alarm"
is :="Alarm":</p>
```

If the value of "Alarm" in the CPU is "2", the following text will be displayed on the HTML page:

'The current value of "Alarm" is Tank is empty' because the enum definition assigns the string "Tank is empty" to the numerical value 2.

Example of writing PLC tags

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' --><!-- AWP_In_Variable Name='"Alarm"'
Enum='AlarmEnum' -->...
<form method="POST">
<p><input type="hidden" name='"Alarm"' value="Tank is full" /></p>
<p><input type="submit" value='Set Tank is full' /></p>
</form>
```

Because the enum definition assigns the string "Tank is full" to the numerical value "1", the value "1" is written to the PLC tag "Alarm".

Definition of fragments

Fragments

Fragments are "logical sections" of a web page to be processed by the CPU individually.

Fragments are usually complete pages but can also be individual elements such as files (for example, images) or complete documents.

Defining fragments

```
<!-- AWP_Start_Fragment Name="<Name>" [Type="<Type>"] [ID="<Id>"]
[Mode=<Mode>]-->
```

The start of a fragment is specified by this command. A fragment runs to the start of the next fragment or to the end of the file.

- <Name> Indicates the name of the fragment.
  The name must start with a letter [a-zA-Z] or an underscore (_). Letters, underscores or numbers [0-9] can follow after this first character.
- <Type> Indicates the type of the fragment.
  - "manual" The user program is informed of the request for a fragment; the web page to be returned can be changed by the user program.
  - "automatic" The page is automatically processed (default).
- <id> A numeric ID can be stipulated for the fragment. If no ID is assigned, the fragment is automatically assigned an ID. For manual pages (<Type>=manual) , the fragment can be addressed in the user program of the CPU by this ID.

  Note

  Keep the ID low because the highest ID influences the size of the Web Control DB.

- <Mode> Fragments support the visible and hidden modes.
  - "visible" The fragment is a part of the web page. This mode is preset and can also be omitted.
  - "hidden" The fragment is not part of the web page. However, the fragment will be saved in the Web DB and is available to the user program for inserting in a requested web page. You use an exchange of the fragment ID (Web-Control-DB.fragment_index tag) to insert a "hidden" fragment in the requested web page.

The input document is completely divided into fragments by the "AWP_Start_Fragment" command. "AWP_End_Fragment" is therefore unnecessary.

Without a start fragment command, a file is mapped as a fragment; the fragment name is derived from the file name. If a file is divided into several fragments (by "AWP_Start_Fragment"), the file must begin with the "AWP_Start_Fragment" command.

Importing fragments

You can declare a fragment in an HTML page and import this fragment into other web pages.

Example

A company logo is to be displayed on all web pages of a web application.

There is only one instance of the HTML code for the fragment that displays the company logo. You can import the fragment as often and into as many HTML files as required.

Syntax

```
<!-- AWP_Import_Fragment Name = "<name>"-->
```

- <name> is the name of the fragment to be imported.

Example

HTML code within a web page that declares a fragment:

```
<!-- AWP_Start_Fragment Name = "My_Company_Logo" -->
<p><img src = "compay_logo.jpg"></p>
```

Example

HTML code within another web page that imports the declared fragment:

```
<!-- AWP_Import_Fragment Name = "My_Company_Logo" -->
```

Creating and loading a data block

Requirement

- All source files required for the web application (*.html, *.js, *.png, ...) have been created.
- The source files are located in one folder, but only those source files that are required for the web application. No other files may be located in this folder.

Note

Length of file names and tag names

If you have a comprehensive web application with many files and directories, the generation of the web data blocks may possibly fail. If this happens, the generation is aborted with the message "Text list overflow...". The cause is system-internal size limitations for management information saved in the web data block.

Remedy: Use short file names and short tag names.

Procedure

To create data blocks from the source files for user-defined web pages in STEP 7, proceed as follows:

1. Select the CPU, for example, in the device configuration.
2. Select the properties for user-defined web pages in the inspector window under "Properties > General > Web server".
3. As "HTML source", select the folder that contains the source files for the web application.
4. Enter the HTMP page to be opened on starting the web application as the start HTML page.

5. Enter a name for the application if required.
6. You can supplement a range of file name extensions as "Files with dynamic content" if necessary. Only enter those file name extensions that also contain AWP commands.
7. The number for the Web Control DB and for the fragment DB start number can be kept as long as they are not already being used by your user program.
8. Click on the "Generate" button to create DBs from the source files.
   The generated data blocks are saved in the project navigation in the "System block" folder (in the "Web server" subfolder).
9. In the CPU, select the network view to be loaded and then select the "Download to device" command in the "Online" menu to download the blocks. Compilation of the blocks is implicitly initiated before downloading.
   If errors are reported during this process, you must correct these errors before you can download the configuration.

Structure of the PLC program

Your user program must call the " WWW" instruction to even allow the web application, for example, the user-defined web pages, to be available to the CPU on the standard web pages and to allow them to be called up there.

The Web Control DB you have created from the source files is the input parameter (CTRL_DB) for the "WWW" instruction. The Web Control DB references the content of the user-defined web pages coded in the fragment DB and then receives status and control information.

Calling the "WWW" instruction in the startup program

If you do not want the user program to influence requested web pages, it is sufficient to only call the "WWW" instruction once in a startup OB. This instruction initializes communication between the web server and the CPU.

Calling the "WWW" instruction in the cyclic program

The "WWW" instruction can also be called in an OB processed in cycles (for example, OB 1). This has the advantage of being able to respond to web server requests from within the user program. Manual fragments must be used for this.

In this case, you must evaluate information from the Web Control DB in order to identify the requested web page or the requested fragment. On the other hand, you must set a bit in the user program in order to explicitly release the web page to be returned by the web server after processing the web page request.

The structure of the Web Control DB is described in the following section.

Web Control DB

The Web Control DB (DB 333 by default) is created by STEP 7 and contains information on the structure of user pages, the status of communication and any errors that occur.

Additional fragment DBs are also created as well as the Web Control DB. These fragment DBs (there may also only be one fragment DB) are referenced in the Web Control DB. The fragment

DBs contain the web pages and media data coded in fragments, for example, images. The content of the fragment DB cannot be changed by the user program. It is created automatically and is only for data management.

The status and control tags of the Web Control DB are accessed via symbols.

The following lists the tags of the Web Control DB required for status evaluation and to control interaction.

The Web Control DB provides two types of information:

- Global status information: Not bound to a concrete web page request.
- Request status and control information: Information about queued requests.

Global status information

"WEB-Control_DB".commandstate.init	Activates and initializes the web application.
"WEB-Control_DB".commandstate.deactivate	Deactivates the web application.
"WEB-Control_DB".commandstate.inititializing	The web application is initialized (read Web Control DB, etc.).
"WEB-Control_DB".commandstate.error	Web application could not be initialized. The reason is coded in "WEB-Control_DB".commandstate.last_error .
"WEB-Control_DB".commandstate.deactivating	The web application is closed.
"WEB-Control_DB".commandstate.initialized	The web application has been initialized and is ready.
"WEB-Control_DB".commandstate.last_error	Refer to the next table for a value table of possible errors.

Last_error	Description
1	Fragment DB is inconsistent (does not match the Web Control DB).
2	A web application already exists with this name.
3	Memory problem initializing in the web server.
4	Inconsistent data in the Web Control DB.
5	A fragment DB is not available (not loaded).
6	No AWP ID for a fragment DB.
7	The enum fragment is not available (contains the texts and information on the enum types).
8	An action requested via the command flag in the Web Control DB is prohibited in the current state.
9	Web application is not initialized (if there is no reinitializing after disabling).
10	Web server is disabled.
...	Last_error is reset once the web application has been successfully initialized.

Request status information

Request status information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].idle	Nothing need be done.
"WEB-Control_DB".requesttab[x].waiting	The user program must react to a request from a manual fragment and explicitly initiate further processing in the web browser.
"WEB-Control_DB".requesttab[x].sending	The web server is occupied with processing the request/fragment.
"WEB-Control_DB".requesttab[x].aborting	The TCP connection is closed by the web server.

Request control information

Request control information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].continue	Releases the fragment being processed for transmission. Processing of the next fragment is initiated.
"WEB-Control_DB".requesttab[x].repeat	Releases the fragment being processed for transmission. The fragment is then processed again.
"WEB-Control_DB".requesttab[x].abort	Closes the TCP connection.
"WEB-Control_DB".requesttab[x].finish	Releases the fragment being processed for transmission. Stops further processing of requests (terminates the request).

Example:

The tag for the DB is: "WEB-Control_DB". Whether errors have occurred during initialization of the web application can be determined by requesting bit "WEB-Control_DB".commandstate.error in the user program.

If an error has occurred you can analyze it using the "WEB-Control_DB".commandstate.last_error value.

Interaction with the user program

With the help of manual fragments, you can make sure that the user program reacts synchronously to browser entries so that the returned website can be prepared by the user program.

Fragment type

To react to the received data in the user program the "manual" fragment type must be used for the fragment writing the data (for "manual pages"):

```
<!-- AWP_Start_Fragment Name="testfrag" ID="1" Type="manual" -->
```

The values are always transferred to the Web server of the CPU for automatic and manual pages in the same way:

Example:

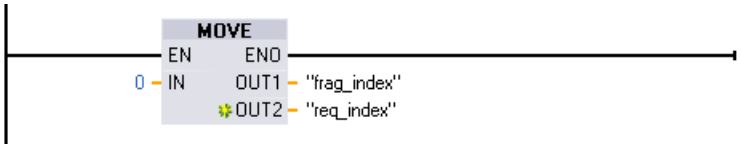
```
<form method="POST" action="">
 <p>
  <input type="submit" value="Set new value">
  <input type="text" name='"Velocity"' size="20">
 </p>
</form>
```

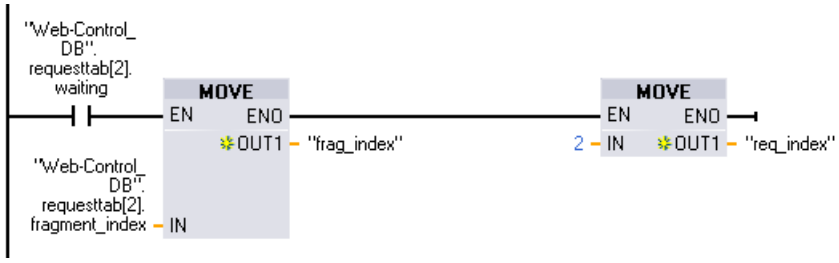
User program for manual fragments

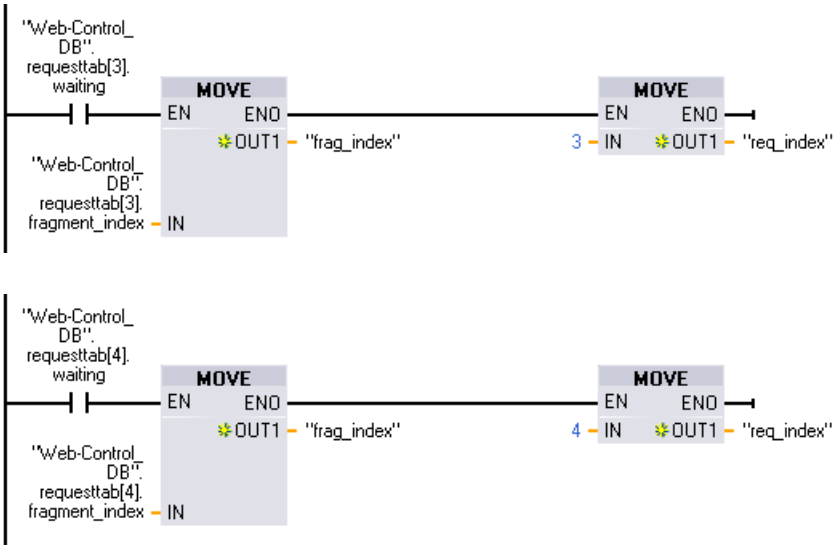
When using manual pages, the "WWW" instruction must be called in cycles in the user program of the CPU.

To react to values entered in the browser, the request - which is made by the manual page to the Web server - must first be evaluated in the user program. The web-control-DB (for example, DB 333) must investigate pending requests to do this. The array that manages four requests is contained in the "requesttab" section of the Web Control DB. Each element of the array contains information about the respective request in a structure.

A simple programming example shows how queued requests are checked based on the tags of the Web Control DB.





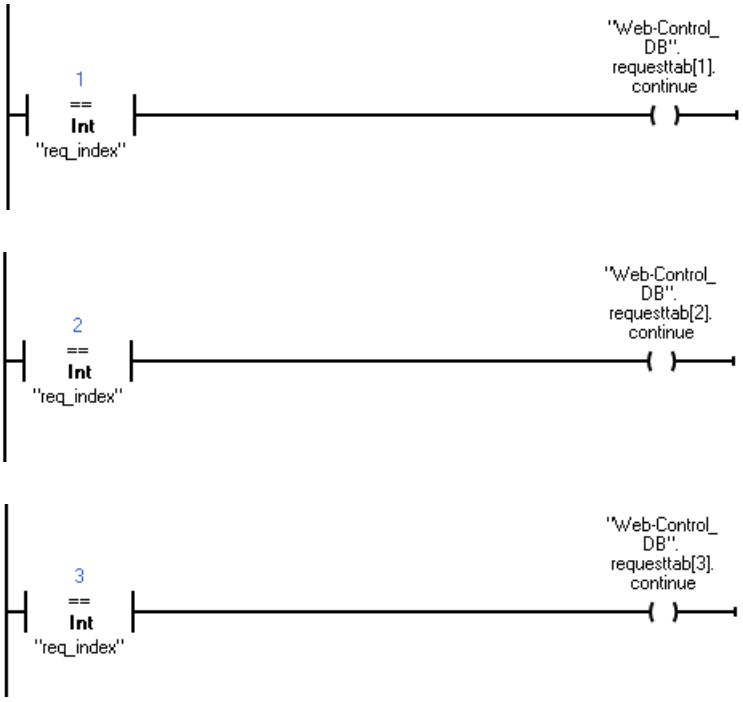


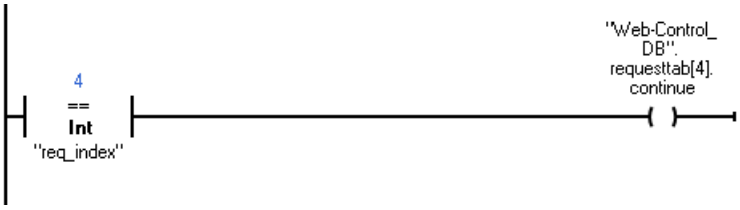
In cases where a request has been made, this program section writes the fragment ID in the #frag_index tag and the request no. (value range 1-4) in the #req_index tag.

Using the information from this, the information transferred in the request can now be processed separately for each fragment ID in the program (for example, plausibility check).

Once processing of the request has been completed by the program, the request must be answered and the appropriate entry is once more reset under"requesttab" of the Web Control DB (for example, DB 333).

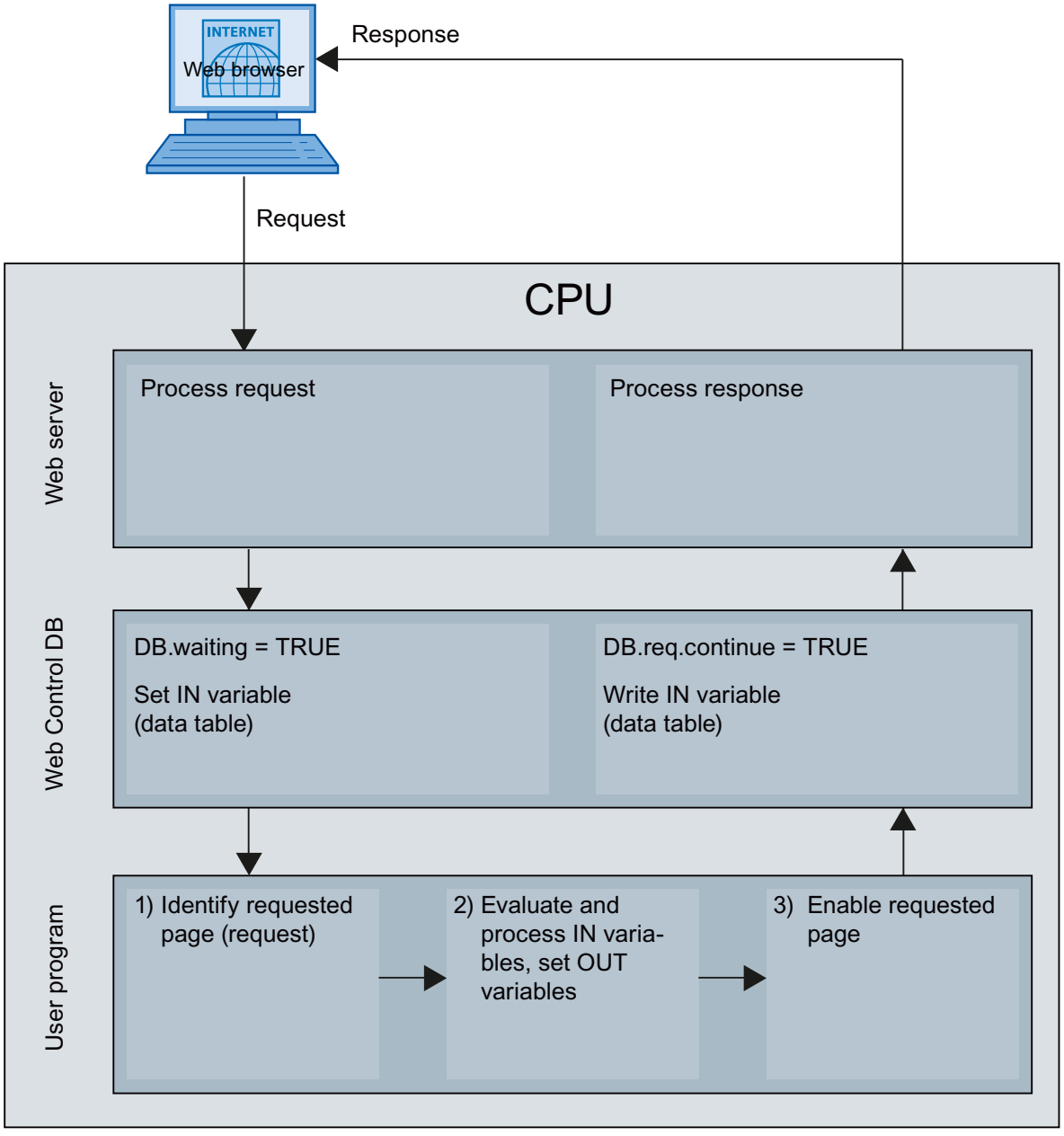
A simple programming example for replying to requests:




4
==
Int
"req_index"
"Web-Control_
DB".
requesttab[4].
continue

Principle sequence of a browser request with interaction from the user program

The following figure shows the simplified, principle sequence of the web browser request on the effects of Web Control DB content and the actions required from the user program until the processed web page is returned (response).



Displaying custom web pages in the browser

Display web pages in browser

Web pages are called from the standard web pages of the web browser.

In addition to the other links in the navigation bar, the standard web pages also have a link to "user pages".

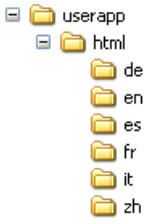
Click on the "user pages" link to open the web browser you have configured as the default HTML page.

Creating custom web pages in several languages

You can make each of your custom web pages available in various languages.

Requirements

The language-dependent HTML; pages must be stored in a folder structure containing folders with the respective language abbreviations:



Specified language abbreviations

Language abbreviations "de", "en", "fr", "es", "it" and "zh" are fixed. Additional language folders or other designated language folders are not supported.

Additional folders within the same folder hierarchy for other files can be created as required; for example, an "img" folder for images and a "script" folder for JavaScript files.

Language switching for custom web pages

Requirements

The HTML pages are contained in the predefined language folders, for example, HTML pages with German text are in the "de" folder, HTML pages with English text are in the "en" folder.

Language switching concept

Language switching is based on a predefined cookie named "siemens_automation_language". If the cookie is set to value "de", at the next web page request or web page update, the web server switches to the web page from the "de" folder.

Similarly, the web server switches to the web page from the "en" folder when the cookie is set to "en".

Example of language switching

The example is structured as follows:

- The language-dependent HTML files with the same name, for example, "langswitch.html" are located in both language folders "de" and "en". The text to be displayed within the two files are German or English, corresponding to the name of the folder.
- There is an additional "script" folder in the folder structure containing the JavaScript file "lang.js". Functions required for language switching are stored in this file .

Structure of the "langswitch.html" file ("de" folder)

Meta data "content language", charset and path to JavaScript file are set in the file header.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="de">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Switch language to German page</title>
<script type="text/javascript" src="script/lang.js" ></script>
<head>
```

Language selection is implemented in the body of the file by the "select" HTML element. The select element initiates a list box and contains the "de" option, labeled as "German" and "en", labeled as "English"; "de" is the default.

The "DoLocalLanguageChange(this)" function is called using the "onchange" event handler. The "this" parameter transmits the select object with the selected option to this function. "onchange" calls the function each time the option is changed.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
         <!-- change language immediately on change of the selection
-->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" selected >Deutsch</option>
                <option value="en" >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Structure of the "langswitch.html" file ("en" folder)

The header of the HTML file with English text is structured similarly to the HTML file with German text.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="en">
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Language switching english page</title>
<script type="text/javascript" src="script/lang.js" ></script>
```

Language selection is also implemented in the body of the file by the "select" HTML element. In contrast to the German HTML file, the English option is already selected as a default and the test or the labels are in English.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
         <!-- change language immediately on change of the selection
-->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" >German</option>
                <option value="en" selected >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Structure of "lang.js" file (in the "script" folder)

The " DoLocalLanguageChange" function is defined in the Java script file and calls the "SetLangCookie" function with the language selection value. SetLangCookie combines the cookie name and cookie value and then sets the cookie by means of the corresponding document.cookie property. The web page must then be reloaded (top.window.location.reload) to allow the web server to react to the setting of the cookie by displaying the required language.

```
function DoLocalLanguageChange(oSelect) {
        SetLangCookie(oSelect.value);
        top.window.location.reload();
    }

 function SetLangCookie(value) {
        var strval = "siemens_automation_language=";
            // this is the cookie by which the web server
            // detects the desired language
            // this name is required by the web server
        strval = strval + value;
        strval = strval + "; path=/ ;";
            // set path to the application, since otherwise
            // path would be set to the requesting page
            // would not get the cookie.
            // The path for user defined applications follows this
sample:
            // path=/awp/<application name>/<pagename>
            // example: path=/awp/myapp/myappstartpage.htm
            //(where myapp is the name of the web application
            // entered in the web server properties of the cpu)
```

```
                /*
                use expiration if this cookie should live longer
                than the current browser session
                var now      = new Date();
                var endttime = new Date(now.getTime() + expiration);
                strval = strval + "; expires=" + endttime.toGMTString()
+ ";";
                */
            document.cookie = strval;
        }
```

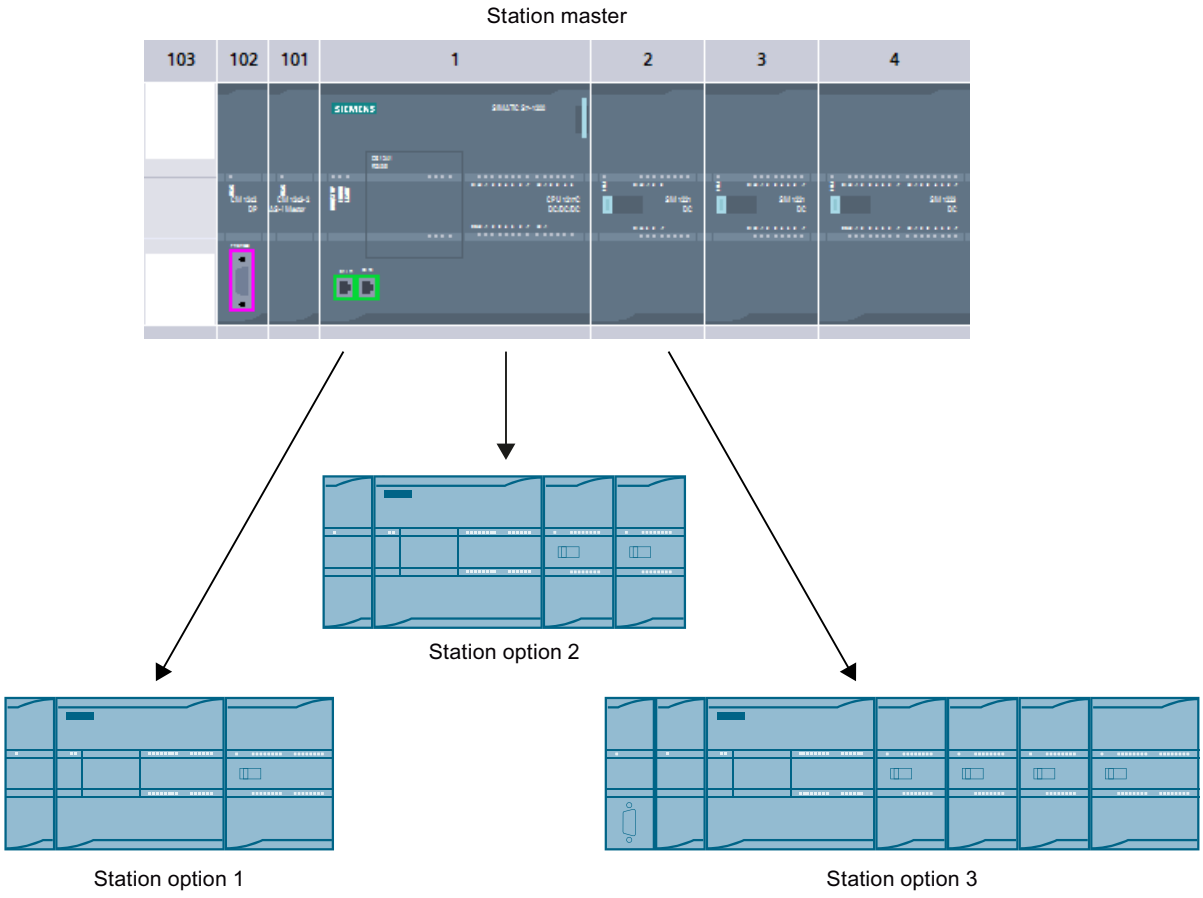
Configuration control for devices

Important information regarding configuration control

Operating principle

With S7-1200 Firmware Version 4.1 or higher, the configuration control enables you to configure the structure of a control system and to work with variants (options) that differ from this configuration.

- All modules that are needed in a set of similar plant units or machines are configured in a master project with a maximum configuration (station master).
- Provision is made in the user program of the master project for various station options for various plant units or machines as well as the selection of a station option. A station option uses, for example, only some of the configured modules and these modules are inserted in varying order.
- An operator selects a station option for a specific plant on-site. The operator does not have to modify the project and thus also does not have to download a modified configuration for this.



A control data record you have programmed in the startup program notifies the CPU as to which modules are missing or located in different slots as compared to the preset configuration. The configuration control has no effect on the parameter assignment of the modules.

Configuration control gives you the flexibility to vary the central installation as long as the real configuration can be derived from a preset maximum configuration.

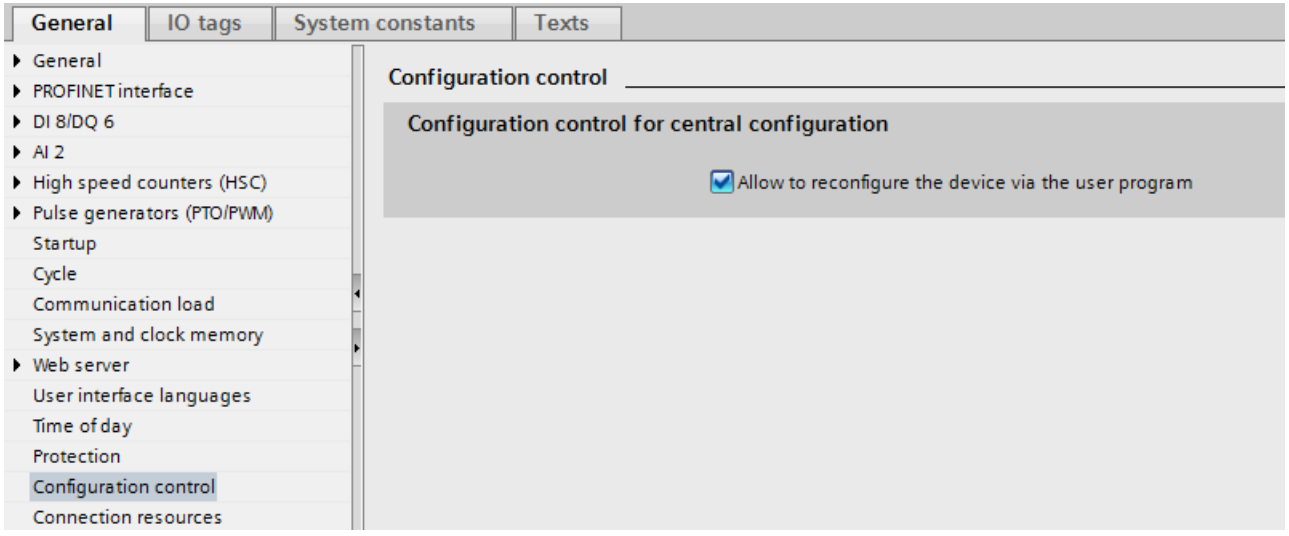
Below you will find a description of how to activate configuration control (CPU parameter assignment) and how to structure the required data record.

Requirement

- STEP 7 Version V13 SP1 or higher
- CPU S7-12XX Firmware Version V4.1 or higher Modules that support the "Configuration control" function also have the "Configuration control" entry in the description (info text) of the hardware catalog.
- Recommendation: Before you load a new program with a modified control data record, perform a memory reset. This action will prevent inconsistent states that may result from the presence of incompatible control data records.
- The startup parameter "Compare preset to actual configuration" is set to "Startup CPU even if mismatch" (default setting).

Required steps

1. Enable the "Allow to reconfigure the device via the user program" parameter when configuring the CPU ("Configuration control" area).



2. Create a control data record (e.g., in a data block) according to the current configuration based on the sample described below for the control data record. The control data record has the number 196. If you want to transfer the control data record as a whole block to the WRREC instruction (input parameter RECORD), note that you must first create a PLC data type containing the structure of the control data record and base the data block on this PLC data type.

ConfDB

	Name	Data type	Start value	...	...	...	...	Comment
1	Static							
2	ConfigControl	Struct			☑	☑		
3	Block_length	USInt	9		☑	☑		Length of control data record, including header
4	Block_ID	USInt	196		☑	☑		Data record number
5	Version	USInt	5		☑	☑		
6	Subversion	USInt	0		☑	☑		
7	Slot_1	USInt	16#FF		☑	☑		Assignment for CPU annex card/Actual annex card
8	Slot_2	USInt	16#FF		☑	☑		Configured slot 2/ Assigned "real" slot
9	Slot_3	USInt	16#FF		☑	☑		Configured slot 3 / Assigned "real" slot
10	Slot_4	USInt	16#FF		☑	☑		Configured slot 4 / Assigned "real" slot
1*	Slot_5	USInt	16#FF		☑	☑		Configured slot 5 / Assigned "real" slot
8	Slot_6	USInt	16#FF		☑	☑		Configured slot 6 / Assigned "real" slot
9	Slot_7	USInt	16#FF		☑	☑		Configured slot 7 / Assigned "real" slot
10	Slot_8	USInt	16#FF		☑	☑		Configured slot 8 / Assigned "real" slot
1*	Slot_9	USInt	16#FF		☑	☑		Configured slot 9 / Assigned "real" slot
8	Slot_101	USInt	16#FF		☑	☑		Configured slot 101 / Assigned "real" slot
9	Slot_102	USInt	16#FF		☑	☑		Configured slot 102 / Assigned "real" slot
10	Slot_103	USInt	16#FF		☑	☑		Configured slot 103 / Assigned "real" slot

3. Transfer the control data record to the CPU in the startup program.
The configuration control for the centrally inserted modules takes effect only after an operating mode change of the CPU from STOP to RUN. For this reason, call the extended WRREC (Write data record) instruction in the startup OB, and transfer the created control data record to the CPU; see next section.

If a valid control data record is not transferred in the startup OB, the control is not ready for operation. The CPU returns from startup to STOP in this case.

Transferring a control data record in the startup program

The CPU processes the WRREC instruction for asynchronous transfer of the control data record. For this reason, you must call WRREC in the startup OB repeatedly in a loop until the output parameter "BUSY" or "DONE" indicate that the data record has been transferred.

Tip: Use the SCL programming language with the REPEAT ... UNTIL instruction for programming the loop.

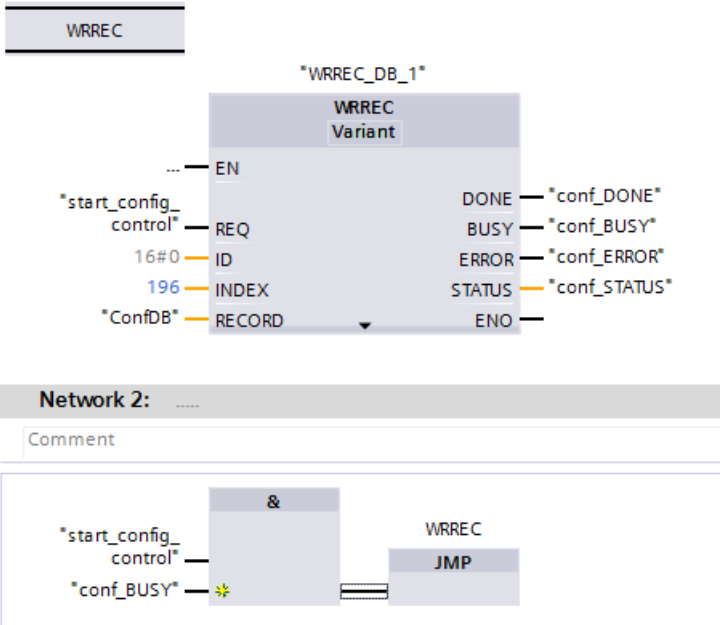
```
REPEAT
    "WRREC_DB"(REQ := "start_config_control",
               ID := 33,
               INDEX := 196,
               LEN := "conf_LEN",
               DONE => "conf_DONE",
               BUSY => "conf_BUSY",
               ERROR => "conf_ERROR",
               RECORD := "ConfDB",
               STATUS => "conf_STATUS");
UNTIL NOT "conf_BUSY"
END_REPEAT;
```

Below, you will find explanations for individual block parameters that you must supply with specific values in the configuration control context. For the remaining block parameters, see also WRREC (Page 3071):

Parameter	Explanation
ID	HW identifier; is always 33 (decimal) for configuration control for centrally arranged modules.
INDEX	Data record number; is always 196 (decimal) for configuration control for centrally arranged modules.
RECORD	Control data record to be transferred. See the section "Control data record" for the structure of the control data record. Tip: The "RECORD" block parameter of the WRREC instruction (V1.1 and higher) is of the "VARIANT" data type and therefore requires a tag with data type. If you store the control data record in a data block, this data block must thus have a data type. The data block created must not be of the "Global-DB" type. Rather, it must be derived from a user data type. Proceed as follows: 1. First, create a new PLC data type (user data type) with the structure of the control data record and name it, for example, "ConfDB". 2. Create a new data block. Select the newly created user data type, e.g., "ConfDB", as the type for this data block.

In graphical programming languages, you realize the loop using instructions for program control.

Example in FBD: Use the LABEL (jump label) and JMP (jump at RLO=1) instructions to program a loop.



Arrangement of the modules

The following table shows the slot number assignment:

Slot	Module	Comment
1	Signal board, communication board, battery board	Slot on front of the CPU
2 - 9	Signal modules	Slots to the right of the CPU
101 - 103	Communications modules	Slots to the left of the CPU

Control data record

A control data record 196 containing a slot assignment is defined for the configuration control.

The following codes apply:

	Module is included in the hardware configuration but is not used in the current configuration.
1 to 9, 101 to 103	Current slot of the module
16#FF (255)	A module is not located in this slot in the hardware configuration.

Byte	Element	Code	Explanation
0	Block length	4 + number of slots	Header
1	Block ID	196	
2	Version	5 (for central I/O)	
3	Subversion	0	
4	Assignment of the CPU expansion board	Expansion board, 0 or 16#FF	Control element Describes in each element which real slot in the device is assigned to the configured slot. The structure of a control element is described in the following section.
5	Assignment of configured slot 2	Real slot, 0 or 16#FF	
...	...	...	
12	Assignment of configured slot 9	Real slot, 0 or 16#FF	
13	Assignment of configured slot 101	Real slot, 0 or 16#FF	In contrast to signal modules, the real slot of the communications modules must correspond to the configured slot.
14	Assignment of configured slot 102	Real slot, 0 or 16#FF	
15	Assignment of configured slot 103	Real slot, 0 or 16#FF	

Structure of a control element

A control element contains the information on which module is inserted in which slot.

The byte numbers represent the configured slots in ascending order (see above):

- Byte 4 stands for the configured slot of the expansion board
- Byte 5 to byte 9 stand for the configured slots 2 to 9
- Byte 13 to byte 15 stand for the configured slots 101 to 103

Which value you must enter in the respective byte results from the following rule:

- If the module is present in the real configuration, enter the real slot number of the module.
  - Example 1: The module configured in slot 2 is located in slot 2.
    Enter value 2 (= actual slot) in byte 5 (= configured slot 2).
  - Example 2: The module configured in slot 3 is located in slot 2.
    Enter value 2 (= actual slot) in byte 6 (= configured slot 3).
- If the module is configured but is missing in the real structure, enter 0 in the byte for the configured slot.
- If a module is not located in this slot in the hardware configuration, enter 16#FF (255) in the byte for the configured slot.

Rules

Observe the following rules:

- The configuration control does not support the repositioning of communications modules. The slot entries in the control data record for slots 101 to 103 must correspond to the real positions of the modules or be defined as not present in the hardware configuration by entering 16#FF (255).
- Slot gaps in the configuration are not allowed. For example, if a signal module is inserted in slot 4 in the real configuration, slots 2 and 3 of the real configuration must also be occupied. The same applies to slots 101 to 103. If a communication module is inserted in slot 102 in the real configuration, slot 101 must also have a communication module inserted in the real configuration.
- If you have enabled configuration control, the CPU is not ready for operation without a control data record. The CPU returns from startup to STOP if a valid control data record is not transferred in the startup OB. The central I/O is not initialized in this case. The cause for the STOP mode is entered in the diagnostics buffer.
- For addressing the WRREC instruction, use the HW identifier 33 (decimal, for the ID block parameter) to write the control data record.
- The control data record is saved retentively in the CPU, which means that it is not necessary to write the control data record 196 again at a restart if the configuration is unchanged. Prior to commissioning, we recommend that a memory reset be performed for the CPU in order to delete any control data record that is present.
- Slot entries in the control data record outside the configured preset configuration are ignored by the CPU.
- Each real slot may only be present once in the control data record.
- A real slot may only be assigned to one configured slot.

Note

Modified configuration

The writing of a control data record with a modified configuration triggers the following automatic reaction by the CPU:

Memory reset with subsequent startup with this modified configuration.

As a result of this reaction, the retentively saved original data record 196 is deleted and the new data record 196 is saved retentively.

Behavior during operation

Effect of the discrepancy between the configured configuration and real configuration:

- For the online display and for the display in the diagnostics buffer (module OK or module faulty), the hardware configuration is always used and not the differing real configuration. Example: A module supplies diagnostic information. This module is configured in slot 4, but is really inserted in slot 3 (missing module; see example in the next section). In the online view, a configured slot 4 is indicated as faulty. In the real configuration, the module in slot 3 indicates an error via an LED display.

If modules are entered as missing in the control data record, the automation system behaves as follows:

- Modules designated as not present in the control data record do not supply diagnostic information and their status is always OK. The value status is OK.
- Direct write access to the outputs or write access to the process image of outputs that are not present: Remains without effect; no access error is signaled.
- Direct read access to the inputs or read access to the process image of inputs that are not present: Value "0" is supplied; no access error is signaled.
- Write data record to module that is not present: Remains without effect; no error is signaled.
- Read data record from module that is not present: An error is signaled because a valid data record cannot be returned.

Error messages

The following error messages are returned if an error occurs during writing of the control data record:

Table 10-66 Error messages

Error code	Meaning
16#80B1	Invalid length; the length information in data record 196 is not correct.
16#80B5	Configuration control parameters not assigned.
16#80E2	Data record was transferred in the wrong OB context. The data record must be transferred in the startup program.
16#80B8	Parameter error; module signals invalid parameters.

See also

VARIANT (Page 1951)

S7-1200 System Manual (http://support.automation.siemens.com/WW/view/en/89851659)

Example of a configuration control

A configuration consisting of a CPU and 3 signal modules is configured in the following.

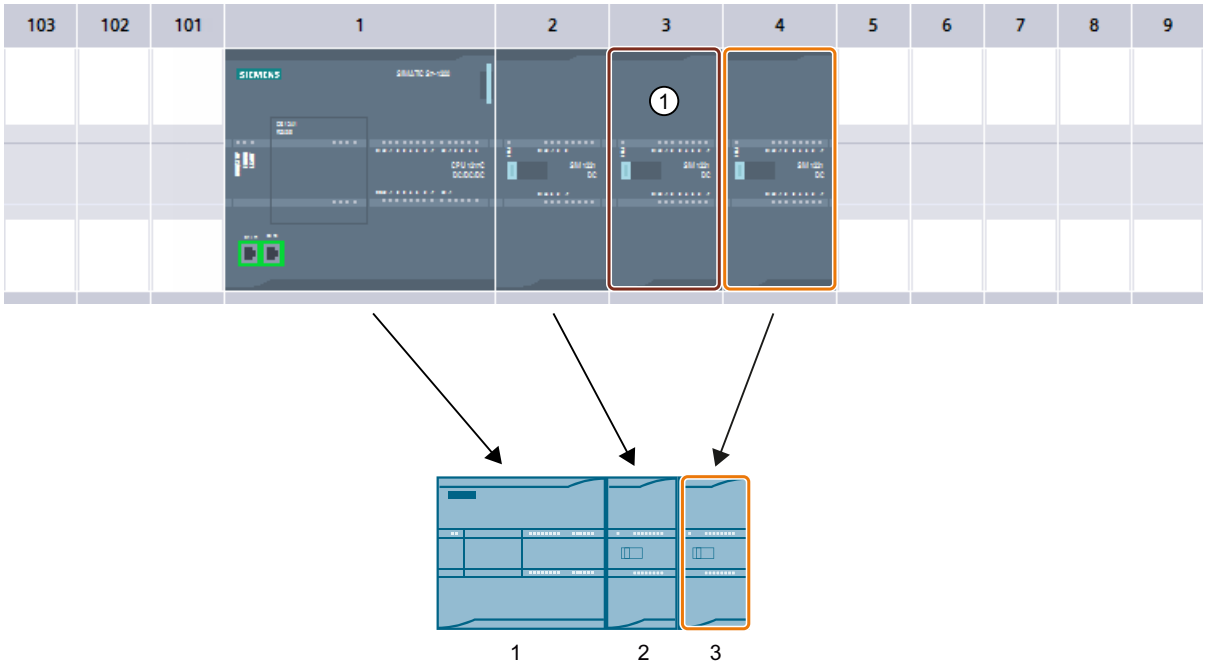
The module at slot 3 is not present in the first configuration expansion and is "hidden" by the configuration control.

In the second configuration expansion, the module that was initially hidden is located in the last slot. The added slot is made known to the CPU by a modified control data record.

Actual configuration with missing module

The specified configuration contains all modules that can be present in a final expansion stage.

The module that is inserted in slot 3 in the specified configuration is missing in the real expanded configuration. For this reason, slot 3 must be coded in the control data record accordingly with "FF_H" (= not present).



ControlDataRecord

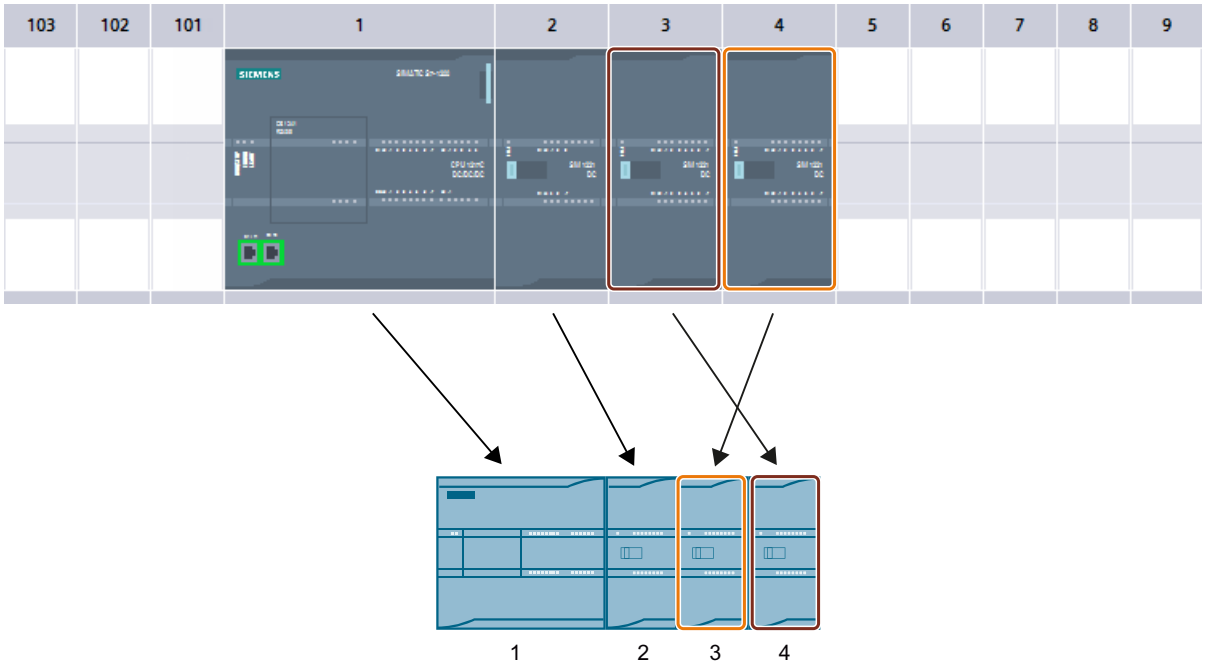
	Name	Data type	Start value	Comment
1	Static			
2	ConfigControl	Struct		
3	Block_length	USInt	16	Length of control data record, including header
4	Block_ID	USInt	196	Data record number
5	Version	USInt	5	
6	Subversion	USInt	0	
7	Slot_1	USInt	16#FF	Assignment for CPU annex card/Actual annex card
8	Slot_2	USInt	2	Configured slot 2 / Assigned "real" slot 2
9	Slot_3	USInt	16#FF	Configured slot 3 / not used ①
10	Slot_4	USInt	3	Configured slot 4 / Assigned "real" slot 3
11	Slot_5	USInt	16#FF	Configured slot 5 / not used
12	Slot_6	USInt	16#FF	Configured slot 6 / not used
13	Slot_7	USInt	16#FF	Configured slot 7 / not used
14	Slot_8	USInt	16#FF	Configured slot 8 / not used
15	Slot_9	USInt	16#FF	Configured slot 9 / not used
16	Slot_101	USInt	16#FF	Configured slot 101 / not used
17	Slot_102	USInt	16#FF	Configured slot 102 / not used
18	Slot_103	USInt	16#FF	Configured slot 103 / not used

① Module is missing in the real configuration

Actual configuration with subsequently added module

The module present in slot 3 in the specified configuration is added to the back of the real configuration by inserting it as the last module in slot 4.

The control data record is adapted accordingly.



ControlDataRecord

		Name	Data type	Start value	Comment
1		Static			
2		ConfigControl	Struct		
3		Block_length	USInt	16	Length of control data record, including header
4		Block_ID	USInt	196	Data record number
5		Version	USInt	5	
6		Subversion	USInt	0	
7		Slot_1	USInt	16#FF	Assignment for CPU annex card/Actual annex card
8		Slot_2	USInt	2	Configured slot 2 / Assigned "real" slot 2
9		Slot_3	USInt	4	Configured slot 3 / Assigned "real" slot 4
10		Slot_4	USInt	3	Configured slot 4 / Assigned "real" slot 3
11		Slot_5	USInt	16#FF	Configured slot 5 / not used
12		Slot_6	USInt	16#FF	Configured slot 6 / not used
13		Slot_7	USInt	16#FF	Configured slot 7 / not used
14		Slot_8	USInt	16#FF	Configured slot 8 / not used
15		Slot_9	USInt	16#FF	Configured slot 9 / not used
16		Slot_101	USInt	16#FF	Configured slot 101 / not used
17		Slot_102	USInt	16#FF	Configured slot 102 / not used
18		Slot_103	USInt	16#FF	Configured slot 103 / not used

Additional configurations

Configuring additional functions

The S7-1200 automation system has numerous additional functions that are useful as integrated CPU functions or available via plug-in modules (for example, communication modules). You can find the description via the following links.

See also

Overview of point-to-point communication (Page 1197)

General information on high-speed counters (Page 1193)

Configuring PID_Compact V1 (Page 7243)

Configuring PID_3Step V1 (Page 7276)

Motion functionality of the CPU S7-1200 (Page 7330)

10.1.4.4 S7-1200 CM/CP

S7-1200 CM/CP

Telecontrol S7-1200

Addressing a double / redundant TCSB system

Addressing of the duplicate or redundant telecontrol server

- **Addressing the main and substitute telecontrol server in TCSB V2**
  With TCSB V2, two independent server PCs can be installed. The second IP address only needs to be configured when these are connected to the Internet via two routers.
- **Addressing the TCSB redundancy group in TCSB V3**
  In the LAN in the master station to which the TCSB server PCs and the DSL router (e.g. SCALANCE M) are connected, the Network Load Balancing (NLB) of the computer operating system will assign a common virtual IP address to the two server PCs.
  This IP address is configured depending on the network setup:
  - If only CP 1243-1 modules without a DSL router are connected, the virtual address assigned by the NLB must be configured in the CPs as the IP address of the telecontrol server.
  - If a DSL router is used, only one IP address will be configured to address the redundant telecontrol server in the stations, the public address of the DSL router.
    Set the port forwarding on the DSL router so that the public IP address (external network) is led to the virtual IP address of the TCSB server PCs (internal network). Only the public IP address is reachable from the Internet. The station does not therefore receive any information telling it which of the two computers of the redundancy group it is connected to.

  If you configure a second IP address, you need to make sure that TCSB is reachable via the IP address of a second router.

Mobile wireless CPs

S7-1200 telecontrol CPs Load, TeleService, project number, station number

Changing the project number or station number for the entire STEP 7 project

If you change the project number or the station number in the "CP identification" parameter group for a telecontrol CP, this parameter is changed for all CPs in the STEP 7 project.

Mobile wireless CPs: Download / TeleService

The following behavior applies to all mobile wireless CPs:

- CP 1242-7 (6GK7 242-7KX30-0XE0)
- CP 1242-7 GPRS V2 (6GK7 242-7KX31-0XE0)
- CP 1243-7 LTE-EU (6GK7 243-7KX30-0XE0)
- CP 1243-7 LTE-USA (6GK7 243-7SX30-0XE0)

Connection resources for TeleService

The TeleService function occupies a connection resource on the engineering station.

The function download to device or upload from device during a TeleService session occupies a second connection resource on the engineering station.

Download to device

Only use the "Download to device" function with the mobile wireless CP via a TeleService connection as follows:

1. Select the CP in STEP 7.
2. Select the "Online" > "Download to device" menu.
3. In the "Extended download" dialog that appears, select the TeleService interface.
4. Download the project data from the "Extended download" dialog.

Upload from device

The "Upload from device" function via a TeleService connection is supported by the mobile wireless CPs with the following TeleService server applications:

- TeleControl Server Basic as of version V3
- TeleService Gateway as of version V3

Operating modes of CP 1242-7

Modes of the CP

The CP 1242-7 allows an S7-1200 to communicate as a GPRS station with a central station or other remote networks via the GSM network. For communication using GPRS, the CP is set to one of the following operating modes:

- **Telecontrol**
  This operating mode of the CP allows the GPRS station to exchange data with the following partners:
  - Communication with the Telecontrol server
    This CP mode allows the GPRS station to exchange data with a telecontrol server. The Telecontrol server is a PC connected to the Internet with the "TCSB" application. It is generally located in the master station and serves to monitor and control the remote GPRS stations. Data can be exchanged with the OPC client of a central control system via the integrated OPC interface.
    The Telecontrol server PC is not configured in STEP 7. The "TCSB" application has its own configuration user interface.
  - Communication with another remote GPRS station
    The message frames are transmitted via the Telecontrol server.
  - Communication with an engineering station (for TeleService)

  Communication with the Telecontrol server is performed via the GSM network and the Internet.
  This operating mode requires a SIM card with GPRS service enabled and a Telecontrol server that can be reached by the CP.
- **GPRS direct**
  This operating mode of the CP is used for direct communication between remote stations via the GSM network. No Telecontrol server is required.
  To allow network nodes in public wireless networks to be directly accessible, these need to be addressed using a fixed address. Here, SIM cards with a fixed IP address are used that allow the stations to address each other directly.
  The possible communications services and security functions (for example VPN) depend on what is offered by the network provider.
  Possible communications partners of the station with a CP 1242-7 in "GPRS direct" mode are:
  - A node that can be reached by the CP via an IP address (S7 station with CP 1242-7)
  - An engineering station (for TeleService)

See also

Connection establishment with a CP 1242-7 (Page 899)

Connection establishment with a CP 1242-7

Connection modes

- "GPRS direct" mode
  There are no different connection modes in the "GPRS direct" mode.
- "Telecontrol" mode
  The CP can be configured for the following connection modes.
  - "Permanent" connection mode
    There is a permanent TCP connection to the telecontrol server. Following connection establishment, there is a permanent TCP connection to the telecontrol server even if data is not transferred permanently.
  - "Temporary" connection mode
    A connection is only established to the telecontrol server when required.

If a TCP connection is established, process data is sent as soon as the telecontrol instructions are called on the CPU.

A connection is always established by the CP. If a connection established by the CP is interrupted, the CP automatically attempts to re-establish the connection.

Triggering connection establishment for permanent stations ("Telecontrol" mode)

In the "Telecontrol" mode, the permanent connection to the telecontrol server is established when the station starts up. If there is an interruption on the connection, the establishment of the connection can be triggered by a wake-up SMS (see below).

Triggering connection establishment for temporary stations ("Telecontrol" mode)

With "temporary" stations, connection establishment can be triggered by the following events:

- Event on the local CPU that needs to be evaluated by the program.
  In terms of the program, two situations need to be distinguished:
  - Events that lead to a single connection establishment (for example alarms or commands from the operator).
  - Expiry of an interval that leads to cyclic connection establishment (for example once daily for data transmission)
- Request by a communications partner (OPC client or S7 station)
  The request from the communications partner leads to the connection being established.
- Request for TeleService by an engineering station
  The request switched by the telecontrol server or TeleService gateway does not need to be evaluated in the program.
- Wake-up SMS of the telecontrol server
  The wake-up SMS can be triggered spontaneously on the telecontrol server. It is also possible to configure cyclic sending on the telecontrol server.

- Telephone wake-up call
  The wake-up call can be sent from a telephone that has a phone number authorized in the STEP 7 project. The telephone must support the CLIP function (transfer of its own call number).
  The connection establishment with the (main) telecontrol server is triggered.
- Telephone wake-up SMS
  The wake-up SMS can be sent from a telephone that has a phone number authorized in the STEP 7 project. The telephone must support the CLIP function (transmission of its own number) and sending of SMS.
  The connection establishment with the telecontrol server specified in the SMS is triggered.

When a temporary station is woken up, all the data is transferred if this has changed since the last data transfer.

Triggering connection establishment in "GPRS direct" mode

In "GPRS direct" mode, a connection establishment is triggered by the following events:

- Event on the local CPU that is evaluated by the program.
- Request by a communications partner (not an engineering station)
  The request in the frame received from the communications partner is evaluated in the program by calling the telecontrol instructions.
- Request for TeleService by an engineering station
  The request switched by the telecontrol server or TeleService gateway does not need to be evaluated in the program.

Right to wake-up by "authorized phone numbers"

The CP only accepts an SMS if the sending communication partner is authorized based on its phone number. These numbers are in configured for the CP in STEP 7 in the "authorized phone numbers" list.

Note

"Authorized phone numbers" in the STEP 7 project

- A phone number entered here gives the sender who transfers this phone number the right to trigger connection establishment.
- If an asterisk (*) is entered in the list, the CP accepts SMS messages from all senders.
- If the list is empty, the CP cannot be woken up for connection establishment.

Wake-up SMS

Depending on the connection type and the triggering server or intermediary TeleService server, the following text must be transferred in the wake-up SMS:

- For telecontrol connections:
  - Text for the wake-up SMS message for establishing a connection to the telecontrol server:
    `TELECONTROL`
  - Text for the wake-up SMS message for establishing a connection to the main telecontrol server:
    `TELECONTROL MAIN`
  - Text for the wake-up SMS message for establishing a connection to the substitute telecontrol server:
    `TELECONTROL BACKUP`

  The configuration of the telecontrol server for the GPRS CP is set in STEP 7 in "Telecontrol interface > Operating mode > main or substitute telecontrol server".

  Note

  Wake-up with a mobile phone

  - One of the texts listed above can be used in a wake-up SMS message.
  - With a wake-up call, the station always connects to the main telecontrol server.

- For TeleService connections:
  - Text for the wake-up SMS message for establishing a connection to the first configured TeleService server:
    `TELESERVICE`
    or
    `TELESERVICE 1`
  - Text for the wake-up SMS message for establishing a connection to the second configured TeleService server:
    `TELESERVICE 2`

  The configuration of the TeleService server for the GPRS CP is set in STEP 7 in "Telecontrol interface > TeleService authorization > 1st or 2nd TeleService server".

Preferred GSM networks

Selection of the preferred mobile wireless networks

The following options are available to select the networks that the mobile wireless CP should preferably dial up:

- Automatic dialup
  The CP dials with highest priority into the mobile wireless network of the contracted network provider set on the SIM card. If a dialing up the contract network fails, the CP dials up other mobile wireless networks with which the network provider has roaming contracts and whose access data is stored on the SIM card.
- Contract network only
  The CP only dials up the mobile wireless network of the contract network provider whose SIM card is plugged into the CP. No roaming.
- Contracted network and alternative network
  The CP dials up the contracted network with the highest priority. If dialing up the contract network is unsuccessful, the CP dials up alternative mobile wireless networks, in decreasing priority, as entered in the list of preferred network providers.
  The alternative networks are entered in the list as "Public Land Mobile Network" (PLMN). PLMN is a construct of Mobile Country Code (MCC) and Mobile Network Code (MNC).
  Example: 26276
  This is the PLMN for the test network of Siemens AG with MCC = 262 and MNC = 76.

CP 1200 with data point configuration

Reconnection delay

"Reconnection delay" parameter

The reconnection delay for connections in telecontrol communication is the waiting time between repeated attempts to establish the connection by the CP when the telecontrol server is not reachable or the connection has aborted. This waiting time avoids continuous connection establishment attempts at short intervals if there are connection problems.

A basic value is configured for the waiting time before the next connection establishment attempt. Starting at the basic value, the current waiting time is doubled after every 3 unsuccessful retries up to a maximum value of 900 s. Range of values for the basic value: 10 to 600 s.

Example: The basic value 20 results in the following intervals (waiting times) between the attempts to re-establish a connection.

- three times 20 s
- three times 40 s
- three times 80 s
- etc. up to max. 900 s

If a second telecontrol server or a second router of the telecontrol server is configured, the CP attempts to connect to the second partner at the 4th attempt. If the second partner is also unreachable, the 7th time the CP attempts to connect to the first partner again and so on.

Note

If the partner cannot be reached, connection establishment via the mobile wireless network can take several minutes. This may depend on the particular network and current network load.

Depending on your contract, costs may result from each connection establishment attempt.

CPU scan cycle

Structure of the CPU scan cycle

The cycle (including the pause) with which the CP scans the memory area of the CPU is made up of the following phases:

- **High-priority read jobs**
  For data points of the type "Input", which are configured with the "High priority" setting in the data point configuration in "General > Priority in the scan cycle", the PLC tags are all read in one scan cycle.
- **Write jobs**
  In every cycle, the values of a certain number of unsolicited write jobs are written to the CPU. The number of tags written per cycle is specified for the CP in the "Communication with the CPU" parameter group with the "Max. number of write jobs" parameter. The tags whose number exceeds this value are then written in the next or one of the following cycles.
- **Low-priority read jobs - proportion**
  For data points of the type "Input", which are configured with the "Low priority" setting in the data point configuration in "General > Priority in the scan cycle", the values of a part of the PLC tags are read in every scan cycle.
  The number of tags read per cycle is specified for the CP in the "Communication with the CPU" parameter group with the "Max. number of read jobs" parameter. The tags that exceed this value and can therefore not be read in one cycle are then read in the next or one of the following cycles.
- **Cycle pause time**
  This is the waiting time between two scan cycles. It is used to reserve adequate time for other processes that access the CPU via the backplane bus of the station.

Duration of the CPU scan cycle

Since no fixed time can be configured for the cycle and since the individual phases cannot be assigned a fixed number of objects, the duration of the scan cycle is variable and can change dynamically.

Datapoint types

During the configuration of the user data to be transferred by the CP, each data point is assigned a protocol-specific data point type. The data point types supported by the CP along with the compatible S7 data types are listed below. They are grouped according to format (memory requirements).

The direction relates to the direction of transfer (monitoring direction = "in", control direction = "out").

CP 1243-1, CP 1242-7 GPRS V2, CP 1243-7 LTE-(EU/USA): Supported data point types

Table 10-67 Supported data point types and compatible S7 data types

Format (memory requirements)	Data point type	S7 data types	Operand area
Bit	Digital input	BOOL	I, Q, M, DB
	Digital output	BOOL	I, Q, M, DB
	Command output (CP 1243-1 only)	BOOL	I, Q, M, DB
Byte	Digital input	BYTE, CHAR	I, Q, M, DB
	Digital output	BYTE, CHAR	I, Q, M, DB
Integer with sign (16 bits)	Analog input	INT	I, Q, M, DB
	Analog output	INT	I, Q, M, DB
Counter (16 bits)	Counter input	WORD	I, Q, M, DB
Integer with sign (32 bits)	Analog input	DINT	I, Q, M, DB
	Analog output	DINT	I, Q, M, DB
Counter (32 bits)	Counter input	DWORD, UDINT	I, Q, M, DB
Floating-point number with sign (32 bits)	Analog input	REAL	Q, M, DB
	Analog output	REAL	Q, M, DB
Floating-point number with sign (64 bits)	Analog input	LREAL	Q, M, DB
	Analog output	LREAL	Q, M, DB
Block of data (1 .. 64 bytes)	Data	ARRAY [1]	DB
	Data	ARRAY [1]	DB

[1] For the possible formats of the ARRAY data type, refer to the following section.

Block of data (ARRAY)

With the ARRAY data type, contiguous memory areas up to a size of 64 bytes can be transferred.

Compatible components of ARRAY are the following uniform S7 data types with a size between 1 and 32 bytes:

- BYTE, CHAR (in total up to 64 times per block of data)
- INT (in total up to 32 times per block of data)
- DINT, UDINT, REAL (in total up to 16 times per block of data)

If the array is modified later, the data point must be recreated.

Time stamp in UTC format

Time stamps are transferred in UTC format (48 bits) and contain the time difference in milliseconds since 01.01.1970.

CP 1243-1 DNP3: Supported data point types

Table 10-68 Supported data point types, DNP3 object groups, variants and compatible S7 data types

Format (memory requirements)	Data point type	DNP3 object group [variations]	Direction	S7 data types	Operand area
Bit	Binary Input	**1** [1, 2]	in	BOOL	I, Q, M
	Binary Input Event	**2** [1, 2]	in	BOOL	I, Q, M
	Binary Output [1]	**10** [2]	out		
	Binary Output Event [1]	**11** [1, 2]	out		
	Binary Command	**12** [1]	out	BOOL	I, Q, M
Integer (16 bits)	Counter Static	**20** [2]	in	UINT, WORD	I, Q, M
	Frozen Counter [2]	**21** [2, 6]	in		
	Counter Event	**22** [2, 6]	in	UINT, WORD	I, Q, M
	Frozen Counter Event [3]	**23** [2, 6]	in		
	Analog Input	**30** [1]	in	INT	I, Q, M
	Analog Input Event	**32** [2, 4]	in	INT	I, Q, M
	Analog Output Status [4]	**40** [2]	out		
	Analog Output	**41** [2]	out	INT	I, Q, M
	Analog Output Event [4]	**42** [2, 4]	out		
Integer (32 bits)	Counter Static	**20** [1]	in	UDINT, DWORD	I, Q, M
	Frozen Counter [2]	**21** [1, 5]	in		
	Counter Event	**22** [1, 5]	in	UDINT, DWORD	I, Q, M
	Frozen Counter Event [3]	**23** [1, 5]	in		
	Analog Input	**30** [2]	in	DINT	I, Q, M
	Analog Input Event	**32** [1, 3]	in	DINT	I, Q, M
	Analog Output Status [4]	**40** [1, 3]	out		
	Analog Output	**41** [1]	out	DINT	I, Q, M
	Analog Output Event [4]	**42** [1]	out		
Floating-point number (32 bits)	Analog Input	**30** [5]	in	REAL	Q, M
	Analog Input Event	**32** [5, 7]	in	REAL	Q, M
	Analog Output Status [4]	**40** [3]	out		
	Analog Output	**41** [3]	out	REAL	Q, M
	Analog Output Event [4]	**42** [5, 7]	out		
Floating-point number (64 bits)	Analog Input	**30** [6]	in	LREAL	Q, M
	Analog Input Event	**32** [6, 8]	in	LREAL	Q, M
	Analog Output	**41** [4]	out	LREAL	Q, M
	Analog Output Event [4]	**42** [6, 8]	out		

Format (memory requirements)	Data point type	DNP3 object group [variations]	Direction	S7 data types	Operand area
Block of data (1...64 bytes) [5]	Octet String	**110** [-]	in	[5]	I, Q, M
	Octet String Event [5]	**111** [-]	in	[5]	I, Q, M

[1] This object group can be configured in the Data point editor of STEP 7 using object group 12.

[2] This object group can be configured in the Data point editor of STEP 7 using object group 20.

[3] This object group can be configured in the Data point editor of STEP 7 using object group 22.

[4] This object group can be configured in the Data point editor of STEP 7 using object group 41.

[5] With these data point types, contiguous memory areas up to a size of 64 bytes can be transferred. All S7 data types with a size between 1 and 64 bytes are compatible. If the array is modified later, the data point must be recreated.

Configuration and feedback of binary and analog values

- **Binary values**
  You configure binary values (inputs) using object groups 1 and 2.
  The object groups 10 and 11 are used to feed back the current values in the process image of the CPU to the control system.
- **Analog values**
  You configure analog values using the object groups 30, 32 and 41 depending on the transfer direction.
  The object groups 40 and 42 are used to feed back current values in the process image of the CPU to the control system.

CP 1243-1 IEC: Supported data point types

Table 10-69 Supported data point types, IEC types and compatible S7 data types

Format (memory requirements)	Data point type	IEC type	Direction	S7 data types	Operand area
Bit	Single point information	<1>	in	BOOL	I, Q, M
	Single point information with time tag	<30>	in	BOOL	I, Q, M
	Single command	<45>	out	BOOL	I, Q, M
Byte	Step position information	<5>	in	BYTE	I, Q, M
	Step position information with time tag	<32>	in	BYTE	I, Q, M
Integer (16 bits)	Measured value, normalized value	<9>	in	INT	I, Q, M
	Measured value, normalized value with time tag	<34>	in	INT	I, Q, M
	Measured value, scaled value	<11>	in	INT	I, Q, M
	Measured value, scaled value with time tag	<35>	in	INT	I, Q, M
	Set point command, normalised value	<48>	out	INT	I, Q, M
	Set point command, scaled value	<49>	out	INT	I, Q, M

Format (memory requirements)	Data point type	IEC type	Direction	S7 data types	Operand area
Integer (32 bits)	Bitstring of 32 bits	<7>	in	DWORD, UDINT	I, Q, M
	Bitstring of 32 bits with time tag	<33>	in	DWORD, UDINT	I, Q, M
	Integrated totals	<15>	in	DWORD, UDINT	I, Q, M
	Integrated totals with time tag	<37>	in	DWORD, UDINT	I, Q, M
Floating-point number (32 bits)	Measured value, short floating point number	<13>	in	REAL	Q, M
	Measured value, short floating point number with time tag	<36>	in	REAL	Q, M
	Set point command, short floating point number	<50>	out	REAL	Q, M
Block of data (1...32 Bit) [1]	Bitstring of 32 bits [1]	<7>	in	[1]	I, Q, M
	Bitstring of 32 bits with time tag [1]	<33>	in	[1]	I, Q, M
	Bitstring of 32 bits [1]	<51>	out	[1]	I, Q, M

[1] With these data point types, contiguous memory areas up to a size of 32 bits can be transferred. All S7 data types with a size between 1 and 32 bits are compatible. If the array is modified later, the data point must be recreated.

Process image, type of transmission, event classes, triggers

Storage of values

The values of all data points are stored in the image memory of the CP. Values in the image memory are transferred only after being called by TCSB (CP 1243-1, CP 1242-7 GPRS V2, CP 1243-7 LTE) or the master (CP 1243-1 DNP3 / CP 1243-1 IEC).

Events are also stored in the send buffer and can be transferred unsolicited.

The image memory, the process image of the CP

The image memory is the process image of the CP. All the current values of the configured data points are stored in the image memory. New values of a data point overwrite the last stored value in the image memory.

The values are sent after querying the communications partner, see "Transfer after call" in the section "Types of transmission" below.

The send buffer

The send buffer of the CP is the memory for the individual values of data points that are configured as an event. The send buffer has the following size:

- CP 1242-7 GPRS V2 / CP 1243-7 LTE / CP 1243-1 / CP 1243-1 DNP3: Max. 64000 events
- CP 1243-1 IEC: Max. 65535 events

The capacity of the send buffer is divided up equally for all enabled partners.

If the connection to a communications partner is interrupted, the individual values of the events are retained in the buffer. When the connection returns, the buffered values are sent. The frame memory operates chronologically; in other words, the oldest frames are sent first (FIFO principle).

If a frame was transferred to the communications partner, the transferred values are deleted from the send buffer.

If frames cannot be transferred for a longer period of time and the send buffer is threatening to overflow, the response is as follows:

- CP 1243-1 / CP 1242-7 GPRS V2 / CP 1243-7 LTE
  The forced image mode
  If the send buffer reaches a fill level of 80%, the CP changes to the forced image mode. New values from data points configured as an event are no longer added to the send buffer but rather they overwrite older existing values in the image memory.
  When the connection to the communications partner returns, the CP changes back to the send buffer mode if the fill level of the send buffer has fallen below 50%.
- CP 1243-1 DNP3 / CP 1243-1 IEC
  If a send buffer fill level of 100% is reached, the oldest values are overwritten.

Configuration of data points as events

Data points are configured as a static value or as an event using the "Type of transmission" parameter (see below):

- No event (static value)
  The values of data points that are not configured as an event ("Transfer after call") are entered in the image memory of the CP and transferred to the communications partner when it requests this value.
- Event
  The values of data points configured as an event are entered in the image memory and also in the send buffer of the CP.
  The values of events are saved in the following situations:
  - The configured trigger conditions are fulfilled (data point configuration > "Trigger" tab, see below)
  - The value of a status bit of the status identifiers changes.

Status identifiers: Generating an event on a status change

With data points that are configured as an event, the change to the status bit leads to an event being generated, refer also to the section Status IDs of data points (Page 910).

Example: When the value of a data point configured as an event is updated during startup of the station by reading the CPU data for the first time, the status "RESTART" of this data point changes (bit status change 1 → 0). This leads to generation of an event.

Type of transmission

Depending on your CP type, you have the following transmission types available:

- **Transfer after call**
  The current value of the data point is entered in the image memory of the CP. A new value overwrites the older value in the image memory. After being called by the communications partner, the current value at the time is transferred.
- **Event class ...**
  The value is entered in the send buffer as an event and transferred unsolicited to the communications partner.
  The configurable event classes of the various CPs are described in the following sections.

Event classes with the CP 1243-1 / CP 1242-7 GPRS V2 / CP 1243-7 LTE

The process data of the various event classes is handled as follows:

- **Every value triggered**
  Each value change is entered in the send buffer in chronological order.
- **Current value triggered**
  Only the last, current value is entered in the send buffer. It overwrites the value stored there previously.

Event classes with a CP 1243-1 DNP3

The process data of the various event classes is handled as follows:

- **Event class 1 / event class 2**
  Classes according to the DNP3 protocol: Event, class 1 / class 2
  Each value change is entered in the send buffer in chronological order.
  The evaluation of the classification (1 or 2) must be handled by the master.
- **Event class 3**
  Class according to the DNP3 protocol: Static event, class 3
  Only the current value at the time the trigger condition was met is entered in the send buffer and overwrites the last value stored there.

Event classes with a CP 1243-1 IEC

The process data of the various event classes is handled as follows:

- **Event class 1**
  Classes according to the IEC protocol: Event, class 1
  Each value change is entered in the send buffer in chronological order.
- **Event class 3**
  Class according to the IEC protocol: Static event, class 3
  Only the current value at the time the trigger condition was met is entered in the send buffer and overwrites the last value stored there.

Trigger

Various trigger types are available for starting event-driven transfer:

- **Threshold value trigger**
  The value of the data point is transferred when this reaches a certain threshold. The threshold is calculated as the difference compared with the last stored value, refer to the section Threshold value trigger (Page 912).
- **Time trigger**
  The value of the data point is transferred at configurable intervals or at a specific time of day.
- **Event trigger**
  The value of the data point is transferred when a configurable trigger signal is fired. For the trigger signal, the edge change (0 → 1) of a trigger tag is evaluated that is set by the user program. When necessary, a separate trigger tag can be configured for each data point.
  Resetting the trigger tag in the bit memory area / DB:
  If the memory area of the trigger tag is in the bit memory or in a data block, the trigger tag is reset to zero when the data point value is transferred.

You specify whether the value of a data point is transferred to the communications partner immediately after the trigger fires or after a delay in the "Transmission mode" parameter.

Transmission mode

The transmission mode of a frame is set in the "Trigger" tab of the data point. With the two options, you specify whether frames of events are sent immediately or following a delay:

- Spontaneous
  The value is transferred immediately.
- Conditional spontaneous
  The value is transferred only when one of the two following conditions is fulfilled:
  - The communications partner queries the station.
  - The value of another event with the transmission mode "Unsolicited" is transferred.

Status IDs of data points

Status identifiers

The status identifiers of the data points listed in the following tables are transferred along with the value in each frame to the communications partner. They can be evaluated by the communications partner.

Depending on the CP type, the CP sets different status identifiers for each data point. They are transferred in 1 or 2 bytes.

The meaning of the individual status bits relates to the value of the bit in the "Bit status" row in the tables.

CP 1243-1, CP 1242-7 GPRS V2, CP 1243-7 LTE-EU, CP 1243-7 LTE-USA

The status identifiers are transferred in 2 bytes. Byte 1 is not used.

Table 10-70 Byte assignment of status byte 0

Bit	7	6	5	4	3	2	1	0
Flag name	-	NON_ EXISTENT	Substituted	LOCAL_ FORCED	CARRY	OVER_ RANGE	RESTART	ONLINE
Meaning	-	Data point does not exist or S7 address unreachable	Substitute value	Local operator control	Counted value overflow before reading the value	Limit value of the analog preprocessing overshot / undershot	Value not updated after start	Value is invalid
Bit status	*(always 0)*	1	1	1	1	1	1	1

CP 1243-1 DNP3

The status identifiers are transferred in 1 byte.

Table 10-71 Bit assignment of the status byte

Bit	7	6	5	4	3	2	1	0
Flag name	-	-	-	LOCAL_ FORCED	DISCONTI NUITY	OVER_ RANGE	RESTART	ONLINE
Meaning	-	-	-	Local operator control	Counted value overflow before reading the value	Limit value of the analog preprocessing overshot / undershot	Value not updated after start	Value is invalid
Bit status	*(always 0)*	*(always 0)*	*(always 0)*	1	1	1	1	1

CP 1243-1 IEC

The status identifiers are transferred in 1 byte.

Table 10-72 Bit assignment of the status byte

Bit	7	6	5	4	3	2	1	0
Flag name	-	-	SB substituted	-	CY carry	OV overflow	NT not topical	IV invalid
Meaning	-	-	Substitute value	-	Counted value overflow before reading the value	Value range exceeded, analog value	Value not updated	Value is invalid
Bit status	*(always 0)*	*(always 0)*	1	*(always 0)*	1	1	1	0

Rules for configuring the data point index:

Configuration of the data point index (CP 1243-1 IEC)

Data point indexes assigned more than once in a CP are indicated as errors in the consistency check and prevent the project being saved.

Configuration of the data point index (CP 1243-1 DNP3)

On a CP, data point indexes must be unique within each of the following object groups:

- Binary Input / Binary Input Event
- Binary Output / Binary Command
- Counter / Counter Event
- Analog Input / Analog Input Event
- Analog Output
- Octet String / Octet String Event

Indexes of two data points in different object groups can be identical.

Configuration of the data point index (CP 1243-1 / CP 1243-7 LTE / CP 1242-7 GPRS V2)

Within a CP, the indexes of the data point classes must comply with the following rules:

- Input
  The index of a data point of the type input must be unique throughout all data point types (digital inputs, analog inputs etc.).
- Output
  - A data point of the type output can have the same index as a data point of the type input.
  - Several data points of the type output can have the same index.

Note

Data points for the inter-station communication with a CP in another S7 station

Note that for inter-station communication, the indexes of the two corresponding data points (data point pair) must be identical for the sending and receiving CP.

Threshold value trigger

The CP calculates the value for the threshold value trigger after the analog value preprocessing, refer to the section Analog value preprocessing (Page 914).

Threshold value trigger: How the integrattion calculation works

To calculate the threshold value trigger, the integration method is used.

In the integration threshold value calculation, it is not the absolute value of the deviation of the process value from the last stored value that is evaluated but rather the amount of the integrated deviation.

The calculation cycle

The integration threshold value calculation works with a cyclic comparison of the integrated current value with the last stored value. The calculation cycle in which the two values are compared is 500 milliseconds.
(Note: The calculation cycle must not be confused with the scan cycle of the CPU memory areas).

The deviations of the current process value are totaled in each calculation cycle. The trigger is set only when the totaled value reaches the configured value of the threshold value trigger and a new process value is entered in the send buffer.

The method is explained based on the following example in which a threshold value of 2.0 is configured.

Table 10-73 Example of the integration calculation of a threshold value configured with 2.0

Time [s] (calculation cycle)	Process value stored in the send buffer	Current process value	Absolute deviation from the stored value	Integrated deviation
0	**20,0**	**20,0**	0	0
0,5		20,3	+0,3	0,3
1,0		19,8	-0,2	0,1
1,5		20,2	+0,2	0,3
2,0		20,5	+0,5	0,8
2,5		20,3	+0,3	1,1
3,0		20,4	+0,4	1,5
3,5	**20,5**	**20,5**	+0,5	**2,0**
4,0		20,4	-0,1	-0,1
4,5		20,1	-0,4	-0,5
5,0		19,9	-0,6	-1,1
5,5		20,1	-0,4	-1,5
6,0	**19,9**	**19,9**	-0,6	**-2,1**

In this example, a value of 2.0 was configured for the threshold value trigger.

With the changes in the process value shown in the example, the threshold value trigger fires twice, if the value 2.0 is reached:

- At the time 3.5 s: The value of the integrated deviation is at 2.0. The new process value stored in the send buffer is 20.5.
- At the time 6.0 s: The value of the integrated deviation is at 2.1. The new process value stored in the send buffer is 19.9.

In this example, if a deviation of the process value of approximately 0.5 should fire the trigger, then with the behavior of the process value shown here a threshold value of approximately 1.5 ... 2.5 would need to be configured.

Analog value preprocessing

CPs with data point configuration support analog value preprocessing with some or all of the functions described below.

Sequence of processing Threshold value trigger and Analog value preprocessing

Note

Threshold value trigger: Calculation only after "Analog value preprocessing"

Note that the analog value preprocessing is performed before the check for a configured threshold value.

This affects the value that is configured for the threshold value trigger.

Restricted preprocessing options if mean value generation is configured

If you configure mean value generation for an analog value event, the following preprocessing options are not available:

- Unipolar transfer
- Fault suppression time
- Smoothing

No Threshold value trigger if Mean value generation is configured

If mean value generation is configured, no threshold value trigger can be configured for the analog value event involved.

Analog inputs that are configured as an event are processed on the CP in the following sequence:

Sequence of analog value processing

1. Reading the data from the input area of the CPU
2. Analog value preprocessing (part 1)
   Processing involves the following steps:
   - Mean value generation
     - Mean value generation configured: Calculation and then continued at point 4.
     - No mean value generation configured: Continue with "Unipolar transfer".
   - Unipolar transfer (if configured)
   - Fault suppression time (if configured)
   - Smoothing (if configured)
3. Threshold value calculation (if Threshold value trigger is configured)
4. Analog value preprocessing (part 2)
   - Set limit value 'low' / Set limit value 'high' (if configured)
5. Storage of the value in the send buffer
   Transfer of the value to the partner if trigger and threshold value conditions are met.

Unipolar transfer

With unipolar transfer, negative values are corrected to zero. This can be desirable if values from the underrange should not be transferred as real measured values.

Exception: The value -32768 / 8000_h for wire break of live zero inputs is transferred.

Unipolar transfer cannot be configured at the same time as mean value generation.

Mean value generation

With this parameter, acquired analog values are transferred as mean values.

The current values of an analog data point are acquired cyclically and totaled. The number of acquired values per time unit depends on the read cycle of the CPU and the CPU scan cycle of the CP. The mean value is calculated from the accumulated values as soon as the transfer is triggered by a time trigger. Following this, the accumulation starts again so that the next mean value can be calculated.

The mean value can also be calculated if the transmission of the analog value message is triggered by a request from the communications partner. The duration of the mean value calculation period is then the time from the last transmission (for example triggered by the trigger) to the time of the request. Once again, the accumulation restarts so that the next mean value can be calculated.

Overflow range / underflow range

Acquisition of a value in the overflow or underflow range results in the mean calculation being stopped immediately. The value 32767 / $7FFF_h$ or -32768 / 8000_h is saved as an invalid mean value for the current mean value calculation period and sent when the next analog value message is triggered. The calculation of a new mean value is then started. If the analog value remains in the overflow or underflow range, this new value is again saved immediately as an invalid mean value and sent when the next frame is triggered.

Note

Fault suppression time > 0 configured

If you have configured an error suppression time and then enable mean value generation, the value of the error suppression time is grayed out but no longer used. If mean value generation is enabled, the error suppression time is set to 0 (zero) internally.

Smoothing factor

Analog values that fluctuate quickly can be evened out using the smoothing function.

The smoothing factors are calculated according to the following formula as with S7 analog input modules.

$$y_n = \frac{x_n + (k-1)y_{n-1}}{k}$$

where
y_n = smoothed value in the current cycle

x_n = value acquired in the current cycle n
k = smoothing factor

The following values can be configured for the module as the smoothing factor.

- 1 = No smoothing
- 4 = Weak smoothing
- 32 = Medium smoothing
- 64 = Strong smoothing

The smoothing factor cannot be configured at the same time as mean value generation.

Fault suppression time

An analog value in the overflow range (32767 / $7FFF_h$) or underflow range (-32768 / 8000_h) is not transferred for the duration of the fault suppression time. This also applies to live zero inputs. The value in the overflow/underflow range is only sent after the fault suppression time has elapsed, if it is still pending.

If the value returns to the measuring range before the fault suppression time elapses, the current value is transferred immediately.

A typical use case for this parameter is the suppression of peak current values when starting up powerful motors that would otherwise be signaled to the control center as a disruption.

The suppression is adjusted to analog values that are acquired by the S7 analog input modules as raw values. These modules return the specified values for the overflow or underflow range for all input ranges (also for live zero inputs).

The fault suppression time cannot be configured at the same time as mean value generation.

Recommendation for finished values that were preprocessed by the CPU:

If the CPU makes preprocessed finished values available in bit memory or in a data block, suppression is only possible or useful if these finished values also adopt the values listed above 32767 / $7FFF_h$ or -32768 / 8000_h in the overflow or underflow range. If this is not the case, the parameter should not be enabled for preprocessed values.

Set limit value 'low' / Set limit value 'high'

In these two input boxes, you can set a limit value in the direction of the start of the measuring range or in the direction of the end of the measuring range. You can also evaluate the limit values, for example as the start or end of the measuring range.

If the limit value is overshot or undershot, the status identifier "OVER_RANGE" of the data point is set. This status identifiers are described in the section Status IDs of data points (Page 910).

The "OVER_RANGE" bit of the status identifier of the data point is set as follows when the relevant analog value is transferred:

- Limit value 'high':
  - If the limit value is exceeded: OVER_RANGE = 1
  - If the value falls below the limit value: OVER_RANGE = 0
- Limit value 'low':
  - If the value falls below the limit value : OVER_RANGE = 1
  - If the value then exceeds the limit value: OVER_RANGE = 0

Requirements for the function

- Configuration of the threshold trigger for this data point
- PLC tag in the bit memory operand area or data area
  The analog value data point must be linked to a PLC tag in the bit memory or data area (data block). For hardware modules (input operand area) limit value configuration is not possible.

The configuration of limit values is pointless for measured values that have already been preprocessed on the CPU.

Configuration of the limit value

The value to be configured as a whole decimal number therefore depends on the value range of the PLC tags and the raw value of the analog module:

Division	Raw value of the of PLC tags *			Module output [mA]			Measuring range [%]
	Decimal		Hexadeci-mal	0 - 20 (unipolar)	-20 - +20 (bipolar)	4 - 20 (life zero)	
	16 bits	32 bits	16 bits				
Overflow	32767	2147483647	7FFF	> 23.515	> 23.515	> 22.810	> 117.593
Overrange	32511 ... 27649	2130769779 ... 1812067105	7EFF ... 6C01	23.515 ... 20.001	23.515 ... 20.001	22.810 ... 20.001	117.593 ... 100.004
Nominal range (unipolar / life zero)	27648 ... 0	1811994624 ... 0	6C00 ... 0000	20 ... 0		20 ... 4	100 ... 0
Nominal range (bipo-lar)	27648 ... 0 ... -27648	1811994624 ... 0 ... -1811994625	6C00 ... 0000 9400		20 ... 0 ... -20		100 ... 0 ... -100
Underrange (unipolar / life zero)	-1 ... -4864	-1 ... -318729855	FFFF ... ED00	-0.001 ... -3.518		3.999 ... 1.185	-0.004 ... -17.59

Division	Raw value of the of PLC tags *			Module output [mA]			Measuring range [%]
	Decimal		Hexadeci-mal	0 - 20 (unipolar)	-20 - +20 (bipolar)	4 - 20 (life zero)	
	16 bits	32 bits	16 bits				
Underrange (bipolar)	-27649 ... -32512	-1812067105 ... -2130769779	93FF ... 8100		-20.001 ... -23.516		-100.004 ... -117.593
Undershoot / wire break	-32768	−2147483648	8000	< -3.518		< 1.185	< -17.593

* The raw values of the measured values relate to the values of 16-bit or 32-bit PLC tags.

Note

Evaluation of the value even when the option is disabled

If you enable one or both options and configure a value and then disable the option later, the grayed out value is nevertheless evaluated.

To disable the two options, delete the previously configured values limit values from the input boxes and then disable the relevant option.

Recommendation for quickly fluctuating analog values:

If the analog value fluctuates quickly, it may be useful to smooth the analog value first if limit values are configured. If the analog value fluctuates close to a limit value for a longer period of time, with a smoothed value you avoid a status change each time the value exceeds/falls below the limit value and so triggers a transfer.

Processing status of e-mails / SMS

Enable status identifier / External status

If the option is enabled, the CP writes a processing status about the sent e-mail / SMS to a PLC tag.

The meaning of the statuses is as follows:

Table 10-74 SMS: Meaning of the status ID output in hexadecimal format

Status	Meaning
0000	Transfer completed free of errors
8001	Error in the transfer, possible causes: • SIM card invalid. • No network • Wrong destination phone number (number not reachable)

Table 10-75 E-mails: Meaning of the processing status output in hexadecimal format

Status	Meaning
0000	Transfer completed free of errors
82xx	Other error message from the e-mail server Apart from the leading "8", the status corresponds to the three-digit error number of the SMTP protocol.
8401	No channel available Possible cause: There is already an e-mail connection via the CP. A second connection cannot be set up at the same time.
8403	No TCP/IP connection could be established to the SMTP server.
8405	The SMTP server has denied the login request.
8406	An internal SSL error or a problem with the structure of the certificate was detected by the SMTP client.
8407	Request to use SSL was denied.
8408	The client could not obtain a socket for creating a TCP/IP connection to the mail server.
8409	It is not possible to write via the connection. Possible cause: The communications partner reset the connection or the connection aborted.
8410	It is not possible to read via the connection. Possible cause: The communications partner terminated the connection or the connection was aborted.
8411	Sending the e-mail failed. Cause: There was not enough memory space for sending.
8412	The configured DNS server could not resolve specified domain name.
8413	Due to an internal error in the DNS subsystem, the domain name could not be resolved.
8414	An empty character string was specified as the domain name.
8415	An internal error occurred in the cURL module. Execution was aborted.
8416	An internal error occurred in the SMTP module. Execution was aborted.
8417	Requests to SMTP on a channel already being used or invalid channel ID. Execution was aborted.
8418	Sending the e-mail was aborted. Possible cause: Execution time exceeded.
8419	The channel was interrupted and cannot be used before the connection is terminated.
8420	Certificate chain from the server could not be verified with the root certificate of the CP.
8421	Internal error occurred. Execution was stopped.
8450	Action not executed: Mailbox not available / unreachable. Try again later.
84xx	Other error message from the e-mail server Apart from the leading "8", the status corresponds to the three-digit error number of the SMTP protocol.
8500	Syntax error: Command unknown. This also includes the error of having a command chain that is too long. The cause may be that the e-mail server does not support the LOGIN authentication method. Try sending e-mails without authentication (no user name).
8501	Syntax error. Check the following configuration data: Message configuration > Message parameters: • Recipient address ("To" or "Cc").
8502	Syntax error. Check the following configuration data: Message configuration > Message parameters: • Email address (sender)

Status	Meaning
8535	SMTP authentication incomplete. Check the "User name" and "Password" parameters in the CP configuration.
8550	SMTP server cannot be reached. You have no access rights. Check the following configuration data: • CP configuration > E-mail configuration: – User name – Password – Email address (sender) • Message configuration > Message parameters: – Recipient address ("To" or "Cc").
8554	Transfer failed
85xx	Other error message from the e-mail server Apart from the leading "8", the status corresponds to the three-digit error number of the SMTP protocol.

Time-of-day synchronization

Time-of-day synchronization - mode

How the time synchronization modes work

For the time-of-day synchronization of the S7-1200 CPs for telecontrol applications, various methods are used.

There should only ever be 1 time master in a station if this can be set.

- **Time-of-day synchronization by the telecontrol server**
  With CPs that communicate with a telecontrol server, this method is always used. The time of day of the CP is synchronized by the communications partner; in other words, by the telecontrol server.
- **NTP - Network Time Protocol**
  With the NTP method, the CP as NTP client sends time-of-day queries at regular intervals to one or more NTP servers in the subnet (LAN). Based on the replies from the server, the most reliable and most accurate time is calculated and the time of day on the CP is synchronized.
  The advantage of this mode is that it allows the time to be synchronized across subnets. The IP addresses of up to four NTP servers need to be configured. The update interval defines the interval between the time queries (in seconds). The value of the interval ranges between 10 seconds and one day.
  In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred; this corresponds to GMT (Greenwich Mean Time).

Time-of-day synchronization with NTP

NTP mode

In NTP mode, the CP sends time-of-day queries at regular intervals to one or more NTP servers. From the responses of the servers, the CP selects the most accurate time of day.

The advantage of this mode is that it allows the time to be synchronized across subnets.

In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred. This corresponds to GMT (Greenwich Mean Time).

NTP - configuration

Configuration of time-of-day synchronization of the CP using NTP

The IP addresses of up to four NTP servers can be configured.

The update interval specifies the synchronization cycle of the time-of-day queries to the NTP server. The range of values is between 10 seconds and 1 day (86400 seconds).

Time-of-day synchronization of the CPU

CPs with data point configuration can make their time of day available to the CPU via a PLC tag; refer to the "Communication with the CPU" parameter group.

CPUs provide the option of requesting the time of day automatically from an NTP server. If this option is used on the CPU, the time of day on the CPU obtained directly from the NTP server is overwritten again by the CP time of day. In this case, it makes sense to activate time-of-day synchronization only on one device.

"Accept time of day from non-synchronized NTP servers" option

If the option is enabled, the CP also accepts the time-of-day from non-synchronized NTP servers with stratum 16.

If the option is disabled, the response is as follows: If the CP receives a time-of-day frame from an unsynchronized NTP server with stratum 16, the time of day is not set according to the frame. In this case, none of the NTP servers is displayed as "NTP master" in the diagnostics; but rather only as being "reachable".

TeleService with mobile wireless CPs

TeleService via the mobile wireless network

Communication path for TeleService via the mobile wireless network

With TeleService for remote S7 stations with a mobile wireless CP, the connection is always established via an intermediary between the engineering station and remote S7 station.

This intermediary can be either:

- A telecontrol server
  The telecontrol server can be a separate PC or installed on the engineering station in the form of the "TCSB" application.
- A TeleService server
  A TeleService server is used when no telecontrol server is available.

The telecontrol server or TeleService server can be connected to the engineering station via LAN or Internet and the TeleService function can be called from there.

Requirements for TeleService via mobile wireless

- A telecontrol server or a TeleService server
- The STEP 7 project with the required stations

Note on project engineering

The telecontrol server and TeleService server are not configured in STEP 7.

Establishing a TeleService connection via the mobile wireless network

Establishing a connection for TeleService via mobile wireless

The request to establish a connection is triggered by the engineering station and transmitted by a wake-up SMS to the station. The mobile wireless CP in the S7-1200 station establishes a connection to the engineering station via the mobile wireless network and the Internet.

Starting the TeleService via mobile wireless

Start the TeleService via mobile wireless as follows:

1. In the project on the authorized engineering station, select the remote S7 station with which you want to establish a TeleService connection via mobile wireless.
2. Alternatively, open the "Connect online" dialog box as follows:
   – "Connect online" button
   – "Connect online" shortcut menu (right mouse button)
   – "Online" > "Connect online" menu

   The "Connect online" dialog box opens.
3. Choose interface type "TeleService mobile wireless" in the "Type of PG/PC interface" drop-down list.
4. In the "PG/PC interface" drop-down list, select the "Mobile wireless TeleService board" option if it is not automatically displayed.

5. Click on the "Connect" icon next to the "PG/PC interface" drop-down list.
   The "Establish remote connection" dialog box opens.
6. Make the necessary settings in the "Establish remote connection" dialog.
   Details of this are contained in the tooltip cascade of STEP 7.

The following details are required to successfully establish a connection:

Details required to establish a connection with the S7 station

The following information is necessary in the Establish remote connection" dialog:

- IP address or DNS name of the telecontrol server
- TCP port number of the telecontrol server or the DSL router via which the connection between the engineering station and the remote S7 station is running.
- Server password of the ES to authenticate the engineering station at the telecontrol server
  Only required if a group-specific password has been configured in the "TCSB" application.
- TeleService user name
  See CP configuration in STEP 7.
- TeleService password
  See CP configuration in STEP 7.
- Access ID of the CP
  Only required when there are several mobile wireless CPs in the station. See CP configuration in STEP 7.

Status

Connection statuses of TeleService via mobile wireless

The connection statuses described below can be displayed in the "Establish remote connection" dialog box.

When opening the dialog

- Not connected
  There is no connection to the remote S7 station. Connection establishment has not yet started.

Click the "Go online" button

If connection establishment was started by clicking the "Connect" button, the following statuses are displayed one after the other after the connection has been established:

- Connect to telecontrol server
  The engineering station connects to the telecontrol server.
- Wait for S7 station
  The wake-up SMS has been sent to the remote station. Wait for reply from station.

- Authentication at S7 station
  The S7 station has established an IP connection with the engineering station via GPRS and Internet and is checking the received logon and authentication data.
- Connected
  The station has successfully established the connection to the engineering station.

If connection establishment is unsuccessful

The following states may be displayed if the connection cannot be successfully established:

- Telecontrol server not accessible
  Possible causes:
  - The connection between the engineering station and the telecontrol server has been interrupted.
  - The telecontrol server is switched off.
- Wrong server password
  Cause: The wrong server password for logging on and authentication was entered at the telecontrol server.
- S7 station not responding
  Possible causes:
  - Faulty GSM communication between the telecontrol server and the station.
  - There is a disrupted connection between the mobile wireless network and the Internet.
  - Faulty Internet connection.
  - The telecontrol server could not send a wake-up SMS.
  - The CP has not received a wake-up SMS.
  - The sender of the SMS was not configured in the list of authorized wake-up call numbers.
- Wrong TeleService user name or TeleService password
  Possible causes:
  - An incorrect TeleService user name or incorrect TeleService password was entered in the authentication dialog for the mobile wireless CP.
  - The TeleService user name or TeleService password was not configured in STEP 7.
- All TeleService access points are in use.
- The CP is not known to the telecontrol server.
  Cause: The CP originates from a STEP 7 project that does not match the project of the telecontrol server.
- No resources available for TeleService on the CP: Please contact the hotline.
- Protocol error
  Cause: Wrong frame or frame from wrong subscriber. Please contact the hotline.

10.1.4.5 SCALANCE X, W and M

Configuring SCALANCE X / W / M

Legal notice

Qualified personnel

The product/system belonging to this documentation may only be handled by qualified personnel for the intended purpose taking into account the documentation relating to the intended purpose and, in particular, the safety and warning notices it contains. Due to training and experience, **qualified personnel** is capable of recognizing risks and avoiding possible dangers when handling these products/systems.

Configuring SCALANCE X

Useful information

VLAN

Network definition regardless of the spatial location of the nodes

VLAN (Virtual Local Area Network) divides a physical network into several logical networks that are shielded from each other. Here, devices are grouped together to form logical groups. Only nodes of the same VLAN can address each other. Since multicast and broadcast frames are only forwarded within the particular VLAN, they are also known as broadcast domains.

The particular advantage of VLANs is the reduced network load for the nodes and network segments of other VLANs.

To identify which packet belongs to which VLAN, the frame is expanded by 4 bytes (VLAN tagging (Page 927)). This expansion includes not only the VLAN ID but also priority information.

Options for the VLAN assignment

There are various options for the assignment to VLANs:

- Port-based VLAN
  Each port of a device is assigned a VLAN ID. You configure port-based VLAN in "Layer 2 > VLAN > Port-based VLAN".
- Protocol-based VLAN
  Each port of a device is assigned a protocol group. You can configure protocol-based VLAN in "Layer 2 > VLAN > Protocol Based VLAN Port"
- Subnet-based VLAN
  The IP address of the device is assigned a VLAN ID. You configure subnet-based VLAN in "Layer 2 > VLAN > IPv4 Subnet Based VLAN".

processing the VLAN assignment

If more than one VLAN assignment is created on the device, the assignments are processed in the following order:

1. Subnet-based VLAN
2. Protocol-based VLAN
3. Port-based VLAN

The frame is first examined for the IP address. If a rule on the "IPv4 Subnet Based VLAN" tab applies, the frame is sent to the corresponding VLAN. If no rule applies, the protocol type of the frame is examined. If a rule on the "Protocol Based VLAN Port" tab applies, the frame is sent to the corresponding VLAN. If no rule applies, the frame is sent via the port-based VLAN. The rules for the port-based VLAN are specified on the "Port Based VLAN" tab.

See also

General (Page 1012)

GVRP (Page 1014)

Port-based VLAN (Page 1015)

Protocol-based VLAN group (Page 1016)

Protocol-based VLAN port (Page 1017)

IPv4 subnet-based VLAN (Page 1017)

VLAN tagging

Expansion of the Ethernet frames by four bytes

For CoS (Class of Service, frame priority) and VLAN (virtual network), the IEEE 802.1 Q standard defined the expansion of Ethernet frames by adding the VLAN tag.

Note

The VLAN tag increases the permitted total length of the frame from 1518 to 1522 bytes. With the IE switches, the standard MTU size is 1536 bytes. The MTU size can be changed to values from 64 to 9216 bytes.
The end nodes on the networks must be checked to find out whether they can process this length / this frame type. If this is not the case, only frames of the standard length may be sent to these nodes.

The additional 4 bytes are located in the header of the Ethernet frame between the source address and the Ethernet type / length field:

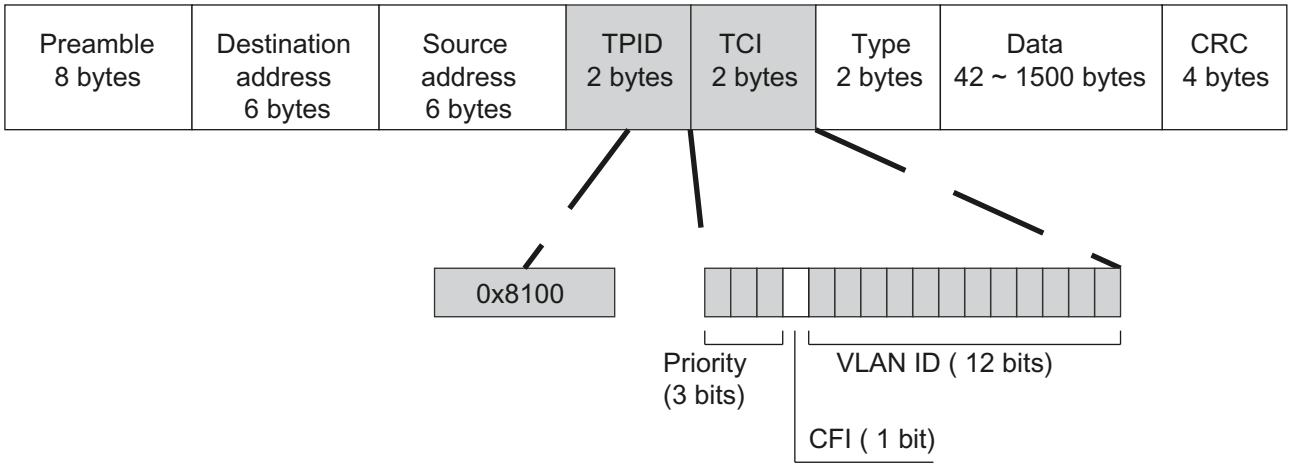


Figure 10-2 Structure of the expanded Ethernet frame

The additional bytes contain the tag protocol identifier (TPID) and the tag control information (TCI).

Tag protocol identifier (TPID)

The first 2 bytes form the Tag Protocol Identifier (TPID) and always have the value 0x8100. This value specifies that the data packet contains VLAN information or priority information.

Tag Control Information (TCI)

The 2 bytes of the Tag Control Information (TCI) contain the following information:

CoS prioritization

The tagged frame has 3 bits for the priority that is also known as Class of Service (CoS). The priority according to IEEE 802.1p is as follows:

CoS bits	Type of data
000	Non time-critical data traffic (less then best effort [basic setting])
001	Normal data traffic (best effort [background])
010	Reserved (standard)
011	Reserved (excellent effort)
100	Data transfer with max. 100 ms delay
101	Guaranteed service, interactive multimedia
110	Guaranteed service, interactive voice transmission
111	Reserved

The prioritization of the data packets is possible only if there is a queue in the components in which they can buffer data packets with lower priority.

The device has eight parallel queues in which the frames with different priorities can be processed. First, the frames with the highest priority ("Strict Priority" method) are processed. This method ensures that the frames with the highest priority are sent even if there is heavy data traffic.

Canonical Format Identifier (CFI)

The CFI is required for compatibility between Ethernet and the token Ring.
The values have the following meaning:

Value	Meaning
0	The format of the MAC address is canonical. In the canonical representation of the MAC address, the least significant bit is transferred first. Standard-setting for Ethernet switches.
1	The format of the MAC address is not canonical.

VLAN ID

In the 12-bit data field, up to 4096 VLAN IDs can be formed. The following conventions apply:

VLAN ID	Meaning
0	The frame contains only priority information (priority tagged frames) and no valid VLAN identifier.
1 - 4094	Valid VLAN identifier, the frame is assigned to a VLAN and can also include priority information.
4095	Reserved

SNMP

Introduction

With the aid of the Simple Network Management Protocol (SNMP), you monitor and control network elements from a central station, for example routers or switches. SNMP controls the communication between the monitored devices and the monitoring station.

Tasks of SNMP:

- Monitoring of network components
- Remote control and remote parameter assignment of network components
- Error detection and error notification

In versions v1 and v2c, SNMP has no security mechanisms. Each user in the network can access data and also change parameter assignments using suitable software.

For the simple control of access rights without security aspects, community strings are used.

The community string is transferred along with the query. If the community string is correct, the SNMP agent responds and sends the requested data. If the community string is not correct, the SNMP agent discards the query. Define different community strings for read and write permissions. The community strings are transferred in plain text.

Standard values of the community strings:

- public
  has only read permissions
- private
  has read and write permissions

Note

Because the SNMP community strings are used for access protection, do not use the standard values "public" or "private". Change these values following the initial commissioning.

Further simple protection mechanisms at the device level:

- Allowed Host
  The IP addresses of the monitoring systems are known to the monitored system.
- Read Only
  If you assign "Read Only" to a monitored device, monitoring stations can only read out data but cannot modify it.

SNMP data packets are not encrypted and can easily be read by others.

The central station is also known as the management station. An SNMP agent is installed on the devices to be monitored with which the management station exchanges data.

The management station sends data packets of the following type:

- GET
  Request for a data record from the agent
- GETNEXT
  Calls up the next data record.
- GETBULK (available as of SNMPv2)
  Requests multiple data records at one time, for example several rows of a table.
- SET
  Contains parameter assignment data for the relevant device.

The SNMP agent sends data packets of the following type:

- RESPONSE
  The agent returns the data requested by the manager.
- TRAP
  If a certain event occurs, the SNMP agent itself sends traps.

SNMPv1 and SNMPv2 and SNMPv3 use UDP (User Datagram Protocol). The data is described in a Management Information Base (MIB).

SNMPv3

Compared with the previous versions SNMPv1 and SNMPv2. SNMPv3 introduces an extensive security concept.

SNMPv3 supports:

- Fully encrypted user authentication
- Encryption of the entire data traffic
- Access control of the MIB objects at the user/group level

Spanning Tree

Avoiding loops on redundant connections

The spanning tree algorithm allows network structures to be created in which there are several connections between two stations. Spanning tree prevents loops being formed in the network by allowing only one path and disabling the other (redundant) ports for data traffic. If there is an interruption, the data can be sent over an alternative path. The functionality of the spanning tree algorithm is based on the exchange of configuration and topology change frames.

Definition of the network topology using the configuration frames

The devices exchange configuration frames known as BPDUs (Bridge Protocol Data Unit) with each other to calculate the topology. The root bridge is selected and the network topology created using these frames. BPDUs also bring about the status change of the root ports.

The root bridge is the bridge that controls the spanning tree algorithm for all involved components.

Once the root bridge has been specified, each device sets a root port. The root port is the port with the lowest path costs to the root bridge.

Response to changes in the network topology

If nodes are added to a network or drop out of the network, this can affect the optimum path selection for data packets. To be able to respond to such changes, the root bridge sends configuration messages at regular intervals. The interval between two configuration messages can be set with the "Hello Time" parameter.

Keeping configuration information up to date

With the "Max Age" parameter, you set the maximum age of configuration information. If a bridge has information that is older than the time set in Max Age, it discards the message and initiates recalculation of the paths.

New configuration data is not used immediately by a bridge but only after the period specified in the "Forward Delay" parameter. This ensures that operation is only started with the new topology after all the bridges have the required information.

RSTP, MSTP, CIST

Rapid Spanning Tree Protocol (RSTP)

One disadvantage of STP is that if there is a disruption or a device fails, the network needs to reconfigure itself: The devices start to negotiate new paths only when the interruption occurs. This can take up to 30 seconds. Fur this reason, STP was expanded to create the "Rapid Spanning Tree Protocol" (RSTP, IEEE 802.1w). This differs from STP essentially in that the devices are already collecting information about alternative routes during normal operation and do not need to gather this information after a disruption has occurred. This means that the reconfiguration time for an RSTP controlled network can be reduced to a few seconds.
This is achieved by using the following functions:

- Edge ports (end node port)
  Edge ports are ports connected to an end device.
  A port that is defined as an edge port is activated immediately after connection establishment. If a spanning tree BPDU is received at an edge port, the port loses its role as edge port and it takes part in (R)STP again. If no further BPDU is received after a certain time has elapsed (3 x hello time), the port returns to the edge port status.
- Point-to-point (direct communication between two neighboring devices)

  By directly linking the devices, a status change (reconfiguration of the ports) can be made without any delays.
- Alternate port (substitute for the root port)

  A substitute for the root port is configured. If the connection to the root bridge is lost, the device can establish a connection over the alternate port without any delay due to reconfiguration.
- Reaction to events

  Rapid spanning tree reacts to events, for example an aborted connection, without delay. There is no waiting for timers as in spanning tree.
- Counter for the maximum bridge hops
  The number of bridge hops a package is allowed to make before it automatically becomes invalid.
  In principle, therefore with rapid spanning tree, alternatives for many parameters are preconfigured and certain properties of the network structure taken into account to reduce the reconfiguration time.

Multiple Spanning Tree Protocol (MSTP)

The Multiple Spanning Tree Protocol (MSTP) is a further development of the Rapid Spanning Tree Protocol. Among other things, it provides the option of operating several RSTP instances within different VLANs or VLAN groups and, for example, making paths available within the individual VLANs that the single Rapid Spanning Tree Protocol would globally block.

Note

Default setting

HTTP is enabled as default on the device.

Common and internal Spanning Tree (CIST)

CIST identifies the internal instance used by the switch that is comparable in principle with an internal RSTP instance.

Routing function

Introduction

The term routing describes the specification of routes for communication between different networks; in other words, how does a data packet from subnet A get to subnet B.

SCALANCE X supports the following routing functions:

- Static routing
  With static routing, the routes are entered manually in the routing table.
- Router redundancy
  With standardized VRRP (Virtual Router Redundancy Protocol), the availability of important gateways is increased by redundant routers.
- Dynamic routing
  The entries in the routing table are dynamic and are updated continuously. The entries are created with one of the following dynamic routing protocols:
  - OSPFv2
  - RIPv2

Static routing

The route is entered manually in the routing table. Enter the route in the routing table on the "Layer 3 > Routes" page.

OSPFv2

Dynamic routing with OSPFv2

OSPF (Open Shortest Path First) is a cost-based routing protocol. To calculate the shortest and most cost-effective route, the Short Path First algorithm by Dijkstra is used. OSPF was developed by the IETF (Internet Engineering Task Force).
You configure OSPFv2 in "Layer 3 > OSPFv2".

OSPFv2 divides an autonomous system (AS) into different areas.

Areas in OSPF

The following areas exist:

- Backbone
  The backbone area is area 0.0.0.0. All other areas are connected to this area. The backbone area is connected either directly or via virtual connections with other areas.
  All routing information is available in the backbone area. As a result, the backbone area is responsible for forwarding information between different areas.
- Stub Area
  This area contains the routes within its area within the autonomous system and the standard route out of the autonomous system. The destinations outside this autonomous system are assigned to the standard route.
- Totally Stubby Area
  This area knows only the routes within its area and the standard route out of the area.
- Not So Stubby Area (NSSA)
  This area can forward (redistribute) packets from other autonomous systems into the areas of its own autonomous system. The packets are further distributed by the NSSA router.

Routers of OSPF

OSPF distinguishes the following router types:

- Internal router (IR)
  All OSPF interfaces of the router are assigned to the same area.
- Area Border Router (ABR)
  The OSPF interfaces of the router are assigned to different areas. One OSPF interface is assigned to the backbone area. Where possible, routes are grouped together.
- Backbone Router (BR)
  At least one of the OSPF interfaces is assigned to the backbone area.
- Autonomous System Area Border Router (ASBR)
  One interface of the router is connected to a different AS, for example an AS that uses the routing protocol RIP.

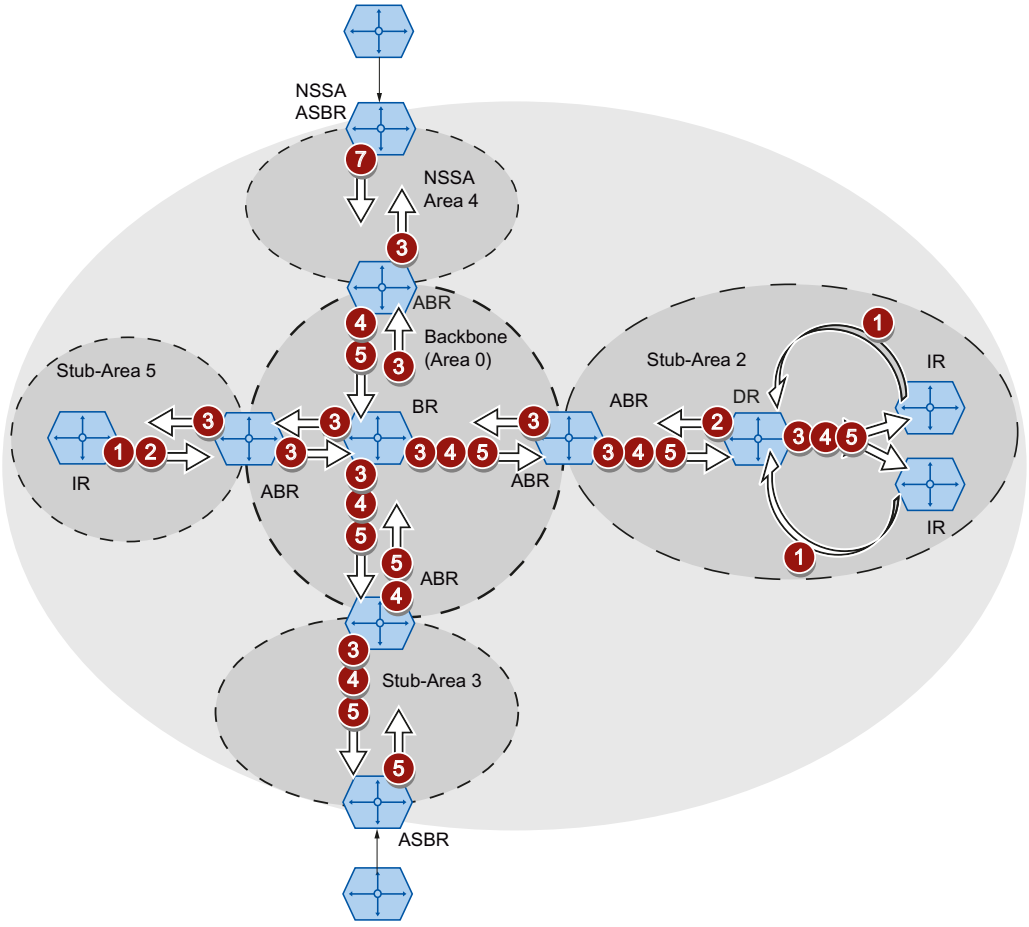
Virtual connection

Each area must be connected to the backbone area. In some situations a direct physical connection is not possible. In this case, a router of the relevant area must be connected to a backbone router via a virtual connection.

LSA types

Within the autonomous system, packets are exchanged that contain information about the connections of a router and the connection status message. The packets are also known as LSAs (Link State Advertisements). The LSAs are always sent from the router to the neighbor router.

If there are changes in the network, LSAs are sent to all routers in the network. The information depends on the LSA type.



1. **Router LSA (LSA Type 1)**
   The LSA Type 1 is only sent within an area. For each active connection of the router that belongs to the area in consideration, an LSA Type 1 is generated. The LSA Type 1 contains information about the status and the costs of the connection, for example IP address, network mask, network type
2. **Network LSA (LSA Type 2)**
   The LSA Type 2 is sent only within an area. For each network that belongs to the relevant area, the router generates an LSA Type 2. If several routers are interconnected in a network, the LSA Type 2 is sent by the designated router (DR). The LSA Type 2 includes the network address, the network mask and a list of routers that are connected to the network
3. **Summary LSA (LSA Type 3 / LSA Type 4)**
   The Summary LSA is generated by the area border router and sent into the area. The
4. Summary LSA contains information about routes outside the area but inside the AS. Where possible, the routes are grouped together.
   - Summary LSA (LSA Type 3)
     The LSA Type 3 describes the routes to the networks and advertises the standard route to the areas.
   - AS Summary LSA (LSA Type 4)
     The LSA Type 4 describes the routes to the ASBR.
5. **External LSA (LSA Type 5 / LSA Type 7)**
   The External LSA is generated by the ASBR. The LSA type depends on the area.
7. - AS External LSA (LSA Type 5)
     The LSA Type 5 is sent by the AS border router into the areas of the autonomous system except the Stub and NSSA areas. The LSA contains information about routes to a network in another AS. The routes are either created manually or learned externally. The ASBR uses LSA Type 5 to distribute standard routes to the backbone area.
   - NSSA External LSA (LSA Type 7)
     The LSA Type 7 is generated by the AS border router of an NSSA. The router is also known as the NSSA ASBR. The LSA Type 7 is sent only within the NSSA. If the P bit in LSA Type 7 = 1, these LSAs are converted to LSA Type 5 by the ABR and sent to the backbone area.

Establishing the neighborhood

The router runs through the following statuses to establish a connection to the neighbor router.

1. Attempt state / Init state
   The router activates OSPF and begins to send and receive Hello packets. Based on the received Hello packets, the router learns which OSPF routers are in its vicinity. The router checks the content of the Hello packet. The Hello packet also contains the list of the neighbor routers (neighbor table) of the "sender".
2. Two way state
   If, for example, the ID of the area, the area type and the settings for the times match, a connection (adjacency) can be established to the neighbor. In a point-to-point network, the connection is established directly. If several neighbor routers can be reached in a network, the designated router (DR) and the designated backup router (DBR) are identified based on Hello packets. The router with the highest router priority becomes the designated router. If two routers have the same router priority, the router with the lower router ID becomes the designated router. The router establishes a connection to the designated router.
3. Exchangestart state
   The neighbor routers decide which router starts communication. The router with the higher router ID becomes master.
4. Exchange state
   The neighbor routers send packets that describe the content of their neighborhood database. The neighborhood database (link state database - LSDB) contains information on the topology of the network.
5. Loading state
   The router completes the received information. If the router still has questions relating to the status of a specific connection, it sends a link state request. The neighbor router sends a response (link state update). The response contains a suitable LSA. The router confirms receipt of the response (link state acknowledge).
6. Full State
   The information exchange with the neighbor router is completed. The neighborhood database of the neighbor router is the same. Based on the Short Path First algorithm, the router calculates a route to every destination. The route is entered in the routing table.

Check the neighborhood

The Hello packets are only used to establish the neighborhood relations. Hello packets are used to check the connection to the neighbor router by sending them cyclically. If no Hello packet is received within a certain interval (dead interval), the connection to the neighbor is marked as "down". The relevant entries are deleted.

Updating the neighborhood database

Once the neighborhood database is established, LSAs are sent to all routers in the network if there are changes in the topology.

RIPv2

Dynamic routing with RIPv2

The Routing Information Protocol (RIPv2) is used to create routing tables automatically. RIPv2 is used in autonomous systems (AS) with a maximum of 15 routers. It is based on the Distance-Vector algorithm.

RIPv2 was developed by the IETF (Internet Engineering Task Force) and is described in RFC 2453.
You configure RIPv2 in "Layer 3 > RIPv2".

Setting up a routing table

Since a router initially only knows its directly connected networks, it sends a request to its direct neighbor routers. As the reply, it receives the routing tables of the neighbor routers. Based on the information it receives, the router set up its own routing table.

The routing table contains entries for all possible destinations. Each entry includes the distance to the destination and the first router on the route.

The distance is also known as the metric. This indicates the number of routers to be passed through on the route to the destination (hop count). The maximum distance is 15 routers (hops).

Updating the routing table

Once the routing table is set up, the router sends its routing table to each direct neighbor router at intervals of 30 seconds.

The router compares new routing information with its existing routing table. If the new information includes shorter routes, the existing routes are overwritten. The router only keeps the shortest route to a destination.

Checking neighbor routers

If a router does not receive messages from a neighbor router for longer than 180 seconds, it marks the router as being invalid. The router assigns the metric 16 for the neighbor router.

Link aggregation

Link aggregation

With link aggregation, several parallel physical connections with the same transmission speed are grouped together to form a logical connection with a higher transmission speed. This method based on IEEE 802.3ad is also known as port trunking or channel bundling.

Link aggregation works only with full duplex connections with the same transmission speed in point-to-point mode. This achieves multiplication of the bandwidth or transmission speed. If part of the connection fails, the data traffic is handled via the remaining parts of the connection.

To control and monitor, the Link Aggregation Control Layer (LACL) and the Link Aggregation Control Protocol (LACP) are used.

Authentication

Authentication method

You can configure the authentication methods "802,1x" and "MAC Authentication" as well as the "Guest VLAN" option separately for each port.

The functions have a hierarchical order. If all three functions are enabled, an attempt is made initially to authenticate the end device using "802.1x". If this authentication is unsuccessful, "MAC Authentication" is started. If this authentication is also unsuccessful, the end device is enabled for communication in the "Guest VLAN". ""Guest VLAN" can only be used if at least one authentication method is active.

The two authentication methods depend on the end device. If the end device supports EAP (Extensible Authentication Protocol), it can be authenticated using the "802.1x" method. If the end device does not support EAP, it can be authenticated using "MAC Authentication". In this case, the IE switch adopts the role of the end device and uses the MAC address of the device as the authentication parameter.

802.1x

The "802.1x" authentication method works as follows:
An end device that supports EAP sends authentication information to the IE switch. The IE switch forwards the information to the authentication server. The authentication server checks the information and allows or denies the end device access to the network.

MAC Authentication

The "MAC Authentication" authentication method works as follows:
As soon as the IE switch receives a frame from the end device, the IE switch sends a query to the RADIUS server to allow or deny the end device access to the network.

Editing properties and parameters

Editing options

You have the following options for editing properties and parameters:

- Hardware and network editor
  Once you have inserted the network component, you can edit the properties and parameters, for example the device name. You will find more detailed information in "Hardware and network editor".
- Web Based Management (WBM)
  The parameters and properties are accessible using the supplied HTML pages (WBM pages). Each WBM page has its own help page that describes the properties and parameters. For further information, refer to the configuration manual "SCALANCE XM400/X-500 Web Based Management". You will find the MIB at Siemens Industry Automation and Drives Service & Support on the Internet under the entry ID 67428305 (http://support.automation.siemens.com/WW/view/en/67428305).
- Command Line Interface
  All the configuration settings for the device can be made using the CLI. The CLI provides the same options as Web Based Management (WBM). For further information, refer to the configuration manual "SCALANCE XM400/X-500 Command Line Interface". You will find the MIB at Siemens Industry Automation and Drives Service & Support on the Internet under the entry ID 67430663 (http://support.automation.siemens.com/WW/view/en/67430663).

Availability

The availability of the settings depends

- on the port type
- on the configuration mode

SCALANCE X distinguishes the following port types:

- Switch port
- Router port

The following configuration modes are available:

- Offline configuration
  You can configure a PC station the first time offline. In this mode only the settings that require no connection to the device are available.
- Online configuration mode
  If there is a connection to the device, there are additional pages available in the Inspector window. These pages include the note "This page is only available if there is an online connection to the device". With some settings, the additional information "only available online" is included.

Creating and deleting an entry

As an example, an entry will be created and deleted on the Syslog client. The procedure is valid on all pages.

Creating a new entry

1. Select the device in the network or device view.
2. Open the properties of the device in the Inspector window.
3. In the Inspector window go to System > Syslog client.
4. Enter the IP address for the Syslog server.
5. Click in the table.
6. Select the entry "New entry" in the shortcut menu.
   A new entry is created in the table.

Deleting an entry

1. Select the required entry in the table.
2. Select the "Delete" entry in the shortcut menu.

Buttons you require often

- **Refresh the display with "Refresh"**
  Pages that display current parameters have a "Refresh" button at the lower edge of the page. Click this button to request up-to-date information from the device for the current page.
- **Save settings for all ports with "Transfer to table"**
  Pages on which you can configure several ports have 2 tables. In the first table, you can make the settings for all ports and transfer these to the second table. In the last column of the first table, there is the "Transfer to table" button. Click this button to save the settings you have made for all ports.

Assigning the IP address

Configuration options

When shipped and after restoring to the factory settings, the device does not have an IP address.

The following options are available to assign an IP address to the device:

- DHCP (default setting)
- Primary Setup Tool
  For further information, refer to the configuration manual "Primary Setup Tool". You will find the MIB at Siemens Industry Automation and Drives Service & Support on the Internet under the entry ID 19440762 (http://support.automation.siemens.com/WW/view/en/19440762).

- STEP 7
- CLI via the serial interface
  For further information, refer to the configuration manual "SCALANCE XM-400/XR-500 Command Line Interface". You will find the configuration manual on the Internet pages of Siemens Industry Automation and Drives Service & Support (http://support.automation.siemens.com/WW/view/en/48805144/133300).

Requirement

- "Set IP address on device" is enabled in the properties of the device. You will find more detailed information in "Addressing PROFINET devices".

Configuring SCALANCE XR-500 as IO device

With a SCALANCE XR500 configured as a PROFINET IO device and assigned to an IO controller, the functions "Compile" and "Download to device" only download the data to the switch that can also be configured in Web Based Management (WBM) (layer 2, layer 3, system, security).

If you want to use the "Compile" or "Download to device" functions for the PROFINET IO device data of the XR500, first select the assigned IO controller.

Show information

Versions

Note

The page is only available if there is an online connection to the device.

This page shows the versions of the hardware and software of the device.

Description

Table 1 has the following columns:

- **Hardware**
  Shows the device being used.
- **Name**
  Shows the name of the device or module.
- **Revision**
  Shows the hardware version of the device.
- **Order ID**
  Shows the order number of the device or module.

Table 2 has the following columns:

- **Software**
  - Firmware
    Shows the current firmware version. If a new firmware file was downloaded and the device has not yet restarted, the firmware version of the downloaded firmware file is displayed here. After the next restart, the downloaded firmware is activated and used.
  - Bootloader
    Shows the version of the boot software stored on the device.
- **Description**
  Shows the short description of the software.
- **Version**
  Shows the version number of the software version.
- **Date**
  Shows the date on which the software version was created.

I&M

Note

The page is only available if there is an online connection to the device.

This page contains information about device-specific vendor and maintenance data such as the order number, serial number, version number etc. You cannot configure anything on this page.

Description of the displayed values

The table has the following rows:

- **Manufacturer ID**
  Shows the manufacturer ID.
- **Order number**
  Shows the order number.
- **Serial number**
  Shows the serial number.
- **Hardware revision**
  Shows the hardware version.
- **Software version**
  Shows the software version.
- **Revision counter**
  Shows the revision counter: Counter for revisions since the initial commissioning
- **Revision Date**
  Date and time of the last revision

- **Function tag**
  Shows the function tag (plant designation) of the device. The plant designation (HID) is created during configuration of the device with HW Config of STEP 7.
- **Location tag**
  Shows the location tag of the device. The location identifier (LID) is created during configuration of the device with HW Config of STEP 7.

ARP table

Note

The page is only available if there is an online connection to the device.

Assignment of MAC address and IP address

With the Address Resolution Protocol (ARP), there is a unique assignment of MAC address to IPv4 addresses. This assignment is kept by each network node in its own separate ARP table. The page shows the ARP table of the device.

Displayed values

The table has the following columns:

- **Interface**
  Shows the interface via which the row entry was learnt.
- **MAC address**
  Shows the MAC address of the target device.
- **IP address**
  Shows the IP address of the target device.
- **Media Type**
  Shows the type of connection.
  - Dynamic
    The device recognized the address data automatically.
  - Static
    The addresses were entered as static addresses

Log table

Logging events

Note

The page is only available if there is an online connection to the device.

The device allows you to log events that occur, some of which you can specify in " System > Events". This, for example, allows you to record when an authentication attempt failed or when the connection status of a port has changed. The content of the events log table is retained even when the device is turned off.

Settings

- **Severity Filters**
  You can filter the entries in the table according to severity. To display all the entries, enable or disable all settings.
  - Info
    Information (6)
  - Warning
    Warnings (4)
  - Critical
    Critical (2)

Displayed values

The table has the following columns:

- **Restart**
  Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.
- **System up time**
  Shows the time the device has been running since the last restart when the described event occurred.
  If the system time is set, the time is also displayed at which the event occurred.
- **System Time**
  Shows the date and time of the device.
- **Severity**
  Shows the severity.
- **Log Message**
  Displays a brief description of the event that has occurred.

Buttons

- **Delete**
  Click this button to delete the content of the event log file. The display is also cleared. The restart counter is only reset after you have restored the device to the factory settings and restarted the device.

  Note

  The number of entries in this table is restricted to 400. When this number is reached, the oldest entries are overwritten. The table remains permanently in memory.

Fault

Error status

Note

The page is only available if there is an online connection to the device.

This page displays any errors that occur. If there are no more unanswered error/fault messages, the fault LED goes off.

The time calculation always begins after the last system start. When the system is restarted, a new entry with the type of restart is created in the fault memory.

Description of the displayed values

- **No. of Signaled Faults**
  Shows the number of reported faults.

The table contains the following columns:

- **Fault Time**
  Shows the time the device has been running since the last restart when the described error/ fault occurred.
- **Fault description**
  Display of the fault state for the device.
- **Clear Fault State**
  When you can delete faults, the "Clear Fault State" button is enabled.

Redundancy

Spanning Tree

Note

The page is only available if there is an online connection to the device.

The page shows the current information about the Spanning Tree and the settings of the root bridge.

- If Spanning Tree is turned off, only the basic information about this device is displayed.
- If Spanning Tree is activated, the following is displayed:
  - Parameters of the selected instance
  - Parameters for the ports of the selected instance

  The information depends on the selected Spanning Tree mode.

Displayed values

- **Spanning Tree Mode**
  Shows the set mode. You specify the mode in "Layer 2 > MSTP > General".
  The following values are possible:
  - '-'
  - STP
  - RSTP
  - MSTP
- **Instance ID**
  Shows the number of the instance. The parameter depends on the configured mode.
- **Bridge Priority / Root Priority**
  Which device becomes the root bridge is decided by the bridge priority. The bridge with the highest priority (in other words, with the lowest value for this parameter) becomes the root bridge. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames. The value for the bridge priority is a whole multiple of 4096.
- **Bridge address / root address**
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Cost**
  Shows the path costs from the device to the root bridge.
  In MSTP mode, the path costs as far as the root bridge of the CIST are displayed.
- **Bridge Status**
  Shows the status of the bridge, e.g. whether or not the device is the root bridge.
- **Regional root priority** (available only with MSTP)
  For a description, see Bridge priority / Root priority.
- **Regional root address** (available only with MSTP)
  Shows the MAC address of the device.
- **Regional root costs** (only available with MSTP)
  The path costs from this device to the regional root bridge.

The table has the following columns:

- **Port**
  Shows the port via which the device communicates.
- **Role**
  Shows the status of the port. The following values are possible:
  - Disabled
    The port was removed manually from the spanning tree and will no longer be taken into account by the spanning tree.
  - Designated
    The ports leading away from the root bridge.
  - Alternate
    The port with an alternative route to a network segment
  - Backup
    If a switch has several ports to the same network segment, the "poorer" port becomes the backup port.
  - Root
    The port that provides the best route to the root bridge.
  - Master
    This port points to a root bridge located outside the MST region.
- **Status**
  Displays the current status of the port. The values are only displayed. The parameter depends on the configured protocol. The following statuses are possible:
  - Discarding
    The port receives BPDU frames. Other incoming or outgoing frames are discarded.
  - Listening
    The port receives and sends BPDU frames. The port is involved in the spanning tree algorithm. Other outgoing and incoming frames are discarded.
  - Learning
    The port actively learns the topology; in other words, the node addresses. Other outgoing and incoming frames are discarded.
  - Forwarding
    Following the reconfiguration time, the port is active in the network. The port receives and sends data frames.
- **Oper. Version**
  Describes the type of spanning tree in which the port operates
- **Priority**
  If the path calculated by the spanning tree is possible over several ports of a device, the port with the highest priority (in other words the lowest value for this parameter) is selected. A value between 0 and 240 can be entered for the priority in steps of 16. If you enter a value that cannot be divided by 16, the value is automatically adapted.

- **Path costs**
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected. If several ports of a device have the same value, the port with the lowest port number will be selected.
  If the value "Cost Calc." is "0", the automatically calculated value is displayed. Otherwise, the value of "Cost Calc." is displayed.
  The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000.
- **Edge type**
  Shows the type of the connection. The following values are possible:
  - Edge Port

    An edge port is connected to this port.
  - No Edge Port
    There is a spanning tree or rapid spanning tree device at this port.
- **P.t.P. Type**
  Shows the type of point-to-point link. The following values are possible:
  - P.t.P.
    With half duplex, a point-to-point link is assumed.
  - Shared Media

    With a full duplex connection, a point-to-point link is not assumed.

VRRP statistics

Introduction

Note

The page is only available if there is an online connection to the device.

This page shows the statistics of the VRRP protocol and all configured virtual routers.

Description of the displayed values

The following fields are displayed:

- VRID errors
  Shows how many VRRP frames containing an unsupported VRID were received.
- Version errors
  Shows how many VRRP frames containing an invalid version number were received.
- Checksum errors
  Shows how many VRRP frames containing an invalid checksum were received.

The table has the following columns:

- Interfaces
  Interface to which the settings relate.
- VRID
  Shows the ID of the virtual router.
  Valid values are 1 to 255.
- Change to master status
  Shows how often this virtual router changed to the "Master" role.
- Advertisements recd.
  Shows how often a VRRP frame was received that contained a bad address list.
- Advertisement Interval Errors
  Shows how many bad VRRP packets were received whose interval does not match the value set locally.
- IP TTL errors
  Shows how many bad VRRP packets were received whose TTL (Time to live) value in the IP header is incorrect.
- Prio 0 received
  Shows how many VRRP packets with priority 0 were received. VRRP packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- Prio 0 sent
  Shows how many VRRP packets with priority 0 were sent. Packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- Invalid Type
  Shows how often a VRRP packet was received that had the wrong type.
- Address List Errors
  Shows the number of errors in the address list.
- Invalid Auth. Type
  Shows how many bad VRRP packets were received whose authentication type was not type 0. Type 0 means "no authentication".

- **Auth. Type Mismatch**
  Shows how many bad VRRP frames were received whose authentication type does not match.
- **Frame length error**
  Shows how many bad VRRP frames were received whose length is not correct.

Ring Redundancy

Note

The page is only available if there is an online connection to the device.

Automatic Redundancy Detection (ARD) is the default when the device ships. If you want to use the previous High Speed Redundancy Protocol (HRP), HRP must be configured.

- Reconfiguration time of the frame traffic following a failover in MRP: 200 ms
- Reconfiguration time of the frame traffic following a failover in HRP: 300 ms

This page informs you about the values currently set for the device.

Description

The following fields are displayed:

- **Redundancy Function**
  The following values are shown:
  - Disabled
    Ring redundancy is disabled on the device.
  - MRP Manager
    The device is operating as MRP manager.
  - MRP Client
    The device is operating as MRP client.
  - HRP Client
    The device is operating as HRP client.
  - HRP Manager
    The device is operating as HRP manager.
- **RM Status**
  The "RM Status" column shows whether or not the IE switch is operating as redundancy manager and whether it has opened or closed the ring in this role.
  - Passive
    The IE switch is operating as redundancy manager and has opened the ring; in other words, the line of switches connected to the ring ports is operating problem free. The passive status is also displayed if the IE switch is not operating as the redundancy manager (RM Function Disabled).
  - Active
    The IE switch is operating as redundancy manager and has closed the ring; in other words, the line of switches connected to the ring ports is interrupted (problem). The redundancy manager connects its ring ports through and restores an uninterrupted linear topology.
  - If media redundancy in ring topologies is completely disabled, ring ports configured last are displayed and the text "Ring Redundancy disabled" is displayed.
- **Observer Status**
  Shows the current status of the observer.
- **Ring ports**
  Ring ports 1 and ring port 2 show the ports used for redundancy.
- **No. of Changes to RM Active State**
  Shows how often the device as redundancy manager switched to the active status, i.e. closed the ring.
  If the redundancy function is disabled or the device is not the HRP/MRP manager, the text "Redundancy manager disabled" appears.
- **Max. delay of the RM test frames [ms]**
  Shows the maximum delay for test frames of the redundancy manager.
  If the redundancy function is disabled or the device is not the HRP/MRP manager, the text "Redundancy manager disabled" appears.

Standby

Note

The page is only available if there is an online connection to the device.

This page shows the standby status of the device.

For more information on Ethernet cabling and the topological location of master and slave, refer to the "Industrial Ethernet Network Manual".

Description

The following fields are displayed:

- **Standby ports**
  Shows the ports being used.
- **Standby name**
  Shows the name for the standby connection. The master/slave device pair is defined by this name (both must be in the same ring). This is achieved by entering the same name on two devices in the ring. You can select any name to suit your purposes, however, you can only use the name for one pair of devices in the entire network.
- **Standby Function**
  Shows whether the standby connection is activated or deactivated.
  - Master
    The device has a connection to the partner device and is operating as master. In normal operation, the standby port of this device is active.
  - Slave
    The device has a connection to the partner device and is operating as slave. In normal operation, the standby port of this device is inactive.
  - Disabled
    The standby link is disabled. The device is operating neither as master nor slave. The port configured as a standby port works as a normal port without standby function.
  - Waiting for Connection...
    No connection has yet been established to the partner device. The standby port is inactive. In this case, either the configuration on the partner device is inconsistent (for example incorrect connection name, standby link disabled) or there is a physical fault (for example device failure, link down).
  - Connection Lost
    The existing connection to the partner device has been lost. In this case, either the configuration on the partner device was modified (for example a different connection name, standby link disabled) or there is a physical fault (for example device failure, link down).

- **Standby status**
  Shows the status of the standby port:
  - Active
    The standby port of this device is active; in other words is enabled for frame traffic.
  - Passive
    The standby port of this device is inactive; in other words is blocked for frame traffic.
  - "-"
    The standby function is disabled.
- **No. of Changes to Standby Active State**
  Shows how often the IE switch has changed the standby status from "Passive" to "Active". If the connection of a standby port fails on the standby master, the IE switch changes to the "Active" status.
  Following each restart on the device, the counters are automatically reset.

Ethernet Statistics

Interface statistics

Interface statistics

The page shows the statistics from the interface table of the Management Information Base (MIB).

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- **Received bytes**
  Shows the number of received bytes.
- **Sent bytes**
  Shows the number of sent bytes.
- **Received unicast frames**
  Shows the number of received unicast frames.
- **Received non-unicast frames**
  Shows the number of received frames that are not of the type unicast.
- **Sent unicast frames**
  Shows the number of sent unicast frames.
- **Sent non-unicast frames**
  Shows the number of sent frames that are not of the type unicast.

Frame length

Frames sorted by length

This page displays how many frames of which size were sent and received at each port.

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations.

  Note

  Display of frame statistics

  In the statistics relating to frame lengths, note that both incoming and outgoing frames are counted.

- **Frame lengths**
  The other columns after the port number contain the absolute numbers of frames according to their frame length.
  The following frame lengths are distinguished:
  - 64 bytes
  - 65 - 127 bytes
  - 128 - 255 bytes
  - 256 - 511 bytes
  - 512 - 1023 bytes
  - 1024 - max.

Button

Reset Counter

Click "Reset Counter" to reset all counters. The counters are also reset by a restart.

Frame type

Received frames sorted by type

This page displays how many frames of the type "unicast", "multicast", and "broadcast" were received at each port.

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations.
- **Unicast / Multicast / Broadcast**
  The other columns after the port number contain the absolute numbers of the incoming frames according to their frame type "unicast", "multicast" and "broadcast".

Button

Reset Counter

Click "Reset Counter" to reset all counters. The counters are also reset by a restart.

Packet Errors

Received bad frames

This page shows how many bad frames were received per port.

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations.
- **Error types**
  The other columns after the port number contain the absolute numbers of the incoming frames according to their error type.
  In the columns of the table, a distinction is made according to the following error types:
  - CRC
    Frames whose content does not match the CRC checksum.
  - Undersize
    Frames with a length less than 64 bytes.
  - Oversize
    Frames discarded because they were too long.
  - Fragments
    Frames with a length less than 64 bytes and a bad CRC checksum.
  - Jabbers
    VLAN-tagged frames with an incorrect CRC checksum that were discarded because they were too long.
  - Collisions
    Collisions that were detected.

Button

Reset Counter

Click "Reset Counter" to reset all counters. The counters are also reset by a restart.

History

Samples of the statistics

The page shows samples from each port with information from the statistics.

Note

The page is only available if there is an online connection to the device.

Settings

- **Port**
  Select the port for which the history will be displayed.

Displayed values

- **Entries**
  Maximum number of samples that can be saved at the same time.
- **Interval [s]**
  Interval after which the current status of the statistics will be saved as a sample.

The table has the following columns:

- **Sample**
  Number of the sample
- **Time for the Sample**
  System up time at which the sample was taken.
- **Unicast**
  Number of received unicast frames.
- **Multicast**
  Number of received multicast frames.
- **Broadcast**
  Number of received broadcast frames.
- **CRC**
  Number of frames with a bad CRC checksum.
- **Undersize**
  Number of frames that are shorter than 64 bytes.
- **Oversize**
  Number of frames discarded because they were too long.
- **Fragments**
  Number of frames that are shorter than 64 bytes and have a bad CRC checksum.
- **Jabbers**
  Number of frames with a VLAN tag that have a bad CRC checksum and will be discarded because they are too long.
- **Collisions**
  Number of collisions of received frames.
- **Utilization**
  Utilization of the port during a sample.

Unicast

This page shows the learnt and static unicast MAC addresses.

Description

- **VLAN ID**
  Shows the VLAN-ID assigned to this MAC address.
- **MAC Address**
  Shows the MAC address of the node that the device has learned or the user has configured.

- **Status**
  Shows the status of each address entry:
  - Learnt
    The specified address was learned by receiving a frame from this node and will be deleted when the aging time expires if no further packets are received from this node.
  - Static
    Configured by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the switch is restarted.
- **Port**
  Shows the port via which the node with the specified address can be reached. Frames received by the device whose destination address matches this address will be forwarded to this port.

Multicast

This page shows the learnt and static multicast MAC addresses.

Description

- **VLAN ID**
  Shows VLAN ID of the VLAN to which the MAC multicast address is assigned.
- **MAC Address**
  Shows the MAC multicast address that the device has learned or the user has configured.
- **Status**
  Shows the status of each address entry. The following information is possible:
  - static
    The address was entered statically by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the device is restarted. These must be deleted by the user.
  - IGMP
    The destination port for this address was obtained by IGMP configuration.
  - GMRP
    The destination port for this address was registered by a received GMRP frame.

LLDP

Note

The page is only available if there is an online connection to the device.

Identifying the network topology

Using LLDP, network components exchange information about the topology of the network.

Description of the displayed values

The table contains the following columns:

- **Device ID**
  Device ID of the connected device.
- **Local interface**
  Port at which the IE switch received the information.
- **Hold Time**
  An entry remains stored in the MIB for the time specified here. If the IE switch does not receive any new information from the connected device during this time, the entry is deleted.
- **Capability**
  Shows the properties of the connected device:
  - (R) Router
  - (B) Bridge
  - (T) Telephone
  - (C) DOCSIS cable device
  - (W) WLAN access point
  - (P) Repeater
  - (S) Station
  - (O) Other
- **Port ID**
  Port of the device with which the IE switch is connected.

Routing

Routing table

This page shows the routing table of the device.

Description

The table has the following columns:

- **Destination network**
  Shows the destination address of this route.
- **Subnet mask**
  Shows the subnet mask of this route.
- **Gateway**
  Shows the gateway for this route.
- **Interface**
  Shows the interface for this route.

- **Metric**
  Shows the metric of the route. The higher value, the longer the packets require to their destination.
- **Routing protocol**
  Shows the routing protocol from which the entry in the routing table originates. The following entries are possible:
  - Connected: Connected routes
  - Static: Static routes
  - RIP: Routes using RIP
  - OSPF: Routes using OSPF
  - other: Other routes

OSPFv2 interfaces

Overview

Note

The page is only available if there is an online connection to the device.

This page shows the configuration of the OSPF interface.

Description of the displayed values

The table has the following columns:

- **IP address**
  Shows the IP address of the OSPF interface
- **Area ID**
  Shows the Area ID to which the OSPF interface belongs.

- **Interface Status**
  Shows the status of the WLAN interface:
  - Down
    The interface is not available.
  - Loopback
    Loopback interface
  - Waiting
    Startup and negotiation of the interface.
  - Point to Point
    Point-to-point connection
  - Designated Router
    The router is a designated router and generates a network LSA.
  - Backup D. Router
    The router is backup for the designated router.
  - Other D. Router
    The interface is started up. The router is neither a designated router nor a backup designated router.
- **OSPF status**
  Shows the status of OSPF.
  - Enabled: OSPF is enabled on the interface.
  - Disabled: OSPF is disabled on the interface.
- **Designated Router**
  Shows the IP address of the designated router for this OSPF interface.
- **Backup Designated Router**
  Shows the IP address of the designated backup router for this OSPF interface.
- **Events**
  Shows the number of status changes of OSPF.

OSPFv2 neighbors

Note

The page is only available if there is an online connection to the device.

This page shows the dynamically detected neighbor routers in the relevant networks.

Description of the displayed values

The table has the following columns:

- **IP address**
  Shows the IP address of the neighbor router in this network.
- **Router ID**
  Shows the ID of the neighbor router. The two addresses can match.

- **Status**
  Shows the status of the neighbor router. The status can adopt the following values:
  - unknown
    Status of the neighbor router is unknown.
  - down
    The neighbor router cannot be reached.
  - attempt and init
    Short-lived statuses during initialization
  - two-way
    Two-way receipt of Hello packets. Specification of the designated router and the designated backup router.
  - exchangestart, exchange and loading
    Status during exchange of the LSAs
  - full
    The database is complete and synchronized within the area. The routes can now be detected.

  Note

  Normal status

  If the partner router is a designated router or a designated backup router, the status is "full". Otherwise the status is "two-way".

- **Assoc. Area Type**
  Shows the area type via which the neighbor-neighbor relation is maintained. The following area types exist:
  - Normal
  - Stub
  - NSSA
  - Backbone
- **Priority**
  Shows the priority of the neighbor router. This is only significant when selecting the designated router on a network. For virtual neighbor routers, this information is irrelevant.
- **Hello Suppr.**
  Shows whether there are suppressed Hello packets to the virtual neighbor router.
  - no: There are no suppressed Hello packets (default).
  - yes: There are suppressed Hello packets.
- **Hello Queue**
  Shows the length of the queue with Hello packets still to be transmitted.
- **Events**
  Shows the number of status changes.

OSPFv2 virtual neighbors

Note

The page is only available if there is an online connection to the device.

This page shows the configured virtual neighbor routers.

Overview

The table has the following columns:

- **IP address**
  Shows the IP address of the virtual neighbor router in this network.
- **Router ID**
  Shows the router ID of the virtual neighbor router.
- **Status**
  Shows the status of the neighbor router. The status can adopt the following values:
  - unknown
    Status of the neighbor router is unknown.
  - down
    The neighbor router cannot be reached.
  - attempt and init
    Short-lived statuses during initialization
  - two-way
    Two-way receipt of Hello packets. Specification of the designated router and the designated backup router.
  - exchangestart, exchange and loading
    Status during exchange of the LSAs
  - full
    The database is complete and synchronized within the area. The routes can now be detected.

    Note

    Normal status

    If the partner router is a designated router or a designated backup router, the status is "full". Otherwise the status is "two-way".
- **Trans. Area ID**
  Shows the ID of the area via which the virtual neighborhood relation exists.
- **Hello Suppr.**
  Shows whether there are suppressed Hello packets to the virtual neighbor router.
  - no: There are no suppressed Hello packets (default)
  - yes: There are suppressed Hello packets.

- **Hello Queue**
  Shows the length of the queue with Hello packets still to be transmitted.
- **Events**
  Shows the number of status changes.

OSPFv2 LSDB

Overview

Note

The page is only available if there is an online connection to the device.

The link state database is the central database for managing all links within an area. It consists of the link state advertisements (LSAs). The most important data of these LSAs is shown on the this WBM page.

Description of the displayed boxes

The table has the following columns:

- **Area ID**
  Shows the ID of the area to which the LSA belongs. If the LSA is an external connection, '-' is displayed.
- **LSA type**
  Shows the LSA type. The following values are possible:
  - unknown
    LSA type is unknown.
  - Router
    The router LSA (Type 1) is sent by the OSPF router within an area. The LSA contains information about the status of all router interfaces.
  - Network
    The network LSA (Type 2) is sent by the designated router within an area. The LSA contains a list of routers connected to the network.
  - NSSA External
    The NSSA external LSA (Type 7) is sent by the NSSA-ASBR within an NSSA. The NSSA-ASBR receives LSAs of Type 5 and converts the information to LSAs of Type 7. The NSSA router can forward these LSAs within an NSSA.
  - Summary
    The summary LSA (Type 3) is sent by the ABR within an area. The LSA contains information about routes to other networks.
  - AS Summary
    The AS summary LSA (Type 4) is sent by the area border router within an area. The LSA contains information about routes to other autonomous systems.
  - AS External
    The AS external LSA (Type 5) is sent by the AS border router within an autonomous system. The LSA contains information about routes from one network to another.
- **Link State ID**
  Shows the ID of the LSA.
- **Router ID**
  Shows the ID of the router that sent this LSA.
- **Sequence number**
  Shows the sequence number of the LSA. Each time an LSA is renewed, this sequence number is incremented by one.

RIPv2 statistics

Overview

Note

The page is only available if there is an online connection to the device.

This page shows the statistics of the RIP interface.

Description of the displayed values

The table has the following columns:

- **IP address**
  Shows the IP address of the RIPv2 interface
- **Bad packets**
  Number of received RIP packets that were deleted and therefore ignored.
- **Bad Routes**
  Number of routes of valid RIP packets that could not be taken into consideration.
- **Updates Sent**
  Shows how often the router has sent its routing table to its neighbor routers.

Configuring system functions

Configuration

On this page, you specify the services with which the device can be accessed. With some services, there are further configuration pages on which more detailed settings can be made.

Settings

- **Telnet Server**
  Enable or disable the "Telnet Server" service for unencrypted access to the CLI.
- **SSH server**
  Enable or disable the "SSH Server" service for encrypted access to the CLI.
- **HTTPS server only**
  Enable or disable access using HTTP.
- **DNS client**
  Enable or disable depending on whether the IE switch should operate as a DNS client. You can configure other settings in "System > DNS Client".
- **SMTP Client**
  Enable or disable the SMTP client. You can configure other settings in "System > SMTP Client".
- **Syslog Client**
  Enable or disable the system event client. You can configure other settings in "System > Syslog client".

- **DCP Server**
  Specify whether or not the device can be accessed with DCP (Discovery and Configuration Protocol):
  - "-" (disabled)
    DCP is disabled. Device parameters can neither be read nor modified.
  - Read/write access
    With DCP, device parameters can be both read and modified.
  - Read-Only
    With DCP, device parameters can be read but cannot be modified.
- **Time**
  Select the required setting. The following settings are possible:
  - Manual
    The system time is set manually.
  - SNTP Client
    The system time is set via an SNTP server. You can configure other settings in "System > System time > SNTP client".
  - NTP Client
    The system time is set via an NTP server. You can configure other settings in "System > System time > NTP client".
  - SIMATIC Time
    The system time is set using a SIMATIC time transmitter. You can configure other settings in "System > System Time SIMATIC Time Client".
  - PTP Client
    The system time is set via a PTP server. You can configure other settings in "System > System Time > PTP Client".
    This function is only available for SCALANCE X500.
- **SNMP**
  Select the required protocol. The following settings are possible:
  - "-" (SNMP disabled)
    Access to device parameters using SNMP is not possible.
  - SNMPv1/v2c/v3
    Access to device parameters is possible with SNMP versions 1, 2c or 3. You can configure other settings in "System > SNMP > General".
  - SNMPv3
    Access to device parameters is possible with SNMP version 3. You can configure other settings in "System > SNMP > General".
- **SNMPv1/v2 Read-Only**
  Enable or disable write access to SNMP variables with SNMPv1/v2c.
- **SNMPv1 traps**
  Enable or disable the sending of SNMP traps (alarm frames). You can configure other settings in "System > SNMP > Traps".

- **NFC (Near Field Communikation)**
  Enable or disable the "NFC" function.
  This function is only available for SCALANCE XM400.
- **Configuration mode**
  Select the required mode. The following modes are possible:
  - Automatic save
    Automatic save mode. Approximately 1 minute after the last parameter change or when you restart the device, the configuration is automatically saved.
  - Trial
    In the "Trial" configuration mode, although changes are adopted, they are not saved in the configuration file (startup configuration).
    To save changes in the configuration file, use the "Write startup config" button. The button is displayed when you set this configuration mode. The display area also shows the message "Trial mode active - Press "Write Startup Config" button to make your settings persistent" as soon as there are unsaved modifications. This message can be seen on every page until the changes made have either been saved or the device has been restarted.

General

Device

This page contains the general device information.

Settings

- **Current System Time**(Only available online)
  Shows the current system time. The system time is either set by the user or by a time-of-day frame: either SINEC H1 time-of-day frame, NTP or SNTP. (readonly)
- **System Up Time**(only available online)
  Shows the operating time of the device since the last restart. (readonly)
- **Device Type**(only available online)
  Shows the type designation of the device. (readonly)
- **System Name**
  You can enter the name of the device. The entered name is displayed in the selection area. A maximum of 255 characters are possible. The system name is also displayed in the CLI input prompt. The number of characters in the CLI input prompt is limited. The system name is truncated after 16 characters.

- **System Contact**
  You can enter the name of a contact person responsible for managing the device.
- **Location**
  You can enter the location of the device. The entered installation location is displayed in the selection area.

Note

The ASCII characters 0x20 to 0x7e are used in the input boxes.

At the start and end of the input boxes"System Name", "System Contact" and "System Location", the characters "<", ">" and "space" are not permitted.

Coordinates

On this page, you configure the geographic coordinates (latitude, longitude and the height above the ellipsoid according to WGS84). These boxes are purely for information with a maximum length of 32 characters.

Getting the coordinates

Use suitable maps for obtaining the geographic coordinates of the device.

The geographic coordinates can also be obtained using a GPS receiver. The geographic coordinates of these devices are normally displayed directly and only need to be entered in the input boxes of this page.

Settings

- **Latitude**
  Enter the north or south latitude for the location of the device.
  For example, +49° 1´ 31.67" means that the device is located at 49 degrees, 1 arc minute and 31.67 arc seconds north latitude.
  A south latitude is shown by a preceding minus character.
  You can also append the letters N (north latitude) or S (south latitude) to the numeric information (49° 1´ 31.67" N).
- **Longitude**
  Enter the value of the eastern or western longitude of the location of the device.
  For example, +8° 20´ 58.73" means that the device is located at 8 degrees, 20 minutes and 58.73 seconds east.
  A western longitude is indicated by a preceding minus sign.
  You can also add the letter E (eastern longitude) or W (western longitude) to the numeric information (8° 20´ 58.73" E).
- **Height**
  Enter the value of the height above sea level in meters.
  For example, 158 m means that the device is located at a height of 158 m above sea level.
  Heights below sea level (for example the Dead Sea) are indicated by a preceding minus sign.

Agent IP

Configuration of the IP addresses

Note

The page is only available if there is an online connection to the device.

On this page, you set the IP configuration for the device.

With devices with more than one IP interface, this call references the "Subnets > Configuration" menu item in the "Layer 3" menu and the configuration of the TIA interface there.

Note

The IP address of the in-band port and the out-band port must belong to different subnets.

Settings

- **IP Address**
  In the input boxes under "In band", enter the IP address, subnet mask and the default gateway. In the input boxes under "Out band", enter the IP address, subnet mask and the default gateway.
  If you change the IP address, the Web browser should automatically set itself to the new address. If this does not happen, enter the new address in the Web browser manually.
- **Subnet mask**
  Here, in "In-band", you enter the subnet mask of the in-band port and in "Out-band" the subnet mask of the out-of-band port.
- **Default Gateway**
  If the device is required to communicate with devices (diagnostics stations, e-mail servers etc.) in another subnet, enter the IP address of the default gateway here. The out-of-band port it is not accessible from a different subnet.
- **Agent VLAN ID**
  From the drop-down list, select the VLAN ID for the in-band management. You can only select VLANs that have already been configured.

  Note

  Changing the agent VLAN ID

  If the configuration PC is connected directly to the device via Ethernet and you change the agent VLAN ID, the device is no longer reachable via Ethernet following the change.

- **MAC Address**
  Shows the MAC address of the device. The MAC address is linked to the hardware and cannot be modified.

DNS client

The DNS () server (Domain Name System) assigns a domain name to an IP address so that a device can be uniquely identified.

If this function is enabled, the IE switch can communicate with a DNS server as a DNS client.

Note

The DNS client function can only be used if there is a DNS server in the network.

Description

The page contains the following boxes:

- **DNS Client**
  Enable or disable depending on whether the IE switch should operate as a DNS client.
- **DNS Server IP Address**
  Enter the IP address of the DNS server.

Restart

On this page, there is a button with which you can restart the device and various options for resetting to the device defaults.

Note

The page is only available if there is an online connection to the device.

Note

Note the following points about restarting a device:

- You can only restart the device with administrator privileges.
- A device should only be restarted with the buttons of this page or with the appropriate CLI commands and not by a power cycle on the device.
- Any modifications you have made only become active on the device after clicking the "Set Values" button on the relevant page. If the device is in "Trial Mode", configuration modifications must be saved manually before a restart. In "Automatic Save" mode, the last changes are saved automatically before a restart.

Settings

The following options are available for restarting:

- **Restart**
  Click this button to restart the system. You must confirm the restart in a dialog box. During a restart, the device is reinitialized, the internal firmware is reloaded, and the device runs a self-test. The learned entries in the address table are deleted. You can leave the browser window open while the device restarts. You then need to log in again:
- **Restore Factory Defaults and Restart**
  Click this button to restart the system. You must confirm the restart in a dialog box. During a restart, the device is reinitialized, the internal firmware is reloaded, and the device runs a self-test. The learned entries in the address table are deleted. You can leave the browser window open while the device restarts. You then need to log in again.

Note

By resetting all the defaults to the factory settings, the IP address and the passwords are also lost. Following this, the device can only be accessed using the Primary Setup Tool or using DHCP.

With the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

Load & save

Uploading and saving using HTTP

Loading and saving data via HTTP

On this page, you store device data in an external file on your client PC or load such data from an external file from the PC to the devices. This means, for example, that you can also load new firmware from a file located on your client PC.

Note

The page is only available if there is an online connection to the device.
This page is available both for connections using HTTP and for connections using HTTPS.

Note

Incompatibility with previous firmware versions with/without PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed.

In this situation, if a PLUG is inserted in the device, following the restart, this has the status "Not Accepted" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data. If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using System > PLUG.

Note

Configuration files and Trial mode/Automatic Save

In "Automatic Save" mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In Trial mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write Startup Config" button on the "System > Configuration" WBM page to save changes in the configuration file.

Settings

The table has the following columns:

- **Type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.
- **Load**
  With this button, you can upload files to the device. The button can be enabled, if this function is supported by the file type.
- **Save**
  With this button, you can save files from the device. The button can only be enabled if this function is supported by the file type and the file exists on the device.
- **Delete**
  With this button, you can delete files from the device. The button can only be enabled if this function is supported by the file type and the file exists on the device.

Loading new firmware

After successfully downloading the firmware, you will be requested to restart the device.

Restart the device and continue to configure the newly started firmware.

Procedure

Loading files using HTTP

1. Start the upload function by clicking the one of the "Load" buttons.
   The dialog for uploading a file opens.
2. Go to the file you want to upload.
3. Click the "Open" button in the dialog.
   The file is now uploaded.
4. When the message "Restart required" appears, click the "Yes" button to trigger the restart.
   If you click the "No" button, there is no device restart. The changes only take effect after a restart.

Saving files using HTTP

1. Start the save function by clicking the one of the "Save" buttons.
2. You will be prompted to select a storage location and a name for the file. Or you accept the proposed file name. To make the selection, use the dialog in your browser. After making your selection, click the "Save" button.

Deleting files using HTTP

1. Start the delete function by clicking the one of the "Delete" buttons.
   The file will be deleted.

Reusing configuration data

If several devices are to receive the same configuration and the IP addresses are assigned using DHCP, the effort for configuration can be reduced by saving and reading in the configuration data.

Follow the steps below to reuse configuration data:

1. Save the configuration data of a configured device on your PC.
2. Download this configuration file to all other devices you want to configure in this way.
3. If individual settings are necessary for specific devices, these must be made online on the relevant device.

Configuration data has a checksum. If you edit the files, you can no longer upload them to the IE switch.

Uploading and saving using TFTP

Loading and saving data via a TFTP server

On this page, you can configure the TFTP server and the file names. You can also store device data in an external file on your client PC or load such data from an external file from the PC to the devices. This means, for example, that you can also load new firmware from a file located on your client PC.

Note

Incompatibility with previous firmware versions with/without PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed.
In this situation, if a PLUG is inserted in the device, following the restart, this has the status "Not Accepted" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data. If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually on the "System > PLUG page.

Note

Configuration files and Trial mode/Automatic Save

In "Automatic Save" configuration mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In the "Trial" configuration mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write Startup Config" button on the "System > Configuration" WBM page to save changes in the configuration files.

Settings

- **TFTP Server IP Address**
  Here, enter the IP address of the TFTP server with which you exchange data.
- **TFTP Server Port**
  Here, enter the port of the TFTP server via which data exchange will be handled. If necessary, you can change the default value 69 to your own requirements.

The table has the following columns:

- **File type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.
- **File name**
  Enter a file name.
- **Select action** (only available online)
  Select the action from the drop-down list. The selection depends on the selected file type, for example the log file can only be saved.
  The following actions are possible:
  - Save file
    With this selection, you save a file on the TFTP server.
  - Load file
    With this selection, you load a file from the TFTP server.

Events

Configuration

On this page, you specify how the device reacts to events.

Settings

- **Signaling Contact Response**
  Select the behavior of the signaling contact. The following reactions are possible:
  - conventional
    Default setting for the signaling contact. An error/fault is displayed by the fault LED and the signaling contact is opened. When the error/fault state no longer exists, the fault LED goes off and the signaling contact is closed.
  - User-defined
    The way the signaling contact works does not depend on the error/fault that has occurred. The signaling contact can be opened or closed as required by user actions.
- **Signaling Contact Status**
  Select the status of the signaling contact. The following statuses are possible:
  - close
    Signaling contact is closed.
  - open
    Signaling contact is opened.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **E-mail/Trap/Log table/Syslog/Errors**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Event**
  The "Event" column contains the following values:
  - Cold/Warm Start
    The device was turned on or restarted by the user.
  - Link Change
    This event occurs only when the port status is monitored and has changed, see "System > Fault Monitoring > Link Change".
  - Authentication Failure
    This event occurs when attempting access with a bad password.
  - RMON Alarm
    An alarm or event has occurred relating to the remote monitoring of the system.
  - Power Supply Switchover
    This event occurs only when power supply lines 1 and 2 are monitored. It indicates that there was a change to line 1 or line 2, see System > Fault monitoring > Power supply".
  - RM status change
    The status of the redundancy manager has changed
  - Spanning Tree Topology Change
    The STP or RSTP or MSTP topology has changed.
  - Fault State Change
    The fault state has changed. The fault state can relate to the activated port monitoring, the response of the signaling contact or the power supply monitoring.
  - VRRP State Change (only with routing using VRRP)
    The status of the virtual router has changed
  - Loop Detection
    A loop was detected in the network segment.
  - OSPF State Change (only with routing using OSFP)
    The status of OSPF has changed
- **E-mail**
  The device sends an e-mail. This is only possible if the SMTP server is set up and the "SMTP client" function is enabled.
- **Trap**
  The device sends an SNMP trap. This is only possible if "SNMPv1 traps" is enabled in "System > Configuration".
- **Log table**
  The device writes an entry in the log table.
- **Syslog**
  The device writes an entry to the system log server. This is only possible if the system log server is set up and the "Syslog client" function is enabled.
- **Fault**
  The device triggers a fault. The error LED lights up

Severity filter

On this page, set the threshold levels for sending system event notifications.

Settings

The table has the following columns:

- **Client Type**
  Select the client type for which you want to make settings:
  - E-mail
    Sending system event messages by e-mail
  - Log table
    Entry of system events in the log table.
  - Syslog
    Entering system events in the Syslog file
- **Severity**
  Select the required level. The following settings are possible:
  - Info
    System events are processed as of the severity level "Info".
  - Warning
    System events are processed as of the severity level "Warning".
  - Critical
    System events are processed as of the severity level "Critical".

SMTP client

The device provides the option of automatically sending an e-mail if an alarm event occurs (for example to the network administrator). The e-mail contains the identification of the sending device, a description of the cause of the alarm in plain language, and a time stamp. This allows centralized network monitoring to be set up for networks with few nodes based on an E-mail system. When an e-mail error message is received, the WBM can be started by the Internet browser using the identification of the sender to read out further diagnostics information.

On this page, you can configure up to three SMTP servers and the corresponding e-mail addresses.

Note

Depending on the properties and configuration of the SMTP server, it may be necessary to adapt the "Sender E-Mail Address" input box for the e-mails. Check with the administrator of the SMTP server.

Settings

- **SMTP Client**
  Enable or disable the SMTP client.
- **Email address of the sender**
  Enter the name of the sender to be included in the e-mail, for example the device name. This setting applies to all configured SMTP servers.
- **SMTP Port**
  Enter the port via which your SMTP server can be reached.
- **SMTP Server IP Address**
  Enter the IP address of the SMTP server.

This table contains the following columns:

- **SMTP server address**
  Shows the SMTP server IP address.
- **Email address of the recipient**
  Enter the e-mail address to which the device sends an e-mail if a fault occurs.

Syntax of email addresses

The following conditions apply to e-mail addresses:

- Alphanumeric characters are permitted.
- The following special characters are permitted:
  - @
  - _ (underscore)
  - - (hyphen)
  - . (period)
- A @ character must be included.
- Only one @ may be used.
- The characters "@" and "." must not be the first or last character.

DHCP client

If the DHCP mode is activated, the DHCP client starts a DHCP request to a configured DHCP server and is assigned an IP address as the response. The server manages an address range from which it assigns IP addresses. It is also possible to configure the server so that the client always receives the same IP address in response to its request.

Settings

- **DHCP client configuration request (opt. 66, 67)**
  Select this option if you want the DHCP client to use options 66 and 67 to download and then enable a configuration file.
- **DHCP Mode**
  Select the DHCP mode. The following modes are possible:
  - via MAC Address
    Identification is based on the MAC address.
  - via DHCP Client ID
    Identification is based on a freely defined DHCP client ID.
  - via System Name
    Identification is based on the device name. If the device name is 255 characters long, the last character is not used for identification.

This table contains the following columns:

- **Interface**
  Shows the available interfaces.
- **DHCP**
  Enable or disable the DHCP client.

SNMP

General

On this page, you make the basic settings for SNMP. Enable the functions you want to use.

Settings

- **SNMP**
  Select the SNMP protocol. The following settings are possible:
  - "-" (disabled)
    SNMP is disabled.
  - SNMPv1/v2c/v3
    SNMPv1/v2c/v3 is supported.
  - SNMPv3
    Only SNMPv3 is supported.
- **SNMPv1/v2 Read-Only**
  If you enable this option, SNMPv1/v2c can only read the SNMP variables.

Note

Community String

For security reasons, do not use the standard values "public" or "private". Change the community strings following the initial installation.

- **SNMPv1/v2c Read Community String**
  Enter the community string for access of the SNMP protocol.
- **SNMPv1/v2c Read/Write Community String**
  Enter the community string for read and write access of the SNMP protocol.
- **SNMPv1 Traps**
  Enable or disable the sending of SNMP traps (alarm frames). On the "Trap" tab, specify the IP addresses of the devices to which SNMP traps will be sent.
- **SNMPv1/v2c Trap Community String**
  Enter the community string for sending SNMPv1/v2 messages.

Traps

If an alarm event occurs, a device can send SNMP traps (alarm frames) to up to ten different management stations at the same time. Traps are only sent if events specified in "Events > Configuration" occur.

Note

SNMP traps are sent only when the "SNMPv1 Traps" setting was selected in "SNMP > General".

Settings

- **IP Address**
  Enter the IP address of the stations to which the device sends SNMP traps

The table has the following columns:

- **IP Address**
  Shows the IP addresses of the stations to which the device sends SNMP traps.
- **Trap**
  Enable or disable the sending of SNMP traps. Stations that are entered but not selected do not receive SNMP traps.

v3 Groups

Security settings and assigning permissions

SNMP version 3 allows permissions to be assigned, authentication, and encryption at protocol level. The security levels and read/write permissions are assigned according to groups. The settings automatically apply to every member of a group.

Settings

- **Group Name**
  Enter the name of the group. This name must have at least two characters, the maximum length is 32 characters.
- **Security Level**
  Select the security level (authentication, encryption) valid for the selected group. You have the following options for the security levels:
  - no Auth/no Priv
    No authentication enabled / no encryption enabled.
  - Auth/no Priv
    Authentication enabled / no encryption enabled.
  - Auth/Priv
    Authentication enabled / encryption enabled.

The table has the following columns:

- **Group Name**
  Shows the defined group names.

  Note

  Once a group name and the security level have been specified, they can no longer be modified after the group is created. If you want to change the group name or the security level , you will need to delete the group and recreate it and reconfigure it with the new name.

- **Security Level**
  Shows the configured security level.
- **Read**
  Enable or disable read access.
- **Write**
  Enable or disable write access.

  Note

  For write access to work, you also need to enable read access.

- **Persistence**
  Shows whether or not the group is assigned to an SNMPv3 user. If the group is not assigned to an SNMPv3 user, no automatic saving is triggered and the configured group disappears again after restarting the device.
  - Yes
    The group is assigned to an SNMPV3 user.
  - No
    The group is not assigned to an SNMPV3 user.

v3 users

User-specific security settings

On the WBM page, you can create new SNMPv3 users and modify or delete existing users. The user-based security model works with the concept of the user name; in other words, a user ID is added to every frame. This user name and the applicable security settings are checked by both the sender and recipient.

Settings

- User name
  Enter a freely selectable user name. After you have entered the data, you can no longer modify the name.

The table has the following columns:

- User name
  Shows the created users.
- Group Name
  Select the group to which the user will be assigned.
- Authentication Protocol
  Select the authentication protocol. Can only be enabled, if this group supports the function. The following settings are available:
  - None
  - MD5
  - SHA
- Encryption Protocol
  Specify whether or not the user uses the DES algorithm. Can only be enabled, if the group supports this function.
- Authentication Password
  Enter the authentication password in the first input box.
- Confirm authentication password
  Confirm the password by repeating the entry.
- Encryption Password
  Enter your encryption password.

- **Confirm Encryption Password**
  Confirm the encryption password by repeating the entry.
- **Persistence**
  Shows whether or not the user is assigned to an SNMPv3 group. If the user is not assigned to an SNMPv3 group, no automatic saving is triggered and the configured user disappears again after restarting the device.
  - Yes
    The user is assigned to an SNMPv3 group.
  - No
    The user is not assigned to an SNMPv3 group.

System time

Manual time setting

On this page, you set the date and time of the system. For this setting to be used, enable "Manual time setting".

Note

The page is only available if there is an online connection to the device.

Settings

- **Time Manually**
  Enable or disable the manual time setting.
- **System Time**
  Enter the date and time in the format "MM/DD/YYYY HH:MM:SS". Can only be edited, if manual time setting is enabled.
  After a restart, the time of day begins at 01/01/2000 00:00:00
- **Use PC Time**
  Click the button to use the time setting of the PC.

- **Last Synchronization Time**
  This box is read-only and shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".
- **Last Synchronization Mechanism**
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame.
  - PTP
    Automatic time-of-day synchronization with PTP

DST overview

On this page, you can create new entries for the daylight saving time changeover.

The table provides an overview of the existing entries.

Settings

- **DST No.**
  Shows the number of the entry.
  If you create a new entry, a new line with a unique number is created.
- **Name**
  Shows the name of the entry.
- **Year**
  Shows the year for which the entry was created.
- **Start Date**
  Shows the month, day and time for the start of daylight saving time.
- **End Date**
  Shows the month, day and time for the end of daylight saving time.
- **Type**
  Shows how the daylight saving time changeover is made:
  - Date
    A fixed date is entered for the daylight saving time changeover.
  - Recurring
    A rule was defined for the daylight saving time changeover.

DST configuration

On this page, you can configure the entries for the daylight saving time changeover. As result of the changeover to daylight saving or standard time, the system time for the local time zone is correctly set.

You can define a rule for the daylight saving time changeover or specify a fixed date.

Settings

Note

The content of this page depends on the selection in the "Type" box.

The boxes "DST No.", "Type" and "Name" are always shown.

- **DST No.**
  Select the type of the entry.
- **Type**
  Select how the daylight saving time changeover is made:
  - Date
    You can set a fixed date for the daylight saving time changeover.
    This setting is suitable for regions in which the daylight saving time changeover is not governed by rules.
  - Recurring
    You can define a rule for the daylight saving time changeover.
    This setting is suitable for regions in which the daylight saving time always begins or ends on a certain weekday.
- **Name**
  Enter a name for the entry.

Settings with "Date" selected

You can set a fixed date for the start and end of daylight saving time.

- **Year**
  Enter the year for the daylight saving time changeover.
- **Start Date**
  Enter the following values for the start of daylight saving time:
  - Day
  - Hour
  - Month
- **End Date**
  Enter the following values for the end of daylight saving time:
  - Day
  - Hour
  - Month

Settings with "Recurring" selected

You can create a rule for the daylight saving time changeover.

- **Start Date**
  Enter the following values for the start of daylight saving time:
  - Hour
  - Month
  - Week
    You can select the 1st to 5th or the last week of the month.
  - Weekday
- **End Date**
  Enter the following values for the end of daylight saving time:
  - Hour
  - Month
  - Week
    You can select the 1st to 5th or the last week of the month.
  - Weekday

SNTP client

SNTP (Simple Network Time Protocol) is used for synchronizing the time in the network. The appropriate frames are sent by an SNTP server in the network.

Settings

- **SNTP Client**
  Enable or disable automatic time-of-day synchronization using SNTP.
- **Current System Time** (only available online)
  Shows the values currently set in the system for date and time.
- **Last Synchronization Time** (only available online)
  This box is read-only and shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".

- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time. Settings for daylight-saving and standard time are taken into account in this box by specifying the time offset.
- **SNTP Mode**
  Select the synchronization mode. The following types of synchronization are possible:
  - Poll
    If you select this protocol type, the input boxes "SNTP Server IP Address", "SNTP Server Port" and "Poll Interval" are displayed for further configuration. With this type of synchronization, the device is active and sends a time query to the SNTP server.
  - Listen
    With this type of synchronization, the device is passive and "listens" for SNTP frames that deliver the time of day.
- **SNTP Server IP Address**
  Enter the IP address of the SNTP server.
- **SNTP Server Port**
  Enter the port of the SNTP server.
- **Poll Interval [s]**
  Enter the interval in seconds between two time polls.

NTP client

If you require time-of-day synchronization using NTP, you can make the relevant settings here.

Settings

- **NTP Client**
  Enable or disable automatic time-of-day synchronization using NTP.
- **Current System Time** (only available online)
  Shows the values currently set in the system for date and time.

- **Last Synchronization Time** (only available online)
  This box is read-only and shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".
- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time. Settings for daylight-saving and standard time are taken into account in this box by specifying the time offset.
- **NTP Server IP Address**
  Enter the IP address of the NTP server.
- **NTP Server Port**
  Enter the port of the NTP server.
- **Poll Interval [s]**
  Enter the interval in seconds between two time polls.

SIMATIC Time Client

On this page, you configure time synchronization with the SIMATIC Time Client.

Settings

- **SIMATIC Time Client**
  Enable or disable loop the SIMATIC Time Client.
- **Current System Time** (only available online)
  Shows the values currently set in the system for date and time.

- **Last Synchronization Time** (only available online)
  This box is read-only and shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".
- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP

PTP Client

On this page, you configure time synchronization with PTP (Precision Time Protocol).

This function is only available for SCALANCE X500.

Settings

- **PTP Client**
  Enable or disable time synchronization using PTP. You can configure other settings in "Layer 2 > PTP".
- **Current System Time** (only available online)
  This box is read-only and displays the current system time.
- **Last Synchronization Time** (only available online)
  This box is read-only and shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".

- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time.

Auto logout

On this page, set the times after which there is an automatic logout from WBM or the CLI following user in activity.

Note

No automatic logout from the CLI

If the connection is not terminated after the set time, check the setting of the "keepalive" mechanism on the Telnet client.

If the interval is shorter than the configured time, the connection is kept alive although no user data is transferred. Example: you selected 300 seconds for the automatic logout and 120 seconds is set for the keepalive function. In this case, a packet is sent every 120 seconds that keeps the connection up.

- Disable the keepalive mechanism (interval time = 0)
  or
- Set the interval high enough so that the underlying connection is terminated when there is inactivity.

Settings

- **Web Based Management [s]**
  Enter the time in seconds for the automatic logout from WBM. If you enter the value 0, the automatic logout is disabled.
- **CLI (TELNET, SSH, Serial) [s]**
  Enter the time in seconds for the automatic logout from the CLI. If you enter the value 0, the automatic logout is disabled.

Button

The "Select/Set" button has the following functions:

- Change display mode
- Enable redundancy manager
- Restore to factory defaults
- Fault mask and LED display

You will find a detailed description of the individual functions available with the buttons in the operating instructions of the specific device.

Settings

- **Restore Factory Defaults**
  Enables or disables the "Restore Factory Defaults" function for the Select/Set button.

Caution

Button function "Restore Factory Defaults" active during startup

If you have disabled this function in your configuration, disabling is only valid during operation. When restarting, for example after power down, the function is active until the configuration is loaded so that the device can inadvertently be reset to the factory settings. This may cause unwanted disruption in network operation since the device needs to be reconfigured if this occurs. An inserted PLUG is also deleted and returned to the status as shipped

- **Redundancy Manager**
  Enables or disables the redundancy manager function for the Select/Set button.
- **Set Fault Mask**
  Enable or disable the function "Define fault mask via the LED display" with the Select/set button. This function only works in display mode D.

Syslog client

Syslog according to RFC 3164 is used for transferring short, unencrypted text messages over UDP in the IP network. This requires a system log server.

Requirements for sending log entries:

- The system log function is enabled on the device.
- The system log function is enabled for the relevant event.
- There is a system log server in your network that receives the log entries. Since this is a UDP connection, there is no acknowledgment to the sender.
- The IP address of the system log server is entered on the device.

Settings

- **Syslog Client**
  Enable or disable the system log function.
- **Syslog Server IP Address**
  Enter the IP address of the system log server.

This table contains the following columns

- **Server address**
  Shows the IP address of the system log server.
- **Server Port**
  Enter the port of the Syslog server being used.

Ports

Port overview

The page shows the configuration for the data transfer for all ports of the device.

Settings

- **Port**
  Shows the configurable ports.
- **Port name**
  Shows the name of the port.
- **MAC Address** (only available online)
  Shows the MAC address of the port.
- **Mode** (only available online)
  Shows the transfer parameters of the port
- **Negotiation**
  Shows whether the automatic configuration is enabled or disabled.
- **Flow Ctrl. Type**
  Shows whether flow control is enabled or disabled for the port.
- **Flow Ctrl.**
  Shows whether or not flow control is working on this port.
- **MTU**
  Shows the maximum packet size.
- **Port Type** (only with routing)
  Shows the type of the port. The following types are possible:
  - Router Port
  - Switch Port VLAN Hybrid
  - Switch-Port VLAN Trunk
- **Status**
  Shows whether the port is on or off. Data traffic is possible only over an enabled port.

- **Combo Port Media Type**(SCALANCE XM400 only)
  Shows the mode of the combo port:
  - auto
  - rj45
  - sfp
- **Link** (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - Up
    The port has a valid link to the network, a link integrity signal is being received.
  - Down
    The link is down, for example because the connected device is turned off.

Configuration

On this page, you configure the ports of the device

Settings

- **Port**
  Select the port to be configured. The port is made up of the port number and the slot number, for example port 0.1 is slot 0, port 1.
- **Status**
  Specify whether the port is enabled or disabled.
  - enabled
    The port is enabled. Data traffic is possible only over an enabled port.
  - disabled
    The port is disabled but the connection remains.
  - Link down
    The port is disabled and the connection to the partner device is terminated.
- **Port name**
  Enter a name for the port.
- **MAC Address** (only available online)
  Shows the MAC address of the port.
- **Mode Type**
  Shows the transmission speed and the transmission method of the port. You make the settings for "Autonegotiation" and "Transmission rate" in the port options.

  Note

  Before the port and partner port can communicate with each other, the settings must match at both ends.

- **Mode** (only available online)
  Shows which transmission speed and which transmission mode is currently being used.

- **Negotiation**
  Shows whether the automatic configuration of the connection to the partner port is enabled or disabled.
- **Flow Ctrl. Type**
  Shows whether or not flow control is working on this port.

  Note

  Turning flow control on/off with autonegotiation

  Flow control can only be enabled or disabled if the "autonegotiation" function is turned off. The function cannot enabled again afterwards.

- **Flow Ctrl.**
  Shows whether or not flow control is working on this port.
- **MTU**
  Enter the packet size.
- **Port Type** (only with routing)
  Select the type of the port:
  - Router Port
    The port is an IP interface. It does not support layer 2 functions.
  - Switch Port VLAN Hybrid
    The port sends tagged and untagged frames. It is not automatically a member of a VLAN.
  - Switch-Port VLAN Trunk
    The port only sends tagged frames and is automatically a member of all VLANs.
- **Combo Port Media Type**(SCALANCE XM400 only)
  Specify the mode for the combo port:
  - auto
    If you select this mode, the SFP transceiver port has priority. As soon as an SFP transceiver is plugged in, an existing connection at the fixed RJ-45 port is terminated. If no SFC transceiver is plugged in, a connection can be established via the fixed RJ-45 port.
  - rj45
    If you select this mode, the fixed RJ-45 port is used regardless of the SFP transceiver port.
  - sfp
    If you select this mode, the SFP transceiver port is used regardless of the built-in RJ-45 port.

  The factory setting is auto mode.
- **Link** (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - Up
    The port has a valid link to the network, a link integrity signal is being received.
  - Down
    The link is down, for example because the connected device is turned off.

Changing the port configuration

Note

Optical ports only work with the full duplex mode and at maximum transmission rate. As a result, the following settings cannot be made for optical ports:

- Automatic configuration
- Transmission speed
- Transmission technique

Note

With various automatic functions, the device prevents or reduces the effect on other ports and priority classes (Class of Service) if a port is overloaded. This can mean that frames are discarded even when flow control is enabled.

Port overload occurs when the device receives more frames than it can send, for example as the result of different transmission speeds.

Fault monitoring

Power supply

Configure whether or not the power supply should be monitored by the messaging system. Depending on the hardware variant, there are one or two power connectors (Supply 1 / Supply 2). With a redundant power supply, configure the monitoring separately for each individual feed-in line.

If there is no power on one of the monitored lines (Supply 1 or Supply 2) or when the voltage is too low, a fault is signaled by the signaling system.

Note

You will find the permitted operating voltage limits in the compact operating instructions of the device.

An error causes the fault LED to light up on the device. Depending on the configuration, the error may trigger a trap, an e-mail or an entry in the event log table.

Settings

- **Line 1**
  Enable or disable the monitoring of power connector 1.
- **Line 2**
  Enable or disable the monitoring of power connector 2.

Link Change

On this page, you configure whether or not an error message is triggered if there is a status change on a network connection.

If connection monitoring is enabled, an error is signaled

- when there should be a link on a port and this is missing.
- or when there should not be a link on a port and a link is detected.

An error causes the signaling contact to trigger and the fault LED to light up on the device. Depending on the configuration, the error may trigger a trap, an e-mail or an entry in the event log table.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns

- **Port**
  Shows the available ports.
- **Setting**
  Select the setting. You have the following options:
  - Up
    Error handling is triggered when the port changes to the active status.
    (From "Link down" to "Link up")
  - Down
    Error handling is triggered when the port changes to the inactive status.
    (From "Link up" to "Link down")
  - "-" (disabled)
    The error handling is not triggered.

Redundancy

On this page, you configure whether or not an error message is triggered if there is a status change on a network connection.

Setting

- **Redundancy loss (HRP only)**
  Enable or disable connection monitoring. If the redundancy of the connection is lost, an error is signaled.

PNIO

On this page, you configure the device response to PROFINET input and output.

Description

The page contains the following boxes:

- **PNIO runtime mode**
  Shows the status of the PNIO operation.
- **PNIO runtime mode for next startup**
  Select the runtime mode that will be active the next time the device starts up.
- **PNIO AR status**
  This box shows the PROFINET IO application relation status; in other words, whether or not the IE switch is connected "online" or "offline" with a PROFINET controller.
  In this context, online means that a connection to a PROFINET IO controller exists, that the controller has downloaded its configuration data to the IE switch and that the device can send status data to the PROFINET IO controller. In this status known as "in data exchange", the parameters set with the PROFINET IO controller cannot be configured on the IE switch.
- **PNIO Device Name**
  Enter the PROFINET IO device name.
- **Allow PNIO Data Exchange**
  Enable or disable PNIO data exchange
- **Simulate PNIO Data Exchange**
  Enable or disable the simulation of PNIO data exchange.

PLUG

PLUG configuration

Note

The page is only available if there is an online connection to the device.

Notice

Do not remove or insert a C-PLUG / KEY-PLUG during operation!

A PLUG may only be removed or inserted when the device is turned off.
The device checks whether or not a PLUG is present at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid KEY-PLUG was inserted in the device, the device changes to a defined error state following the restart.

Information about the configuration of the C-PLUG / KEY-PLUG

This page provides detailed information about the configuration stored on the C-PLUG or KEY-PLUG. It is also possible to reset the PLUG to "factory defaults" or to load it with new contents.

Note

The action is only executed after you click the "Set Values" button.

The action cannot be undone.

If you decide against executing the function after making your selection, click the "Refresh" button. As a result the data of this page is read from the device again and the selection is canceled.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Settings

The table has the following rows:

- **Status**
  Shows the status of the PLUG. The following are possible:
  - ACCEPTED
    There is a PLUG with a valid and suitable configuration in the device.
  - NOT ACCEPTED
    Invalid or incompatible configuration on the inserted PLUG.
  - NOT PRESENT
    There is no C-PLUG or KEY-PLUG inserted in the device.
  - FACTORY
    PLUG is inserted and does not contain a configuration. This status is also displayed when the PLUG was formatted during operation.
  - MISSING
    There is no PLUG inserted. Functions are configured on the device for which a license is required.
- **Device Family**
  Shows the SIMATIC NET product line that used the C-PLUG or KEY-PLUG previously.

- **Device Type**
  Shows the device type within the product line that used the C-PLUG or KEY-PLUG previously.
- **Configuration Revision**
  The version of the configuration structure. This information relates to the configuration options supported by the device and has nothing to do with the concrete hardware configuration. This revision information does not therefore change if you add or remove additional components (modules or extenders), it can, however, change if you update the firmware.
- **File System**
  Displays the type of file system on the PLUG.

  Notice

  New file system UBI

  As of firmware version 3.0, UBI is the standard file system for the C-PLUG or KEY-PLUG. If a C-PLUG with the previous file system IECP is detected in such a device, this C-PLUG will be formatted for the UBI file system and the data will be rewritten to the C-PLUG.

  The file system is also changed following a firmware update to V3.0. A downgrade to the previous version of the corresponding software is then a problem. The firmware can neither read nor write the C-PLUG or KEY-PLUG and it is not even possible to "Erase PLUG to factory default".

- **File System Size [bytes]**
  Shows the maximum storage capacity of the file system on the C-PLUG.
- **File System Usage**
  Displays the storage space in use in the file system of the C-PLUG.
- **Info String**
  Shows additional information about the device that used the PLUG previously, for example, order number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.
- **Modify PLUG**
  Select the required setting.
  - Write current configuration to PLUG
    This option is available only if the status of the PLUG is "NOT ACCEPTED" or FACTORY. The configuration in the internal flash memory of the device is copied to the PLUG.
  - Erase PLUG to factory default
    Deletes all data from the C-PLUG and triggers low-level formatting.

PLUG license

Note

The page is only available if there is an online connection to the device.

Notice

Do not remove or insert a C-PLUG / KEY-PLUG during operation!

A PLUG may only be removed or inserted when the device is turned off.
The device checks whether or not a PLUG is present at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid KEY-PLUG was inserted in the device, the device changes to a defined error state following the restart.

If the device was configured at some time with a PLUG, the device can no longer be used without this PLUG. To be able to use the device again, reset the device to the factory settings.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Information about the license of the KEY-PLUG

A C-PLUG can only store the configuration of a device. In addition to the configuration, a KEY-PLUG also contains a license that enables certain functions of your SIMATIC NET device.

This page provides detailed information about the license on the KEY-PLUG. In this example, the KEY-PLUG contains the data for enabling the layer 3 functions of the device.

Displayed values

- **Status**
  Shows the status of the KEY-PLUG. The following are possible:
  - ACCEPTED
    The KEY-PLUG in the device contains a suitable and valid license.
  - NOT ACCEPTED
    The license of the inserted KEY-PLUG is not valid.
  - NOT PRESENT
    No KEY-PLUG is inserted in the device.
  - MISSING
    There is no KEY-PLUG or a C-PLUG with the status "FACTORY" inserted in the device. Functions are configured on the device for which a license is required.
  - WRONG
    The inserted KEY-PLUG is not suitable for the device.
  - UNKNOWN
    Unknown content of the KEY-PLUG.
  - DEFECTIVE
    The content of the KEY-PLUG contains errors.
- **Order ID**
  Shows the order number of the KEY-PLUG. The KEY-PLUG is available for various functional enhancements and for various target systems.
- **Serial Number**
  Shows the serial number of the KEY-PLUG.
- **Info String**
  Shows additional information about the device that used the KEY-PLUG previously, for example, order number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.

Note

When you save the configuration, the information about whether or not a KEY-PLUG was inserted in the device at the time is also saved. This configuration can then only work if a KEY-PLUG with the same order number / license is inserted.

Ping

Reachability of an address in an IP network

Note

The page is only available if there is an online connection to the device.

With the ping function, you can check whether a certain IP address is reachable in the network.

Settings

- **IP Address**
  Enter the IP address of the device.
- **Repeat**
  Enter the number of ping requests.
- **Ping**
  Click this button to start the ping function.
- **Ping Output**
  This box shows the output of the ping function.
- **Delete**
  Click this button to delete the ping output.

PoE

General

On this page, you specify the maximum power for the power sourcing equipment (PSE).

PSE (power sourcing equipment)

The SCALANCE X-500 represents a PSE (Power Sourcing Equipment).

With a SCALANCE XM400, you can use the "Power over Ethernet" function via the port extender PE408PoE. Each group of four ports with PoE capability is known as a PSE. The numbering of the PSEs is fixed and does not change with the number of plugged-in PoE port extenders or the slot. A maximum of 4 PSEs are possible.

Setting

- **PSE**
  Shows the number of the PoE power supply.
- **Maximum Power**
  Maximum power that a PSE provides to supply PoE devices.
- **Allocated Power**
  Sum of the power reserved by the PoE devices according to the "Classification".
- **Used power**
  Sum of the power used by the end devices.
- **Usage Threshold [%]**
  When the power being used by the end devices exceeds the percentage shown here, an event is triggered.

Port

Settings for the ports

For each individual PoE port, you can specify whether or not the power will be supplied via Ethernet. You can also set a priority for each connected powered device (PD). Devices for which a high priority was set, take preference over other devices for the power supply.

Settings

The table has the following columns:

- Port
  Shows the configurable PoE ports.
  The port is made up of the port number and the slot number, for example port 0.1 is slot 0, port 1.
- Setting
  Enable the PoE power supply for this port or interrupt it.
- Priority
  Select which priority this port will have for the power supply.
  The following settings are possible, in ascending order of relevance:
  - low
    low priority
  - high
    medium priority
  - critical
    high priority

  If the same priority is set for two ports, the port with the lower port number will be preferred when necessary.
- Type
  Here, you can enter a string to describe the connected device in greater detail.
- Classification
  The classification specifies the class of the device. From this, it is possible to recognize the maximum power of the device.

- **Status**
  Shows the current status of the port.
  The following states are possible:
  - disabled
    The PoE power supply is deactivated for this port.
  - delivering
    The PoE power supply is activated for this port and a device is connected.
  - searching
    The PoE power supply is activated for this port but there is no device connected.

  Note

  If a device is connected to a port with PoE capability, a check is made to determine whether the power of the port is adequate for the connected device. If the power of the port is inadequate, although PoE is enabled in "Setting", the port nevertheless has the status "disabled". This means that the port was disabled by the PoE power management.

- **Power [mW]**
  Shows the power that the SCALANCE provides at this port.
- **Voltage [V]**
  Shows the voltage applied to this port.
- **Current [mA]**
  Shows the current with which a device connected to this port is supplied.

Port diagnostics

Cable tester

Note

The page is only available if there is an online connection to the device.

With this page, each individual Ethernet port can run independent fault diagnostics on the cable. This test is performed without needing to remove the cable, connect a cable tester and install a loopback module at the other end. Short-circuits and cable breaks can be localized to within a few meters.

Note

Please note that this test is permitted only when no data connection is established on the port to be tested.

Settings

- **Port**
  Select the required port from the drop-down list.
- **Run Test**
  Activates error diagnostics. The result is shown in the table.

This table contains the following columns:

- **Pair**
  Shows the wire pair in the cable.

  Note

  Wire pairs

  Wire pairs 4-5 and 7-8 of 10/100 Mbps network cables are not used.

  In 1000 Mbps or gigabit Ethernet, all 4 wire pairs are used.

  The wire pair assignment - pin assignment is as follows (DIN 50173):

  Pair 1 = pin 4-5

  Pair 2 = pin 1-2

  Pair 3 = pin 3-6

  Pair 4 = pin 7-8

- **Status**
  Displays the status of the cable.
- **Distance [m]**
  Displays the distance to the cable end, cable break, or short-circuit.

SFP diagnostics

On this page, you run independent error diagnostics for each individual SFP port. This test is performed without needing to remove the cable, connect a cable tester or install a loopback module at the other end.

Note

Please note that this test is permitted only when no data connection is established on the port to be tested.

If, however, there is a data connection to the port to be tested, this is briefly interrupted.

Automatic re-establishment of the connection can fail and then needs to be done manually.

Description

The page contains the following boxes:

- **Port**
  Select the required port from the drop-down list.

The values are shown in the following boxes:

- **Name**
  Shows the name of the interface.
- **Model**
  Shows the type of interface.
- **Revision**
  Shows the hardware version of the SFP.
- **Serial**
  Shows the serial number of the SFP.
- **Nominal Bit Rate [Mbps]**
  Shows the nominal bit rate of the interface.
- **Max. Link (50.0/125um) [m]**
  Shows the maximum distance in meters that is possible with this medium.
- **Max. Link (62.5/125um) [m]**
  Shows the maximum distance in meters that is possible with this medium.

The following table shows the values of the SFP transceiver used in this port:

- **Temperature [°C]**
  Shows the temperature of the interface.
- **Voltage [V]**
  Shows the voltage applied to the interface [V].
- **Current [mA]**
  Shows the current consumption of the interface [mA].
- **RX Power [mW]**
  Shows the receive power of the interface [mW].
- **Tx Power [mW]**
  Shows the transmit power of the interface [mW].
- **Current**
  Shows the current value.
- **Low**
  Shows the lowest value.
- **High**
  Shows the highest value.

Configuring layer 2 functions

Configuration

The functions of layer 2 are configured on this page. With some functions, there are further configuration pages on which more detailed settings can be made. You can also check the settings on the configuration pages.

Settings

- **Protocol Based VLAN**
  Enable or disable protocol-based VLAN. You can configure other settings in "Layer 2 > VLAN".
- **Subnet Based VLAN**
  Enable or disable subnet-based VLAN. You can configure other settings in "Layer 2 > VLAN".
- **Dynamic MAC Aging**
  Enable or disable the "Aging" mechanism. You can configure other settings in "Layer 2 > Dynamic MAC Aging".
- **Redundancy Type**
  The following settings are available:
  - **"-" (disabled)**
    The redundancy function is disabled.
  - **Ring**
    Enables ring redundancy. You can configure other settings in "Layer 2 > Ring Redundancy > Ring".
  - **Spanning Tree**
    If you select this option, you specify the required redundancy mode in "Redundancy Mode".

- **Redundancy Mode**
  If you select "Spanning Tree" for the "Redundancy Type", the following options are then available:
  - **STP**
    Enables the Spanning Tree Protocol (STP). Typical reconfiguration times with spanning tree are between 20 and 30 seconds. You can configure other settings in "Layer 2 > Spanning Tree".
  - **RSTP**
    Enables the Rapid Spanning Tree Protocol (RSTP). If a spanning tree frame is detected at a port, this port reverts from RSTP to spanning tree. You can configure other settings in "Layer 2 > Spanning Tree".

    Note

    When using RSTP (Rapid Spanning Tree Protocol), loops involving duplication of frames or frames being overtaken may occur briefly. If this is not acceptable in your particular application, use the slower standard spanning tree mechanism.
  - **MSTP**
    Enables the Multiple Spanning Tree Protocol (MSTP). You can configure other settings in "Layer 2 > Spanning Tree".

  If you select "Ring" for the "Redundancy Type", the following options are then available:
  - **Automatic Redundancy Detection**
    Select this setting to create an automatic configuration of the redundancy mode. In this mode, the device automatically detects whether or not there is a device with the "HRP Manager" role in the ring. If this is the case, the device adopts the role of "HRP Client". If no HRP manager is found, all devices with the "Automatic Redundancy Detection" or "MRP Auto Manager" setting negotiate among themselves to establish which device adopts the role of "MRP Manager". The device with the lowest MAC address will always become "MRP Manager". The other devices automatically set themselves to redundancy type "MRP Client".
  - **MRP Auto-Manager**
    Automatic media redundancy manager
  - **MRP Client**
    Media redundancy client
  - **HRP Client**
    High Speed Redundancy Protocol client
  - **HRP Manager**
    High Speed Redundancy Protocol manager
- **Standby**
  Enable or disable the "Standby redundancy" function. You can configure other settings in "Layer 2 > Ring Redundancy > Standby".
- **Passive Listening**
  Enable or disable the "passive listening" function.

- **RMON**
  If you select this check box, Remote Monitoring (RMON) allows diagnostics data to be collected on the device, prepared and read out using SNMP by a network management station that also supports RMON. This diagnostic data, for example port-related load trends, allow problems in the network to be detected early and eliminated.
- **Dynamic Multicast**
  The following settings are possible:
  - **"-" (disabled)**
  - **IGMP Snooping**
    Enables IGMP (Internet Group Management Protocol). You can configure other settings in "Layer 2 > Multicast > IGMP".
  - **GMRP**
    Enables GMRP (GARP Multicast Registration Protocol). You can configure other settings in "Layer 2 > Multicast > GMRP".

    Note

    GMRP and IGMP cannot operate at the same time.
- **GVRP**
  Enable or disable "GVRP" (GARP VLAN Registration Protocol). You can configure other settings in "Layer 2 > VLAN > GVRP".
- **Mirroring**
  Enable or disable port mirroring. You can configure other settings in "Layer 2 > Mirroring > Port".
- **Loop Detection**
  Enable or disable loop detection. You can configure other settings in "Layer 2 > Loop Detection".
- **PTP**
  Specify how the device will process PTP messages. You can configure other settings in "Layer 2 > PTP".
  - Off
    The device does not process any PTP messages. PTP messages are, however, forwarded according to the rules of the switch.
  - transparent
    The device adopts the function of a transparent clock and forwards PTP messages to other nodes while at the same time making entries in the correction field of the PTP message.

QoS

CoS map

On this page, CoS priorities are assigned to certain queues (traffic queues).

Settings

- **CoS**
  Shows the CoS priority of the incoming packets.
- **Queue**
  Select the forwarding queue (send priority) that is assigned the CoS priority.
  The higher the number of the queue, the higher the send priority.

DSCP map

On this page, DSCP settings are assigned to various queues (traffic queues).

Settings

- **DSCP**
  Shows the DSCP priority of the incoming packets.
- **Queue**
  Select the forwarding queue (send priority) that is assigned the DSCP value. The higher the queue number, the higher the send priority.

Load limitation

Limiting the transfer rate of incoming and outgoing data

On this page, you configure the load limitation (maximum number of data packets per second) for the individual ports. You can specify the category of frame for which these limit values will apply.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Limit Ingress Unicast (DLF) / Limit Ingress Broadcast / Limit Ingress Multicast**
  Select the required setting.
  - enabled: Enables the function.
  - disabled: Disables the function
  - No Change: The setting in table 2 remains unchanged
- **Total Ingress Rate [pkts/s]**
  Specify the maximum number of incoming packets processed by the device. If "No Change" is entered, the entry in table 2 remains unchanged.

- **Egress Rate [Kb/s]**
  Specify the data rate for all outgoing frames. If "No Change" is entered, the entry in table 2 remains unchanged
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the port to which the settings relate.
- **Limit Ingress Unicast (DLF)**
  Enable or disable the data rate for limiting incoming unicast frames with an unresolvable address (Destination lookup failure).
- **Limit Ingress Broadcast**
  Enable or disable the data rate for limiting incoming broadcast frames.
- **Limit Ingress Multicast**
  Enable or disable the data rate for limiting incoming multicast frames.
- **Total Ingress Rate [pkts/s]**
  Specify the maximum number of incoming packets processed by the device.
- **Egress Rate [Kb/s]**
- Specify the data rate for all outgoing frames.

Note

Rounding of the values, deviation from desired value

When you enter the values for transmission rates, note that the WBM rounds to correct values.

If values are configured for total ingress transmission rate and egress transmission rate, the actual values in operation can exceed or fall below the set values by 10%.

VLAN

General

On this page, you define the VLAN and specify the use of the ports.

Note

Changing the "Agent VLAN ID"

If the configuration PC is connected directly to the device via Ethernet and you change the "agent VLAN ID", the device is no longer reachable via Ethernet following the change.

Important rules for VLANs

Make sure you keep to the following rules when configuring and operating your VLANs:

- Frames with the VLAN ID "0" are handled as untagged frames but retain their priority value.
- As default, all ports on the device send frames without a VLAN tag to ensure that the end node can receive these frames.
- With SCALANCE X devices, the VLAN ID "1" is the default on all ports.
- If an end node is connected to a port, outgoing frames should be sent without a tag (static access port). If, however, there is a further switch at this port, the frame should have a tag added (trunk port).
- With a trunk port, the VLAN assignment is dynamic. Static configurations can only be created if, in addition to the trunk port property, the port is also entered statically as a member in the VLANs involved. An example of a static configuration is the assignment of the multicast groups in certain VLANs.

Settings

- **VLAN ID**
  Enter the VLAN ID.

The table has the following columns:

- **VLAN ID**
  Shows the VLAN ID. The VLAN ID is assigned once when you create a new data record and can then no longer be changed. To make a change, the entire data record must be deleted and created again.
- **Name**
  Enter a name for the VLAN. The name only provides information and has no effect on the configuration.
  The VLAN name can be a maximum of 32 characters long.

- **Status**
  Shows the status of the entry. Here, static means that the address was entered as a static address by the user. The entry GVRP means that the configuration was registered by a GVRP frame. This is, however, only possible if GVRP was enabled for the device.
- **List of ports**
  Specify the use of the port. The following options are available:
  - "-"
  - The port is not a member of the specified VLAN.
    With a new definition, all ports have the identifier "-".
  - M
    The port is a member of the VLAN. Frames sent in this VLAN are forwarded with the corresponding VLAN tag.
  - R
    The port is a member of the VLAN. A GVRP frame is used for the registration.
  - U (upper case)
    The port is an untagged member of the VLAN. Frames sent in this VLAN are forwarded without the VLAN tag. Frames without a VLAN tag are sent from this port.
  - u (lower case)
    The port is an untagged member of the VLAN, but the VLAN is not configured as a port VLAN. Frames sent in this VLAN are forwarded without the VLAN tag.
  - F
    The port is not a member of the specified VLAN and it is not possible for the VLAN to be registered dynamically at this port using GVRP. You can configure other settings in "Layer 2 > VLAN > Port-based VLAN".

GVRP

Configuration of GVRP functionality

Using GVRP frame, a different device can register at the port of the device for a specific VID. A different device, can, for example be an end device or a switch. The device can also send GVRP frames via this port.

Settings

- **GVRP**
  Enable or disable the "GVRP" function.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **Setting**
  Select the setting. You have the following setting options:
  - enabled
    Enables the sending of GVRP frames.
  - disabled
    Disables the sending of GVRP frames.
  - No Change
    No change in table 2.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2. If you have selected "No Change", the content of table 2 will not be changed.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Enable or disable the sending GVRP frames.

Port-based VLAN

Processing received frames

On this page, you specify the configuration of the port properties for receiving frames.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Priority / Port VID / Acceptable Frames / Ingress Filtering**
  Select the setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports and link aggregations.
- **Priority**
  Select the priority assigned to untagged frames.
  The CoS priority (Class of Service) used in the VLAN tag. If a frame is received without a tag, it will be assigned this priority. This priority specifies how the frame is further processed compared with other frames.
  There are a total of eight priorities with values 0 to 7, where 7 represents the highest priority (IEEE 802.1p Port Priority).
- **Port VID**
  Select the VLAN ID. Only VLAN IDs defined on the "VLAN > General" page can be selected.
  If a received frame does not have a VLAN tag, it has a tag with the VLAN ID specified here added to it and is sent according to the rules at the port.
- **Acceptable Frames**
  Specify which types of frames will be accepted. The following alternatives are possible:
  - Tagged Frames Only
    The device discards all untagged frames. Otherwise, the forwarding rules apply according to the configuration.
  - All
    The device forwards all frames
- **Ingress Filtering**
  Specify whether the VID of received frames is evaluated.
  The following options are available:
  - enabled
    The VLAN ID of received frames decides whether they are forwarded: To forward a VLAN tagged frame, the receiving port must be a member in the same VLAN. Frames from unknown VLANs are discarded at the receiving port.
  - disabled
    All frames are forwarded.

Protocol-based VLAN group

On this page, you specify groups to which a protocol will be assigned.

Settings

- **Protocol Based VLAN**
  Enable or disable the protocol-based VLAN assignment.
- **Protocol Value**
  Enter the hexadecimal protocol value.
  A few examples are shown below:
  - PROFINET: 88:92
  - IP: 08:00
  - Novell: 81:37
  - netbios: f0:f0
  - appletalk: 80:9b
- **Group ID**
  Enter the ID of the group.

The table has the following columns:

- **Protocol Value**
  Shows the file value.
- **Group ID**
  Shows the group ID.

Protocol-based VLAN port

On this page, you specify which protocol and which VLAN is assigned to the individual port.

Settings

- **Port**
  Select the required port. All available ports and the link aggregations can be selected.
- **Group ID**
  Select the group ID from the drop-down list. Specify the ID in "Layer 2 > VLAN > Protocol Based VLAN Group".

The table has the following columns:

- **Port**
  All available ports and the link aggregations are shown.
- **Group ID**
  Shows the group ID assigned to the port.
- **VLAN ID**
  Select the required VLAN ID to be assigned to the port.

IPv4 subnet-based VLAN

On this page, you specify which VLAN ID is assigned to the subnet.

Settings

- **Subnet Based VLAN**
  Enable or disable the subnet-based VLAN assignment
- **Port**
  Select the port. All available ports and the link aggregations can be selected.
- **Subnet Address**
  Enter the IP address of the subnet.
  Example: 192.168.10.0 for the network 192.168.10.x with nodes 192.168.10.1 to 192.168.10.254.
- **Subnet Mask**
  Enter the subnet mask.

The table has the following columns:

- **Port**
  All available ports and the link aggregations are shown.
- **Subnet Address**
  Shows the subnet assigned to the port.
- **Subnet Mask**
  Shows the subnet mask.
- **VLAN ID**
  Select the VLAN ID you want to assign to the port or the subnet.

Mirroring

Basics

Mirroring

The device provides the option of simultaneously channeling incoming or outgoing data streams via other interfaces for analysis or monitoring. This has no effect on the monitored data streams. This procedure is known as mirroring. In this menu section, you enable or disable mirroring and set the parameters.

Mirroring ports

Mirroring a port means that the data traffic at a port (mirrored port) of the IE switch is copied to another port (monitor port). You can mirror one or more ports to a monitor port.

If a protocol analyzer is connected to the monitor port, the data traffic at the mirrored port can be recorded without interrupting the connection. This means that the data traffic can be investigated without being affected. This is possible only if a free port is available on the device as the monitor port.

General

On this page, you can enable or disable the mirroring function and make the basic settings.

Note

If the maximum data rate of the mirrored port is higher than that of the monitor port, data may be lost and the monitor port no longer reflects the data traffic at the mirrored port. Several ports can be mirrored to one monitor port at the same time.

Mirroring a port does not work beyond switch core boundaries.

Disable port mirroring if you want to connect a normal end device to the monitor port.

Settings

- **Mirroring**
  Enable or disable mirroring of the data traffic.
- **Monitor Barrier**
  Enable or disable the option to restrict communication via the monitor port.

  Note

  Monitor Barrier

  If you enable the monitor barrier, the data traffic on the destination port is automatically blocked (broadcast, multicast, unicast, DCP forwarding, LLDP) so that only the mirrored traffic is present. To be able to allow other data traffic again, you need to configure this. The previous statuses of these options are not restored after stopping the monitor barrier and must be reconfigured.

  – enabled
    The monitor port is taken out of normal frame switching.
  – disabled
    Communication via the monitor port is unrestricted.

This table has the following columns:

- **Session ID**
  Enable or disable listening in on incoming packets at the required port.
- **Session Type**
  Specify which data traffic is mirrored. The following options are available:
  - '-'
    None
  - Port Based
    Port-based mirroring You can configure other settings in "Layer 2 > Mirroring > Port".
  - VLAN
    VLAN-based mirroring. You can configure other settings in "Layer 2 > Mirroring > VLAN".
  - MAC ACL
    Mirroring of the MAC Access Control List. You can configure other settings in "Layer 2 > Mirroring > MAC Flow".
  - IP ACL
    Mirroring of the IP Access Control List. You can configure other settings in "Layer 2 > Mirroring > IP Flow".
- **Status**
  Shows whether or not mirroring is active.
- **Dest. Port**
  Select the destination port to be mirrored to in this session.

Port

Mirroring ports

You can only configure the settings on this page if you have already generated a session ID with the session type "Port Based" on the "General" tab.

Settings

- **Session ID**
  Select the session ID. Only session IDs of the session type "Port Based" are available.

The table has the following columns:

- **Port**
  Shows the port to be monitored.
- **Ingress Mirroring**
  Enable or disable mirroring of incoming packets at the required port.
- **Egress Mirroring**
  Enable or disable mirroring of outgoing packets at the required port.

VLAN

VLAN sources of the port mirroring function

You can only configure the settings on this page if you have already generated a session ID with the session type "VLAN" on the "General" tab.

On this page, you specify the VLAN whose incoming data traffic will be mirrored to the monitor port.

Settings

- **Session ID**
  Select the session ID. Only session IDs of the session type "VLAN" are available.
- **VLAN ID**
  Enter a VLAN ID. The VLAN ID can only be assigned once when you create a new data record and can then no longer be changed. To make a change, the entire data record must be deleted and created again.

The table has the following columns:

- **VLAN ID**
  Shows the VLAN ID.

MAC Flow

You can only configure the settings on this page if you have already generated a session ID with the session type "MAC ACL" on the "General" tab.

The MAC ACL filter decides which data is available at the monitor port.

Settings

- **Session ID**
  Select the session ID. Only session IDs of the session type "MAC ACL" are available.

The table has the following columns:

- **ACL Filter Number**
  Shows the number of the ACL filter. You configure the MAC ACL filter in "SecurityPort ACL MAC".
- **Ingress Mirroring**
  Shows whether incoming packets are mirrored.

  Note

  Rules

  A rule selected for ingress mirroring only becomes active if it was configured as a port ingress rule on at least one port. You configure the port ingress rules in "Security> Port MAC IP > Port Ingress Rules".

- **Source MAC**
  Shows the MAC address of the sender.

- **Dest. MAC**
  Shows the MAC address of the recipient.
- **Ingress Port**
  Shows a list of all ports to which this rule applies.
- **Egress Port**
  Shows a list of all ports to which this rule applies.

IP Flow

You can only configure the settings on this page if you have already generated a session ID with the session type "IP ACL" on the "General" tab.

The IP ACL filter decides which data is mirrored to the monitor port.

Settings

- **Session ID**
  Select the session ID. Only session IDs of the session type "IP ACL" are available.

The table has the following columns:

- **ACL Filter Number**
  Shows the number of the ACL filter. You configure the IP ACL filter in "Security > Port ACL IP".
- **Ingress Mirroring**
  Shows whether incoming packets are mirrored.

  Note

  Rules

  A rule selected for ingress mirroring only becomes active if it was configured as a port ingress rule on at least one port. You configure the port ingress rules in "Security> Port ACL IP > Port Ingress Rules".

- **Source IP**
  Shows the IP address of the sender.
- **Source Subnet Mask**
  Shows the subnet mask of the sender.
- **Dest. IP**
  Shows the IP address of the recipient.
- **Dest. Subnet Mask**
  Shows the subnet mask of the recipient.
- **Ingress Port**
  Shows a list of all ingress ports to which this rule applies.
- **Egress Port**
  Shows a list of all egress ports to which this rule applies.

Dynamic MAC Aging

The device automatically learns the source addresses of the connected nodes.

This information is used to forward data frames to the nodes specifically involved. The network load for the other nodes is reduced.
If a device does not receive a frame whose source address matches a learnt address within a certain time, the learnt address is deleted. This mechanism is known as aging. Aging prevents frames being forwarded incorrectly, for example when an end device (for example a programming device) is connected to a different port. If the check box is not enabled, a device does not delete learned addresses automatically.

Settings

- **Dynamic MAC Aging**
  Enable or disable the function for automatic aging of learned MAC addresses:
- **Aging Time [s]**
  Enter the time in seconds. After this time, a learned address is deleted if the device does not receive any further frames from this sender address.

Ring Redundancy

Ring

On this page, you can select the required mode for fast ring redundancy. The "Automatic Redundancy Detection" ring redundancy mode is the default when the device ships.

Note

The ring redundancy cannot be enabled if Spanning Tree is still enabled on the device.

Settings

- **Ring Redundancy**
  Enable or disable ring redundancy.
- **Ring redundancy mode**
  Specify the ring redundancy mode. The following options are available:
  - **"-" (disabled)**
    The redundancy function is disabled.
  - **Automatic Redundancy Detection**
    Select this setting to create an automatic configuration of the redundancy mode. In this mode, the device automatically detects whether or not there is a device with the "HRP Manager" role in the ring. If this is the case, the device adopts the role of "HRP Client". If no HRP manager is found, all devices with the "Automatic Redundancy Detection" or "MRP Auto Manager" setting negotiate among themselves to establish which device adopts the role of "MRP Manager". The device with the lowest MAC address will always become "MRP Manager". The other devices automatically set themselves to redundancy mode "MRP Client".
  - **MRP Auto Manager**
    Devices with the setting "Automatic Redundancy Detection" or "MRP Auto Manager" negotiate among themselves which device will adopt the "MRP Manager" role. The device with the lowest MAC address will always become "MRP Manager". In contrast to the setting "Automatic Redundancy Detection", the devices are not capable of detecting whether or not an HRP manager is in the ring. This means that they never adopt the role of HRP client.
  - **MRP Client**
    In a ring whose devices are configured with MRP, at least one device must be set to the "Automatic Redundancy Detection" or "MRP Auto Manager" mode. You can set the "MRP Client" role for all other devices. If all devices except one are configured as "MRP Client", this one device automatically adopts the role of "MRP Manager".
    Select "MRP Client" mode if you want to operate the device along with components that do not originate from Siemens in the ring.
  - **HRP Client**
    Here, you can select the role "HRP Client".
  - **HRP Manager**
    When you configure an HRP ring, one device must be set as HRP manager. All other devices must be configured as HRP clients.
- **Ring ports**
  Specify the ports to be used as ring ports in media redundancy in ring topologies.
  The ring port you select in the upper drop-down menu is the "Isolated Port" in HRP.

Note

If you restore the factory defaults, the default redundancy mode "Automatic Redundancy Detection" becomes active.

The ring port configuration is also reset to the factory default ports. If other ports were used previously as ring ports, with the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

- **Observer**
  Enable or disable the observer. The "Observer" function is only available in HRP rings. The ring port selected in the upper drop-down menu is connected to the "Isolated Port" of an HRP manager.
  The observer monitors malfunctions of the redundancy manager or incorrect configurations of an HRP ring.
  If the observer is enabled, it can interrupt the connected ring if errors are detected. To do this, the observer switches a ring port to the "blocking" status. When the error is resolved, the observer enables the port again.
- **Restart Observer** (only available online)
  If numerous errors occur in quick succession, the observer no longer enables its port automatically. The ring port remains permanently in the "blocking" status. This is signaled by the error LED and a message text.
  After the errors have been eliminated, you can enable the port again using the "Restart Observer" button.

Standby

Standby manager

The standby manager allows the redundant linking of two HRP rings. To do this, two neighboring devices within a ring must be configured as standby partners. Enable the standby manager for both standby partners and select the port via which the device is connected to the ring you want to link to. For the "Standby Connection Name", a name unique within the ring must be assigned for both partners. This identifies the two modules as standby partners that belong together.

Note

To be able to use the function, HRP must be activated.

Settings

- **Standby**
  Enable or disable the standby manager.

  Note

  If two devices are linked by the standby function, the "Standby" function must be enabled on both devices.

  Note

  The standby manager always requires an activated HRP client.

- **Standby Connection Name**
  Enter the name for the standby connection. This name defines the master/slave device pair. Both must be located in the same ring. This is achieved by entering the same name on two devices in the ring.

  Note

  Make sure that the standby name (for one pair of devices) is used only once in the network.

- **Force device to standby master**
  When selected, the device is configured as standby master regardless of its MAC address.
  - If the setting is enabled for neither of the two devices for which the standby function is enabled, then assuming that no error has occurred, the device with the higher MAC address adopts the role of standby master.
  - If the setting is enabled for both devices or if the setting is only supported by one device, the standby master is also selected based on the MAC address.

  This type of assignment is important in particular when a device is replaced. Depending on the MAC addresses, the previous device with the slave function can take over the role of the standby master.

This table has the following columns:

- **Port**
  Shows the port to which the setting relates.
- **Setting**
  Here, you specify which ports are standby ports. The standby ports are involved in the redirection of data traffic.
  If there are no problems, only the standby ports of the master are enabled and handle to the data traffic into the connected HRP rings (buses). If the master or the Ethernet connection (link) of one of the standby ports of the master fails, all standby ports of the master will be disabled and the standby ports of the slave enabled. As a result, a functioning Ethernet connection to the connected network segments (HRP rings/linear buses) is restored.

Spanning Tree

General

General settings of MSTP

On this page, you configure the settings for MSTP. As default, Rapid Spanning Tree is enabled that can be set to the MSTP, RSTP or STP compatible mode with a switch.

On the configuration pages of these functions, you can make detailed settings.

Depending on the compatibility mode, you can configure the corresponding function on the relevant configuration page.

Settings

- **Spanning Tree**
  Enable or disable Spanning Tree.
- **Protocol Compatibility**
  Select the compatibility mode of Spanning Tree. If, for example, you select RSTP, Spanning Tree behaves like RSTP.
  The following settings are available:
  - STP
  - RSTP
  - MSTP

Using a link aggregation in an MSTP instance

If you want to use a link aggregation in an MSTP instance, follow the steps below during configuration:

1. Create a link aggregation in "Layer 2" > "Link aggregation (Page 1037)".
2. Create an MSTP instance in "Layer 2" > "Spanning Tree" > "MST General (Page 1032)".
3. Configure the link aggregation in "Layer 2" > "Spanning Tree" > "MST port (Page 1033)".

Automatic activation of MRP in redundant topologies

If you connect SCALANCE X switches with redundant network structures in the topology view, MRP is activated automatically on the switches involved.

If there is an existing configuration with other redundancy mechanisms, such as MSTP, this is automatically deactivated.

CIST general

On this page, you configure CIST.

Settings

- **Bridge Priority / Root Priority** (only available online)
  Which device becomes the root bridge is decided based on the bridge priority. The bridge with the highest priority becomes the root bridge. The lower the value, the higher the priority. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames.
- **Bridge Address / Root Address**(only available online)
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Port** (only available online)
  Shows the port via which the switch communicates with the root bridge.
- **Root Cost** (only available online)
  The path costs from this device to the root bridge.
  In MSTP mode, path costs up to the root bridge are shown.
- **Topology Changes** / **Last Topology Change** (only available online)
  The entry for the device shows the number of reconfigurations due to the spanning tree mechanism since the last startup. For the root bridge, the time since the last reconfiguration is displayed as follows:
  - Seconds: sec unit after the number
  - Minutes: min unit after the number
  - Hour: hr unit after the number
- **Bridge Hello Time / Root Hello Time** (only available online)
  Each bridge regularly sends configuration frames (BPDUs). The interval between two such frames is the Hello time. The default for this parameter is 2 seconds.
- **Bridge Forward Delay / Root Forward Delay**
  New configuration data is not used immediately by a bridge but only after the period specified in the parameter. This ensures that operation is only started with the new topology after all the bridges have the required information.
- **Bridge Max Age / Root Max Age** (only available online)
  Bridge Max Age defines the maximum "age" of a received BPDU for it to be accepted as valid by the switch.
- **Bridge Max Hop Count**
  This parameter specifies how many MSTP nodes a BPDU may pass through. If an MSTP BPDU is received and has a "Bridge Max Hop Count" that exceeds the value configured here, it is discarded.
- **Regional root priority** (only available online)
  For a description, see Bridge Priority / Root Priority
- **Regional root address** (only available online)
  The MAC address of the device.
- **Regional root costs** (only available online)
  Shows the path costs from this device to the regional root bridge.

- **Region Name**
  Enter the name of the MSTP region to which this device belongs. As default, the MAC address of the device is entered here. This value must be the same on all devices that belong to the same MSTP region.
- **Region Version**
  Enter the version number of the MSTP region in which the device is located. This value must be the same on all devices that belong to the same MSTP region.
- **Reset Counter** (only available online)
  Click this button to reset the counters on this page to zero.

CIST port

MSTP-CIST port configuration

When the page is called, the table displays the current status of the configuration of the port parameters.

To configure them, click the relevant cells in the port table.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **MSTP Status**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **MSTP Status**
  Specify whether or not the port is integrated in the spanning tree.

  Note

  If you disable the "MSTP Status" option for a port, this may cause the formation of loops. The topology must be kept in mind.

- **Priority**
  Enter the priority of the port. The priority is only evaluated when the path costs are the same. The value must be divisible by 16. If the value that cannot be divided by 16, the value is automatically adapted.
- **Cost Calc.**
  Enter the path cost calculation. If you enter the value "0" here, the automatically calculated value is displayed in the "Path Costs" box.

- **Path Costs** (only available online)
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected as the path. If several ports of a device have the same value for the path costs, the port with the lowest port number is selected.
  If the value for "Cost Calc." is "0", the automatically calculated value is displayed. Otherwise, the value of the "Cost Calc." box is displayed. The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000

  The values can, however, also be set individually.
- **Status** (only available online)
  Displays the current status of the port. The values are only displayed and cannot be configured. The "Status" parameter depends on the configured protocol. The following is possible for status:
  - Disabled

    The port only receives and is not involved in STP, MSTP and RSTP.
  - Discarding
    In the "Discarding" mode, BPDU frames are received. Other incoming or outgoing frames are discarded.
  - Listening

    In this status, BPDUs are both received and sent. The port is involved in the spanning tree algorithm.
  - Learning

    Stage prior to the forwarding status, the port is actively learning the topology (in other words, the node addresses).
  - Forwarding

    Following the reconfiguration time, the port is active in the network; it receives and forwards data frames.

- **Fwd. Trans** (only available online)
  Specifies the number of changes from the "Discarding" status to the "Forwarding" status
- **Edge type**
  Specify the type of edge port. You have the following options:
  - "-"
    Edge port is disabled. The port is treated as a "no edge port".
  - Admin
    Select this option when there is always an end device on this port. Otherwise a reconfiguration of the network will be triggered each time a connection is changed.
  - Auto
    Select this option if you want a connected end device to be detected automatically at this port. When the connection is established the first time, the port is treated as a "no Edge Port".
  - Admin/Auto
    Select these options if you operate a combination of both on this port. When the connection is established the first time, the port is treated as an Edge Port.
- **Edge** (only available online)
  Shows the status of the port.
  - Enabled

    An edge port is connected to this port.
  - Disabled
    There is a spanning tree or rapid spanning tree device at this port.

  With an end device, a switch can change over the port faster without taking into account spanning tree frames. If a spanning tree frame is received despite this setting, the port automatically changes to the "Disabled" setting for switches.

- **P.t.P. Type**
  Select the required option. The selection depends on the port that is set.
  - "-"
    Point to point is calculated automatically. If the port is set to half duplex, a point-to-point link is not assumed.
  - P.t.P.

    Even with half duplex, a point-to-point link is assumed.
  - Shared Media

    Even with a full duplex connection, a point-to-point link is not assumed.

    Note

    Point-to-point connection means a direct connection between two devices. A shared media connection is, for example, a connection to a hub.

- **Hello Time**
  Enter the interval after which the bridge sends configuration BPDUs

  Note

  The port-specific setting of the Hello time is only possible in MSTP compatible mode.

MST General

Multiple Spanning Tree configuration

With MSTP, in addition to RSTP, several VLANs can be managed in a LAN with separate RSTP trees.

Settings

- **MSTP Instance ID**
  Enter the number of the MSTP instance.

The table has the following columns:

- **MSTP Instance ID**
  Shows the number of the MSTP instance.
- **Root Address**
  Shows the MAC address of the root bridge
- **Root Priority**
  Shows the priority of the root bridge.
- **Bridge Priority**
  Enter the bridge priority. The value for the bridge priority is a whole multiple of 4096.
- **VLAN ID**
  Enter the VLAN ID. Here, you can also specify ranges with Start ID, "-", End ID. Several ranges or IDs are separated by ",".

MST port

Configuration of the Multiple Spanning Tree port parameters

On this page, you set the parameters for the ports of the configured multiple spanning tree instances.

Settings

- **MSTP Instance ID**
  Select the ID of the MSTP instance.

Table 1 has the following columns

- **1st column**
  Shows that the setting is valid for all ports.
- **MSTP Status**
  Select the setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2

Table 2 has the following columns:

- **Port**
  Shows all available ports and link aggregations.
- **MSTP Instance ID**
  ID of the MSTP instance.
- **MSTP Status**
  Enable or disable MSTP for this port.
- **Priority**
  Enter the priority of the port. The priority is only evaluated when the path costs are the same. The value must be divisible by 16. If the value that cannot be divided by 16, the value is automatically adapted.
- **Cost Calc.**
  Enter the path cost calculation in the input box. If you enter the value "0" here, the automatically calculated value is displayed in the next box "Path Costs".

- **Path Costs** (only available online)
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected as the path. If several ports of a device have the same value for the path costs, the port with the lowest port number is selected.
  If the value for "Cost Calc." is "0", the automatically calculated value is displayed. Otherwise, the value of the "Cost Calc." box is displayed. The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000

  The values can, however, also be set individually.
- **Status** (only available online)
  Displays the current status of the port. The values are only displayed and cannot be configured. The "Status" parameter depends on the configured protocol. The following is possible for status:
  - Disabled

    The port only receives and is not involved in STP, MSTP and RSTP.
  - Discarding
    In the "Discarding" mode, BPDU frames are received. Other incoming or outgoing frames are discarded.
  - Listening

    In this status, BPDUs are both received and sent. The port is involved in the spanning tree algorithm.
  - Learning

    Stage prior to the forwarding status, the port is actively learning the topology (in other words, the node addresses).
  - Forwarding

    Following the reconfiguration time, the port is active in the network; it receives and forwards data frames.
- **Fwd. Trans** (only available online)
  Specifies the number of changes from the "Discarding" status to the "Forwarding" status

Enhanced Passive Listening Compatibility

Enabling the function

On this page, you enable the expanded compatibility for passive listening.

Settings

- **Enhanced Passive Listening Compatibility**
  Enable or disable this function for the entire device.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Displays the port of the device.
- **Setting**
  - enabled
    Enables the function for all ports of the device.
  - disabled
    Disables the function for all ports of the device.

Loop detection

With the "Loop Detection" function, you specify the ports for which loop detection will be activated. The ports involved send special test frames - the loop detection frames. If these frames are sent back to the device, there is a loop.

A "local loop" involving this device means that the frames are received again at a different port of the same device. If the sent frames are received again at the same port, there is a "remote loop" involving other network components.

Note

A loop is an error in the network structure that needs to be eliminated. The loop detection can help to find the errors more quickly but does not eliminate them. The loop detection is not suitable for increasing network availability by deliberately including loops.

Note

Note that loop detection is only possible at ports that were not configured as ring ports or standby ports.

Settings

- **Loop Detection**
  Enable or disable loop detection.
- **VLAN Loop Detection**
  Enable or disable loop detection in a VLAN.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2
- **Threshold / Remote Reaction / Local Reaction**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Specify how the port handles loop detection frames.

  Note

  Test frames create additional network load. We recommend that you only configure individual switches, for example at branch points of the ring, as "sender" and the others as "forwarder".

  – sender
    Loop detection frames are sent out and forwarded.
  – forwarder
    Loop detection frames from other devices are forwarded.
  – blocked
    The forwarding of loop detection frames is blocked.
- **Threshold**
  Specify the number of received loop detection frames as of which a loop is assumed.
- **Remote Reaction**
  Specify how the port will react if a remote loop occurs. Select one of the two options from the drop-down list:
  – no action
    A loop has no effect on the port.
  – disable
    The port is blocked.

- **Local Reaction**
  Specify how the port will react if a local loop occurs. Select one of the two options from the drop-down list:
  - no action
    A loop has no effect on the port.
  - disable
    The port is blocked
- **Status** (only available online)
  This box shows whether loop detection is enabled or disabled for this port.
- **Source Port** (only available online)
  Shows the output port of the loop detection frame that triggered the last reaction
- **Source VLAN** (only available online)
  Shows the VLAN ID of the loop detection frame that triggered the last reaction.
  This is only possible if "VLAN Support Enabled" was selected earlier on the "Loop Detection Configuration" page.
- **Reset** (only available online)
  After a loop in the network has been eliminated, click this button to reset the port again

Link aggregation

Bundling network connections for redundancy and higher bandwidth

A link aggregation according to IEEE 802.3ad allows several connections between neighboring devices to be bundled to achieve higher bandwidths and to protect against failure.

Ports on both partner devices are included in link aggregations and the devices are then connected via these ports. To assign ports (in other words links) correctly to a partner device, the Link Aggregation Control Protocol (LACP) from the IEEE 802.3ad standard is used.

Up to 8 link aggregations can be defined. A maximum of 8 ports can be assigned to each link aggregation.

You can also use link aggregations within an MSTP instance, see "Layer 2" > "Spanning Tree" > "General (Page 1027)".

Settings

The table has the following columns:

- **Port**
  Shows the virtual port number of this link aggregation. This identifier is assigned internally by the firmware.
- **Link Aggregation Name**
  Enter the name for the link aggregation. This name can be specified by the user during configuration. The name is not absolutely necessary but can be useful to distinguish between the various link aggregations.
- **MAC Address**
  Shows the MAC address.

- **Status**
  Enable or disable link aggregation.
- **MTU**
  Specify the packet size.
- **LACP**
  - on
    Enables the sending of LACP frames.
  - off
    Disables the sending of LACP frames.
- **Frame Distribution**
  Set the type of distribution of packets on the individual links of an aggregation.
  - Destination&Source Mac
    The distribution is based on a combination of the destination and source MAC address.
  - Destination&Source IP-MAC
    The distribution is based on a combination of the destination and source IP address and MAC address.
- **VLAN Mode**
  Specify how the link aggregation is entered in a VLAN:
  - Hybrid
    The link aggregation sends tagged and untagged frames. It is not automatically a member of a VLAN.
  - Trunk
    The link aggregation only sends tagged frames and is automatically a member of all VLANs.
- **Port**
  Shows the ports that belong to this link aggregation. The following values can be selected from the drop-down list:
  - "-" (disabled)
    Link aggregation is disabled.
  - "a" (active)
    The port sends LACP frames and is only involved in the link aggregation when LACP frames are received.
  - "p" (passive)
    The port is only involved in the link aggregation when LACP frames are received.
  - "o" (on)
    The port is involved in the link aggregation and does not send any LACP frames.

  Note

  Within a "link aggregation", only ports with the following configuration are possible:
  - all ports with "o"
  - all ports with "a" or "p".

DCP forwarding

Applications

The DCP protocol is used by STEP 7 and the PST Tool for configuration and diagnostics. When shipped, DCP is enabled on all ports; in other words, DCP frames are forwarded at all ports. With this option, you can disable the sending of these frames for individual ports, for example to prevent individual parts of the network from being configured with the PST Tool or to divide the full network into smaller parts for configuration and diagnostics.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Specify whether the port should block or forward outgoing DCP frames. The following options are available:
  - forward
    DCP frames are forwarded via this port.
  - block
    No outgoing DCP frames are forwarded via this port. It is nevertheless still possible to receive via this port.

LLDP

Applications

PROFINET uses the LLDP protocol for topology diagnostics. In the default setting, LLDP is enabled for all ports; in other words, LLDP frames are sent and received on all ports. With this function, you have the option of enabling or disabling sending and/or receiving per port.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the port.
- **Setting**
  Specify the LLDP functionality. The following options are available:
  - Rx
    This port can only receive LLDP frames.
  - Tx
    This port can only send LLDP frames.
  - Rx & Tx
    This port can receive and send LLDP frames.
  - "-" (disabled)
    This port can neither receive nor send LLDP frames.

Unicast

Filter

Address filtering

This page shows the current content of the unicast filter table. This table lists the source addresses of unicast address frames. The displayed entries were made statically, in other words, the user set them.

On this page, you also define the static unicast filters.

Settings

- **VLAN ID**
  Select the VLAN ID in which you configure a new static MAC address. If nothing is set, "VLAN1" is set as the basic setting.
- **MAC Address**
  Enter the MAC address here.

This table contains the following columns:

- **VLAN ID**
  Shows the VLAN-ID assigned to this MAC address.
- **MAC Address**
  Shows the MAC address of the node that the device has learned or the user has configured.
- **Status**
  Shows the status of each address entry:
  - static
    Configured by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the switch is restarted.
- **Port**
  Shows the port via which the node with the specified address can be reached. Frames received by the device whose destination address matches this address will be forwarded to this port.

Note

You can only specify **one** port for unicast addresses.

Learning

Starting/stopping learning

Note

The page is only available if there is an online connection to the device.

With the automatic learning function, all connected devices are automatically entered in the unicast filter table. As long as the "Start learning" function is enabled, all learned unicast addresses are created immediately as static unicast entries.
The learning process is ended only after clicking the "Stop learning" button. With this method, learning can take a few minutes or several hours in larger networks before all nodes have really been learned. Only nodes that send packets during the learning phase are found.
By subsequently enabling the Port Lock function, only packets from the nodes known after the end of the learning phase (static unicast entries) will be accepted at the relevant ports.

Note

If the Port Lock function was already active on individual ports prior to the automatic learning phase, no addresses will be learned on these ports. This makes it possible to restrict learning to certain ports. To do this, first enable the Port Lock function of the ports that are not intended to learn addresses.

Settings

- **Start learning**
  Click the "Start learning" button to start the learning phase.
  The device now enters the addresses of connected devices until you stop the function.
- **Stop learning**
  Click the "Stop Learning" button to stop the learning phase.
  The learned entries are stored.
- **Deleting all static unicast addresses**
  Click the "Clear all static unicast addresses" button to delete all static entries.
  In large networks with numerous nodes, automatic learning may lead to a lot of undesired static entries. To avoid having to delete these individually, this button can be used to delete all static entries. This function is disabled during automatic learning.

Note

Depending on the number of entries involved, deleting may take some time.

Locked ports

Activating the access control

On this page, you can block individual ports for unknown nodes.

If the Port Lock function is enabled, packets arriving at this port from unknown MAC addresses are discarded immediately. Packets from known nodes are accepted by the port.
Since ports with the port lock function enabled cannot learn any MAC addresses, learned addresses on these ports are automatically deleted after the port lock function is enabled.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Enable or disable access the port lock function for the port.

Blocking

Blocking the forwarding of unknown unicast frames

On this page, you can block the forwarding of unknown unicast frames for individual ports.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Enable or disable the blocking of unicast frames

Note

Ring redundancy / standby

If ring redundancy or standby is enabled, the ports configured for this are not included in the blocking.

Multicast

Groups

Multicast applications

In the majority of cases, a frame is sent with a unicast address to a particular recipient. If an application sends the same data to several recipients, the amount of data can be reduced by sending the data using one multicast address. For some applications, there are fixed multicast addresses (NTP, IETF1 Audio, IETF1 Video etc.).

Reducing network load

In contrast to unicast frames, multicast frames represent a higher load for the device. Generally, multicast frames are sent to all ports. There are three ways of reducing the load caused by multicast frames:

- Static entry of the addresses in the multicast filter table.
- Dynamic entry of the addresses by listening in on IGMP parameter assignment frames (IGMP Configuration).
- Active dynamic assignment of addresses by GMRP frames.

The result of all these methods is that multicast frames are sent only to ports for which an appropriate address is entered.

"Multicast" shows the multicast frames currently entered in the filter table and their destination ports. The displayed entries were made statically, in other words, the user set them.

Settings

- VLAN ID
  Select the VLAN ID to be assigned to the MAC multicast address.
- MAC Address
  Here you enter a new MAC multicast address you want to configure.

The table has the following columns:

- VLAN ID
  Shows VLAN ID of the VLAN to which the MAC multicast address is assigned.
- MAC Address
  Shows the MAC multicast address that the device has learned or the user has configured.
- Status
  Shows the status of each address entry. The following information is possible:
  - static
    The address was entered statically by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the device is restarted. These must be deleted by the user.

IGMP

Function

IE switches support "IGMP snooping" and the IGMP querier function. If "IGMP snooping" is enabled, IGMP frames are evaluated and the multicast filter table is updated with this information. If "IGMP Querier is also enabled, IE switches also send IGMP queries that trigger responses from IGMP-compliant nodes.

IGMP Snooping Aging Time

In this menu, you can configure the aging time for IGMP Configuration. When the time elapses, entries created by IGMP are deleted from the address table if they are not updated by a new IGMP frame.

This applies to all ports; a port-specific configuration is not possible.

IGMP Snooping Aging Time depending on the querier

SCALANCE XR500 as IGMP querier:

If a SCALANCE XR500 is used as an IGMP querier, the query interval is 125 seconds. For the "IGMP Snooping Aging Time", set at least 250 seconds.

Other IGMP queriers

If a different IGMP querier is used, the value of the "IGMP Snooping Aging Time" should be at least twice as long as the query interval.

Settings

- **IGMP Snooping**
  Enable or disable IGMP (Internet Group Management Protocol). The function allows the assignment of IP addresses to multicast groups. If the check box is selected, IGMP entries are included in the table and IGMP frames are forwarded.
- **IGMP Snooping Aging Time**
  Enter the value for the aging time in seconds in this box.
- **IGMP Querier**
  Enable or disable "IGMP Querier". The device sends IGMP queries.

GMRP

Enabling GMRP

By selecting the check box, you specify whether or not GMRP is used for each individual port. If "GMRP" is disabled for a port, no registrations are made for it and it cannot send GMRP frames.

Settings

- **GMRP**
  Enable or disable the GMRP function.

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports and the link aggregations.
- **Setting**
- Enable or disable GMRP for the port or for the link aggregation.

Blocking

On this page, you can block the forwarding of unknown multicast frames for individual ports.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Enable or disable the blocking of multicast frames.

Broadcast

Blocking the forwarding of broadcast frames

On this page, you can block the forwarding of broadcast frames for individual ports.

Note

Some communication protocols work only with the support of broadcast. In these cases, blocking can lead to loss of data communication. Block broadcast only when you are sure that you do not need it.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  All available ports and the link aggregations are shown.
- **Setting**
  Enable or disable the blocking of broadcast frames.

PTP

General

Note

PTP is only available with SCALANCE X500.

The Precision Time Protocol (PTP) complying with IEEE 1588v2 allows the time-of-day synchronization of devices connected to the ports of a device. These devices forward the synchronization frames through the network using the "Transparent Clock" (TC) mechanism. The connection mechanisms "end-to-end" and "peer-to-peer" are supported.

To use IEEE 1588v2, enable this function and configure every port that is on the synchronization path as well as ports that are blocked due to redundancy mechanisms. IEEE 1588v2 can also be used with redundancy mechanisms in the ring such as HRP, standby linking of rings, MRP and RSTP.

Setting

- **1588 Mode**
  You can make the following settings:
  - **off**
    The device does not process any PTP messages. PTP messages are, however, forwarded according to the rules of the device.
  - **transparent**
    The device adopts the function of a transparent clock and forwards PTP messages to other nodes while at the same time making entries in the correction field of the PTP message.

TC general

1588 Transparent Clock

On this page, you specify the general settings for PTP.

Settings

- **Delay Mechanism**
  Specify the delay mechanism the device will work with:
  - End-to-end(delay request response mechanism will be used)
  - Peer-to-peer (peer delay mechanism will be used)
- **Domain Number**
  Enter the identification number for the time domain. Synchronization is only for the devices within the domain. The device ignores PTP messages with a different domain number. A SCALANCE device can only be assigned to one synchronization domain.

TC port

On this page, you specify the ports that can process PTP messages.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **Setting / Transport Mechanism**
  Select the setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Enable or disable PTP. When enabled, the port processes PTP messages.
- **Faulty Flag**
  Shows error status relating to PTP.
  - **true**
    An error occurred.
  - **false**
    No error has occurred on this port.
- **Transport Mechanism**
  Specifies the protocol for transferring the PTP messages. This protocol must also be supported by the communications partner of the port
  - Ethernet
  - UDP IPv4

RMON

Statistics

Statistics

On this page you can specify the ports for which statistics are displayed.

Settings

- **RMON**
  If you select this check box, Remote Monitoring (RMON) allows diagnostics data to be collected on the device, prepared and read out using SNMP by a network management station that also supports RMON. This diagnostic data, for example port-related load trends, allow problems in the network to be detected early and eliminated.
- **Port**
  Select the ports for which statistics will be displayed.

The table has the following columns:

- **1st column**
  Select the check box of the ports for which no more statistics will be displayed.
- **Port**
  Shows the ports for which statistics will be displayed.

History

Samples of the statistics

On this page, you can specify whether or not samples of the statistics are saved for a port. You can specify how many entries should be saved and at which intervals samples should be taken.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Entries**
  Enter the maximum number of samples to be stored at the same time. If "No Change" is entered, the entry in table 2 remains unchanged.
- **Interval [s]**
  Enter the interval after which the current status of the statistics will be saved as a sample. If "No Change" is entered, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the port to which the settings relate.
- **Setting**
  Enable the port for which the history will be displayed.
- **Entries**
  Enter the maximum number of samples to be stored at the same time.
- **Interval [s]**
  Enter the interval after which the current status of the statistics will be saved as a sample.

Configuring layer 3 functions

Configuration

The page contains the overview of the layer 3 functions of the device. To use the "Routing", "VRRP", "OSPF" and "RIP" functions, the device requires a KEY-PLUG.

Settings

- **Routing** (only available on layer 3)
  Enable or disable the "Routing" function.
- **DHCP Relay Agent**
  Enable or disable the DHCP relay agent. You can configure other settings in "Layer 3 > DHCP Relay Agent".
- **VRRP** (only available on layer 3)
  Enable or disable routing using VRRP. To use VRRP, first enable the "Routing" function. You can configure other settings in "Layer 3 > VRRP".
- **OSPF** (only available on layer 3)
  Enable or disable routing using OSPF. To use OSPF , first enable the "Routing" function. You can configure other settings in "Layer 3 > OSPFv2".
- **RIP** (only available on layer 3)
  Enable or disable routing using RIP. To use RIO , first enable the "Routing" function. You can configure other settings in "Layer 3 > RIPv2".

Subnets

Overview

Subnet

The page shows the subnets for the selected interface. If more than one subnet is available on an interface, the first entry of this interface is of the address type "Primary". All other subnets have the address type "Secondary".

On this page, you can create further subnets "Layer 3 > Subnets > Overview". In "Layer 3 > Subnets > Configuration", you configure the settings of the created subnets.

Settings

The following settings are possible for subnets of the "Secondary" address type:

- **Interface**
  Select the interface on which you want to configure another subnet.

The table has the following columns:

- **Interface**
  Shows the interface.
- **TIA Interface**
  Shows whether or not the entry is a TIA interface.
- **Interface Name**
  Shows the name of the interface.
- **MAC Address**
  Shows the MAC address.

- **IP Address**
  Shows the IP address of the subnet.
- **Subnet Mask**
  Shows the subnet mask.
- **Address Type**
  Shows the address type. The following values are possible:
  - Primary
    The first IP address that was configured on an IP interface.
  - Secondary
    All other IP addresses that were configured on an interface.
- **IP Assignment Method**
  Shows how the IP address is assigned.
  - Static
    The IP address is static. You enter the IP settings in "IP Address" and "Subnet Mask".
  - Automatic (DHCP)
    The device obtains a dynamic IP address from a DHCP server.
- **Address Collision Detection Status**
  Shows the status of the "Detection of address collisions" function.
  - Idle
    The interface is not enabled and does not have an IP address.
  - Starting
    This status indicates the start-up phase. In this phase, the device initially sends a query as to whether the planned IP address already exists. If the address is not yet been assigned, the device sends the message that it is using this IP address as of now.
  - Conflict
    The interface is not enabled. The interface is attempting to use an IP address that is already been assigned.
  - Defending
    The interface uses a unique IP address. Another interface is attempting to use the same IP address.
  - Active
    The interface uses a unique IP address. There are no collisions.
  - Not supported
    The function for detection of address collisions is not supported.

Configuration

On this page, you configure the subnet. You create the subnet in "Layer 3 > Subnets > Overview".

Settings

- **Interface (Name)**
  Select the required interface.
- **Interface Name**
  Enter the name of the interface, e.g. eth0; P3.
- **MAC Address** (only available online)
  Shows the MAC address of the selected interface
- **DHCP**
  Enable or disable the DHCP client for this interface.
- **IP Address**
  Enter the IP address of the subnet. The IP address must not be used more than once.
- **Subnet Mask**
  Enter the subnet mask of the subnet you are creating. Subnets on different interfaces must not overlap.
- **Address Type**
  Shows the address type. The following values are possible:
  - Primary
    The first subnet of the interface.
  - Secondary
    All other subnets of the interface.
- **Address Collision Detection**
  Enable or disable the function.
  If new IP addresses become active in the network, this function checks whether this can result in address collisions.
  With this function, IP addresses that will be assigned twice can be detected.

  Note

  The function does not run a cyclic check.

- **TIA Interface**
  Enable or disable the setting.
  The following conditions apply to the TIA interface:
  - Only interfaces with the address type "Primary" can be enabled as the TIA interface.
  - There must only ever be one TIA interface.
  - There can only be one TIA interface.
  - A TIA interface is always a VLAN interface.

TIA interface and PROFINET interface

The IP address of the TIA interface is linked to the IP address located in the parameter group "Properties > General > PROFINET interface > Ethernet addresses".

Both IP addresses always have the same value. If you change the value of one IP address, the other IP address changes accordingly.

TIA Interface

The TIA interface is the interface via which all PROFINET functions of the IE switch are handled.

TIA interface is the name of the IP address for PROFINET from the perspective of the IE switch.

PROFINET interface

You make the IP setting of the PROFINET device in TIA under PROFINET interface.

PROFINET interface is the name of the IP address for PROFINET from the perspective of the TIA Portal.

Routes

Static route

On this page, you create the static routes. During automatic adaptations, static routes are not taken into account and need to be adapted manually.

Settings

- **Destination Network**
  Enter the network address of the destination.
- **Subnet Mask**
  Enter the corresponding subnet mask.
- **Gateway**
  Enter the IP address of the next gateway.
- **Metric**
  Enter the metric for the route. The metric corresponds to the quality of a connection, for example speed, costs.

The table has the following columns:

- **Destination Network**
  Shows the network address of the destination.
- **Subnet mask**
  Shows the corresponding subnet mask.
- **Gateway**
  Shows the IP address of the next gateway.
- **Interface** (only available online)
  Shows the interface of the route.
- **Metric**
  Enter the metric for the route. When creating the route, "not used" is entered automatically. The metric corresponds to the quality of a connection, for example speed, costs. If there are several equal routes, the route with the lowest metric value is used.
- **Status** (only available online)
  Shows the status of the route.

Route maps

General

Route maps

With route maps, you control how routing information is further processed. You can filter routing information and specify whether the information is further processed, modified or discarded.

Route maps operate according to the following principle:

- Routing information is compared with the filters of the route maps.
- The comparison is continued until the filters of a route map match the properties of an item of information.
- The information is then processed according to the route map settings:
  - The routing information is discarded.
  - The properties of the routing information are changed.

Settings

- **Name**
  Enter the name for the route map.
- **Sequence No**
  Enter a number for the route map.
  The number specifies the order in which the route maps are processed.

The table has the following columns:

- **Name**
  Shows the name of the route map.
- **Sequence No**
  Shows the sequence number of the route map.
- **Action**
  Specify what happens to the routing information that matches the settings of the route map:
  - permit
    The routing information is further processed according to the settings you make in the "Settings" tab.
  - deny
    The routing information is discarded.

Interface and value match

On this page, you specify whether or not the routing information for a route map is filtered according to interfaces or metric.

Settings

- **Route Map (name/seq. no.)**
  Select a route map.
  The created route maps are available to you.
- **Type**
  Select the basis for the filtering:
  - Interface
  - Metric
  - Tag
- **Interface**
  Select an interface.
  This box is active only if you have selected the "Interface" entry in the "Type" drop-down list.
- **Metric**
  Enter a value for the metric.
  This box is active only if you have selected the "Metric" entry in the "Type" drop-down list.
- **Tag**
  Enter a value for the tag.
  This box is active only if you have selected the Tag entry in the "Type" drop-down list.

The table has the following columns:

- **Type**
  Shows the selected type:
  - Interface
  - Metric
  - Tag
- **Value**
  Shows the selected interface or the value of the metric.

Destination match

On this page, you specify whether or not the routing information for a route map is filtered based on the destination IP address.

Settings

- **Route Map (name/seq. no.)**
  Select a route map.
- **IP Address**
  Enter the IP address of the destination on which the filtering is based.
- **Subnet Mask**
  Enter the subnet mask of the destination on which the filtering is based.

The table has the following columns:

- **IP Address**
  Shows the IP address of the destination.
- **Subnet mask/prefix**
  Shows the subnet mask of the destination.

Next hop match

On this page, you specify whether or not the filtering for a route map will be based on the router to which the routing information is sent next.

Settings

- **Route Map (name/seq. no.)**
  Select a route map.
- **IP Address**
  Enter the IP address of the router to which the routing information will be sent next.

The table has the following columns:

- **IP Address**
  Shows the IP address of the next router.

Settings

On this page, you specify whether or not the routing information for a route map will be changed.

If, for example, you have filtered based on a certain metric, you can change the value of the metric here. The routing information is then forwarded with the new value.

Settings

- **Route Map (name/seq. no.)**
  Select a route map.

The table has the following columns:

- **Name**
  Shows the name of the route map.
- **Sequence No**
  Shows the sequence number of the route map.
- **Metric**
  Enter the new value for the metric with which the routing information will be forwarded.
- **Tag**
  Enter the new value for the tag with which the routing information will be forwarded.

DHCP Relay Agent

General

If the DHCP server is in a different network, the device cannot reach the DHCP server. The DHCP relay agent intercedes between a DHCP server and the device. The DHCP relay agent forwards the port number of the device with the DHCP query to the DHCP server. If a DHCP server is unreachable, the device can switch to a different DHCP server.

Settings

- **DHCP Relay Agent (Opt. 82)**
  Enable or disable the DHCP relay agent.
- **Server IP Address**
  Enter the IP address of the DHCP server.

The table has the following columns:

- **Server IP address**
  Shows the IP address of the DHCP server.

Option

Parameters of the DHCP relay agent

On this page, you can specify parameters for the DHCP server, for example the circuit ID. The circuit ID describes the origin of the DHCP query, for example which port received the DHCP query.

You specify the DHCP server on the "General" tab.

Settings

Global configuration

- **Circuit ID router index**
  When enabled, the router index is added to the generated circuit ID.
- **Circuit ID recipient VLAN ID**
  When enabled, the VLAN ID is added to the generated circuit ID.
- **Circuit ID receiving port**
  When enabled, the receiving port is added to the generated circuit ID.

  Note

  You need to select a least one option.

- **Remote ID**
  Shows the device ID.

Interface-specific configuration

- **Interface**
  Select the required interface.

The table has the following columns:

- **Interface**
  Shows the selected interface.
- **Remote ID type**
  Select the type of the device ID. You have the following options:
  - IP Address
    The current IP address of the interface will be used as the device identifier and copied to the "Remote ID" box.

    Note

    No automatic updating

    There is no link between the "Remote ID" box and the currently set IP address.

    If the IP address is changed, the new IP address is not entered in the "Remote ID" box automatically. Only the value of the "Remote ID Type" box changes to "Free Text". To use the current IP address again, select "IP Address" again in the "Remote ID Type" box.

  - MAC Address
    The MAC address of the device is used as the device ID.
  - Free Text
    If you use "Free Text", you can enter the device name as the device identifier in the "Remote ID" box.
- **Remote ID**
  The box can only be edited if you select the entry "Free Text" for "Remote ID Type".
  The remote IP must be unique.
- **Circuit ID Type**
  Select the type of circuit ID from the drop-down list. You have the following options:
  - Predefined
    The circuit ID is created automatically based on the router index, VLAN ID or port.
  - Free Number
    If you use "Free Number", you can enter the ID for "Circuit ID".
- **Circuit ID**
  Enter the circuit ID.
  The box can only be edited if you select the "Free Number" entry for the "Circuit ID Type".

VRRP

Router

On this page, you create virtual routers. You can configure other parameters in "Layer 3 > VRRP > Configuration".

Note

- VRRP is available only on layer 3.
- VRRP can only be used in conjunction with VLAN interfaces. Router ports are not supported.
- Select "VRRP" to configure VRRP

Settings

The following settings are available:

- **VRRP**
  Enable or disable routing using VRRP.
- **React to pings on virtual interfaces**
  When enabled, the virtual IP addresses also reply to the ping.
- **Interface**
  Select the interface operating as virtual router.
- **VRID**
  Enter the ID of the virtual router in the input box. This ID defines the group of routers that form a virtual router (VR). In the group, this is the same. it can no longer be used for other groups.

The table has the following columns:

- **Interface**
  Shows the Interface that functions as the virtual router.
- **VRID**
  Shows the ID of the virtual router.
- **Virtual MAC Address**
  Shows the virtual MAC address of the virtual router.
- **Primary IP Address**
  Shows the primary IP address in this VLAN. The entry 0.0.0.0 means that the "Primary" address on this VLAN is used. Otherwise all IP addresses configured in this VLAN in "Layer 3 > Subnets" are valid values.

- **Router Status**
  Shows the current status of the virtual router. Possible values are as follows:
  - Master
    The router is the master router and handles the routing functionality for all assigned IP addresses.
  - Backup
    The router is the backup router. If the master router fails, the backup router takes over the tasks of the master router.
  - Initialize
    The virtual router has just been turned on. It will soon change to the "Master" or "Backup" status.
- **Master IP Address** (only available online)
  Shows the IP address of the master router.
- **Priority**
  Shows the priority of the virtual router.
  The current master router is automatically given 255. All other priorities can be distributed freely among the VRRP routers. The higher the priority, the earlier the VRRP router becomes "Master".
- **Advert. Internal**
  Shows the interval at which the master router sends VRRP packets.
- **Preempt**
  Shows the precedence of a router when changing roles between backup and master.
  - yes
    This router has precedence when changing roles.
  - no
    This router does not have precedence when changing roles.

Configuration

Introduction

On this page, you configure the virtual router.

Note

VRRP is available only on layer 3.

Description of the displayed values

The page contains the following boxes:

- **Interface / VRID**
  Select the ID of the virtual router to be configured.
- **Primary IP Address**
  Select the primary IP address. If the router becomes master router, the router uses this IP address.

  Note

  If you only configure one subnet on this VLAN, no entry is necessary. The entry is 0.0.0.0. If you configure more than one subnet on the VLAN and you want a specific IP address to be used as the source address for VRRP packets, select the IP address from the drop-down list. Otherwise, the IP address with priority will be used.

- **Master**
  If this option is enabled, the primary IP address is entered for "Associated IP Address". This means that the highest priority IP address of the VRRP router is used as the virtual IP address of the virtual master router. The option must be disabled for the backup routers in this group and the IP address of the router in "Assigned IP address" must be used.
- **Priority**
  Enter the priority of this virtual router.
  The current master router is always given 255. All other priorities can be distributed freely among the redundant routers. The higher the priority, the earlier the router becomes "Master".
- **Advertisement Interval [s]**
  Enter the interval in seconds after which a master router sends a VRRP packet again.
- **Preempt lower priority master**
  Allow the precedence when changing roles between Backup and Master based on the selection process.

Address overview

Overview

This page shows which IP addresses the virtual router monitors.

Note

VRRP is available only on layer 3.

Settings

The table has the following columns:

- **Interface**
  Shows the Interface that functions as the virtual router.
- **VRID**
  Shows the ID of this virtual router.
- **Number of Addresses**
  Shows the number of IP addresses.
- **Associated IP Address (1) ... Associated IP Address (4)**
  Shows the router IP addresses monitored by this virtual router. If a router takes over the role of master, the routing function is taken over by this router for all these IP addresses.

Address configuration

Note

VRRP is available only on layer 3.

Settings

The following settings are possible:

- **Interface / VRID**
  Select the required virtual router.
- **Associated IP Address**
  Enter the IP address that the virtual router will monitor.

The table has the following columns:

- **Associated IP Address**
  Shows the IP addresses that the virtual router monitors.

OSPFv2

Configuration

Introduction

On this page, you configure routing with OSPF.

Note

OSPF is available only on layer 3.

Settings

- **OSPFv2**
  Enable or disable routing using OSPF.
- **Router ID**
  Enter the name of one of the OSPF interfaces. The name is entered in the IP address format and does not need to match the local IP address.
- **OSPFv2 RFC1583 Compatibility**
  Enable the option if you still have old OSPF routers in operation that are not compatible with RFC 2328.
- **Border Router**
  Shows the status of the OSPF router. If the local system is an active member in at least 2 areas, this is an area border router.
- **AS Border Router**
  Specify whether the router is an AS border router. An AS border router intercedes between multiple autonomous systems, for example if you have an additional RIP network. An AS border router is also necessary to add and to distribute static routes.
- **New LSA Received** (only available online)
  Shows the number of received LSAs.
  Updates and local LSAs are not counted.
- **New LSA Configured** (only available online)
  Number of different LSAs sent by this local system.
- **External LSA Maximum**
  To limit the number of entries of external LSAs in the database, enter the maximum number of external LSAs.
- **Exit Interval [s]**
  Enter the interval after which the OSPF router once again attempts to come out of the overflow status. A 0 means that the OSPF router attempts to exit the overflow status only following a restart.
- **Inbound Filter**
  Select a route map that filters inbound routes.

- **Redistribute Routes**
  Specify which known routes are distributed using OSPF.
  The following types of route exist:
  - Default
  - Connected
  - Static
  - RIPv2

  Note

  The settings can only be enabled on an AS border router. Enabling "Default" and "Static", in particular, can cause problems if they are enabled at too many points in the network, for example, forwarding loops.

- **Route Map**
  Select a route map that filters which routes are forwarded using OSPF.

Areas

Overview

An autonomous system can be divided into smaller areas.

On this page, you can view, create, modify or delete the areas of the router.

Note

OSPF is available only on layer 3.

Settings

- **Area ID**
  Enter the ID of the area. The database is synchronized for all routers of an area.
  Input format: x.x.x.x
  x = 0 ... 255

This table contains the following columns:

- **Area ID**
  Shows the ID of the area.

- **Area Type**
  Select the area type from the drop-down list.
  - Normal
  - Stub
  - NSSA
  - Backbone
- **Summary**
  Specify whether summary LSAs are generated for this area.
  - Summary: Summary LSAs are sent to the area.
  - No Summary: Summary LSAs are not sent to the area.
- **Metric**
  Shows the costs of the default route for the area types "Stub" and "NSSA". The initial value depends on the assigned OSPF interface.
  - The metric can be edited for the area types "Stub" and "NSSA". It specifies the costs for the default route via which external networks can be reached.
  - The metric cannot be edited for the area type "Normal".

Area range

Creating a new OSPFv2 area range

On this page, networks can be grouped together under one area ID. The method is used only with area border routers. This means that an area border router only propagates one route for each address area to the outside.

Note

OSPF is available only on layer 3.

Settings

- **Area ID**
  Select the area ID. You configure the ID in "Layer 3 > OSPFv2 > Areas".
- **Subnet Address**
  Enter the address of the network that will be grouped.
- **Subnet Mask**
  Enter the subnet mask of the network that will be grouped.

This table contains the following columns:

- **Area ID**
  Shows the ID of the area.
- **Subnet Address**
  Shows the address of the network that will be grouped with other networks.

- **Subnet Mask**
  Shows the subnet mask of the network that will be grouped with other networks.
- **Advertise**
  Enable this option to advertise the grouped network.

Interfaces

Overview

On this page, you can configure OSPF interfaces.

Note

OSPF is available only on layer 3.

Settings

- **IP Address**
  Select the IP address of the OSPF interface.
- **Area ID**
  Select the area ID of the area with which the OSPF interface is connected.

  Note

  For secondary address types, select the same Area ID as for the corresponding primary address type.
  The information whether an address type is primary or secondary can be found in the "Address Type" column on the "Subnets > Overview" page.

This table contains the following columns:

- **IP Address**
  Displays the IP address of the OSPF interface.
- **Area ID**
  Select the area ID with which the OSPF interface is connected.
- **OSPF Status**
  Specify whether or not OSPF is active on the interface.
  - Enabled: OSPF is enabled on the interface.
  - Disabled: OSPF is disabled on the interface
- **Metric**
  Enter the costs for the OSPF interface.
- **Priority**
  Enter the router priority. The priority is only relevant for selecting the designated router. This parameter can be selected differently on routers within the same subnet.
- **Trans. Delay**
  Enter the required delay when sending a connection update.

- **Retrans. Delay**
  Enter the time after which an OSPF packet is transferred again if no confirmation was received.
- **Hello Interval**
  Enter the interval between two Hello packets.
- **Dead Interval**
  Enter the interval after which the neighbor router is marked as "failed" if no more Hello packets are received from it during this time.

Interface authentication

Configuring the interface login

On this page, you define the authentication of the interface.

Note

OSPF is available only on layer 3.

Settings

- **OSPF Interface**
  Select the OSPF interface for which you want to configure authentication.
- **Authentication type**
  Select the authentication method of the OSPF interface. You have the following options:
  - None: No authentication
  - simple: Authentication using an unencrypted password
  - MD5: Authentication using MD5

Simple authentication

- **Password**
  For "Simple authentication, enter the password if you have selected this type of authentication.
- **Confirmation**
  Confirm the entered password.

MD5 authentication

- **Authentication Key ID**
  If you have selected this type of authentication, enter the ID for the MD5 authentication under which the password is used as the key. Since the key ID is transferred with the protocol, the same key must be stored under the same key ID on all neighboring routers.

The table has the following columns:

- Authentication Key ID
  Can only be edited if you set the MD5 authentication method. It is only possible to use several keys there.
- MD5 Key
  Enter the MD5 key.
- Confirm MD5 Key
  Confirm the entered key.
- Youngest Key ID (only available online)
  Shows whether or not the MD5 key is the latest key ID.

Virtual links

Overview

Due to the protocol, each area border router must have access to the backbone area. If a router is not connected directly to the backbone area, a virtual link to it is created.

Note

OSPF is available only on layer 3.

Note

Note that when creating a virtual link both the transit area and the backbone area must already be configured.

the virtual link must be configured identically at both ends.

Note

- Virtual links are not effective unless the device is an ABR. (only available online)
  This is displayed when at least one virtual link entry is configured and the device is not an area border router.

Settings

- Neighbor Router ID
  Enter the ID of the neighbor router at the other end of the virtual connection.
- Transit Area ID
  Select the ID of the area that connects the two routers.

This table contains the following columns:

- Transit Area ID
  Shows the ID via which the two routers are connected.
- Neighbor Router ID
  Shows the ID of the neighbor router at the other end of the virtual connection.

- **Virt. Link Status**
  Specify the status of the virtual link. The following statuses are possible:
  - down: The virtual link is inactive.
  - point-to-point: The virtual link is active.
- **Trans. Delay**
  Enter the required delay when sending a link update packet (in seconds).
- **Retrans. Delay**
  Enter the time after which a packet is transferred again if no confirmation was received (in seconds).
- **Hello Interval**
  Enter the interval between two Hello packets (in seconds).
- **Dead Interval**
  Enter the interval after which the neighbor router counts as "failed" if no more Hello packets are received from it during this time (in seconds).

Virtual Link Authentication

Configuring the virtual link login

On this page, you define the authentication for the virtual link.

Note

OSPF is available only on layer 3.

Settings

- **Virtual link (area/neighbor)**
  Select the virtual link for which you want to configure authentication.
- **Authentication type**
  Select the authentication method of the OSPF interface. You have the following options:
  - None: No authentication
  - simple: Authentication using an unencrypted password.
  - MD5: Authentication using MD5

Simple authentication

- **Password**
  Enter the password for "simple authentication".
- **Confirmation**
  Confirm the entered password.

MD5 authentication

- **Authentication Key ID**
  Enter the ID for MD5 authentication with which the password will be used as a key. Since the key ID is transferred with the protocol, the same key must be stored under the same key ID on all neighboring routers.

The table has the following columns:

- **Authentication Key ID**
  Can only be edited if you set the MD5 authentication method. It is only possible to use several keys there.
- **MD5 Key**
  Enter the MD5 key.
- **Confirm MD5 Key**
  Confirm the entered key.
- **Youngest Key ID**
  Shows whether or not the MD5 key is the latest key ID.

RIPv2

Configuration

On this page, you configure routing with RIP.

Note

RIPv2 is available only on layer 3.

Settings

- **RIPv2**
  Enable or disable routing using RIPv2.
- **Inbound Filter**
  Select a route map that filters inbound routes.
- **Redistribute Routes**
  Specify which known routes are distributed using RIPv2.
  The following types of route exist:
  - Static Default
  - Connected
  - Static
  - OSPF
- **Route Map**
  Select a route map that filters which routes are forwarded using RIPv2.

Interfaces

Overview

On this page, you can configure RIPv2 interfaces.

Note

RIPv2 is available only on layer 3.

Settings

- **IP Address**
  Select the IP address of the RIPv2 interface.

This table contains the following columns:

- **IP Address**
  Displays the IP address of the RIPv2 interface.
- **Send Updates**
  Select the way in which updates are sent:
  - no send
    No updates are sent.
  - RIPv1
    Updates for RIPv1 are sent.
  - RIPv1-compat.
    RIPv2 updates are sent as broadcasts according to the rules of RIPv1.
  - RIPv2
    Updates for RIPv2 are sent as multicasts.
  - RIPv1 demand/RIPv2 demand
    RIP packets are sent only as a response to an explicit query.
- **Receive Updates**
  Select the form in which received updates are accepted:
  - no receive
    No updates are received.
  - RIPv1
    Only updates of RIPv1 are received.
  - RIPv2
    Only updates of RIPv2 are received.
  - RIPv1/v2
    Updates of RIPv1 and RIPv2 are received.
- **Default Metric**
  Enter the costs for the RIPv2 interface.

Configuring security functions

Passwords

Note

The page is only available if there is an online connection to the device.

Only the administrator can change the device passwords for administrator and users.

The factory settings for the passwords when the devices ship are as follows:

- Administrator: admin
- User: user

Note

If you log on the first time or log on after a "Reset to factory defaults and restart", you will be prompted to change the password.

Note

Changing the password in "Trial" configuration mode

Even if you change the password in "Trial" configuration mode, this change is saved immediately.

Settings

- **Current Admin Password**
  Enter the valid administrator password.
- **User name**
  Select the required user. The following options are available:
  - Administrator: admin
  - User: user
- **New Password**
  Enter the new password.
- **Password Confirmation**
  Confirm the new password.

AAA

General

The term "AAA" used in the menu stands for "Authentication, Authorization, Accounting" and is used to identify and allow network nodes, to provide them with the appropriate services and to determine the amount of use.

On this page, you configure the login.

Settings

- **Login Authentication**
  Specify how the login is made:
  - Local
    Login with local user name and password.
  - RADIUS
    Login using a RADIUS server.

RADIUS client

The term "AAA" used in the menu stands for "Authentication, Authorization, Accounting" and is used to identify and allow network nodes, to provide them with the appropriate services and to determine the amount of use.

Authentication over an external server

The concept of RADIUS is based on an external authentication server. An end device can only access the network after the device has verified the logon data of the device with the authentication server. Both the end device and the authentication server must support the EAP protocol (Extensive Authentication Protocol).

Each column of the table contains access data for one server. In the search order, the primary server is queried first. If the primary server cannot be reached, secondary servers are queried in the order in which they are entered.

If no server responds, there is no authentication. The client has no access to the network although a link is indicated at the port.

Settings

The table has the following columns:

- **Server IP Address**
  Enter the IP address of the RADIUS server.
- **Server Port**
  Here, enter the input port on the RADIUS server. As default, input port 1812 is set.
- **Shared Secret**
  Enter your access ID.
- **Confirm Shared Secret**
  Enter your access ID again as confirmation.
- **Max. Retrans**
  Enter the maximum number of query attempts before another configured RADIUS server is queried or the login counts as having failed. As default, 3 is set.

- **Primary Server**
  Specify whether or not this server is the primary server. You can select one of the options "yes" or "no".
- **Status**
  Enable or disable the RADIUS server.

Note

You can configure a maximum of three servers on this page.

802.1x-Authenticator

The term "AAA" used in the menu stands for "Authentication, Authorization, Accounting" and is used to identify and allow network nodes, to provide them with the appropriate services and to determine the amount of use.

Enabling authentication for individual ports

By selecting the check box, you specify whether or not network access protection according to IEEE 802.1x is enabled on this port.

Settings

- **MAC Authentication**
  Enable or disable MAC authentication for the device.
- **Guest VLAN**
  Enable or disable the "Guest VLAN" function for the device.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **802.1x Auth. Control**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **802.1x Re-authentication**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **MAC Authentication**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Adopt RADIUS VLAN Assignment**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **MAC Auth. Max Permitted Addresses**
  Enter how many end devices are allowed to be connected to the port at the same time. If "No Change" is entered, the entry in table 2 remains unchanged.

- **Guest VLAN**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Guest VLAN ID**
  Enter the VLAN ID of the port. If "No Change" is entered, the entry in table 2 remains unchanged.
- **Guest VLAN max. Permitted Addresses**
  Enter how many end devices are allowed in the guest VLAN at the same time.
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the port to which the setting relates.
  If a configuration is not possible for a port, it is displayed grayed out and you cannot modify the settings.
- **802.1x Auth. Control**
  Specify the authentication of the port:
  - Force Unauthorized
    Data traffic via the port is blocked.
  - Force Authorized
    Data traffic via the port is allowed without any restrictions.
    Default setting
  - Auto
    End devices are authenticated on the port with the "802.1x" method.
    The data traffic via the port is permitted or blocked depending on the authentication result.
- **802.1x Re-authentication**
  Enable this option if you want reauthentication of an already authenticated end device to be repeated cyclically.
- **MAC Authentication**
  Enable this option if you want end devices to be authenticated with the "MAC Authentication" method.
- **Adopt RADIUS VLAN Assignment**
  The RADIUS server informs the IE switch of the VLAN to which the port belongs.
  Enable this option if you want the information of the server to be taken into account. The port then belongs to the corresponding VLAN.
  If the option is disabled, the VLAN information is discarded.
- **MAC Auth. Max Permitted Addresses**
  Enter how many end devices are allowed to be connected to the port at the same time.
- **Guest VLAN**
  Enable this option if you want the end device to be permitted in the guest VLAN if authentication fails.
- **Guest VLAN ID**
  Enter the VLAN ID of the port.

- **Guest VLAN max. Permitted Addresses**
  Enter how many end devices are allowed in the guest VLAN at the same time.
- **MAC Auth. Currently Permitted Addresses**
  Shows the number of currently connected end devices.
- **MAC Auth. Currently Blocked Addresses**
  Shows the number of currently blocked end devices.
- **Guest VLAN Currently Permitted Addresses**
  Shows how many end devices are currently allowed in the guest VLAN.

Port ACL MAC

Rule configuration

On this page, you specify the ACL rules for the MAC-based ACL.

Settings

The table has the following columns:

- **Rule Number**
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- **Source MAC**
  Enter the unicast MAC address of the source.
  Dest. MAC
  Enter the unicast MAC address of the destination.
- **Action**
  Select the action. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- **Ingress Port**
  Shows a list of all ingress ports to which this rule applies.
- **Egress Port**
  Shows a list of all egress ports to which this rule applies.

Note

If you enter the address "00:00:00:00:00:00" for the source and/or destination MAC address, the rule created in this way applies to all source or destination MAC addresses.

Port ingress rules

On this page, you specify the ACL rule according to which incoming frames are handled by the port.

Settings

- **Ports**
  Select the required port.
- **Add Rule**
  Select the ACL rule to be assigned to the port. You specify the ACL rule on the "Rules Configuration" page.
- **Add**
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- **Remove Rule**
  Select the ACL rule to be deleted.
- **Remove**
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- **Rule Order**
  Shows the order of the ACL rules.
- **Rule Number**
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- **Source MAC**
  Shows the unicast MAC address of the source.
- **Dest. MAC**
  Shows the unicast MAC address of the destination.
- **Action**
  Select the action. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.

Port egress rules

On this page, you specify the ACL rule according to which outgoing frames are handled by the port.

Description

- **Ports**
  Select the required port.
- **Add Rule**
  Select the ACL rule to be assigned to the port. You specify the ACL rule on the "Rules Configuration" page.
- **Add**
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- **Remove Rule**
  Select the ACL rule to be deleted.
- **Remove**
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- **Rule Order**
  Shows the order of the ACL rules.
- **Rule Number**
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- **Source MAC**
  Shows the unicast MAC address of the source.
- **Dest. MAC**
  Shows the unicast MAC address of the destination.
- **Action**
  Select the action. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.

Port ACL IP

Rule configuration

On this page, you specify the rules for the IP-based ACL.

Settings

The table has the following columns:

- **Rule Number**
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- **Source IP**
  Enter the IP address of the source.
- **Source Subnet Mask**
  Enter the subnet mask in which the source is located.
- **Dest. IP**
  Enter the IP address of the destination.
- **Dest. Subnet Mask**
  Enter the subnet mask in which the destination is located.
- **Action**
  Select the action from the drop-down list. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- **Ingress Port**
  Shows a list of all ingress ports to which this rule applies.
- **Egress Port**
  Shows a list of all egress ports to which this rule applies.

Protocol configuration

On this page, you specify the rules for protocols.

Settings

The table has the following columns:

- **Rule Number**
  Shows the number of the protocol rule. When you create a rule, a new row with a unique number is created.
- **Protocol**
  Select the protocol for which this rule is valid.
- **Protocol Number**
  Enter a protocol number to define further protocols.
  This box can only be edited if you have set "Other Protocol" for the protocol.
- **Source Port Min.**
  Enter the lowest possible port number of the source port.
  This box can only be edited if you have set "TCP" or "UDP" for the protocol.

- **Source Port Max.**
  Enter the highest possible port number of the source port.
  This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- **Dest. Port Min.**
  Enter the lowest possible port number of the destination port.
  This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- **Dest. Port Max.**
  Enter the highest possible port number of the destination port.
  This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- **Message Type**
  Enter a message type to decide the format of the message.
  This box can only be edited if you have set "ICMP" for the protocol.
- **Message Code**
  Enter a message code to specify the function of the message.
  This box can only be edited if you have set "ICMP" for the protocol.
- **DSCP**
  Enter a value for classifying the priority.
  This box cannot be edited if you have set "ICMP" for the protocol.

Port ingress rules

On this page, you specify the ACL rule according to which incoming frames are handled by the port.

Settings

- **Ports**
  Select the required port.
- **Add Rule**
  Select the ACL rule to be assigned to the port. You specify the ACL rule on the "Rules Configuration" page.
- **Add**
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- **Remove Rule**
  Select the ACL rule to be deleted.
- **Remove**
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- **Rule Order**
  Shows the order of the ACL rules.
- **Rule Number**
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.

- **Protocol**
  Shows the protocol for which this rule is valid.
- **Protocol Number**
  Shows the protocol number.
- **Source IP**
  Shows the IP address of the source.
- **Source Subnet Mask**
  Shows the subnet mask in which the source is located.
- **Dest. IP**
  Enter the IP address of the destination.
- **Dest. Subnet Mask**
  Enter the subnet mask in which the destination is located.
- **Action**
  Select the action. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- **Source Port Min.**
  Shows the lowest possible port number of the source port.
- **Source Port Max.**
  Shows the highest possible port number of the source port.
- **Dest. Port Min.**
  Shows the lowest possible port number of the destination port.
- **Dest. Port Max.**
  Shows the highest possible port number of the destination port.
- **Message Type**
  Shows the message type.
- **Message Code**
  Shows the message code.
- **DSCP**
  Shows the value for classifying the priority.

Port egress rules

On this page, you specify the ACL rule according to which outgoing frames are handled by the port.

Settings

- **Ports**
  Select the required port.
- **Add Rule**
  Select the ACL rule to be assigned to the port. You specify the ACL rule on the "Rules Configuration" page.
- **Add**
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- **Remove Rule**
  Select the ACL rule to be deleted.
- **Remove**
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- **Rule Order**
  Shows the order of the ACL rules.
- **Rule Number**
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- **Protocol**
  Shows the protocol for which this rule is valid.
- **Protocol Number**
  Shows the protocol number.
- **Source IP**
  Shows the IP address of the source.
- **Source Subnet Mask**
  Shows the subnet mask in which the source is located.
- **Dest. IP**
  Enter the IP address of the destination.
- **Dest. Subnet Mask**
  Enter the subnet mask in which the destination is located.
- **Action**
  Select the action from the drop-down list. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- **Source Port Min.**
  Shows the lowest possible port number of the source port.
- **Source Port Max.**
  Shows the highest possible port number of the source port.

- **Dest. Port Min.**
  Shows the lowest possible port number of the destination port.
- **Dest. Port Max.**
  Shows the highest possible port number of the destination port.
- **Message Type**
  Shows the message type.
- **Message Code**
  Shows the message code.
- **DSCP**
  Shows the value for classifying the priority.

Management ACL

On this page, you can increase the security of your device. To specify which station with which IP address is allowed to access your device, configure the IP address or an entire address range.

You can select the protocols and the ports of the station with which it is allowed to access the device. You define the VLAN in which the station may be located. This ensures that only certain stations within a VLAN have access to the device.

Note

Note that a bad configuration may mean that you can no longer access the device.

Settings

- **Management ACL**
  Enable or disable the function.
- **IP Address**
  Enter the IP address or the network address to which the rule will apply. If you use the IP address 0.0.0.0, the settings apply to all IP addresses.
- **Subnet Mask**
  Enter the subnet mask. The subnet mask 255.255.255.255 is for a specific IP address. If you want to allow a subnet, for example a C subnet, enter 255.255.255.0. The subnet mask 0.0.0.0 applies to all subnets.

The table has the following columns:

- **Rule Order**
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- **IP Address**
  Shows the IP address.
- **Subnet Mask**
  Shows the subnet mask.

- **VLANs Allowed**
  Enter the number of the VLAN from which access is allowed.
  Only stations located in these VLANs can access the device. If this input box remains empty, there is no restriction.
  You can enter several individual VLANs and VLAN ranges separated by commas, for example 1,5,10-12.
- **SNMP**
  Specify whether the station or the IP address can access the device using the SNMP protocol.
- **TELNET**
  Specify whether the station or the IP address can access the device using the TELNET protocol.
- **HTTP**
  Specify whether the station or the IP address can access the device using the HTTP protocol.
- **HTTPS**
  Specify whether the station or the IP address can access the device using the HTTPS protocol.
- **SSH**
  Specify whether the station or the IP address can access the device using the SSH protocol.
- **Px.y**
  Specify whether the station or the IP address can access this device via this port.

10.1.4.6 Configuring PROFIBUS DP

The basics of configuring a DP master system

Distributed I/O

DP master systems that consist of a DP master and DP slaves which are connected via a bus and communicate with one another via the PROFIBUS DP protocol are referred to as distributed I/O.

Firmware version of the S7-1200 CPU

Use of the PROFIBUS functions with the S7-1200, requires CPUs with firmware version 2.0 or higher.

Configuring distributed I/O

Since DP masters and DP slaves are different devices, these instructions only provide a basic configuration procedure. However, the process for configuring distributed I/O is practically identical to that of non-distributed configuration.

Creating the DP master system in the network view

After you have used dragged a DP master and a DP slave (for example, a CM 1243-5 and a CM 1243-5) from the hardware catalog to the network view, connect them both to a PROFIBUS subnet.

Additional information

Observe additional information on the scope of functions in the manuals of the respective device.

DP slaves within the hardware catalog

DP slaves within the hardware catalog

You will find the DP slaves in the "Distributed I/O" folder of the hardware catalog. Compact and modular DP slaves are located there:

- Compact DP slaves
  Modules with integrated digital/analog inputs and outputs, for example, ET 200L
- Modular DP slaves
  (Interface modules with S7 modules assigned, for example, ET 200M

The available DP master and the desired functionality will determine which DP slaves can be used.

I slaves within the hardware catalog

The CM 1242-5 is, for example, an DP slave that can be configured as intelligent DP slave. You can find it in the hardware catalog at:

- CM 1242-5
  "PLC > SIMATIC S7 1200 > Communication module > PROFIBUS"

DP/DP coupler in the hardware catalog

Introduction

A DP/DP coupler connects two PROFIBUS DP networks as a gateway so that the DP master from one network can transfer data to the DP master of the other network.

The maximum amount of data that can be transferred is 244 bytes input data and 244 bytes output data.

DP/DP coupler in the hardware catalog

Details of a DP/DP coupler as gateway between two DM master systems are contained in the hardware catalog in the folder "Other field devices > PROFIBUS-DP > Gateways".

Configuring the DP/DP coupler

DP/DP couplers are configured in both PROFIBUS networks, each in their own master systems.

The input and output areas of both networks must thereby be matched to one another. The output data from one end of the DP/DP coupler are accepted as input data at the other respective end and vice versa.

Configurations involving PROFIBUS DP

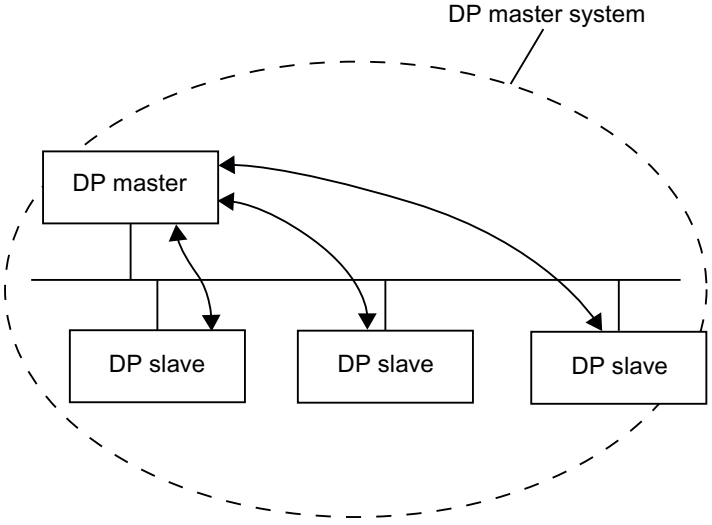
Configurations involving basic DP slaves

Communication between DP master and DP slave

In the case of a configuration involving simple DP slaves, data is exchanged between the DP master and simple DP slaves, i.e. with I/O modules via the DP master. The DP master sequentially polls each DP slave configured within the DP master system and contained in its polling list. In doing so, it transfers the output data to the slaves and receives their input data in return.

Mono-master system

The configuration with only one DP master is also described as mono-master system. A single DP master with its associated DP slaves is connected to a physical PROFIBUS DP subnet.



Configurations involving intelligent DP slaves

Definition

DP slaves that feature their own preprocessing program are referred to as intelligent DP slaves (I-slaves).

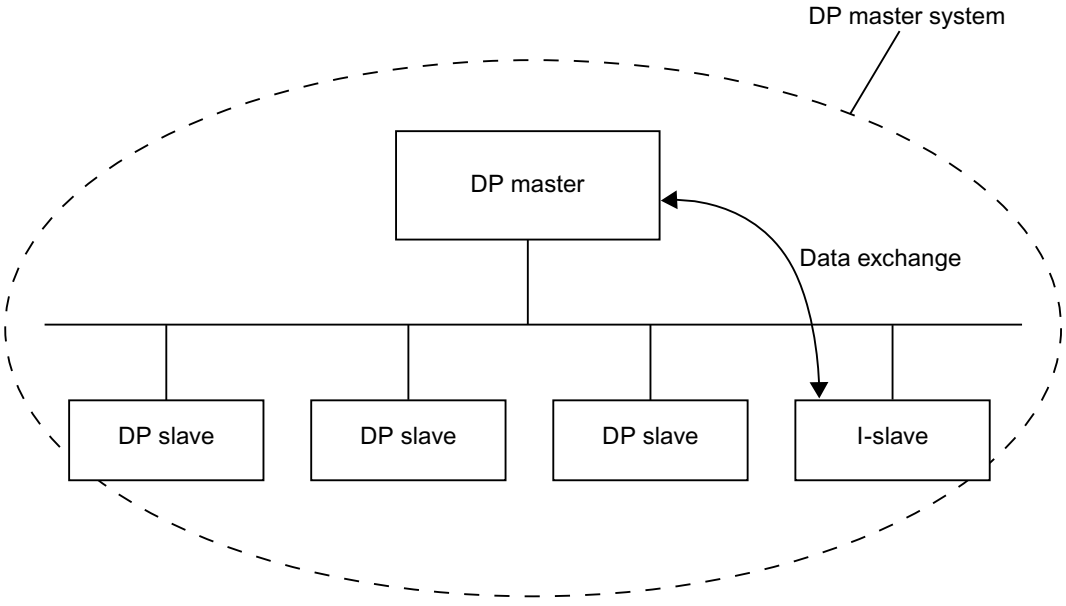
CM 1242-5 is an intelligent DP slave.

I-slave <> DP master data exchange

A higher-level automation system processes the automation task, which is broken down into sub-tasks. The sub-tasks are processed in the lower-level automation systems. The necessary control tasks are processed separately and efficiently in the CPUs as preprocessing programs.

In the case of configurations involving intelligent DP slaves, the DP master only accesses the operand area of the I-slave's CPU, and not the I/O modules of the intelligent DP slave. The operand area must not be assigned to any actual I/O modules in the I-slave. The assignment must be made during I-slave configuration.

The addresses of the data to be exchanged between the master and slave are configured in the transfer area of the I-slave.



Configuring distributed I/O systems

Hint: Quick configuration of master systems

If the DP master system has several DP slaves, use drag-and-drop operation to assign to the master in one step all DP slaves that were placed.

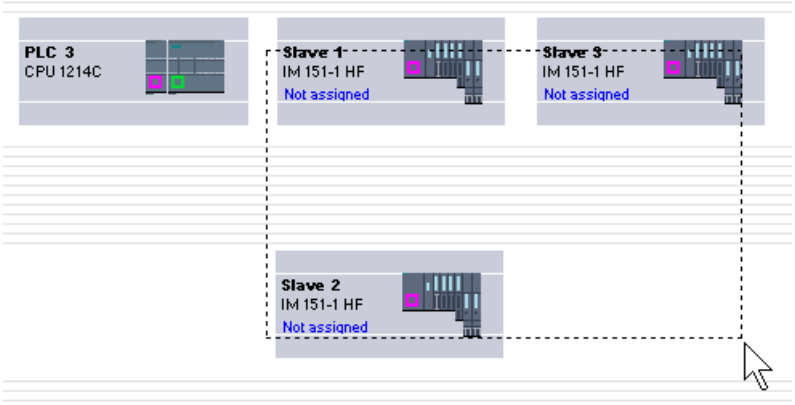
Requirements

DP master and DP slaves are placed in the network view.

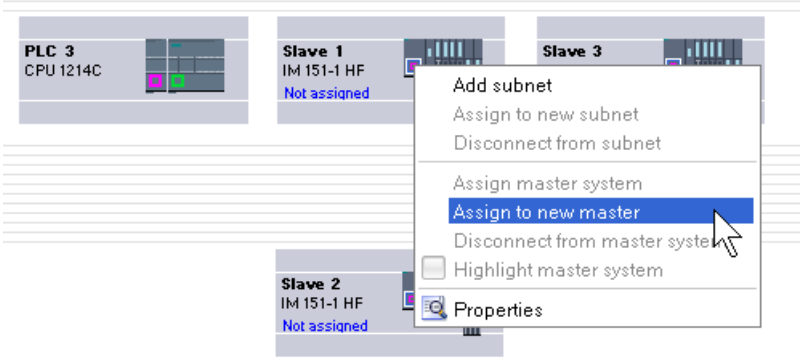
Assigning DP slaves to a DP master system

To do this, follow these steps:

1. Select an appropriate zoom factor so that you can see as many DP slaves as possible in the network view.
2. Arrange the DP slaves in a maximum of two rows.
3. Select all DP interfaces with the mouse cursor (not all devices!). This only works if you begin to drag the mouse cursor outside of the first DP slave and release the mouse button at the last DP slave (selection with the lasso).



4. Select the shortcut menu "Assign to new master" and select the corresponding DP interface for the DP master in the subsequent dialog.



5. The DP slaves are automatically networked with the DP master and combine with it to form a DP master system.

Note

When a DP master system is highlighted, you can double-click on a DP slave in the hardware catalog and thereby quickly add additional DP slaves. This will result in the DP slave being added to the highlighted DP master system automatically.

Creating a DP master system

Introduction

To create a DP master system, you need to have one DP master and at least one DP slave. As soon as you connect a DP master to a DP slave, a master-slave link is established.

DP master

You can use any of the following devices as a DP master:

- CM 1243-5

Requirement

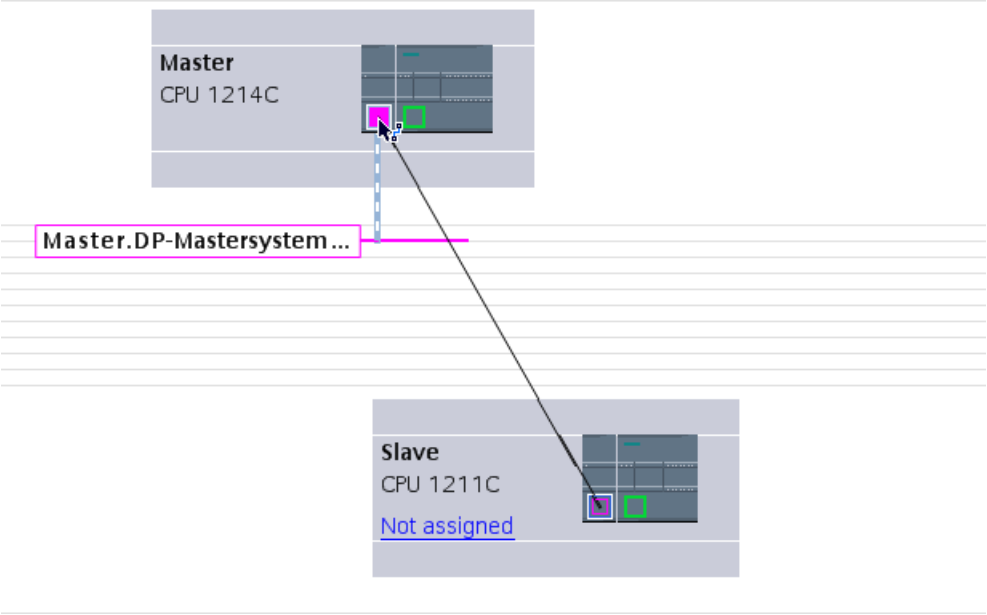
- You must be in the network view.
- The hardware catalog is open.

Procedure

To create a DP master system, follow these steps:

1. Select a DP master from the hardware catalog.
2. Pull the DP master onto the free area within the network view.
3. Right-click on the DP master's DP interface.
4. Select "Create master system" from the shortcut menu.
   A DP master system with one DP master will be created as a single node.

If you connect a DP slave's DP interface to that of the DP master, the DP slave will be added to the master system.



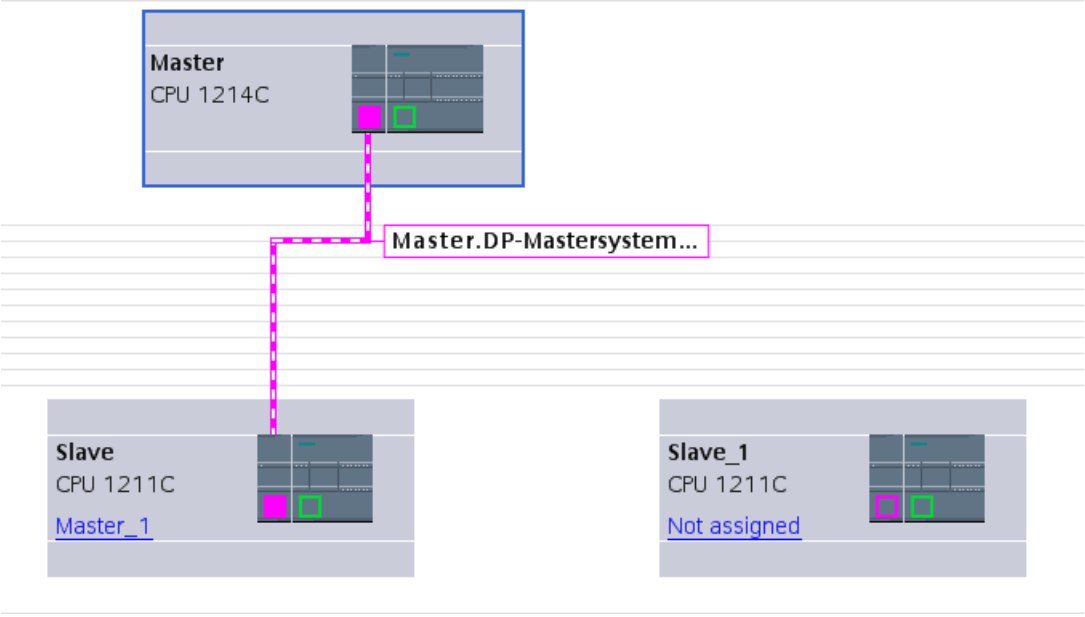
Assuming that you have already placed both a DP master and a DP slave within the network view, you can drag-and-drop to connect the two and thereby create a DP master system. To do so, follow these steps:

1. Click on the DP interface of either the DP master or DP slave.
2. Hold down the mouse button and draw a connecting line between this DP interface and that of the desired communication partner.

This will create a subnet with one DP master system between the DP master and DP slave.

DP master display on the DP slave

When you connect a DP slave to a DP master, the name of the DP master is displayed on the DP slave as a hyperlink. Click the hyperlink to select the associated DP master.

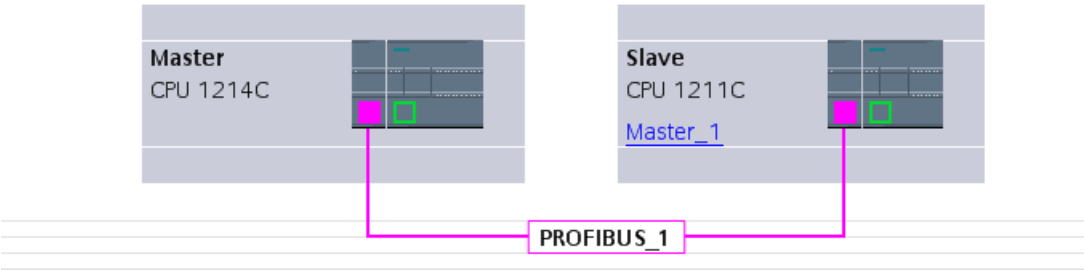


Highlighting applied to the DP master system

When you create a new DP master system, highlighting will be applied to it. This enables you to identify quickly which devices belong to the DP master system. You can also highlight a DP master system yourself by moving the mouse pointer over a subnet. This will result in the names of the available DP master systems being displayed. Click the required DP master system to highlight it.

There are various ways of removing the highlighting from a DP master system:

- Highlight a different master system.
- Click on the drawing pin with the name of the master system in the top right-hand corner of the network view.



Editing DP master systems and interfaces

Introduction

Once you have created a DP master system, you also have the option of disconnecting the DP master system from its components. This can result in subnets with DP slaves but without DP master.

Generally, there is no need to edit the interfaces of a DP master.

You can change the name and number on the DP master system.

Disconnection of master or slaves from the DP master system

If you have configured a PROFIBUS CP as a DP master with master system, you can then disconnect the DP master system from the DP master. After this, the device will no longer be connected to the DP master system.

Disconnecting the subnet from a DP master effectively eliminates the master system in the sense that it is no longer assigned to a DP master. However, the individual DP slaves are still interconnected via the subnet.

If you delete the DP slaves or disconnect them from the master system, the master system is then retained on the DP master.

Requirement

- You must be in the network view.
- There has to be a DP master system with one DP master and at least one DP slave.

Disconnecting the DP master from the DP master system

To disconnect the DP master system, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Disconnect from master system" from the shortcut menu.

The selected DP master will be disconnected from the DP master system. A subnet with the DP slaves will be retained.

Adding a DP master to the DP master system

To reassign a DP master to a subnet, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Create master system" from the shortcut menu.
3. Draw connecting lines between the new DP master system and the DP interfaces of the DP slaves.

The DP master combines with the DP slaves to recreate a DP master system.

Editing the properties of a DP master system

To edit the properties of a DP master system, proceed as follows:

1. Move the mouse pointer over a subnet with a DP master system.
2. A message will appear displaying the available DP master systems. Click the one you want to edit. The DP master system will now be color-highlighted.
3. Click on the highlighted DP master system.
4. In the inspector window, edit the DP master system attributes under "Properties > General".

Note

If you click a subnet when no DP master system is highlighted, you will be able to edit the properties of the entire subnet under "Properties" in the inspector window.

Adding DP slaves to the master system and configuring them

In the network view, you can add various DP slaves from the hardware catalog directly by using the drag-and-drop function or by double-clicking.

DP slaves

For configuration purposes, DP slaves are broken down into the following categories:

- Compact DP slaves
  (Modules with integrated digital/analog inputs and outputs, for example, ET 200L)
- Modular DP slaves
  (interface modules with S5 or S7 modules assigned, for example ET 200M)
- Intelligent DP slaves (I slaves)
  (CM 1242-5 or ET 200S with IM 151-7 CPU)

Rules

- Your DP master system should only contain one DP master, but it may contain one or or more DP slaves.
- You may only configure as many DP slaves in a DP master system as are permitted for the specific DP master.

Note

When configuring the DP master system, remember to observe the DP master technical data (max. number of nodes, max. number of slots, max. quantity of user data). User data restrictions may possibly prevent you from being able to use the maximum number of nodes that is theoretically possible.

Requirements

- You must be in the network view.
- A DP master system must have been created.

Adding a DP slave to the DP master system

To add a DP slave from the hardware catalog to the DP master system, follow these steps:

1. Select a DP slave from the hardware catalog.
2. Drag-and-drop the DP slave from the hardware catalog into the network view.
3. Draw a connecting line between the DP master's DP interface or a highlighted DP master system and the DP interface of the new DP slave.

A DP master system will be created and the DP slave will be connected to the DP master automatically.

Note

When a DP master system is highlighted, you can double-click the required DP slave in the hardware catalog. This will result in the DP slave being added to the highlighted DP master system automatically.

Disconnecting a DP slave from the DP master system

To disconnect a DP slave from the DP master system, follow these steps:

1. In the network view, right-click on the DP slave's DP interface.
2. From the shortcut menu, select the method for disconnecting from the DP master system:
   - "Disconnect from subnet": The PROFIBUS connection is broken and the device is no longer connected to the DP master system or a subnet.
   - "Disconnect from master system": The DP slave remains connected to the subnet, but is no longer assigned to the DP master system as a DP slave.

The selected DP slave will be disconnected from the DP master system.

Assigning a DP slave to a new DP master system

To assign an existing DP slave to a new DP master system, follow these steps:

1. Right-click on the DP slave's DP interface.
2. From the shortcut menu, select "Assign to new master".
   It does not matter whether the DP slave concerned is already assigned to another DP master system.
3. From the selection list, select the DP master whose DP master system is to have the DP slave connected to it.
   The selected DP slave is now assigned to a new DP master system.

The "Assign to new subnet" function works in a similar way in that it allows you to connect a DP slave to a new subnet. However, in this case, the DP slave will not be connected to an existing DP master system.

Configuring a DP slave

To configure a DP slave, follow these steps:

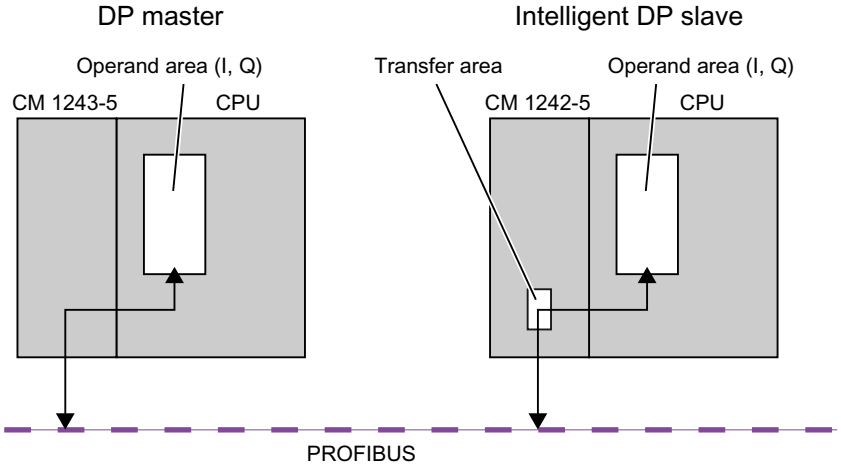
1. Switch to the DP slave's device view.
2. Select the module you want.
3. Configure the DP slave in the Inspector window.

Configuring intelligent DP slaves

Adding an I-slave to a DP master system

Introduction

One of the characteristics of an intelligent DP slave (I-slave) is that the DP master is not provided with I/O data directly by a real I/O, but by a preprocessing CPU. Together with the CP, this preprocessing CPU forms the I-slave.



Difference: DP slave v. intelligent DP slave

In the case of a DP slave, the DP master accesses the distributed I/O directly.

In the case of an intelligent DP slave, the DP master actually accesses a transfer area in the I/O address space of the preprocessing CPU instead of accessing the connected I/O of the intelligent DP slave. The user program running on the preprocessing CPU is responsible for ensuring data exchange between the address area and I/O.

Note

The I/O areas configured for data exchange between the DP master and DP slaves must not be used by I/O modules.

Applications

Configurations involving intelligent DP slaves: I-slave <> DP master data exchange

Procedure

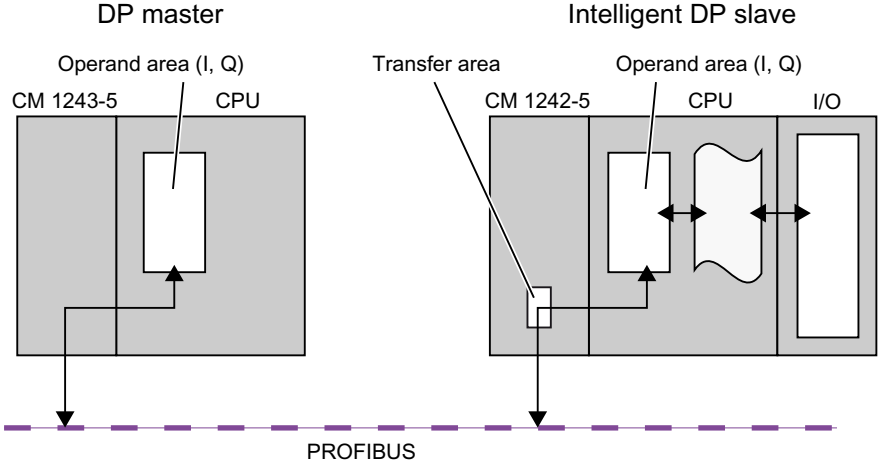
To add an I-slave to a DP master system, follow these steps:

1. In the network view, drag from the hardware catalog to a station one CM 1242-5 as an I-slave and one CM 1243-5 as a DP master.
2. Draw a connecting line between the DP interfaces of both devices.
   This way you connect the I-slave with a DP master in a DP master system.
   Result: You have now set up a DP master system with one DP master and one I-slave.

Configuring access to I slave data

Data access

The following applies to the CP 1242-5 in its function as I-slave: The addresses for the data transfer area and the address for the I/O modules in the I-slave differ. This means that the start address occupied by an I/O module can no longer be used for the transfer memory. If the higher-level DP master is to access the data of an I/O module in the I-slave, you must configure this data exchange between the I/O module and transfer area in the I-slave user program.



Configuration of the transfer area for the CM 1242-5 (transfer area)

With CM 1242-5, the transfer area for the cyclic PROFIBUS data exchange is configured as transfer area in the parameter group "PROFIBUS interface > Mode > I Slave Communication".

Direct data access from CPU to CPU

Direct data access from CPU to CPU via PROFIBUS is supported by the S7-1200 PROFIBUS CMs only via the PUT/GET services.

Configuring DP slaves as distributed I/O devices

Configuring an ET 200S

Slot rules for configuring an ET 200S

The following rules apply when configuring an ET 200S:

- Do not leave any gaps when inserting the ET 200S modules.
- Slot 1: only for PM-E or PM-D Power Modules.
- To the left of an Electronics Module (EM): an EM or a Power Module (PM-E or PM-D) only.
- To the left of Motor Starter (MS): an MS, a PM-D, PM-D Fx (1..x..4) Power Module or a PM-X Power Module only.
- To the left of a PM-X: a motor starter or a PM-D only
- Up to 63 modules and one IM Interface Module are permitted

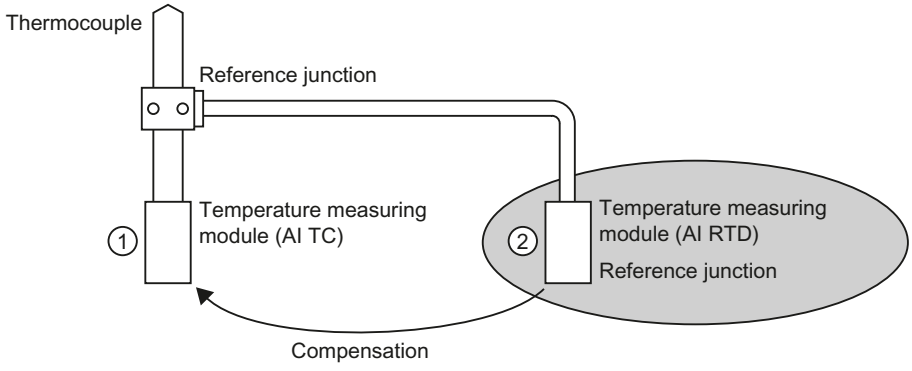
Note

Remember to ensure that the correct PM-E and EM voltage ranges are assigned.

Configuring reference junctions

A reference junction is the connection of a thermocouple to a supply line (generally in the terminal box). The voltage that occurs due to the effects of temperature falsifies the temperature value measured by the module.

On the ET 200S, one channel of the AI RTD module can be programmed as a reference junction. Other AI TC modules can correct their measured values using the temperature at the reference junction as measured by this module.



① Configuring the AI TC:
→ Selection of the reference junction used

② Configuring of the AI RTD:
→ Activation of the reference junction
→ Specifying the slot and channel of the AI RTD

Special characteristics to be noted when assigning parameters for reference junctions

The process of assigning parameters for reference junctions will be explained based on the example of a resistance thermometer with a Pt 100 climatic range that is used for measuring the reference junction temperature.

To assign parameters for the reference junction, follow these steps:

1. In the ET 200S device view, insert an analog electronics module, for example a 2AI RTD HF.
2. Select the module on the rack.
3. Under "Properties > Inputs" in the inspector window, set a channel for the reference junctions function to the "RTD-4L Pt 100 climatic range" measuring range.
4. Select the ET 200S.
5. Under "Properties > Module parameters > Reference junctions" in the inspector window, select the "Reference junction" check box and specify the slot and channel number of the relevant RTD module.
6. Insert the analog electronics module for measuring the temperature using a thermocouple (TC module) and parameterize it with the reference junction number of the RTD module.

Additional information

For additional information on the various types and uses of ET 200S modules, please refer to the operating instructions and the manual titled "ET 200S Distributed I/O System".

For additional information on analog value processing, please see the documentation for the ET 200S distributed I/O system.

Packing addresses

Introduction

DP slaves and I/O devices from the ET 200S family are configured in the same way as other modular DP slaves and I/O devices. As well as supporting all the standard modular DP slave and I/O device functions, the ET 200S also offers the "Pack addresses" function:

When digital electronics modules requiring an address space of 2 or 4 bits are inserted into the device view, they will initially be spread over a total area of 1 byte. However, the address area actually occupied can be compressed after configuration using the "Pack addresses" function.

	Initial state	After "Pack addresses"
Module	I address	I address
2DI (2-bit)	Byte 10	10.0...10.1
4DI (4-bit)	Byte 11	10.2...10.5

Requirements

- You are in the device view.
- An ET 200S, for example an IM 151-1, must be present.
- A pair of digital electronics modules, for example 2DI AC120V ST, must be inserted into the slots.

Packing addresses

To pack addressed, follow these steps:

1. Select the electronics modules whose addresses are to be packed. The following options are available for selecting multiple electronics modules:
   - Press and hold down <Shift> or <Ctrl> while clicking the relevant electronics modules.
   - Click off the rack and select the required electronics modules by drawing round them with the mouse.
2. Click "Pack addresses" in the shortcut menu for the selected electronics modules.

The address areas for inputs, outputs and motor starters are packed separately. The packed addresses will be displayed in the I address and Q address columns of the device overview.

How packed addresses are generated and structured

If you make use of the "Pack addresses" function, the addresses of the selected electronics modules will be packed in accordance with the following rules:

- The start of the address area is determined by the lowest address of the selected electronics modules: X.0
- If the bit address is not "0", then the next (free) byte address as of which the selected area can be compacted will be selected automatically: (X+n).0
- If no coherent area exists, the addresses will be automatically packed into any available address gaps.

Electronics modules with packed addresses and the same byte address form a packing group.

Unpacking addresses

To unpack addressed, follow these steps:

1. Select one or more electronics modules with packed addresses.
2. Click "Unpack addresses" in the shortcut menu for the selected electronics modules.

The packing groups of the selected electronics modules will be disbanded and the packed addresses for the relevant electronics modules unpacked.

The packing group will also be disbanded and the packed addresses unpacked in the following cases: if you delete electronics modules from a packing group, move electronics modules out of a packing group or insert electronics modules on a free slot within a packing group.

The start addresses of the unpacked electronics modules will be assigned to the next available byte addresses in each case.

Special characteristics of electronics modules with packed addresses

The following special characteristics apply to electronics modules with packed addresses:

- As far as the CPU is concerned, there is no way of assigning a slot for the electronics module. Consequently, the instruction GADR_LGC (SFC 5) outputs error information W#16#8099 "Slot not configured" for the actual slot of the electronic module.
- The instruction LGC_GADR (SFC 49) and SZL-ID W#16#xy91 "module status information" cannot be evaluated for an electronics module.
- The electronics module receives an additional diagnostics address via the DPV1 function, because otherwise the packed addresses would prevent interrupts from being assigned as far as the CPU is concerned.
- The "Insert/remove interrupt" is not possible. This is because the "Pack addresses" and "Insert/remove interrupt" functions are mutually exclusive.

Configuring option handling with standby modules

You can use the option handling function to prepare the ET 200S with PROFIBUS interface for future expansions (options). This section describes option handling with standby modules.

You do this by assembling, wiring, configuring, and programming the maximum configuration envisaged for the ET 200S and by using cost-effective standby modules (138-4AA00 or 138-4AA10) during assembly until it becomes time to replace them with the necessary electronics modules.

Note

The ET 200S can be completely factory-wired with the master cabling, as no connection exists between a standby module and the terminals of the terminal module (and, in turn, to the process).

Requirement

- ET 200S interface module
  - IM 151-1 STANDARD (6ES7 151-1AA03-0AB0 or higher)
  - IM 151-1 FO STANDARD (6ES7 151-1AB02-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the numbered check boxes for the slots that are initially to accommodate the standby modules prior to the future electronics modules.
3. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).

The assembled standby modules can be replaced with the configured modules at a later date without having to modify the configuration.

Note

The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 STANDARD Interface Module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/ AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of standby modules, please refer to the documentation for the ET 200S distributed I/O system.

How option handling works during startup

If "Startup when expected/actual config. differ" is not available, the ET 200S will still start up if a standby module is inserted instead of the configured electronics module and option handling has been activated for the slot concerned.

How option handling works during operation

During operation, the option handling mode varies in accordance with the following:

- Option handling enabled for a slot:
  Either the standby module (option) or the configured electronics module can be plugged into this slot. If any other kind of module is present on this slot, diagnostics will be signaled (no module/incorrect module).
- Option handling disabled for a slot:
  Only the configured electronics module can be plugged into this slot. If any other kind of module is present, diagnostics will be signaled (no module/incorrect module).

Standby module substitute values

- Substitute value for digital inputs: 0
- Substitute value for analog inputs: 0x7FFF

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 2 to 63:

- Bit value = 0: Option handling parameterized. Standby modules are permitted.
- Bit value = 1: Option handling canceled. Standby modules are not permitted on this slot.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: This slot has the standby module, an incorrect module or no module plugged into it.
- Bit value = 1: The configured module is plugged into this slot.

See also

Which modules support option handling? (http://support.automation.siemens.com/WW/view/en/22564754)

Configuring option handling without standby modules

You can use the option handling function to prepare the ET 200S for future expansions (options) without installing standby modules. This section describes option handling without standby modules.

Note

ET 200S with PROFINET interface

This description refers to the ET 200S with PROFIBUS interface. In principle, option handling for ET 200S with PROFINET interface functions as described here without standby modules. PN interface modules are to be used instead of the DP interface modules listed here. You can find additional information about option handling for ET 200S with PROFINET interface in the corresponding manuals.

Requirement

- ET 200S interface module
  - IM 151-1 HIGH FEATURE (6ES7151-1BA02 or higher)
  - IM 151-1 STANDARD (6ES7 151-1AA05-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).
3. Configure the slave's maximum configuration. The selection/clearing of options is controlled via the user program.

   Note

   The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 interface module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

   Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

   Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/ AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of standby modules, please refer to the documentation for the ET 200S distributed I/O system.

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 1 to 63:

- Bit value = 0: Slot is not available in the actual configuration.
- Bit value = 1: Slot is available in the actual configuration.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: Slot belongs to an option that is not available or module status is faulty.
- Bit value = 1: The configured module is plugged into this slot.

See also

Sample applications for ET 200S, option handling without standby module (http://support.automation.siemens.com/WW/view/en/29430270)

Configuring the ET 200S in DPV1 mode

PROFIBUS DPV1 enables you to access extended PROFIBUS functions.

Requirement

- You must be in network view.
- A DP master with DPV1 functionality must be available.
- A master-slave connection must be established with PROFIBUS.

Procedure

To switch the DP slave over to DPV1, follow these steps:

1. Select the DP slave.
2. Under "Properties > Module parameters" in the Inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list.

or

1. Select the DP master.
2. In the I/O communications table, select the row with the connection between the DP master and the desired DP slave.
3. Under "Properties > Module parameters" in the Inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list.

Special characteristics

The parameters are subject to interdependencies, which are outlined below:

Parameter	DPV0 mode	DPV1 mode
Operation when target configuration does not match actual configuration	Fully available	Fully available
Diagnostics interrupt (OB 82)	Not available, not set	Fully available
Hardware interrupt (OB 40 to 47)	Not available, not set	Fully available
Insert/remove interrupt (OB 83)	Not available, not set	Only available when addresses are not packed. "Startup when target configuration does not match actual configuration" is activated automatically along with an insert/remove interrupt.

Interrupts in the case of modules with packed addresses

If the module is capable of triggering interrupts and the bit address is not equal to 0 because of packed addresses, you will need to assign a diagnostics address in the ET 200S address dialog.

The diagnostics address is required for the purpose of assigning a DPV1 interrupt to the module as an interrupt trigger. The CPU will only be able to assign an interrupt correctly and store information on the interrupt in the interrupt OB start information/in the diagnostics buffer if the module concerned has this "unpacked" address. In this context, the CPU cannot make use of "packed" addresses.

From the point of view of interrupt processing (interrupt OB), this means the module will have the assigned diagnostics address, but from the point of view of processing the input and output data in the user program, the module will have the packed addresses.

Note

When the module addresses are packed, the insert/remove interrupt for the ET 200S is unavailable.

Using GSD files

GSD revisions

What you need to know about GSD revisions

The properties of DP slaves are made available to configuration tools by means of GSD files.

Functional enhancements in the area of the distributed I/O will have an effect on the GSD specification, for example, they will require the definition of new keywords.

This results in the versioning of the specification. In the case of GSD files, the version of the specification on which a GSD file is based is called a "GSD revision".

From GSD revision 1, the GSD revision must be included as a keyword "GSD_revision" in GSD files. GSD files without this keyword will therefore be interpreted by configuration tools as GSD revision "0".

GSD files can be interpreted up to GSD revision 5. This means that DP slaves that support the following functions, for example, will be supported:

- Diagnostic alarms for interrupt blocks
- Isochronous mode and constant bus cycle time
- SYNC/FREEZE
- Clock synchronization for DP slaves

Installing the GSD file

Introduction

A GSD file (generic station description file) contains all the DP slave properties. If you want to configure a DP slave that does not appear in the hardware catalog, you must install the GSD file provided by the manufacturer. DP slaves installed via GSD files are displayed in the hardware catalog and can then be selected and configured.

Requirement

- The hardware and network editor is closed.
- You have access to the required GSD files in a directory on the hard disk.

Procedure

To install a GSD file, follow these steps:

1. In the "Options" menu, select the "Install general station description file (GSD)" command.
2. In the "Install general station description file" dialog box, select the folder in which the GSD files are stored.
3. Choose one or more files from the list of displayed GSD files.

4. Click on the "Install" button.
5. To create a log file for the installation, click on the "Save log file" button.
   Any problems during the installation can be tracked down using the log file.

You will find the new DP slave installed by means of the GSD file in a new folder in the hardware catalog.

See also

Overview of hardware and network editor (Page 535)

Deleting GSD file

Introduction

You can delete installed DP slaves using GSD files. These DP slaves are then no longer displayed in the hardware catalog.

Requirement

- The hardware and network editor is closed.
- The hardware catalog contains DP slaves installed via GSD files.

Procedure

To delete a GSD file, follow these steps:

1. In the "Options" menu, select the "Install general station description file (GSD)" command.
2. In the "Install general station description file" dialog box, select the folder in which the GSD file is stored.
3. Select the file that is to be deleted from the list of displayed GSD files.
4. Click the "Delete" button.

The selected GSD file was deleted and the DP slave is no longer located in the hardware catalog.

Configuring GSD-based DP slave

DP slaves that you have inserted through installation of a GSD file can be selected as usual via the hardware catalog and inserted in the network view. If you want to insert the modules of the GSD-based DP slaves, you must take into account some particular details.

Requirement

- You have installed a DP slave using a GSD file.
- You have inserted the head module in the network view in the usual manner.

- The device overview opens in the device view.
- The hardware catalog is open.

Procedure

To add the modules of a GSD-based DP slave, proceed as follows:

1. In the hardware catalog, navigate to the modules of the GSD-based DP slave.
   GSD-based DP slaves, also referred to as DP standard slaves, can be found in the "Other field devices" folder of the hardware catalog.
2. Select the desired module.
3. Use drag-and-drop to move the module to a free space in the device overview.
4. Select the module in the device overview to edit parameters.

You have now inserted the module in a free slot of the GSD-based DP slave and can edit its parameters.

Note

You can see only the GSD-based DP slave in the graphic area of the device view. The added modules of GSD-based DP slaves are only found in the device overview.

Preset configuration

For modules with an adjustable preset configuration, you can change this configuration in the inspector window under "Properties > Preset configuration".

10.1.4.7 Configurations for PROFINET IO

What you need to know about PROFINET IO

What is PROFINET IO?

PROFINET IO

PROFINET is an Ethernet-based automation standard of PROFIBUS Nutzerorganisation e.V. (PNO) which defines a manufacturer-neutral communication, automation and engineering model.

Objective

The objective of PROFINET is:

- Integrated communication via field bus and Ethernet
- Open, distributed automation
- Use of open standards

Architecture

The PROFIBUS User Organisation e.V. (PNO) has designated the following aspects for PROFINET architecture:

- Communication between controllers as components within distributed systems.
- Communication between field devices, such as I/O devices and drives.

Implementation by Siemens

The demand for "Communication between controllers as components within distributed systems" is implemented by "Component Based Automation" (CBA). Component Based Automation is used to create a distributed automation solution based on prefabricated components and partial solutions.

The demand for "Communication between field devices" is implemented by Siemens with "PROFINET IO". Just as with PROFIBUS DP, the complete configuration and programming of the components involved is possible using the Totally Integrated Automation Portal.

The following sections deal with the configuration of communication between field devices using PROFINET IO.

Overview of RT classes

RT classes in PROFINET IO

PROFINET IO is a scalable, real-time communication system based on Ethernet technology. The scalable approach is reflected in several real-time classes:

- **RT:** Transmission of data in prioritized, non-isochronous Ethernet frames. The required bandwidth is within the free bandwidth area for TCP/IP communication.
- **IRT:** Isochronous transmission of data with high stability for time-critical applications (for example, motion control). The required bandwidth is from the area of bandwidth reserved for cyclic data.

Depending on the device, not all real-time classes are supported.

Which IO controllers and IO devices support which PROFINET functions?

Additional information and overviews

In the following Siemens Industry Online Support article (http://support.automation.siemens.com/WW/view/en/44383954), you will find an overview of the PROFINET IO controllers and IO devices that support the following PROFINET functions:

- Isochronous real-time communication (IRT)
- Prioritized startup
- Media redundancy (MRP)
- PROFIenergy
- Shared device
- I-device
- Isochronous mode of process data

The functions are explained in the following sections, but without naming the respective hardware that supports these functions.

In the hardware catalog, you can also find an overview of the supported functions in the description below the selected components.

You can also find a description of PROFINET in the respective current STEP 7 version here (http://support.automation.siemens.com/WW/view/en/49948856).

Connecting existing bus systems

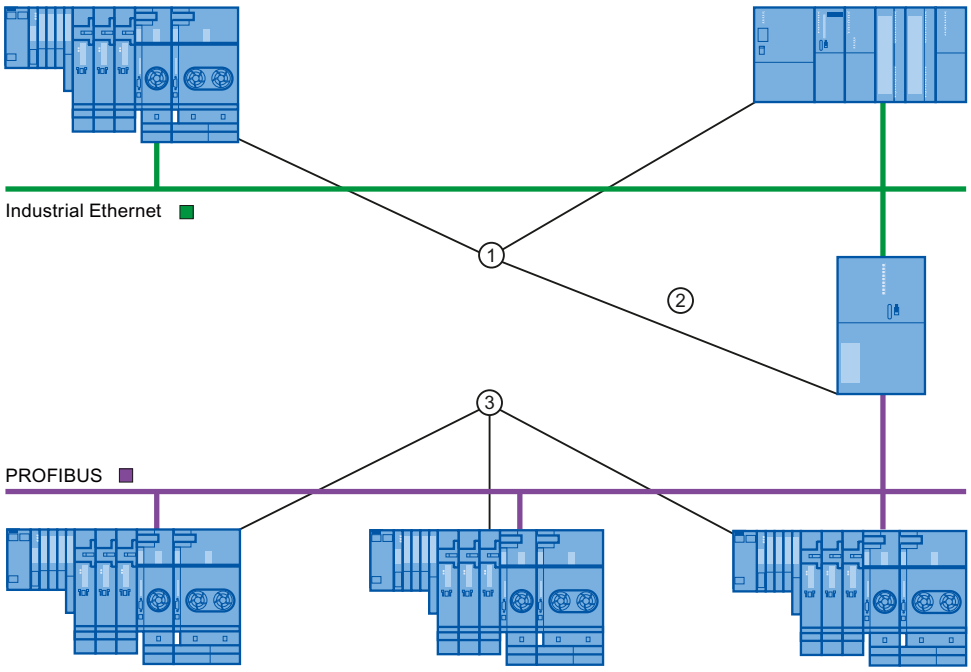
Connection of PROFINET and PROFIBUS

PROFINET IO and PROFIBUS DP can be connected with each other as follows:

- via Industrial Ethernet:
  To connect the two network types, Industrial Ethernet (plant control level) and PROFIBUS (cell level/field level), use the IE/PB link, for example.
- via Industrial Wireless LAN:
  PROFIBUS devices, for example, can be connected to PROFINET IO via a wireless LAN/PB link. This allows existing PROFIBUS configurations to be integrated into PROFINET.

AS interface devices can be connected by an IE/AS-i link PN IO to the interface of a PROFINET device. This allows the existing AS-i network to be integrated into PROFINET.

The following figure shows the connection of a PROFIBUS subnet via a PROFINET device with proxy functions.



① PROFINET devices
② PROFINET device with proxy functions (for example, IE/PB link)
③ PROFIBUS devices

PROFINET device with proxy functions used as proxy for a PROFIBUS device

The PROFINET device with proxy functions is the proxy for a PROFIBUS device on the Ethernet. The proxy functionality enables a PROFIBUS device to communicate with all devices on the PROFINET and not just with its master.

Existing PROFIBUS systems can easily be integrated into PROFINET communication using the proxy functions.

If, for instance, you connect a PROFIBUS device via an IE/PB link to PROFINET, the IE/PB link acts as a proxy for the PROFIBUS components for communicating via PROFINET.

Configuration using IE/PB link PN IO

Configuration using IE/PB Link PN IO

You can use the IE/PB Link PN IO to connect PROFIBUS DP configurations to PROFINET IO.

From the CPU perspective, the PROFIBUS DP slaves are connected to the same network as the IE/PB Link PN IO. These slaves have the same device name and IP address as the IE/PB Link PN IO, but different device numbers. Furthermore, each also has a specific PROFIBUS address.

In the properties of the IE/PB link, the PROFIBUS addresses of the connected DP slaves are displayed in addition to the PROFINET device numbers, as this device has two addressing schemes.

Handling device numbers and PROFIBUS addresses on the master system

During placement, the same number is assigned for the PROFINET device number and the PROFIBUS address.

In the Inspector window under "General Properties > PROFINET device number", you can find an overview of the device numbers used and the PROFIBUS addresses of an IE/PB link. The device number can also be changed here. You can also set that the device number and the PROFIBUS address should or should not always be identical. If the "PROFINET device number=PROFIBUS address" option is activated, you do not have to update the device number when the PROFIBUS address changes.

The PROFIBUS address can be changed in the properties of the PROFIBUS device.

Restrictions

The following restrictions apply to DP slaves configured as described above on the PROFIBUS subnet of an IE/PB link:

- No pluggable IE/PB link
- No pluggable DP/PA link
- No pluggable Y link
- Not CiR-compliant
- No pluggable redundant slaves
- No isochronous transmission / constant bus cycle time can be configured
- SYNC/FREEZE instructions ("DPSYC_FR") of a CPU on the the Ethernet subnet for DP slaves behind the IE/PB-Link are not supported.

See also

Connect the DP slave via the IE/PB Link to a PROFINET IO system (Page 1138)

Configuration using IWLAN/PN link

Maximum number of devices in a IWLAN segment

If an Ethernet subnet is set up as wireless network (IWLAN = Industrial Wireless LAN), cyclic data exchange between IO controllers and IO devices is possible via a wireless route.

On one side of the wireless route there are fixed installed access points (for example, SCALANCE W 788) and on the other side mobile stations (with, for example IWLAN/PB links with PROFIBUS devices).

If the action radius of the mobile stations is large, it may be necessary to install several access points (SCALANCE W 788). Since each access point forms a segment with its wireless range, the IWLAN is made up of a series of segments.

The mobile devices "on the one side" of the wireless link with their IWLAN/PB links can move along the segments.

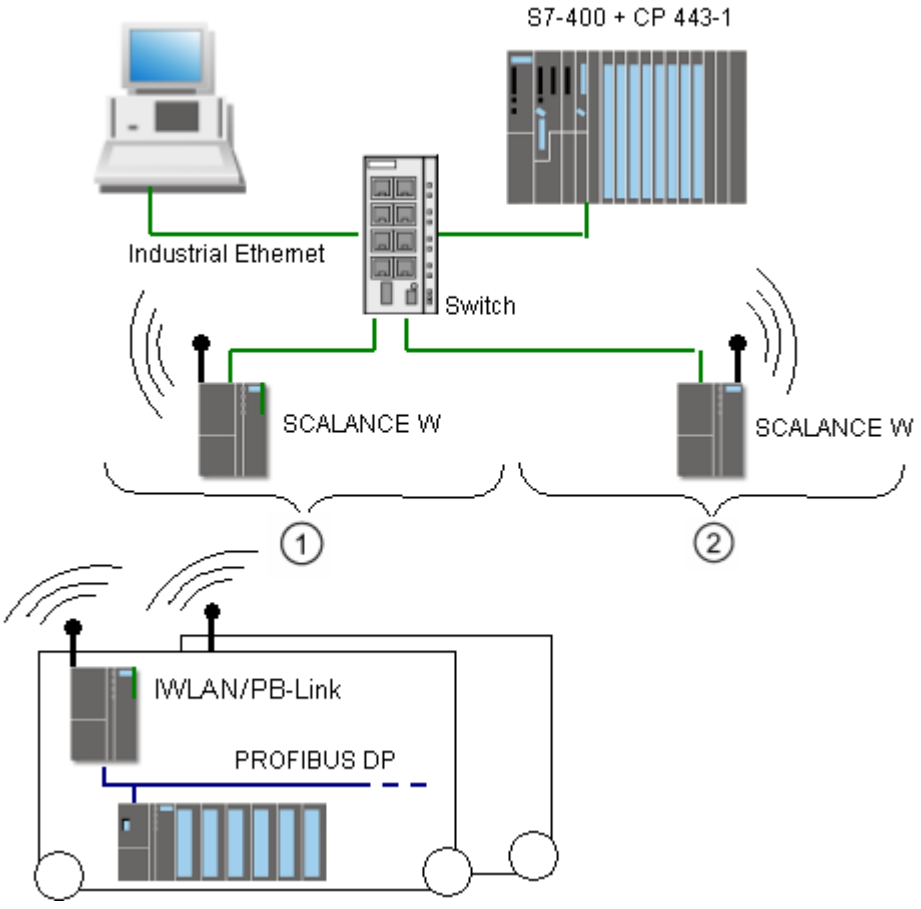
Special feature

If several IWLAN/PB links are located within a segment, they have to share the bandwidth that is available for wireless transmission. This leads to a lengthening of the update time for these devices.

Example

In the following example there are two IO devices (IWLAN/PB link) with a segment.

If no more than a maximum of two IWLAN/PB links are present in a IWLAN segment at the same time, enter a "2".



① Segment 1
② Segment 2

Configure PROFINET IO

Addressing PROFINET devices

Assigning addresses and names to PROFINET devices

In this chapter you will learn which address and naming conventions are valid for the PROFINET devices.

IP addresses

All PROFINET devices work with the TCP/IP protocol and therefore require an IP address for Ethernet operation.

You can set the IP addresses in the module properties. If the network is part of an existing company Ethernet network, ask your network administrator for this data.

The IP addresses of the IO devices are assigned automatically, usually at CPU startup. The IP addresses of the IO devices always have the same subnet mask as the IO controller and are assigned in ascending order, starting at the IP address of the IO controller.

Device names

Before an IO device can be addressed by an IO controller, it must have a device name. This procedure was chosen for PROFINET because names are easier to administer than complex IP addresses.

Both the IO controller as well as IO devices have a device name. When the "Generate PROFINET device name automatically" option is activated, the device name is automatically derived from the name configured for the device (CPU, CP or IM):

- The PROFINET device name is made up of the name of the device (for example, the CPU), the name of the interface (only with multiple PROFINET interfaces) and optionally the name of the IO system:
  <CPU name>.<Name of the interface>.<IO system name>
  You cannot change this name directly. You change the PROFINET device name indirectly, by changing the name of the affected CPU, CP or IM in the general properties of the module. This PROFINET device name is also displayed, for example, in the list of accessible devices. If you want to set the PROFINET device name independently of the module name, you have to deactivate the "Generate PROFINET device name automatically" option.
- A "converted name" is generated from the PROFINET device name. This is the device name that is actually loaded into the device.
  The PROFINET device name is only converted if it does not comply to the rules of IEC 61158-6-10. You cannot change this name directly either.

Rules for the converted name

The rules for the converted name are listed in the following section. If the converted name is **not** different from the name of the module, the name of the module must comply with this rule.

- The name consists of one or more labels , which are separated by a dot [.].
- Restricted to a total of 240 characters (lower case letters, numbers, dash, or dot)
- A name component within the device name, which means a character string between two dots, must not exceed 63 characters.
- A name component consists of the characters [a-z, 0-9].
- The device name must not begin or end with the "-" character.
- The device name must not begin with a number.
- The device name form n.n.n.n (n = 0, ... 999) is not permitted.
- The device name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).

Example of device names

```
device-1.machine-1.plant-1.vendor
```

If you assign this name to a CPU, for example, STEP 7 will not convert it since it conforms to the rules described above.

Device number

In addition to the device name, a device number is also automatically assigned when an IO device is plugged in. You can change this number.

Devices in the PROFINET subnet

In a PROFINET subnet the maximum allowable number of devices is monitored during configuration.

See also

Assigning the device name and IP address (Page 1116)

Retentivity of IP address parameters and device names (Page 1124)

Assigning the device name and IP address

Assigning an IP address and subnet mask for an IO controller the first time

There are various options for this.

During parameter assignment of the PROFINET interface you must specify if the IP address is set in the project (which means in the hardware configuration) or if the IP address is to be set on the device.

Assignment of an IP address	Comments
Option "IP address is set in the project": The IO controller receives the IP address by loading the hardware configuration, for example, via one of the PROFINET interfaces, by means of the PROFIBUS interface or via the MPI interface.	When the hardware configuration is loaded to the IO controller (e.g. CPU), the IP address and the device name, if set, are also loaded. Example PROFINET interface: 1. Connect your programming device/PC to the same network as the relevant PROFINET device. The interface of the PG/PC must be set to TCP/IP (Auto) mode. 2. Have a list of accessible devices displayed. 3. Select the target device by using its MAC address and load the hardware configuration including the configured IP address (IP address is then saved retentively). If your PROFINET device has an MPI or PROFIBUS DP interface, connect your programming device/PC directly to the PROFINET device via the MPI or PROFIBUS DP interface. The configured IP address is applied when the hardware configuration is loaded.
Option "IP address is set directly at the device": • Assign online • Assignment by user program (instruction IP_CONFIG for S7-300/400, T_CONFIG for S7-1200/1500) • Assign via CPU display (S7-1500) • Higher-level IO controller makes assignment (only with I-devices)	If you have selected this option in the properties of the PROFINET interface, the IP address can be assigned by the online and diagnostics editor, by the primary setup tool, or by the user program ("IP_CONFIG" instruction). This option is set automatically if the option "Multiple use IO system" has been enabled in the properties of the PROFINET IO system (standard machine project). In case of an S7-1200-CPU, make sure that access to the CPU is not protected by a password. If the CPU is write-protected, no IP address and no device name can be assigned directly on the device.

Commissioning a PROFINET interface

Further details of how to commission a PROFINET interface can also be found in the operating instructions for the PROFINET devices of the SIMATIC family.

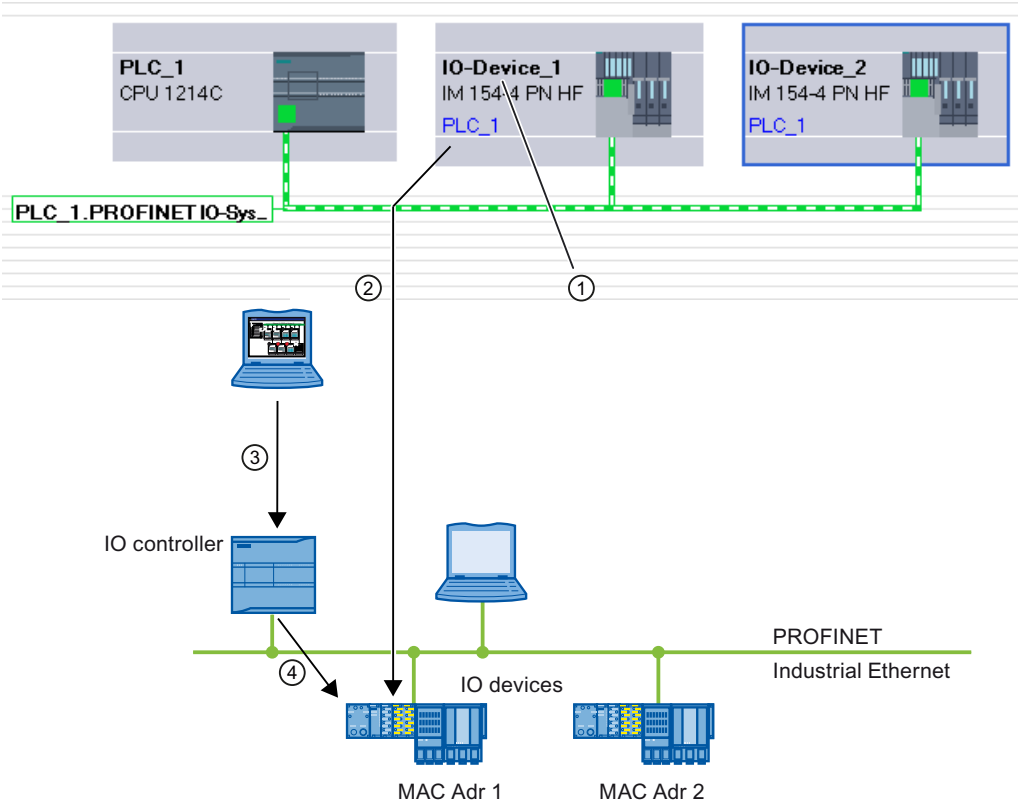
Assignment of device name for IO devices when the "Support device replacement without exchangeable medium" option is selected

For IO controllers with the "Support device replacement without exchangeable medium" option selected, you do not have to assign device names to the IO devices locally, for example in the event of device replacement. Another application is automatic commissioning, in which the CPU automatically assigns the device name and IP address parameters to the IO devices during startup.

Requirement: The ports of the devices are interconnected, and the devices involved support LLDP. The devices have been put into delivery state or - for S7-1500 CPUs version V1.5 and higher - the "Permit overwriting device names of all assigned IO devices" option is selected for the IO controller ("Ethernet addresses" area, "PROFINET" section of the properties of the PROFINET interface).

Assignment of device name and address for an IO device

The following graphic illustrates the process for assigning the device name and address. This process does not apply when the "Support device replacement without exchangeable medium" option is selected.



① Each device receives a name; STEP 7 automatically assigns an IP address.

② From the name, STEP 7 generates a PROFINET device name that you assign to an IO device online (MAC address) and that is written to the device.

③ You load the configuration to the IO controller.

④ The IO controller assigns the appropriate IP address to the IO device with the assigned PROFINET device name during startup.

Changing the device name and IP address

You can change the name and IP address manually. The device name must first be changed in the configuration in order to subsequently assign it to the IO device via the memory card or online with programming device/PC.

Offline with memory card:

1. Place the configured data (device name: for example, turbo-3) for the IO device on the MMC in the PG/PC. Use the command "SIMATIC Card Reader > Save Device Name to Memory Card" in the "Project" menu for this.
2. Then insert the MMC into the IO device. The IO device automatically adopts the configured device name.

Online with programming device/PC:

1. Connect the programming device/PC directly to the Ethernet subnetwork via the PROFINET interface.
2. In the network view, select the subnet or I/O device, and click the "Assign device name" command:
   - Either in the shortcut menu of the selected subnet / I/O device or
   - in the menu bar of the graphic view on the corresponding button.
3. In the "Assign PROFINET device name" dialog box, select the suitable PG/PC interface to connect to the Ethernet subnet. All configured PROFINET device names are in the top drop-down list. Select a PROFINET device name from it and select the IO device to receive this device name from the table at the bottom. You can filter the display of devices in the table according to various criteria.
4. You can easily identify the device using the "Flash LED" button.
5. Click "Assign name".

The IO controller recognizes the IO device by its device name and automatically assigns the configured IP address to it.

IP address assignment for special IO devices

Special IO devices, for example, SCALANCE X, S7-300 CPs, support the option of assigning the IP addresses not from the IO controller during startup. In this case, the IP address is assigned in a different way. The option is called "IP address is set directly at the device". For additional information, refer to the manual of the respective PROFINET device of the SIMATIC device family.

Another special case is the option "IP address is set by the IO controller during runtime" in the "IP protocol" area of the Ethernet address properties of an IO device. This option is set automatically when the option "Multiple use IO system" is selected for a standard machine project in the associated PROFINET IO system. In this case, an adapted IP address is not assigned by the IO controller until the IO controller itself has received a local IP address.

Requirement for additional procedures when assigning IP address and device name

If the IO device, as described above, should not obtain the IP address or device name from the IO controller, proceed as follows:

1. Select device or network view.
2. Open the properties of the respective PROFINET device and select the area "PROFINET interface [X1]" > "Ethernet addresses".
3. Select the option "IP address is set directly at the device" under "IP parameters" or the option "Permit setting of PROFINET device name directly on the device" under "PROFINET".

Rules

If the "IP address/device name is set directly at the device" option is used for a PROFINET device, note the following:

- The subnet part of the IP address of the IO device must match the subnet part of the IP address of the IO controller.
- The corresponding PROFINET device cannot be used as a gateway.

See also

Assigning a name in the online and diagnostics view opened via "Accessible devices" (Page 1392)

Enabling device replacement without exchangeable medium (Page 1137)

Example of the assignment of the device name

In this example you assign device names to a PROFINET IO controller and a PROFINET IO device. To make assignment easier, the device names should also contain the names of the PROFINET IO system.

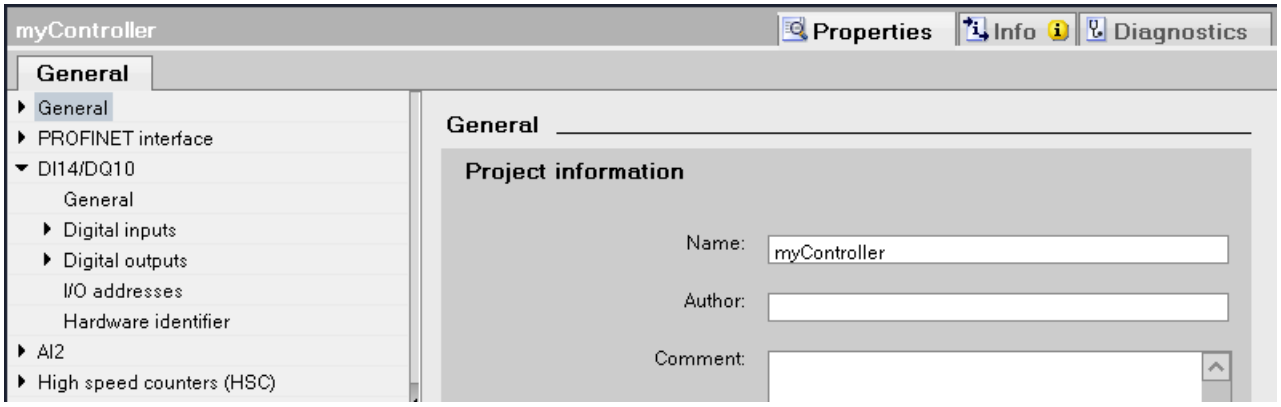
Requirement

- You must be in the network view.
- A CPU 1214C (V2.0 or higher) must be available in the network view.
- An interface module IM 151-3PN exists.
- The PROFINET interfaces of both modules are networked.

Procedure

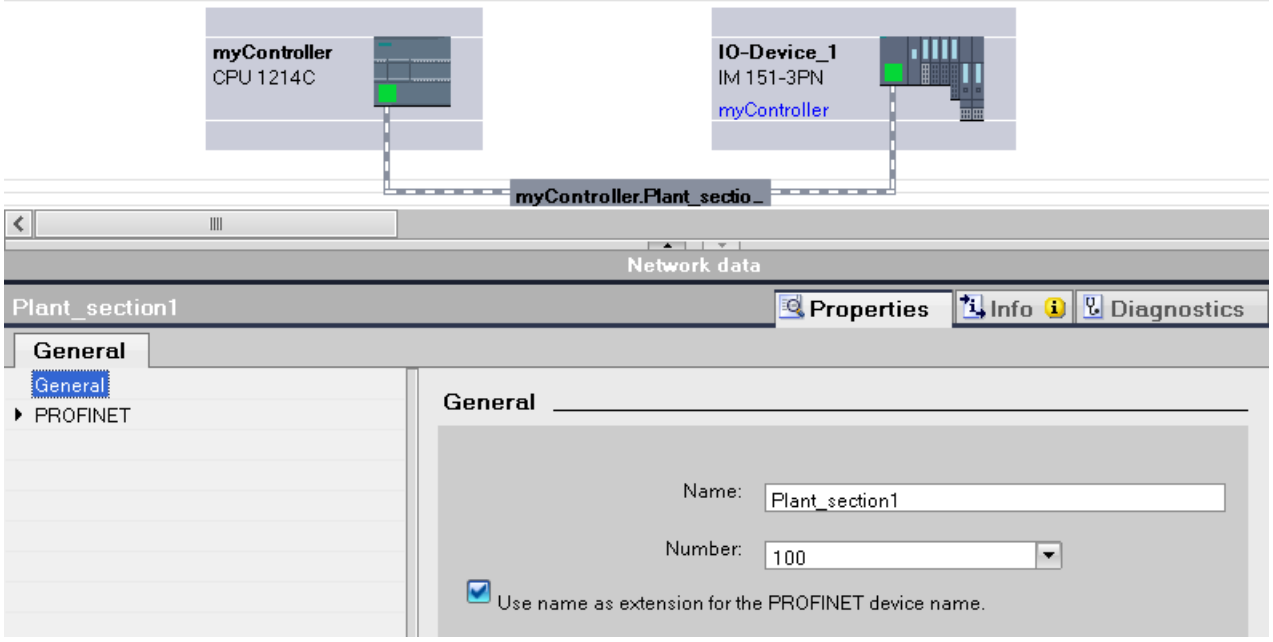
To assign the names, follow these steps:

1. Select the CPU.
   Make sure that you have selected only the CPU and not the complete device!
2. Assign the name "myController" in the Inspector window, under "General".

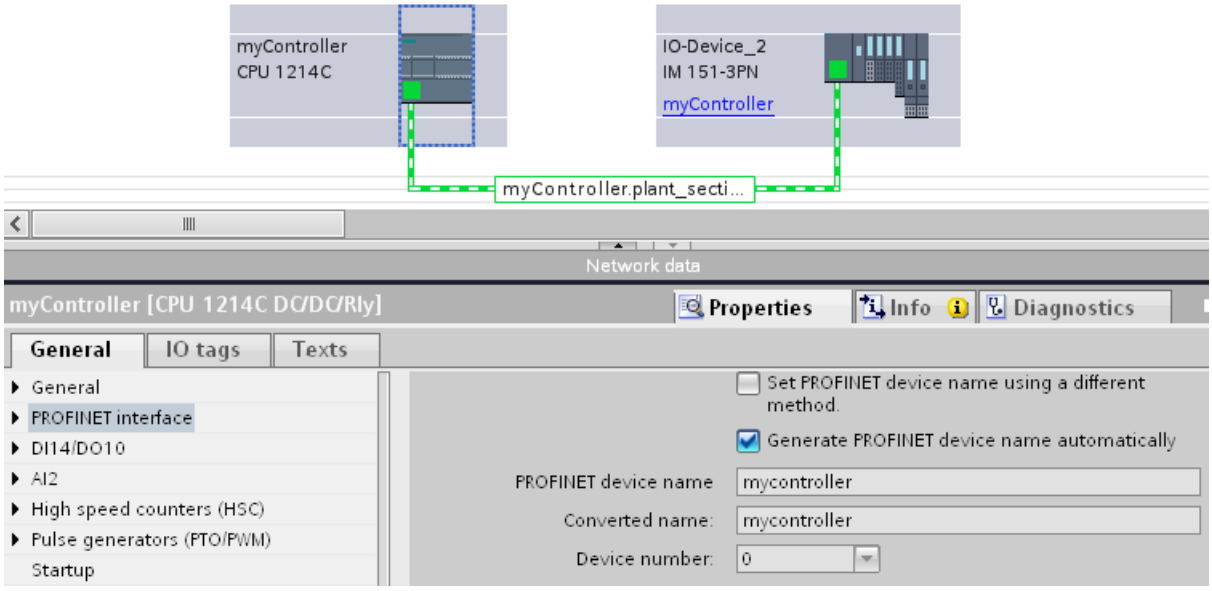


3. Select the interface module.
   Ensure that you have selected only the interface module and not the complete ET 200S device.
4. Assign the name "Device_1" in the Inspector window, under "General".
5. Right-click on the PROFINET IO system and select the "Properties" command.

6. Assign the name "Plant_section1" to the IO system and select the check box "Use name as extension for PROFINET device names".



7. You can find the automatically generated PROFINET device names at the selected device in the Inspector window, at "PROFINET interface".



The PROFINET device name corresponds to the name of the module (with the name of the IO system as extension) with the difference that only lower case text is used.
Background: No distinction is made between upper and lower case ("case insensitive") for the storing of the name.
If you want to specify the device name independently of the module name, you have to deactivate the "Generate PROFINET device name automatically" option. The PROFINET device name can be edited in this case.

The converted name is displayed below. This is the name that is automatically generated from the PROFINET device name and satisfies the DNS conventions. If you work with STEP 7, you do not require this name. This name is displayed here as a check and corresponds to the name that is stored in the device. If you work with other tools that are able to record the data exchange and read the actual device names, then you find the converted names.

Other special features

For PROFINET devices with multiple PROFINET interfaces, the name of the interface is attached to the name of the module, separated by a dot.

Example:

- Name of the module: myController
- Name of the interface: Interface_1
- PROFINET device name: mycontroller.interface_1

Assign device name via memory card

Introduction

You can configure the device names of PROFINET IO devices offline. To do this, store a configured device name on a memory card and then insert the card into the appropriate IO device.

If an IO device has to be completely replaced due to a device defect, the IO controller automatically reconfigures the new device. Using the memory card, a device can be replaced without a programming device.

Requirements

- The programming device has a card reader for memory cards.
- The IO device must support the assignment of the device name via memory card.
- The station and its PROFINET IO system is configured.

Procedure

To store a device name on a memory card, follow these steps:

1. Insert the memory card into the card reader.
2. Select the IO device whose device name is to be assigned by the memory card.
3. Select the "Card reader > Save Device Name to Memory Card" command in the "Project" menu.
   If the memory card is not empty, a message will be issued informing you of this and you will have the option to delete the card.

Retentivity of IP address parameters and device names

The retentivity of IP address parameters (IP address, subnet mask, router setting) and device name depends on how the address is assigned.

The non-retentive, temporary assignment means:

- IP address parameters and device name remain valid for the following time period:
  - Until the next POWER OFF
  - Until the next memory reset
  - Until termination of the online connection (for example, after loading the program) After POWER OFF / POWER ON or a memory reset, the CPU can only be accessed via the MAC address.

If the IP address parameters are not retentive, communication based on the IP protocol can no longer take place after the above described events (for example, after POWER OFF/ POWER ON).

The assignment of a temporary IP address also deletes retentively saved IP address parameters.

Assigning IP address parameters and device name non-retentively

IP address parameters and device name are not retentive in the following cases:

- A temporary IP address that is not retentive is implicitly assigned with the "Accessible devices" function, if the device (e.g. CPU) does not yet have an IP address.
- The device is a "normal" IO controller (i.e., not an I-device), and it is specified in the user program ("IP_Conf" instruction) that the IP address parameters/device name are not to be retentive.

Assigning IP address parameters and device name retentively

IP address parameters and device name are retentive in the following cases:

- In the properties of the PROFINET interface, it is specified that the IP address parameters are set in the project ("Set IP address in the project" option).
- In the properties of the PROFINET interface, it is specified that the IP address is to be set on the device.
  - Once the configuration is loaded, the IP address parameters and the device name are assigned via STEP 7 or a setup tool such as PST (STEP 7: online and diagnostic function "Assign IP address"). The assigned IP address parameters are retentive.
  - The device is a "normal" IO controller (i.e., not an I-device), and it is specified in the user program ("IP_Conf" instruction) that the IP address parameters/device name are to be retentive.

Special features of the I-device

In the properties of the PROFINET interface of the I-device, it is specified that the IP address parameters are to be set on the device. The IP address parameters for the I-device are assigned by the higher-level IO controller.

- If prioritized startup is set, the IP address parameters are retentive.
- If **no** prioritized startup is set, the IP address parameters are not retentive.

Recommendation

If possible, use the "Set IP address in project" option and specify an appropriate IP address. In this case, the IP address is assigned retentively.

Resetting of retentive IP address parameters and device names

Retentive IP address parameters and device names are reset via the online and diagnostic function "Reset to factory settings".

Note

Consequences of reassignment of IP address parameters on top of existing IP parameters

- The temporary assignment of IP address parameters/device names results in a reset of any retentively saved IP address parameters/device names.
- With a fixed assignment of IP address parameters/device names, previously retentively-saved parameters are replaced by the newly assigned parameters.

Note

Reuse of devices

Execute the "Reset to factory settings" before you install a device with retentive IP address parameters and device name in another subnet or system or before you place it in storage.

Creating a PROFINET IO system

A PROFINET IO system is comprised of a PROFINET IO controller and its assigned PROFINET IO devices.

To create a PROFINET IO system you require an IO controller (for example, CPU 1214C) and one or more IO devices (for example, a head module from the distributed I/O family ET 200S).

As soon as you connect an IO controller to an IO device, a controller-device link is established.

Procedure

To create a PROFINET IO system, proceed as follows:

1. Use drag-and-drop to pull an IO controller from the hardware catalog (for example, CPU 1214C) into the free area of the network view.
   The IO controller is created in the project.
2. Use drag-and-drop to move an IO device from the hardware catalog (for example, ET 200S) into the free area of the network view.
3. Click on the PROFINET interface of the IO controller or the IO device.
4. Hold down the mouse button and draw a connecting line between this selected interface and that of the partner device.
   A subnet with an IO system between the IO controller and the IO device is created.
5. If required, adapt the properties of the Ethernet subnet or the IO controller (for example, IP address) under "Properties" in the inspector window.

Handling PROFINET IO systems

Using shortcut menu commands, you can delete PROFINET IO systems, create new ones or even connect the interface to another subnet from within the network view.

An existing PROFINET configuration can thereby be corrected in the network view.

Create new PROFINET IO system for IO controller

To create a new PROFINET IO for an IO controller, proceed as follows:

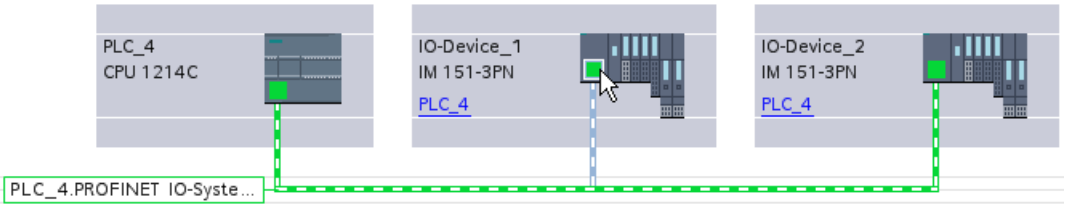
1. Make sure that no IO system is already assigned to the IO controller. If an IO system is already assigned to the IO controller, the "Assign IO system" shortcut menu command is disabled.
2. Select the PROFINET interface and then select the "Assign IO system" shortcut menu command.

A new PROFINET IO system is created at the IO controller and you can assign IO devices to this IO system.

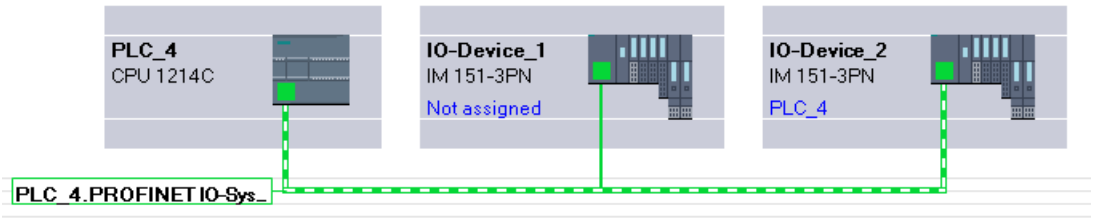
Disconnecting PROFINET IO devices from PROFINET IO system

To disconnect an already networked PROFINET IO device from its PROFINET IO system, follow these steps:

1. Click on the PROFINET interface of an IO device.



2. Select the "Disconnect from IO system" shortcut menu command.
   The IO device that was assigned to this IO system is then no longer assigned to it.



You can create a new IO systems and can assign each of the non-assigned IO devices to an IO controller.

Assign PROFINET IO devices to other IO controllers

Existing PROFINET IO systems can be easily reconfigured in the network view:

1. Select the interface of an IO device and then select the shortcut menu. You have the following options here:
   - Assign a new subnet to the IO device or disconnect it from the existing subnet
   - Assign a new IO controller to the IO device
   - Assign a new IO system to the IO device or disconnect it from the existing subnet
2. To assign another IO controller to the IO device, select the "Assign to new IO controller" shortcut menu command.
   If there is no connection, a subnet is automatically created and the IO device is assigned to the IO system of the new IO controller.

Tip: Quick configuration of IO systems

If the IO system has a lot of IO devices, assign all IO devices placed by drag-and-drop operation to an IO controller on one step.

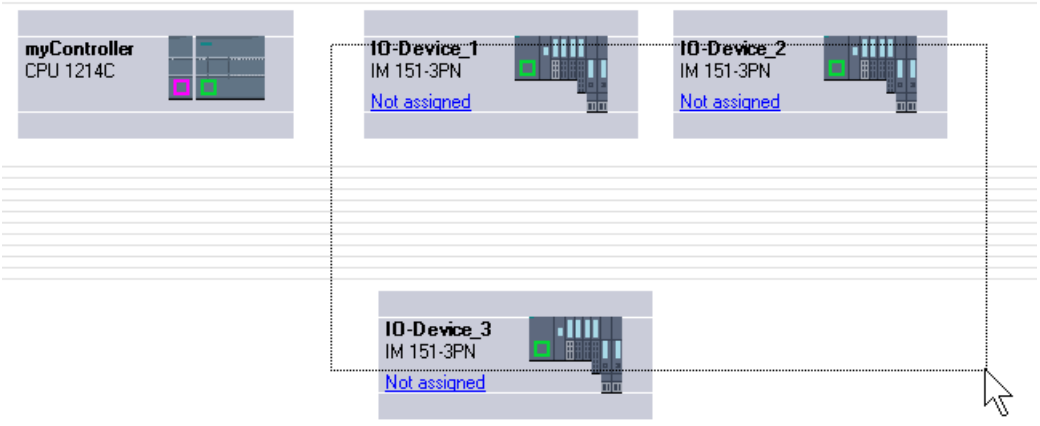
Requirements

IO controller and IO devices are placed in the network view.

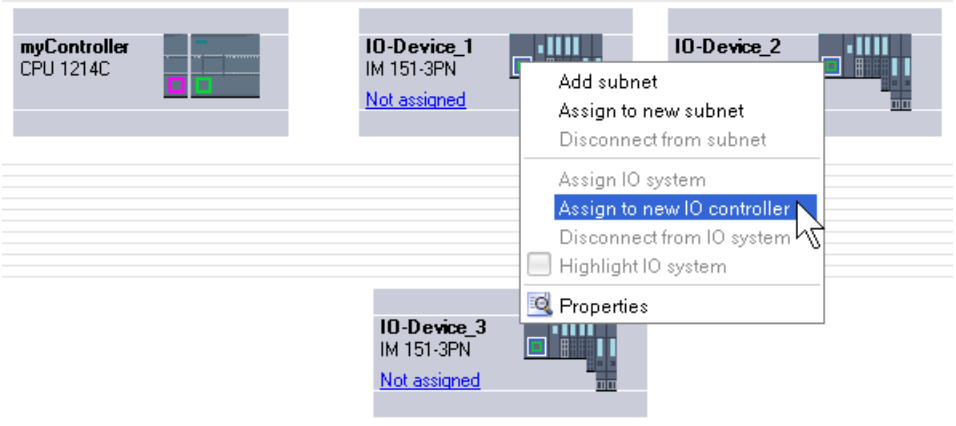
Assign IO devices to an IO system

To do this, follow these steps:

1. Select an appropriate zoom factor so that you can see as many IO devices as possible in the network view.
2. Arrange the IO devices in not more than of two rows.
3. Select all IO interfaces (not all devices) with the mouse cursor. This only works if you begin to drag the mouse cursor outside of the first IO device and release the mouse button at the last IO device (selection with the lasso).



4. Select the shortcut menu "Assign new IO controller" and select the corresponding IO interface of the IO controller in the subsequent dialog.



5. The IO devices are automatically networked with the IO controller and combine with it to form an IO system.

Note

When an IO system is highlighted, you can double-click on an IO device in the hardware catalog and thereby quickly add additional IO devices. Result: The IO device is automatically added to the highlighted IO system.

Interconnecting ports

If an IO device is assigned to an IO controller, this does not yet specify how the ports are connected to each other.

Although a port interconnection is not required to make use of the Ethernet/PROFINET functions, it does offer the following advantages:

- A target topology is specified with the port interconnection. Based on an online-offline comparison, it is possible to conduct a target-actual comparison with all devices that support this function.
- Only with IRT communication: If a port interconnection is configured, STEP 7 can determine the required bandwidth more precisely. As a rule, this leads to a higher performance.

Make sure that no invalid ring structures occur through the interconnection of ports.

Port interconnection is only advisable for devices that support the topology configuration.

Interconnecting ports in the Inspector window

To interconnect ports, follow these steps:

1. Select the Ethernet/PROFINET device or the Ethernet/PROFINET interface.
2. Navigate to the port property "Port interconnection".
   When the Ethernet/PROFINET interface is selected, you can find this setting in the Inspector window as follows: Properties > General > Advanced Options > Port [...] > Port Interconnection.
3. In the "Local port" section, you can find the settings at the local port. In the case of fiber-optic cable you can, for example, set the cable names here.
   In the "Partner port" section, click on the black triangle in the "Partner port" box to display and select the available partner ports.
4. If the port interconnection is a port interconnection with copper as medium and the devices support IRT communication, you can also set cable length and signal transit time.

If the Ethernet/PROFINET interface was disconnected, it is automatically networked by this action. In the properties of the subnet you can set whether this subnet should or should not be used for the networking.

Note

Interconnecting an electric with an optical port

If you want to interconnect an electric and an optical port, you have to decide between RT and IRT communication:

- With RT communication, it is not necessary to configure a media converter.
- With IRT communication, you have to make the interconnection via a media converter.

Information on monitoring the partner port

After you have interconnected the two ports, you receive information on the monitoring of the partner port in a text field. The following displays are possible:

- Monitoring of the partner port is not possible.
- Partner port is being monitored.

It is not possible to monitor the partner port if you, for example, select a deactivated port as partner port. In this case it is not possible to monitor the target topology and the signal propagation time. A device replacement is then only possible with a Micro Memory Card.

See also

Overview (Page 669)

Setting the send clock

Requirements to change the send clock at the PROFINET device

No IRT (Isochronous Realtime) should be configured. In detail, this means:

- No device must be configured at the IO system as a sync slave or sync master.
- All devices at the IO system must be unsynchronized.

If IRT is configured (in other words, if the IO controller is configured as sync master), the send clock can only be configured in the sync domain.

Procedure

To set the send clock on the PROFINET device, follow these steps:

1. Select the PROFINET IO controller in the device or network view.
2. Change the value for the shortest possible update interval in the properties of the PROFINET interface under "PROFINET Interface > Advanced options > Real-time settings > IO communication > Send clock".

The send clock is valid for all PROFINET devices at the IO system. If the synchronization role is set to a value other than "Unsynchronized", you can only set the send clock in the sync domain, in other words, centrally at the PROFINET IO system.

Setting the update time

Update time

An IO device / IO controller in the PROFINET IO system is supplied with new data from the IO controller / IO device within this time period. The update time can be separately configured for each IO device and determines the time interval in which data is transmitted from the IO controller to the IO device (outputs) as well as data from the IO device to the IO controller (inputs).

STEP 7 calculates the update time automatically in the default setting for each IO device of the PROFINET IO system, taking into account the volume of data to be exchanged as well as the set send clock.

Setting the update time

If you do not want to have the update time calculated automatically, you can change the setting.

To change the update time, proceed as follows:

1. Select the PROFINET interface of the IO device in the network or device view.
2. Change the update time in the interface properties under "Advanced options > Realtime settings > IO cycle".
   - To have a suitable update time calculated automatically, select "Automatic".
   - To set the update yourself, select "Can be set" and enter the required update time in ms.
3. if you want to keep the relationship between the send clock and the update time constant, enable the "Adapt update time when send clock changes" option.
   This option ensures that the update time is not set to less than the send clock.

Manual setting of the send clock may result in errors if the available bandwidth is not adequate or when other limits/configuration limits are exceeded (for example, too many nodes are configured).

No update time can be calculated

STEP 7 determines the sequence of the cyclic data exchange based on the configuration information (IO controller properties, IO device properties, number and type of the IO devices, consistency of the cyclic user data...). The cyclic data is packed in frames and sent/received successively at calculated time intervals.

The maximum number/size of the frames and the maximum number of intervals available must be sufficient to "accommodate" all the data. The resulting send/receive interval must also be supported by every PROFINET device.

If the limits relating to the amount of cyclic user data/number of frames or relating to the intervals available are exceeded, STEP 7 cannot calculate an update time.

If there is no common basis for the send/receive interval, calculating the update time is also not possible.

If there is a reason preventing the calculation of the update time, STEP 7 reports the cause when compiling the hardware configuration.

How to eliminate the problem:

- Reduce the number of IO devices
- Reduce the number of modules within the IO devices
- If you are using an IE/PB link: Reduce the number of DP slaves downstream from the IE/PB link
- Use a more powerful IO controller or IE/PB link
- Increase the send clock
  - With RT: in the properties of the IO controller
  - With IRT: in the properties of the sync domain
- Check the device properties of the IO devices ('MinDeviceInterval' and the possible scan rates) due to the common basis for the send/receive interval. Replace unsuitable IO devices. This device properties are stored in the GSD file of the IO device.
- With IRT configuration:
  - Check whether the ports of the sync master and sync slaves are interconnected.
  - Check the order of the IO devices: There can be no unsynchronized device connected between the sync master and sync slave
    (Example of bad configuration: Sync master --- unsynchronized device --- sync slave).
  - Check whether you have configured more than one sync master.
  - Check the bandwidth remaining for RT data. The bandwidth for RT data available for the transfer can be restricted by IRT communication on the same Ethernet subnet.
- When using I-devices:
  It may not be possible to use the set send clock together with the existing I-device configuration.
  - Configure the I-device without lower-level IO devices and activate the setting "Parameter assignment of PN interface by higher-level IO controller".
  - Change the send clock of the IO controller to an even value (... 0.250, 0.500, 1.000, ...).

Identification of the IO devices involved:

You can identify the IO devices involved for which no update time can be calculated in the "I/O communication" table of the PROFINET IO system in the network view. No entry is made in the "Update time" column for the IO device involved (entry "-").

Setting the watchdog time

Watchdog time

You can configure the watchdog time for PROFINET IO devices.

If the IO device is not supplied with input or output data (IO data) by the IO controller within the watchdog time, it switches to the safe state.

Do not enter the watchdog time directly, but as "Accepted number of update cycles when IO data is missing". This makes setting easier because the update time can be shorter or longer, depending on the power of the IO device or the setting.

The resulting watchdog time is automatically calculated from the "Accepted number of update cycles when IO data is missing".

Configuring the watchdog time

To specify the watchdog time, follow these steps:

1. Select the PROFINET interface of the IO device in the network or device view.
2. In the properties of the interface, navigate to "Advanced options > Realtime settings > IO cycle".
3. Select the required number of cycles from the drop-down list "Trigger watchdog after # cycles with missing IO data".

The watchdog time is subsequently calculated automatically based on the preset factor. It must not be more than 1.92 seconds.

Note

The default setting should only be changed in exceptional cases, for example, during the commissioning phase.

Calculated bandwidth for cyclic IO data

Calculated bandwidth for cyclic IO data

Adherence to the maximum available bandwidth for cyclic IO data is monitored by the system. The maximum bandwidth depends on the send clock cycle. If the send clock cycle is greater than or equal to 1 ms, the maximum bandwidth is 0.5 ms. If the send clock cycle is shorter, the maximum available bandwidth is also reduced.

The bandwidth actually required for cyclic IO data is determined by the system based on the number of configured IO devices and IO modules. Furthermore, the required bandwidth depends on the update time that is used.

In general, the calculated bandwidth increases in the following cases:

- There is a greater number of IO devices
- There is a greater number of IO modules
- The update times are shorter.

Maximum bandwidth for cyclic IO data depending on the send clock

The following table shows how the maximum available bandwidth for cyclic IO data reacts based on the send clock:

Send clock cycle	Maximum bandwidth for cyclic IO data
250 µs – 468.75 µs	<< 125 µs
500 µs – 968.75 µs	= send clock / 2
1 – 4 ms	= 500 µs

Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission medium/duplex

Depending on the selected device, you can make the following settings for "Transmission medium/duplex":

- Automatic setting
  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the partner port. The "Enable autonegotiation" option is automatically selected by default.
- TP/ITP at x Mbps full duplex (half duplex)
  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:
  - Autonegotiation enabled
    You can use both cross cable and patch cable.
  - Autonegotiation disabled
    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.
- Deactivated
  Depending on the module type, the drop-down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option is used to activate or deactivate the port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because, with this option, the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can therefore not be optimally set.

Note

When a local port is interconnected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not support the setting, an error message is generated.

GBIT PROFINET interface

The PROFINET interface (X3) of the CPU 1518-4 PN/DP supports a maximum transmission rate of 1000 Mbps (GBIT).

In order to achieve this transmission rate, the following requirements must be met:

- CPU Firmware Version V1.7 or higher.
- Devices on the same PROFINET subnet must also support the 1000 Mbps transmission rate.
- The network infrastructure (network cables and outlets) must be category CAT 5e or better.
- The port options of PROFINET interface X3 must be set as follows:
  - “Transmission rate/Duplex”: Automatic
  - “Autonegotiation”: Activated

See also

Wiring rules for disabled autonegotiation (Page 1135)

Boundaries at the port (Page 1136)

Wiring rules for disabled autonegotiation

Requirement

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission speed
- Autonegotiation incl. autocrossing disabled

This saves you the time required to negotiate the transmission rate during startup.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

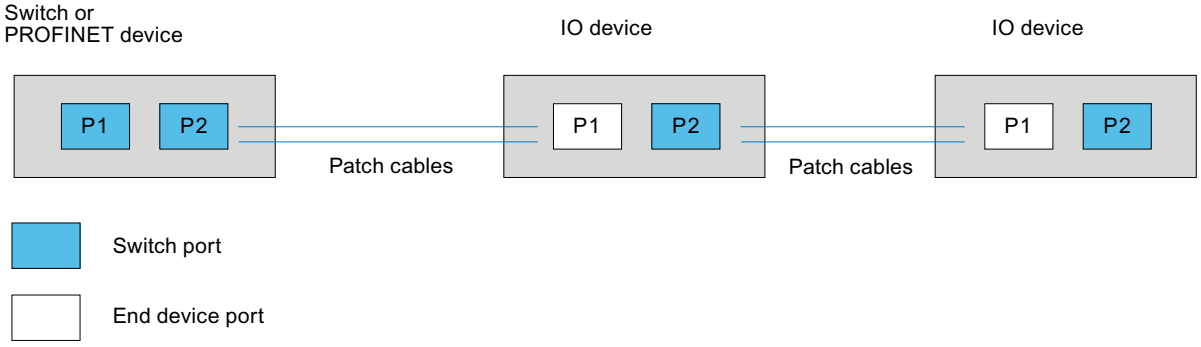
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in a line using a patch cable (one-to-one wiring of both connectors). To do this, you connect port 2 (P2) of the IO device to port 1 (P1) of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirement

To use boundaries, the respective device must have more than one port. If the device for PROFINET does not support boundary settings, the corresponding parameters are disabled.

This applies, for example, to the 1215C CPUs V3; although these have more than one port, they do not support boundary settings.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of detection of accessible devices"
  DCP frames for detection of accessible devices are not forwarded. Devices located behind this port are no longer displayed in the project tree under "Accessible devices". Devices located behind this port can no longer be reached by the CPU.
- "End of topology discovery"
  LLDP frames (Link Layer Discovery Protocol) for topology detection are not forwarded.
- "End of sync domain"
  Sync frames transmitted for synchronization of devices within a sync domain are not forwarded.
  For example, if you operate a PROFINET device with more than two ports in a ring, you should prevent the sync frames from being fed into the ring by setting a sync boundary (at the ports not inside the ring).
  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device of the other sync domain.

Restrictions

The following restrictions must be observed:

- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been defined for the port, the following check boxes cannot be used:
  - "End of detection of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

Enabling device replacement without exchangeable medium

Replacing an IO device without exchangeable medium

It is often necessary to replace IO devices in automation systems. The IO devices are generally assigned a device name by either inserting an exchangeable medium or via the programming device. The IO controller uses this device name to identify the IO device.

Subject to certain conditions, IO devices can also receive their device names without the insertion of an exchangeable medium (e.g., memory card) or without a programming device. For this purpose, the IO controller uses Ethernet mechanisms (LLDP protocol; Link Layer Discovery Protocol) to analyze the relationships between the individual IO devices and the IO controller. From these relationships, the IO controller detects which IO device was replaced and assigns the configured device name to it.

Requirements

- A port interconnection is already configured.
- The affected IO devices in the automation system must support device replacement without exchangeable medium (LLDP protocol).
  If the individual IO devices in the automation system do not support device replacement without exchangeable medium, a corresponding alarm is output for the IO device.

Note

Use only new IO devices as replacement devices, or restore IO devices with an existing parameter assignment to delivery state prior to commissioning.

For S7-1500 CPUs with firmware version V1.5 or higher, it is not necessary to reset IO devices with an existing parameter assignment to delivery state. The condition for this is that the "Permit overwriting of device name" option is enabled for the IO controller ("Ethernet addresses" area, "PROFINET" section of the properties of the PROFINET interface).

Procedure

In order to enable the replacement of an IO device without exchangeable medium, proceed as follows:

1. In the device or network view, select the PROFINET interface of the corresponding IO controller.
2. In the interface properties under "Advanced settings > Interface options", select the "Support device replacement without exchangeable medium" check box.

The option "Support device replacement without exchangeable medium" also permits automatic commissioning, which means you can commission the IO system with the IO devices without assigning their device names in advance.

See also

Assigning the device name and IP address (Page 1116)

Components with the the device replacement without exchangeable medium function (http://support.automation.siemens.com/WW/view/en/36752540)

Connect the DP slave via the IE/PB Link to a PROFINET IO system

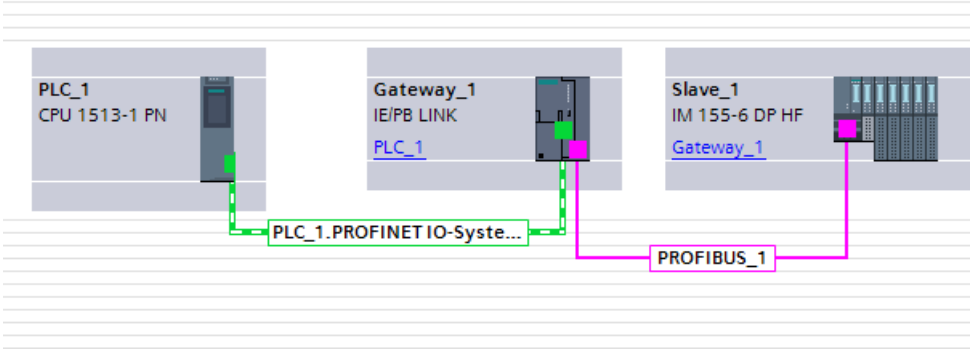
Requirements

- STEP 7 as of V12
- S7-1500 CPU as of firmware version 1.7
- ET 200SP CPU as of firmware version 1.7
- S7-1500 software controller
- S7-300/400 CPU

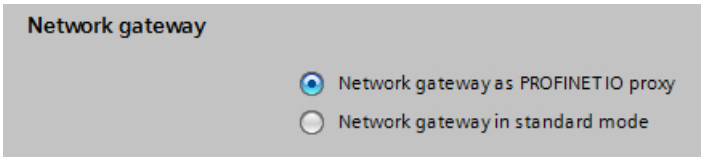
Procedure for connecting a DP slave via an IE/PB Link

To connect a DP slave to a PROFINET IO system via an IE/PB Link in STEP 7, follow these steps:

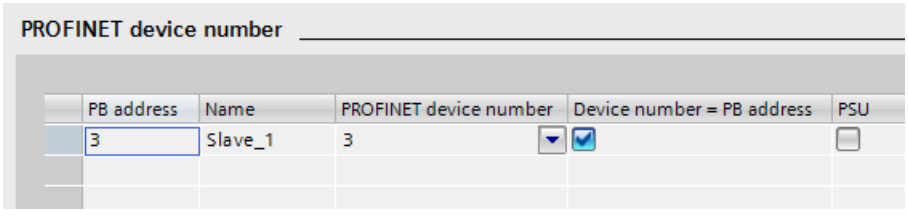
1. Drag a PROFINET CPU, e.g. 1513-1 PN, from the hardware catalog to the network view of STEP 7.
2. Drag an IE/PB Link PN IO from the hardware catalog to the network view of STEP 7. You will find the IE/PB Link PN IO under Network components > Gateways > IE/PB Link PN IO.
3. Assign the IE/PB Link PN IO to the CPU.
4. Drag a PROFIBUS interface module e.g. IM155-6 DP HF, from the hardware catalog to the network view.
5. Assign the interface module to the IE/PB Link.



6. Select the IE/PB Link PN IO in the network view of STEP 7.
7. In the Inspector window, go to the "Gateway" area and select the "Network gateway as PROFINET IO proxy" option.



8. In the PROFINET device number area, you can assign a PROFINET device number for the DP slave.
   If you have selected the "Device number = PB address" check box (default), STEP 7 automatically assigns the device number according to the PROFIBUS address of the slave. In addition, you no longer need to update the device number if the PROFIBUS address changes.



See also

Configuration using IE/PB link PN IO (Page 1112)

Using GSD files

GSD files for IO devices

Basic information on GSD files of IO devices

The properties of PROFINET IO devices are not stored in a keyword-based text file (as for PROFIBUS DP slaves), but in an XML file whose structure and rules are determined by a GSDML schema.

The language used to describe the GSD files is GSDML (Generic Station Description Markup Language). It is defined by the GSDML schema.

A GSDML schema contains validation rules that allow it, for example, to check the syntax of a GSD file. GSDML schemas (as schema files) are acquired by IO device manufacturers from PROFIBUS International.

Functional enhancements in the area of PROFINET IO will have an effect on the GSDML specification and the corresponding schema. A new version of the specification and of the schema is created by the functional enhancement.

Names of GSD files for IO devices

One possible example of a GSD file name for IO devices is:

"GSDML-V1.0-Siemens-ET200S-20030616.xml"

Name component	Explanation
GSDML	String at the start of each GSD file for IO devices
V1.0	Version of the GSDML schema
Siemens	Manufacturer
ET200S	Name of the device
20030616	Version code (date)
.xml	File extension

Versioning of GSD files for IO devices

The version information of GSD files is two-fold:

First, the version of the GSDML schema is indicated. This determines the language scope used by a GSD file.

This is followed by the version, listed as an issue date. The version number of GSD files is incremented, for example, after elimination of an error or introduction of a functional enhancement.

Functional enhancements may result in a new version of the GSDML schema. A new version of a GSDML schema might only be supported with restrictions.

Installing the GSD file

Introduction

A GSD file (general station description file) contains all properties of an IO device. If you want to configure an IO device that is not available in the hardware catalog, you must install the GSD file provided by the manufacturer. IO devices installed via GSD files are displayed in the hardware catalog and can then be selected and configured.

Requirement

- The hardware and network editor is closed.
- You have access to the required GSD files in a directory on the hard disk.

Procedure

To install a GSD file, follow these steps:

1. In the "Options" menu, select the "Install general station description file (GSD)" command.
2. In the "Install general station description file" dialog box, select the folder in which the GSD files are stored.
3. Choose one or more files from the list of displayed GSD files.
4. Click on the "Install" button.
5. To create a log file for the installation, click on the "Save log file" button.
   Any problems during the installation can be tracked down using the log file.

You will find the new IO devices installed by means of GSD files in the hardware catalog under "Additional field devices > PROFINET".

See also

Overview of hardware and network editor (Page 535)

Deleting GSD file

Introduction

You can delete installed DP slaves using GSD files. These DP slaves are then no longer displayed in the hardware catalog.

Requirement

- The hardware and network editor is closed.
- You will find the new IO devices installed by means of GSD files in the hardware catalog under "Additional field devices > PROFINET".

Procedure

To delete a GSD file, follow these steps:

1. In the "Options" menu, select the "Install general station description file (GSD)" command.
2. In the "Install general station description file" dialog box, select the folder in which the GSD file is stored.
3. Select the file that is to be deleted from the list of displayed GSD files.
4. Click the "Delete" button.

The selected GSD file was deleted and the DP slave is no longer located in the hardware catalog.

Changing the revision of a GSD file

Changing the revision of a GSD file

You can change the revision of a GSD file for an IO device:

- Only for the current IO device
- All suitable IO devices within the IO system
- All suitable IO devices within the complete project

First, all existing GSD files for the current IO device are shown. The only difference between the GSD files shown is their revision status. The currently used GSD file is highlighted.

Requirement

- The I/O data is the same for all IO devices whose revision is to be changed.
- The article number has not changed.
- The number of submodules is identical.
- The configuration data has not changed.
- There must be no module or submodule in a slot that is invalid after the new GSD file has been created.

Procedure

To change the revision of one or more IO devices, proceed as follows:

1. Select the IO device whose GSD file revision is to be changed.
2. Click on the "Change revision" button under "General> Catalog information" in the properties of the IO device.
   The "Change revision" dialog box opens.
3. Select the GSD revision you want to use in the "Available revisions" table.
4. Under "Use selected revision for", select the devices whose version are to be changed:
   - Only for the current IO device
   - For all suitable IO devices in the IO system
   - For all suitable IO devices in the project
5. Click the "Apply" button.

10.1.4.8 Bus coupling with PN/PN coupler

Application and function

Application

The PN/PN coupler is used to link two Ethernet subnets with one another and to exchange data. That way use data about input or output address areas or datasets can be used. The maximum size of the transferable input and output data is 1024 bytes. The division into input and output data is preferable, so that e.g. 800 byte input data and 200 byte output data can be configured.

As a device, the PN/PN coupler has two PROFINET interfaces, each of which is linked to one subnet.

In the configuration, two IO Devices are produced from this one PN/PN coupler which means that there is one IO Device for each station with its own subnet. The other part of PN/PN coupler in each case is known as the bus node. Once configuring is complete, the two parts are joined.

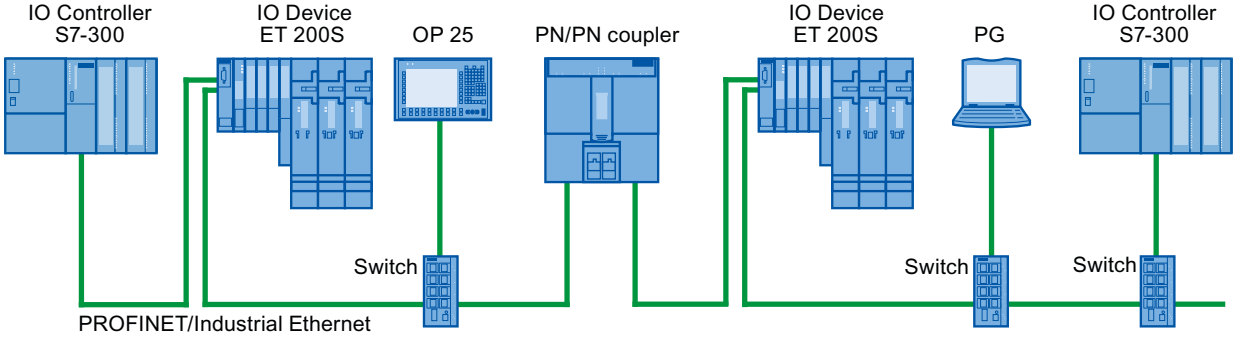


Figure 10-3 Coupling two PROFINET IO subnets with one PN/PN coupler

Additional information

For additional information on "PN/PN couplers", refer to Service & Support on the Internet (http://support.automation.siemens.com/WW/view/en/44319532).

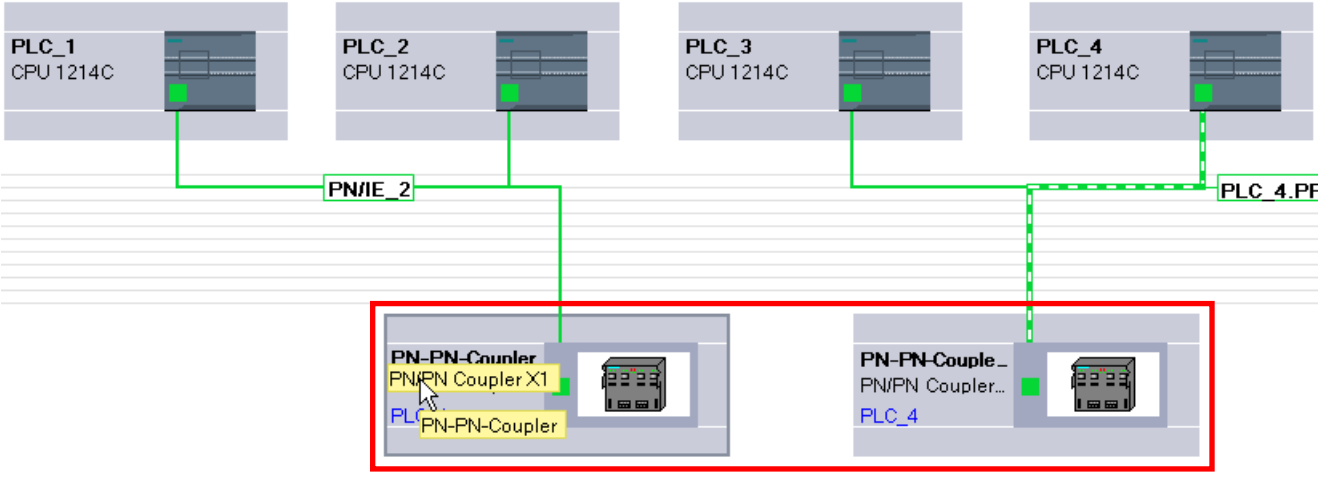
Linking Ethernet subnets

Linking Ethernet subnets with a PN/PN coupler

You can link Ethernet subnets with the standard device PN/PN coupler.

To link Ethernet subnets, follow these steps:

1. Create your Ethernet subnets.
2. Select the standard field devices in the hardware catalog. Find the PN/PN coupler as head module in the "PROFINET IO" folder.
3. In the network view, drag the two components X1 and X2 to the required version of the PN/PN coupler per drag-and-drop operation. The components form a device, but are shown separately to make handling easier.
4. Connect the Ethernet interface of the PN/PN coupler X1 to the first Ethernet subnet.
5. Connect the Ethernet interface of the PN/PN coupler X2 to the second Ethernet subnet. The Ethernet subnets are now linked through the two components of the PN/PN coupler.



10.1.4.9 Integrating external tools

Integrating S7-external tools

Introduction

Tools external to STEP 7 ("Device Tools") with a special call interface (Tool Calling Interface) can be used to configure distributed devices. Such devices are also referred to as "TCI capable".

The performance range of these tools exceeds the possibilities provided within GSD configuration, for example, they can provide expanded graphical input options.

Distributed devices can be as follows:

- PROFIBUS DP slaves
- Modules within a DP slave
- PROFINET IO devices
- Modules within an IO device

Note

Warranty and liability

Siemens accepts no liability for third-party software (device tools) called with the TCI (Tool Calling Interface) or for proper interaction with the associated devices.

Requirement

The call interface of the tool complies with the TCI specification. Parameters and commands are forwarded to the distributed device via this call interface.

Such tools have to be installed using a setup provided by the manufacturer. The "S7-PCT" (Port Configuration Tool) device tool for IO-Link master modules and IO-Link devices is an exception; this is supplied with STEP 7. Special note: After the installation, the tool is not shown in the list of installed software or in the list of software products in the project.

The GSD file of the distributed device that is to be configured with the Device Tool must be installed.

Starting the device tool

The command for starting the device tool is available in the shortcut menu of the TCI-capable device in the shortcut menu of the graphical and tabular device view: "Start device tool".

See also

Starting the SIMATIC S7-PCT (Page 1145)

Starting the SIMATIC S7-PCT

Introduction

The "S7-PCT" (Port Configuration Tool) device tool is installed with STEP 7.

The tool is used to assign parameters to the ports of IO-Link modules such as 4SI IO-Link (S7-1200, ET 200S) or 4IOL+8DI+4DO (ET 200eco PN).

Requirement

You have configured the corresponding CPU, the DP slave or the IO device with an IO-Link module.

Procedure

To start via the graphical device view, follow these steps:

1. Select the IO-Link module in the device view.
2. Select "Start device tool" from the shortcut menu.

OR, to start with the tabular device view, follow these steps:

1. Select the IO-Link module in the device view.
2. Arrange the areas in the work area in such a way that the tabular device overview is visible (it is located between the device view and the inspector window).
3. Select the row with the IO-Link module in the device overview.
4. Select "Start device tool" from the shortcut menu.

Result

The tool starts and you can configure the ports.

See also

Integrating S7-external tools (Page 1144)

10.1.4.10 Loading a configuration

Introduction to loading a configuration

In order to commission a device, identical configurations must be stored on the programming device/PC and on the connected devices. To synchronize the configurations on the programming device/PC and the connected devices, you load a configuration. Configuration data can be loaded in two directions:

- From the programming device/PC to a device
- From the device to a PG/PC

See also

Uploading project data from a device (Page 399)

General information on loading (Page 394)

Downloading a configuration to a device (Page 1147)

Downloading project data to a device (Page 396)

General information on loading to PG/PC (Page 1148)

Special features during startup (Page 1167)

Downloading a configuration to a device

Loading the hardware configuration

After you have inserted a new device in the project and configured it or if you have modified an existing hardware configuration, the next step is to load the current configuration to the device. This ensures that the same configuration is set on the programming device/PC and on the module that is physically present. Use the "Online > Download to device (Page 396)" menu command for this.

In the first download, the complete hardware configuration is downloaded. In subsequent downloads, only changes to the hardware configuration are downloaded.

You have the following options for loading the hardware configuration:

- Loading in the device or network view
- Loading in the project tree
- Loading to an accessible device

Warning

Perform load operation only in STOP mode

Following loading, the machine or process may behave unexpectedly if the parameter assignment is incorrect. A CPU must be set to STOP mode for the load operation to rule out possible damage to equipment or personal injury.

Special considerations for loading isochronous applications

Isochronous applications consist of a hardware configuration part and a software part.

Example: If you change the number of an IO system, the delay time, or the assignment of a process image partition of the isochronous I/O in the hardware configuration, this affects the parameters of the isochronous mode interrupt OB and thus also the software part.

With isochronous applications, you should always load the complete project (hardware and software). With partial loading (loading hardware and software separately at different times), inconsistencies can arise that, for example, can prevent CPU startup or isochronous operation of the application.

See also

General information on loading (Page 394)

Loading a configuration to PG/PC

General information on loading to PG/PC

Introduction

If you bring your PG/PC to a plant and the STEP 7 project used to create the configuration of this plant is not available, load the configuration to a new project on your PG/PC, for example. Use the "Online > Loading the device as new station (hardware and software) (Page 399)" menu command for this.

The list of accessible devices in the project tree is always used when loading a device to your programming device. You can select multiple devices and load them to the project at the same time.

Requirements

- The hardware configuration in the device must be created in TIA Portal, V12 or higher. A hardware configuration present in the device that was created with an older version cannot be loaded and must be upgraded (Page 380).
- Modules present in the device from GSD (ML), HSPs, or service packs must be installed in TIA Portal on the PG/PC.
- A project must be open. This project can be a new (empty) project or an existing project.
- The opened project is in offline mode.

Scope of load operation

The following list shows an overview of the loadable components of a configuration:

- The device (e.g., a CPU) with all I/O modules and all parameter settings
- PROFIBUS master systems and all PROFIBUS-relevant settings
- PROFINET IO systems and all PROFINET-relevant settings
- I devices and I slaves
- Settings for direct data exchange

After loading a CPU, all other modules within the address area of the CPU are also loaded.

The following connections are also loaded when the configuration is loaded:

- S7 connections (including routed connections) in combined PB/IE networks, including via IE-CP or PB-CM interfaces. S7 connections are automatically accepted as configured at one end when a device configuration is loaded, even if the S7 connection was configured at both ends in the original project. When both connection partners are loaded, the connection is joined together again during the next compilation.
- TCP connections via CPU-internal Ethernet interface, UDP/ISOonTCP connections, and TCP, UDP, ISO, and ISOonTCP connections via IE-CP interface
- Connections via the OUC connection parameter assignment for projects from STEP 7 V13 and higher

Note

The hardware configuration loaded to PG/PC is not completely identical to the configuration originally loaded to the device. Note the additional information on loading, in particular with regard to partially loaded configuration data in the case of cross-device communication.

Loading PC systems, such as WinAC or PC-based automation, is not possible.

See also

General information on loading (Page 394)

Loading specific device configurations

Information on loading

When device configurations are loaded to the PG/PC, all assigned parameters are transferred from the device to the project. If the CPU is connected to a subnet, all parameters of the device are loaded and the CPU is displayed in the network view as networked.

Note

CPUs reset to their factory settings do not have any hardware configuration. Therefore, nothing is loaded in this case after "Load to PG/PC" is selected in the "Online" menu.

Loading S7-300/400 configurations

To avoid conflicts when loading a device to an existing project, the following rules must be followed:

- Device names for CPUs, PROFIBUS slaves (DP slaves, I-slaves) and PROFINET devices (IO devices, I devices) must be unique
- The combination of network name, subnet ID, and IP/DP address for modules must be unique

If conflicts occur, the load operation is canceled and an alarm provides information about the problems that occurred. You can then adapt the project correspondingly or install missing components and then repeat the load operation.

Alarm configurations are not loaded to PG/PC.

Loading S7-1200/1500 configurations

Take the following into consideration when loading:

- CPUs of the S7-1200 series with firmware version V1.0 are not supported when loading.
- Device-specific diagnostic interrupts of the S7-1200 are not supported when loading. For device-specific diagnostic interrupts of the S7-1200 to be generated on the PG/PC once again, the hardware configuration must be compiled once again.
- Module comments of the S7-1200/1500 are loaded from the device to the PG/PC if the same project language is set that was used during loading to the device. You can deselect the loading of comments, if desired.

Note

All types of PC systems, such as WinAC, Embedded Controller, CP 1616 or PC CPs, do not support loading to PG/PC.

Loading distributed I/O

The following functionalities and settings of the distributed I/O are loaded:

- DP master systems/IO systems with associated DP masters/IO controllers (CPUs and CPs), DP slaves/IO devices, the utilized modules and their parameter assignment and properties, such as options handling, status bytes, or SYNC/FREEZE
- Connection of process image partitions (PIP) to organization blocks (OB). Applies to the module and OB properties
- Configured hardware interrupts with the associated properties
- DP master systems with I-slave
- CPs as PROFIBUS I-slave or PROFINET I-device
- Direct data exchange

Master-slave relationships between the I-slave/I-device and assigned DP master/IO controller are only established in the project if both the master as well as the I-slave are loaded to the PG. It does not matter whether you load the DP master/IO controller or the I-slave/I-device first. As soon as both devices are loaded, the master-slave/controller-device relationships will also be re-established.

Loading subnets and devices with MPI, PROFIBUS, Ethernet, and PtP

The following special considerations apply to loading subnets and end points of connections for MPI, PROFIBUS, Ethernet, and PtP with their respective connection properties:

- If a device with PROFIBUS interface is loaded, the bus parameters of the device will initially be different from the settings in the original project. The bus parameters will only match those of the original project after all devices involved are loaded and no additional devices are on the same bus.
- Passive communication devices that are not connected as DP slaves or IO devices to a corresponding master system or IO system do not participate in the data exchange. They are therefore not loaded.
- All devices involved should therefore be loaded for cross-device configurations. A warning is output during compiling of the project for missing network stations. Missing routing information due to communication stations that are not loaded is displayed as warning during compilation. If the configuration is loaded from the PG/PC back to the device, different routing information results.

When you compile the project after loading devices to the PG/PC, STEP 7 checks if all devices to which communication relationships have been configured are available. If devices are missing, you receive an alarm with the number of missing communication stations.

Notice

Cross-device communication

When you load a configuration with cross-device communication to the PG/PC, you must also load all corresponding network stations to the PG/PC. If required network stations are missing and the configuration is loaded back to the device, there is no guarantee that cross-device communication is working again.

See also

Upgrading projects (Page 380)

Loading configurations with web server

Information on loading

The hardware configuration of a CPU also contains the Web server settings. A number of restrictions apply to loading a Web server configuration to the PG/PC:

- The assignment of the Web server language and project language is not loaded for S7-300/400. The project texts are not loaded and an alarm is output that no project languages are assigned. The languages assigned in STEP 7 are loaded without restriction for S7-1200/1500 CPU.
- User administration data of the S7-1200/1500 can be loaded but not edited. You can use a check box to select whether you want to use the existing data as read-only data or discard these data and enter new data.
- Watch tables of the Web server are not loaded.

The source files of the user-defined web pages (HTML pages, Java scripts, etc.) are not loaded. The program blocks generated during loading can only be edited if you enter the properties and the HTML page yourself.

See also

Information about the web server (Page 853)

Loading configurations with PROFIBUS

Information on loading

A DP master is loaded to PG/PC. The DP master system and all connected DP slaves are inserted into the project. The respective settings remain unchanged. If a suitable PROFIBUS subnet has already been created, the loaded devices with PROFIBUS interface are connected to the existing subnet.

As a prerequisite for loading DP master systems with standard slaves, the corresponding GSD files must be installed in the TIA Portal and available in the hardware catalog. If a required GSD file is not available in the same version as in the device, differences are identified during the consistency check.

Note

Direct data exchange in a configuration is only loaded if all communication partners involved in the direct data exchange are loaded to the PG/PC.

Isochronous mode

Note the following when loading DP master systems with activated PROFIBUS functionality "Isochronous mode":

- Bus parameters and the settings for isochronous mode are only identical after the device is loaded to the PG/PC if all devices relevant for calculation of isochronous mode are loaded.
- Only mono-master systems with isochronous mode are supported. Therefore, only configurations with just one DP master on the PROFIBUS subnet are loaded.

I-slave

Master-slave relations between DP master and I slave are only established in the project if both the DP master and the I slave are downloaded to the PG. It does not matter whether you load the DP master or the I-slave first. If you load the DP master from a DP master system with connected I-slave, the DP master and its DP slaves are loaded. An I-slave proxy is loaded as dummy module for an involved I-slave.

For I-slave proxy devices:

- DP master can be complied and loaded
- Properties are displayed but cannot be changed
- Diagnostics in the project tree is not carried out

In the network view, proxy devices are represented with a question mark:

In order to edit the I-slave in the project, you must load the I-device from the device to the PG/PC. The I-slave proxy is hereby fully replaced by the I-slave.

It is possible to load an DP master with connected I-slave proxy from the PG/PC to the device.

Note

The replacement of I-slave proxy devices is only possible if the required I-device is available in the hardware catalog.

Loading configurations with PROFINET

Information on loading

If you have selected a CPU in the list of accessible devices and perform an load operation to PG/PC, all IO controllers and IO devices associated with this device along with their IO systems will be loaded. Settings for the topology are also transferred. If there is already a suitable Ethernet network in the project, the loaded devices are integrated into the existing network.

Relationships between the IO controllers and IO devices are only established within the project if both the IO controller as well as the I-device are loaded to the PG. It does not matter whether you load the IO controller or the I-devices first.

Supported functions

The following functionalities and settings are loaded:

- PROFINET configurations (RT and IRT) in IO systems with the associated IO controllers (CPUs and CPs), IO devices, and the modules used
- Logical addresses and interface properties
- Port interconnections
- Isochronous mode
- Sync domains/MRP domains
- Redundancy role "Client" or "Manager" for MRP configurations

Note

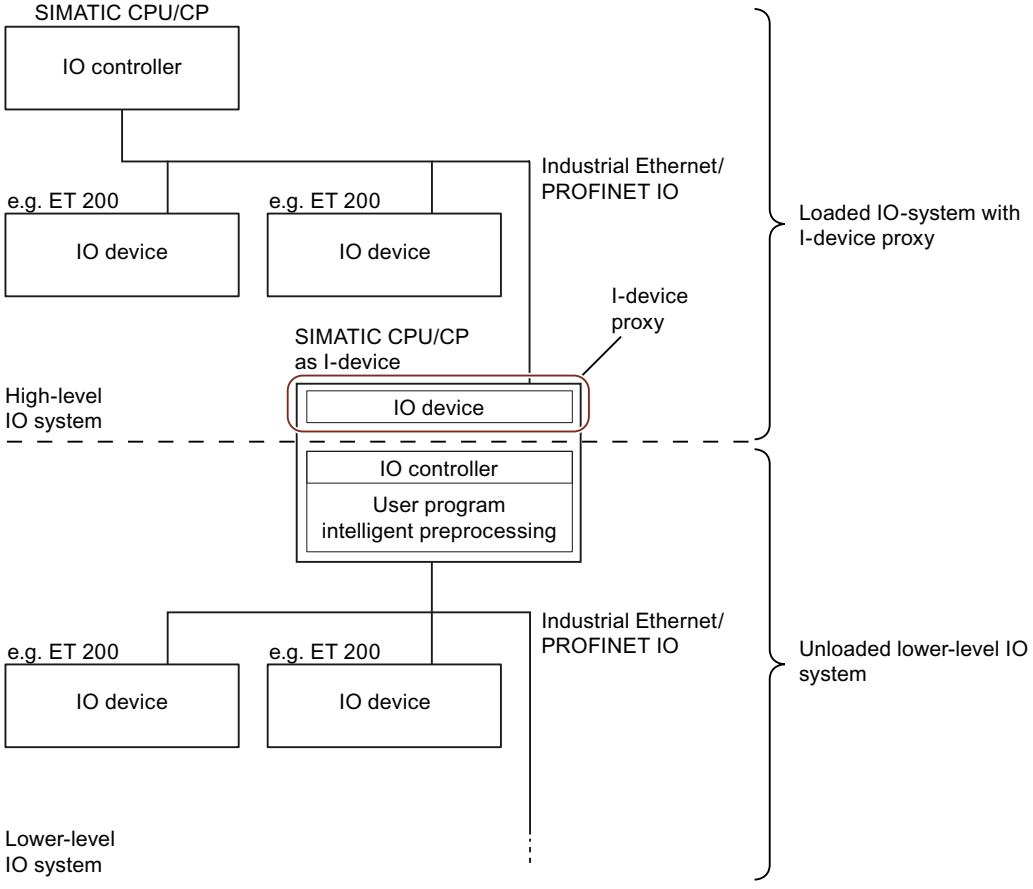
Empty Sync domains and MRP domains are not included in the loading.

GSD-based IO devices

As a prerequisite for loading GSD-based IO devices, the corresponding GSD files must be installed in the TIA Portal and available in the hardware catalog. If a required GSD file is not available in the same version as in the device, differences are identified during the consistency check.

I-device

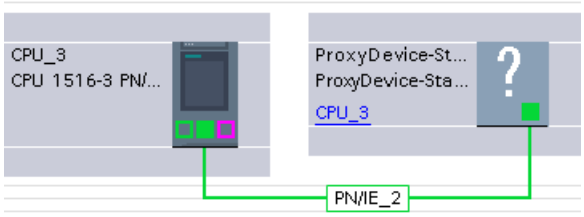
If you load the IO controller from an IO system with connected I-device, the IO controller and its IO devices will be loaded. An I-device proxy is loaded as dummy module for an involved I-device. The CPU parameter assignment, including the parameter assignment of the CPU's "own" (subordinate) IO system, is missing in the I-device proxy. Only the interface to the higher-level IO controller is loaded.



For I-device proxy devices:

- IO controller can be compiled and loaded
- Properties are displayed but cannot be changed
- Diagnostics in the project tree is not carried out

In the network view, proxy devices are represented with a question mark:



In order to edit the I-device in the project, you must load the I-device from the device to the PG/PC. This replaces the I-device proxy with the complete I-device along with its subordinate IO devices.

It is possible to load an IO controller with connected I-device proxy from the PG/PC to the device.

Note

The replacement of I-device proxy devices is only possible if the required I-device is available in the hardware catalog.

An I-device proxy representing a SIMOTION I-device cannot be loaded and replaced.

Configurations with IE/PB link

If one of the following configurations with IE/PB Link PN IO as PROFINET IO device is present, the complete configuration with all subordinate PROFIBUS devices is loaded:

- CPU/CP of S7-300/400
- PC station and connected PROFIBUS master system

The complete configuration consists of the following:

- CPU
- CP configuration
- PROFINET IO system with connected IE/PB link
- PROFIBUS master system of the IE/PB link with connected DP slaves

An example configuration consists of an S7-300 CPU with a CP as PROFINET IO controller. An IE/PB link as IO device is connected to the IO controller. As the PROFIBUS DP master, the IE/PB link polls a PROFIBUS DP slave, e.g., ET 200L. If you load the CPU from the device to the PG/PC, the complete configuration is loaded.

Note

If the IE/PB link is not operated as a PROFINET IO proxy but rather as a gateway in standard mode, the IE/PB link functions as a CPU and can be loaded separately.

Loading Shared Devices

The following applies to loading comments: If an input/output module with the Module-internal Shared Input (MSI) or Module-internal Shared Output (MSO) function consists of only one submodule, the submodule does not have its own comment. Instead, it uses the comment of the input/output module. The module must be subdivided into several submodules for the input/output module and all submodules to have their own comment fields.

Loading HMI devices

Information on loading

We distinguish between the following cases when loading HMI devices to PG/PC:

- HMI devices connected to a DP master as DP slave or to an IO controller as IO device are loaded as DP slave or IO device respectively (for example, PP 17-I PROFIsafe).
- HMI devices in a master system as I-slave or in an IO system as I-device are loaded as I-slave proxies or I-device proxies (Page 1153) (for example, SIMATIC Comfort Panels). The settings of the device proxy are read-only.
- HMI devices are not loaded if they are connected to a subnet (PROFIBUS or PROFINET) but not to a master system or IO system (for example, KP600 Basic color DP).

Going online with loaded configurations

You can go online with the loaded project or with loaded portions of the project.

Requirement

The hardware configuration loaded from the device to the PG/PC has been compiled. The statuses of the central and distributed modules are only correctly displayed after compilation.

Note

If you go online before compiling, the diagnostics icon “?” is displayed (diagnostics not possible). A corresponding alarm is displayed under "Info > General" in the Inspector window.

Dependencies

Depending on how completely the hardware configuration was loaded to the PG, restrictions apply to going online and to diagnostics:

- Completely loaded device with all associated central and distributed modules, such as DP slaves or IO devices:
  Going online and diagnostics are possible.
- Loaded device with connected I-devices/I-slaves:
  - I device/I slave is not loaded: Going online for the device and its modules is possible. For the dependent components of the configuration that are not loaded, device proxies are handled with only minimum diagnostics support. The online status is represented as an icon. The standard diagnostics is shown in the online and diagnostics view. I&M data are not loaded.
  - I device/I slave is also loaded: Going online is possible for all devices; diagnostics is fully supported.

10.1.5 Displaying alarms

10.1.5.1 Overview of the alarm display

The "Alarm display" function can be used to output asynchronous alarms of diagnostics events and user-defined diagnostics alarms as well a alarms from ALARM instructions.

From the alarm display, you can also start the alarm editor with the "Edit alarm" shortcut menu command and then create user diagnostics alarms.

Icons

The following table shows the icons and their functions:

Icon	Function
Archive view	Shows the alarms located in the archive.
Active alarms	Shows the currently active (pending) alarms. Alarms that must be acknowledged are shown in blue lettering.
Ignore	Ignores the arrival of alarms, These alarms are neither shown in the window nor stored in the archive.
Acknowledge	Confirms the selected alarm as read. Alarms requiring acknowledgment are shown in blue lettering.
Clear archive	Deletes all alarms in the archive.

Icon	Function
Export archive	Exports the current alarm archive to a file in xml format.
Multiple lines	Shows the alarms with multiple lines.
Automatically display the last alarm	Always show the last incoming alarm first.

10.1.5.2 Archive view

In the archive view, alarms are displayed and archived according to the time they appear. You can set the size of the archive (between 200 and 3000 alarms) with the menu command "Options > Settings > Online & Diagnostics". If the selected archive size is exceeded, the oldest alarm it contains is deleted.

Alarms that must be acknowledged are displayed in blue lettering and can be acknowledged with the shortcut menu command "Acknowledge alarm(s)".

The archive is constantly updated and does not need to be saved explicitly.

10.1.5.3 Layout of the alarms in the archive view

In the archive view, all events occurring on the selected CPUs are logged. A new entry is created for each individual event and shown as a further row in the table.

Table structure

All attributes of the alarms can be shown as columns. You can show or hide individual columns as well as modify the width and order of the columns. These settings are saved when the project is closed.

You can sort the columns in ascending or descending order. However, this setting is not saved when you close the project.

The alarms can be displayed in one or more rows. In the single row display, only the first row of the multiple-row alarm data is displayed.

The alarms either require acknowledgment or do not require acknowledgment. The alarms requiring acknowledgment that have not yet been acknowledged are highlighted in blue lettering and can be acknowledged either with the button in the toolbar for the particular context or with the shortcut menu command "Acknowledge alarm(s)".

10.1.5.4 Receiving alarms

To allow alarms to be displayed, you must first set the receipt of alarms for each CPU.

Procedure

Toe receive alarms, follow these steps:

1. Double-click on the "Online & Diagnostics" folder of the relevant CPU in project navigation.
2. Click the "Online access" group in the area navigation.
3. Select the option "Receive alarms".

Note

If you select this procedure, alarms are only received after you have re-established an online connection to the device.

Or:

1. Select the relevant CPU in the device, network, or topology view.
2. Select the command "Receive alarms" in the "Online" menu or in the shortcut menu.

Or:

1. Select the CPU in project navigation.
2. Select the command "Receive alarms" in the "Online" menu or in the shortcut menu.

Note

If you select one of the two above-named procedures, you must have first established an online connection to the device.

10.1.5.5 Export archive

To archive alarms, you can export the archive. Follow these steps:

1. Go to the archive view.
2. Click the "Export archive" button.
3. In the dialog that opens, select the path to export the archive.

Result

The archive is saved as an xml file at the location you selected.

10.1.5.6 Clear archive

The archive is organized as a ring buffer, in other words, when it is full, the oldest alarms are deleted from the archive. With the "Clear archive" button, you can delete the entire archive.

Procedure

To clear the archive, follow these steps:

1. Click the "Clear archive" button in the toolbar of the alarm display.

10.1.5.7 "Active alarms" view

The "Active alarms" view is an image of the alarm acknowledgement memory of the selected CPU(s).

10.1.5.8 Layout of the alarms in the "Active alarms" view

The "Active alarms" view represents an image of the alarm acknowledgment memory of the selected CPUs. One entry is shown in the table per active alarm. Events of an alarm ("incoming", "outgoing" and "acknowledged") are displayed in one row.

Table structure

All attributes of the alarms can be shown as columns. You can show or hide individual columns as well as modify the width and order of the columns. These settings are saved when the project is closed.

You can sort the columns in ascending or descending order. However, this setting is not saved when you close the project.

The alarms can be displayed in one or more rows. In the single row display, only the first row of the multiple-row alarm data is displayed.

The alarms either require acknowledgment or do not require acknowledgment. The alarms requiring acknowledgment that have not yet been acknowledged are highlighted in bold print and can be acknowledged either with the button in the toolbar for the specific context or with the shortcut menu command "Acknowledge alarm(s)".

10.1.5.9 Status of the alarms

Depending on whether you are in the "Active alarms" view or the archive view, the displayed alarms may have a different status.

Status of the alarms in the "Active alarms" view

- I: Alarm came
- IA: Alarm came and was acknowledged
- IO: Alarm has gone

If more signal changes occur than can be sent (signal overflow), OV is displayed as the status and the status is shown in red.

Status of the alarms in the archive view

- No information: only with alarms generated by the PG/PC and displayed in the "Archive" tab, for example logon status, connection abort, mode changes
- I: Alarm came
- A: Alarm came and was acknowledged
- O Alarm has gone
- D: The alarm was deleted.

If more signal changes occur than can be sent (signal overflow), OV is displayed as the status and the status is shown in red.

10.1.5.10 Acknowledging alarms

Alarms that must be acknowledged are shown in blue lettering.

Procedure

To acknowledge an alarm, follow these steps:

1. Select the required alarm or alarms from the table.
2. Click the "Acknowledge" button.

Note

You can select more than one alarm to acknowledge at the same time. To do this, hold down the <Ctrl> key and then select the alarms you want to acknowledge.

Result

The selected alarm was acknowledged and is then shown in normal characters.

Note

In the "Active alarms" view, acknowledged alarms that have already gone are no longer displayed.

10.1.5.11 Ignoring alarms

Ignoring alarms

To ignore alarms, follow these steps:

1. Click the "Ignore" button.
   The icon is shown on a gray background.

Result

From this point onwards, all alarms will be ignored. A message is created in the archive view indicating that the display of alarms and events is disabled.

Canceling the ignoring of alarms

To cancel the ignoring of alarms, follow these steps:

1. Click the "Ignore" button.
   The icon is shown on a white background.

Result

All alarms, in other words, even the alarms currently pending on the CPU while the "Ignore alarms" function was active, are displayed again from this point onwards. A message is created in the archive view indicating that the display of alarms and events is enabled again.

10.1.5.12 Sort table in alarm display

Sorting a table in ascending or descending order

To sort the table by a column in ascending or descending order, follow the steps below:

1. Click the table header of a column if you want to sort the column in ascending order.
2. Click again on the same column of the table header to sort the column in descending order.
3. Click a third time on the table header of the same column to cancel the sorting.

10.1.5.13 Keyboard commands in the alarm display

Alarm display

Function	Shortcut keys
Select all alarms	Ctrl+A
Acknowledge all selected alarms	Ctrl+Q

10.1.6 Additional information on configurations

10.1.6.1 Functional description of S7-1200 CPUs

Operating modes

Principles of the operating modes of S7-CPUs

Introduction

Operating modes describe the behavior of the CPU. The following operating modes are possible:

- STARTUP
- RUN
- STOP

In these operating modes, the CPU can communicate via the PN/IE interface, for example.

Other operating modes

If the CPU is not ready for operation, it is in one of following two operating modes:

- Deenergized, i.e. the supply voltage is switched off.
- Defective, which means an internal error has occurred.
  If the "Defective" status is caused by a firmware error, this state is indicated by the status LEDs of the CPU (refer to the description of the CPU). To find out the cause, follow these steps:
  - Turn the power supply switch off and on again.
  - Read out the diagnostics buffer when the CPU starts up and send the data for analysis to Customer Support.

  If the CPU does not start up, replace it.

See also

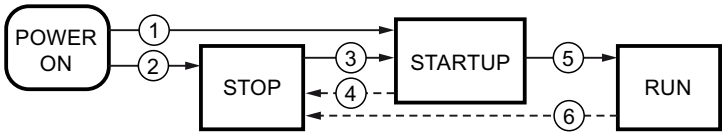
STOP mode (Page 1168)

RUN mode (Page 1168)

Operating mode transitions

Overview

The following figure shows the operating modes and the operating mode transitions of S7-1200 CPUs:



The following table shows the conditions under which the operating modes will change:

No.	Operating mode transition	Conditions
①	POWER ON → STARTUP	After switching on, the CPU goes to "STARTUP" mode if: • "Warm restart" startup type is set, and • the hardware configuration and the program blocks are consistent. Non-retentive memory is cleared and the contents of non-retentive DBs are reset to the initial values of the load memory. Retentive memory and retentive DB contents are retained.
②	POWER ON → STOP	When startup type "No startup" is set, the CPU goes to "STOP" mode after the supply voltage is switched on. Non-retentive memory is cleared and the contents of non-retentive DBs are reset to the initial values of the load memory. Retentive memory and retentive DB contents are retained.
③	STOP → STARTUP	The CPU switches to "STARTUP" mode if: • CPU is set to "RUN" from the programming device, and • the hardware configuration and the program blocks are consistent.
④	STARTUP → STOP	The CPU returns to the "STOP" mode in the following situations: • Error detected during startup. • The CPU is set to "STOP" from the programming device. • A STOP command is processed in the STARTUP OB.
⑤	STARTUP → RUN	If the STARTUP is successful, the CPU switches to "RUN".
⑥	RUN → STOP	The CPU returns to the "STOP" mode in the following situations: • An error is detected that prevents continued processing. • The CPU is set to "STOP" from the programming device. • A STOP command is processed in the user program.

"STARTUP" operating mode

Principles of the STARTUP mode

Function

After turning on the CPU, it executes a startup program before starting to execute the cyclic user program.

By suitably programming startup OBs, you can specify certain initialization variables for your cyclic program in the startup program. There is no rule in terms of the number of startup OBs. That is, you can set up one or several startup OBs in your program, or none at all.

Parameter settings for startup characteristics

You can specify whether the CPU remains in STOP mode or whether a warm restart is run. Over and above this, you can set the response during startup (RUN or previous mode) in the "Startup" group of the CPU properties.

Special characteristics

Note the following points regarding the "STARTUP" mode:

- The startup OBs are executed. All startup OBs you have programmed are executed, regardless of the selected startup mode.
- No time-based program execution can be performed.
- Interrupt controlled program execution limited to:
  - OB 82 (diagnostics interrupt)
- The outputs on the modules are disabled.
- The process image is not updated; direct I/O access to inputs is possible.

See also

Editing properties and parameters (Page 574)

Principles of the operating modes of S7-CPUs (Page 1163)

Organization blocks for startup (Page 1213)

Warm restart (Page 1165)

Warm restart

Function

During a warm restart, all non-retentive bit memory is deleted and non-retentive DB contents are reset to the initial values from load memory. Retentive bit memory and retentive DB contents are retained.

Program execution begins at the call of the first startup OB.

Triggering a warm restart

You can trigger a "Warm restart" using a corresponding menu command on your programming device in the following situations:

- The CPU must be in "STOP" mode.
- After a memory reset
- After downloading a consistent program and a consistent hardware configuration in the "STOP" mode of the CPU.

"POWER ON" triggers a "warm restart" if you have set the following parameters for the startup response:

- Startup type "warm restart - RUN" (regardless of the CPU operating mode prior to POWER OFF).
- "Warm restart - mode prior to POWER OFF" (depending on the CPU operating mode prior to POWER OFF. The CPU must have been in RUN mode prior to this.)

See also

Retentive memory areas (Page 1173)

Startup activities

Overview

The following table shows which activities the CPU performs at STARTUP:

Activities in execution sequence	At warm restart
Clear non-retentive bit memories	Yes
Clear all bit memories	No
Clear the process image output	Yes
Processing startup OBs	Yes
Update the process image input	Yes
Enable outputs after changing to "RUN" mode	Yes

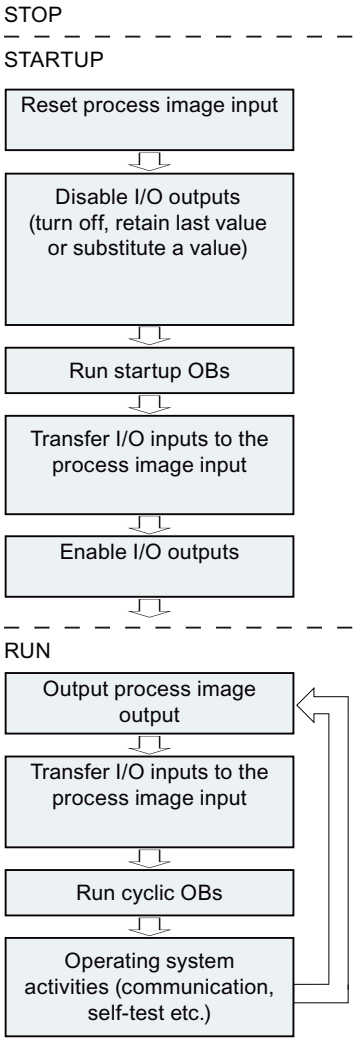
Sequence

The following figure shows the activities of the CPU in "STOP", "STARTUP", and "RUN" modes.

You can use the following measures to specify the state of the I/O outputs in the first cycle of the user program:

- Use assignable output modules to be able to output substitute values or to retain the last value.
- Set default values for outputs in startup OBs.

During the startup, all interrupt events are entered in a queue so that they can be processed later during RUN mode. In RUN mode, hardware interrupts can be processed at any time.



Special features during startup

Response when expected and actual configurations do not match

The expected configuration is represented by the engineering configuration loaded on the CPU. The actual configuration is the actual configuration of the automation system.

If the expected configuration and actual configuration differ, the CPU nevertheless initially changes to RUN.

Canceling a STARTUP

If errors occur during startup, the startup is canceled and the CPU remains in "STOP" mode.

Under the following conditions, a startup will not be performed or will be canceled:

- If an invalid SD card is inserted.
- If no hardware configuration has been downloaded.

See also

Overview of the CPU properties (Page 1185)

RUN mode

Function

In "RUN" mode the cyclic, time-driven, and interrupt-driven program sections execute:

- The process image output is read out.
- The process image input table is read.
- The user program is executed.

Active data exchange between S7-1200 CPUs by means of Open User Communication is only possible in "RUN" mode.

Running the user program

Once the CPU has read the inputs, the cyclic program runs from the first to the last instruction.

If you have configured a minimum cycle time, the CPU will not end the cycle until this minimum cycle time is up even if the user program is completed sooner.

A maximum cycle time is set which you can adjust according to your requirements. This ensures that the cyclic program is completed within a specified time. The system will respond with a time error if the cyclic program is not completed within this time.

Other events such as hardware and diagnostic interrupts can interrupt the cyclic program flow and prolong the cycle time.

See also

Principles of the operating modes of S7-CPUs (Page 1163)

Events and OBs (Page 1177)

STOP mode

Function

In "STOP" mode, the user program is not executed. All outputs are disabled or react according to the parameter settings: They provide a substitute value as set in the parameters or retain the last value output and bring the controlled process to a safe status.

The CPU checks the following points:

- Hardware, for example whether are all modules are available
- Whether the default settings for the CPU are applicable or parameter sets are present
- Whether the general conditions for the programmed startup behavior are correct

See also

Principles of the operating modes of S7-CPUs (Page 1163)

Basics of a memory reset

Function

A memory reset on the CPU is possible only in STOP mode.

When memory is reset, the CPU is changed to an "initial status". This means:

- An existing online connection between your programming device/PC and the CPU is terminated.
- The content of the work memory and the retentive and non-retentive data are deleted.
- The diagnostic buffer, the time, the IP address, the hardware configuration, and active force jobs are retained.
- The load memory (code and data blocks) is then copied to work memory. As a result, the data blocks no longer have current values but their configured initial values.

Memory areas

Useful information on memory cards

How the memory card functions

The SIMATIC Memory Card for a S7-1200 is an SD memory card preformatted by Siemens for the CPU user program.

You may only delete files and folders. If you format the memory card with Windows, for example with a commercially available card reader, you make the memory card unusable as a storage medium for an S7 CPU.

Setting the card type

You can use the memory card as a transfer card, a program card or a firmware update card.

To set the card type, insert the memory card into the card reader of the programming device and select the "Card reader/USB memory" folder from the project tree. In the properties of the selected memory card, designate the card type:

- Program
  If it is used as a program card, you can load the user program on the memory card. In this case, the internal load memory of the device is replaced by the memory card and the internal load memory is erased. The user program is then fully executable from the memory card. If the memory card with the user program is removed, there is no longer a program available.
- Transfer
  If it is used as a transfer card, you can transfer the user program from the memory card to the internal load memory of the CPU. You can then remove the memory card again.
- Firmware card
  Firmware for the S7-1200 modules can be stored on a memory card. Therefore it is possible to perform a firmware update with the help of a specifically prepared memory card. Likewise, a backup copy of firmware for a module can be stored on a memory card.

Transferring objects from the project to a memory card

When the memory card is inserted in the programming device or in an external card reader, you can transfer the following objects from the project tree to the memory card:

- Individual blocks (multiple selection possible)
  In this case a consistent transfer is available, as the dependencies of the blocks to each other is taken into account with block selection.
- PLC
  In this case, all objects relevant to processing are transferred, such as blocks and the hardware configuration on the memory card, just as with downloading.

To perform the transfer, you can move the objects with drag-and-drop or use the command "Card reader/USB memory > Write to memory card" in the "Project" menu.

Transferring objects from the memory card to the project

You can transfer Individual blocks (multiple selection is possible) by dragging them to the project. A hardware configuration cannot be transferred from the memory card to the project.

Updating firmware with a memory card

You can get the latest firmware data on the Internet from the Service & Support pages:

http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/34143537)

Save the firmware files on the hard disk and plug the SIMATIC Memory Card into the card reader of your programming device.

To store the file on the memory card, select the memory card in the "Card Reader/USB memory" folder in the project tree. Select the shortcut menu "Card Reader/USB memory > Create firmware update memory card".

Then follow the instructions in the Service & Support portal for performing a firmware update with your CPU.

Updating the firmware changes the CPU firmware status. If you have used the CPU in the project, you will have to update the CPU already configured to the CPU with the new firmware status by changing devices offline, and adapt and then load the program or configuration.

See also

Replacing a hardware component (Page 574)

Useful information on CPU firmware versions and STEP 7 versions (Page 1235)

Displaying properties of memory cards (Page 469)

Load memory

Function

Each CPU has an internal load memory. The size of this internal load memory depends on the CPU used.

This internal load memory can be replaced by using external memory cards. If there is no memory card inserted, the CPU uses the internal load memory; if a memory card is inserted, the CPU uses the memory card as load memory.

However, the size of the usable external load memory cannot be greater than the internal load memory even if the inserted memory card has more free space.

See also

Using memory cards (Page 467)

Work memory

Function

Work memory is a non-retentive memory area for storing elements of the user program that are relevant for program execution. The user program is executed exclusively in work memory and system memory.

System memory

System memory areas

Function

System memory contains the memory elements that each CPU makes available to the user program, such as the process image and bit memory.

By using appropriate operations in your user program, you address the data directly in the relevant operand area.

The following table shows the operand areas of the system memory:

Operand area	Description	Access via units of the following size:	S7 notation
Process image output	The CPU writes the values from the process image output table to the output modules at the start of the cycle.	Output (bit)	Q
		Output byte	QB
		Output word	QW
		Output double word	QD
Process image input	The CPU reads the inputs from the input modules and saves the values to the process image input table at the start of the cycle.	Input (bit)	I
		Input byte	IB
		Input word	IW
		Input double word	ID
Bit memory	This area provides storage for intermediate results calculated in the program.	Bit memory (bit)	M
		Memory byte	MB
		Memory word	MW
		Memory double word	MD
Data block	Data blocks store information for the program. They can either be defined so that all code blocks can access them (global DBs) or assigned to a specific FB or SFB (instance DB). Requirement: The block attribute "Optimized block access" is not enabled.	Data bit	DBX
		Data byte	DBB
		Data word	DBW
		Data double word	DBD

Operand area	Description	Access via units of the following size:	S7 notation
Local data	This area contains the temporary data of a block while the block is being processed. Requirement: The block attribute "Optimized block access" is not enabled. Recommendation: Access local data (temp) symbolically.	Local data bit	L
		Local data byte	LB
		Local data word	LW
		Local data double word	LD
I/O input area	The I/O input and output areas permit direct access to central and distributed input and output modules.	I/O input bit	<tag>:P
		I/O input byte	
		I/O input word	
		I/O input double word	
I/O output area		I/O output bit	
		I/O output byte	
		I/O output word	
		I/O output double word	

See also

Diagnostics buffer (Page 1176)

Basic principles of process images (Page 1174)

Access to the I/O addresses (Page 1177)

Retentive memory areas

Retentive memory areas

Data loss after power failure can be avoided by marking certain data as retentive. This data is stored in a retentive memory area. A retentive memory area is an area that retains its content following a warm restart, in other words, after cycling the power when the CPU changes from STOP to RUN.

The following data can be assigned retentivity:

- Bit memory: The precise width of the memory can be defined for bit memory in the PLC tag table or in the assignment list.
- Tags of a function block (FB): You can define individual tags as retentive in the interface of an FB if you have enabled optimized block access. Retentivity settings can be defined only in the assigned instance data block if optimized block access has not been activated for the FB.
- Tags of a global data block: You can define retentivity either for individual or for all tags of a global data block depending on the settings for access.
  - Block with optimized access: retentivity can be set for each individual tag.
  - Block with standard access: The retentivity setting applies to all tags of the DB; either all tags are retentive or no tag is retentive.

See also

Warm restart (Page 1165)

process image input/output

Basic principles of process images

Function

When the user program addresses the input (I) and output (O) operand areas, it does not query or change the signal states on the digital signal modules. Instead, it accesses a memory area in the system memory of the CPU. This memory area is referred to as the process image.

Advantages of the process image

Compared with direct access to input and output modules, the main advantage of accessing the process image is that the CPU has a consistent image of the process signals for the duration of one program cycle. If a signal state on an input module changes during program execution, the signal state in the process image is retained until the process image is updated again in the next cycle. The process of repeatedly scanning an input signal within a user program ensures that consistent input information is always available.

Access to the process image also requires far less time than direct access to the signal modules since the process image is located in the internal memory of the CPU.

Updating the process images

Sequence

The operating system updates the process images at cyclic intervals unless defined otherwise in your configuration. The process image input/output is updated in the following order:

1. The internal tasks of the operating system are performed.
2. The process image output (PIQ) table is written to the outputs of the module.
3. The status of inputs is read to the process image input (PII) table.
4. The user program is executed with all the blocks that are called in it.

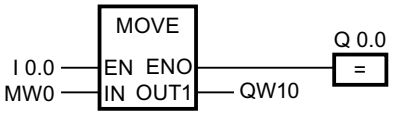
The operating system automatically controls the writing of the process image output to the outputs of the modules and the reading of the process image input.

Special characteristics

You have the option of accessing inputs or outputs directly using direct I/O access.

- If an instruction accesses an output directly and the output address is located in the process image output, the process image of the relevant output is updated.
- If an instruction accesses an output directly and the output address is **not** located in the process image output, the process image of the relevant output is **not** updated.

Example of normal I/O access by way of the process image



Update QW10 in the I/O output area with the value from MW0.

I/O access error during process image updating

If an error occurs during process image updating (I/O access error), the CPU reacts with the default system reaction "Ignore".

See also

Start address of a module (Page 1176)

Access to the I/O addresses (Page 1177)

Startup activities (Page 1166)

Diagnostics buffer

Function

The diagnostics buffer is part of the system memory of the CPU. It contains the errors detected by the CPU or modules with diagnostics capability. It includes the following events:

- Every mode change of the CPU (for example, POWER UP, change to STOP mode, change to RUN mode)
- Every diagnostics interrupt

The diagnostics buffer of the S7-1200-CPU has a capacity of 50 entries of which the last (most recent) 10 entries are retained following power cycling.

Those entries can only be cleared by restoring the CPU to factory defaults.

You can read the content of the diagnostics buffer with the help of the Online and Diagnostics view.

See also

Basic information on the diagnostics buffer (Page 1398)

I/O data area

Start address of a module

Definition

The start address is the lowest byte address of a module. It acts as the initial address of the module user data area.

Configuring module start addresses

The addresses used in the user program and the module start addresses are coordinated when the modules are configured.

In the module properties ("I/O addresses" group), you can change the start addresses that were assigned automatically after the modules were inserted.

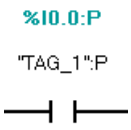
You can also make a setting that decides whether or not the addresses are located in the process image.

Access to the I/O addresses

I/O addresses

If you insert a module in the device view, its user data is located in the process image of the S7-1200 CPU (default). The CPU handles the data exchange between the module and the process image area automatically during the update of the process images.

Append the suffix ":P" to the I/O address if you want the program to access the module directly instead of using the process image.



This could be necessary, for example, during execution of a time-sensitive program which also has to control the outputs within the same cycle.

Basics of program execution

Events and OBs

Events and OBs

The operating system of S7-1200-CPUs is based on events. There are two types of events:

- Events which can start an OB
- Events which cannot start an OB

An event which can start an OB triggers the following reaction:

- It calls the OB you possibly assigned to this event. The event is entered in a queue according to its priority if it is currently not possible to call this OB.
- The default system reaction is triggered if you did not assign an OB to this event.

An event which cannot start an OB triggers the default system reaction for the associated event class.

The user program cycle is therefore based on events, the assignment of OBs to those events, and on the code which is either contained in the OB, or called in the OB.

The following table provides an overview of the events which can start an OB, including the associated event classes and OBs. The table is sorted based on the default OB priority. Priority class 1 is the lowest.

Event class	OB no.	Number of OBs	Start event	OB priority (default)
Cyclic program	1, >= 123	>= 1	Starting or end of the last program cycle OB	1
Startup	100, >= 123	>=0	STOP to RUN transition	1
Time-of-day interrupt	>= 10	Max. 2	Start time has been reached	2
Time-delay interrupt	>= 20	Max. 4	Delay time expired	3
Cyclic interrupt	>= 30		Constant bus cycle time expired	8
Hardware interrupt	>= 40	Max. 50 (more can be used with DETACH and ATTACH)	• Positive edge (max. 16) • Negative edge (max. 16)	18
			• HSC: Count value = reference value (max. 6) • HSC: Count direction changed (max. 6) • HSC: External reset (max. 6)	18
Status interrupt	55	0 or 1	CPU has received status interrupt	4
Update interrupt	56	0 or 1	CPU has received update interrupt	4
Manufacturer- or profile-specific interrupt	57	0 or 1	CPU has received manufacturer-specific or profile-specific interrupt	4
Diagnostic error interrupt	82	0 or 1	Module has detected an error	5
Pull/plug interrupt	83	0 or 1	Removal/insertion of modules of distributed I/O	6
Rack error	86	0 or 1	Error in the I/O system of the distributed I/O	6
Time error	80	0 or 1	• Maximum cycle time exceeded • Called OB is still being executed • Time-of-day interrupt missed • Time-of-day interrupt missed during STOP • Queue overflow • Interrupt loss due to high interrupt load	22

The following table describes events which do not trigger an OB start, including the corresponding reaction of the operating system. The table is sorted based on event priority.

Event class	Event	Event priority	System reaction
Insert/remove central modules	Insert/remove a module	21	STOP
I/O access error during process image update	I/O access error during process image update	22	Ignore
Programming error	Programming error in a block for which you use system reactions provided by the operating system (note: the error handling routine in the block program is executed if you activated local error handling).	23	RUN

Event class	Event	Event priority	System reaction
I/O access error	I/O access error in a block for which you use system reactions provided by the operating system (note: the error handling routine in the block program is executed if you activated local error handling).	24	RUN
Maximum cycle time exceeded twice	Maximum cycle time exceeded twice	27	STOP

Assignment between OBs and events

With the exception of the cyclic program and startup program and event can only be assigned to one OB. However, in certain event classes such as hardware interrupts one and the same OB can be assigned to several events.

The assignment between OBs and events is defined in the hardware configuration. Defined assignments can be changed at runtime by means of ATTACH and DETACH instructions.

OB priority and runtime behavior

S7-1200-CPUs support the priority classes 1 (lowest) to 27 (highest). An OB is assigned the priority of its start event.

OBs are always executed on a priority basis: The OBs with the highest priority are executed first. Events of the same priority are processed in order of occurrence.

As of firmware version V4.0 of the S7-1200 CPUs you can specify in the device configuration, under properties of the CPU, if the OBs are interruptible or not. This parameter assignment has an effect on all OBs with exception of the cycle OBs which are always interruptible.

The following applies to S7-1200 CPUs with firmware version < V4.0:

- Any OB with priority >= 2 will interrupt cyclic program execution.
- An OB of priority 2 to 25 cannot be interrupted by any event of priority group 2 to 25. This rule also applies to events of a priority higher than that of the currently active OB. Such events are processed later.
- A time error (priority 26) will interrupt any other OB.

The following applies to S7-1200 CPUs as of firmware version V4.0:

If you do not configure the OBs as interruptible, an OB is always processed completely even if an event of a higher priority occurs during its runtime. Specifically, this means:

- Any OB with priority >= 2 will interrupt cyclic program execution.
- An OB of priority 2 to 25 cannot be interrupted by any event. This rule also applies to events of a priority higher than that of the currently active OB, which also includes a time error. Such events are processed later.

If you do configure the OBs as interruptible and an event of a higher priority occurs during the runtime of an OB, the running OB is interrupted and the OB associated with the occurring event

is processed. Once this OB has been completed, processing of the interrupted OB continues. Specifically, this means:

- Any OB with priority >= 2 will interrupt cyclic program execution.
- An OB of priority 2 to 25 can be interrupted by any event whose priority is higher than that of the running OB. This is also true for time errors: A time error (priority 26) will interrupt any OB.

OB start information

Certain OBs have start information, while others do not. This is explained in greater detail in the description of the relevant OB.

See also

Event-based program execution (Page 1180)

ATTACH: Attach an OB to an interrupt event (Page 3216)

DETACH: Detach an OB from an interrupt event (Page 3218)

Event-based program execution

OB priority and runtime behavior

S7-1200-CPUs support the priority classes 1 (lowest) to 27 (highest). An OB is assigned the priority of its start event.

Interrupt OBs can only be interrupted by time error interrupts. This rule also applies to events of a priority higher than that of the currently active OB. That is, only one interrupt OB can be active, with exception of the time error interrupt OB.

Any further event of generated while an interrupt OB is being executed is added to a queue in accordance with its priority. Start events within a queue are processed later based on the chronological order of their occurrence.

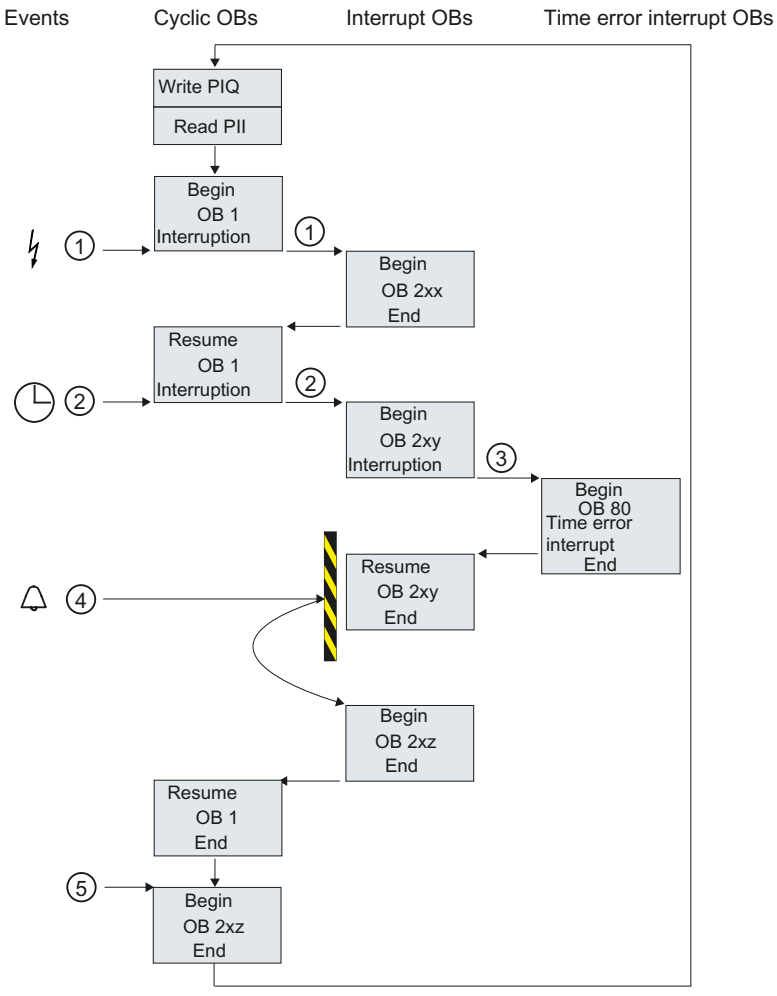
Program execution on the CPU

Cyclic OBs are interrupted by interrupt OBs.

Reactions to events which start an interrupt OB:

- For CPUs up to firmware version V3: Interrupt OBs can only be interrupted by time error interrupt OBs.
- For CPUs as of firmware version V4: Interrupt OBs can be interrupted by interrupt OBs of a higher priority.

The figure below shows the basic procedure in case interrupt OBs cannot be interrupted (behavior up to firmware version V3):



① and ② An event (e.g. a hardware interrupt) calls its associated OB.

A called OB is executed without interruption, including all of its nested blocks. Execution of the cyclic OB is resumed on completion of interrupt processing, provided the queue does not contain any events which trigger an OB start.

③ An interrupt OB can only be interrupted by a time error interrupt OB (OB 80).

④ An new alarm-triggering event occurs during interrupt processing.

Reaction for CPUs up to and including firmware version V3:

This new event is added to a queue. The queued events successively call their corresponding OBs only after execution of the current interrupt OBs was completed and according to the following rules:

- Events are processed in the order of their priority (starting at the highest priority).
- Events of the same priority are processed in chronological order.

Reaction for CPUs as of firmware version V4:

You use a CPU parameter to set the interruptibility for CPUs as of firmware version V4. Default behavior: OBs are interruptible. In this case: If the new event has a higher priority than the currently running OB, the OB started by the new event interrupts the currently running OB.

If you disable the option, the interrupt OBs cannot be interrupted.

⑤ The cyclic OBs are processed one after the other.

Notes on queues

- Every priority class (OBs of the same priority to be called) is assigned a separate queue. The size of those queues is set by default.
- Any new event leading to the overflow of a queue is discarded and therefore lost. A "time error interrupt event" is generated simultaneously. Information identifying the OB that caused the error is included in the start information of the time error interrupt OB (OB 80). A corresponding reaction such as an alarm trigger can be programmed in the time error interrupt OB.

Example of a hardware interrupt event

The function principle of event-oriented program execution in the S7-1200 CPU is described based on the example of a hardware interrupt-triggering module.

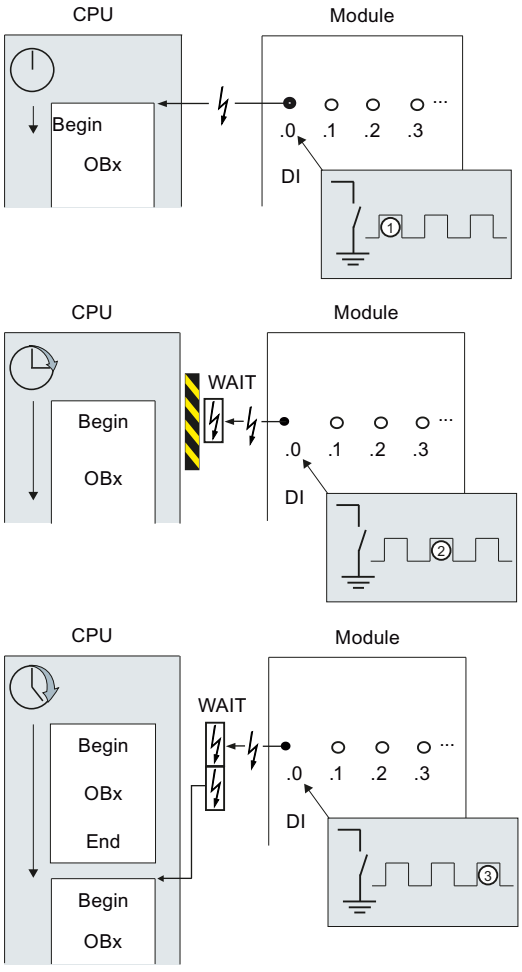
Process events and their priority

Process events are triggered by the I/O (e.g. at a digital input) and initiate a call of the assigned OB in the S7-1200 CPU. OBs assigned to a process event are called hardware interrupt OBs.

Examples of process events and their priority:

- Process events "rising edge" or "falling edge" at an interrupt-triggering module: The hardware interrupt OB started by such an event is always assigned priority 5.
- Process events from a high-speed counter
  - Count value corresponds to the reference value
  - Change count direction
  - External reset of the high-speed counter
    The hardware interrupt OB started by this event is always assigned priority 6.

The figure below shows the chronological sequence of hardware interrupt execution: In the case of two hardware interrupt events in immediate succession, the second hardware interrupt triggering event is held back in the queue until the first OBx has been processed. The next hardware interrupt triggering event can only start the associated OBx when the OBx has been processed. Additional hardware interrupt triggering events are lined up in the queue according to this principle.



Hardware interrupt execution

① A hardware interrupt-triggering event such as a rising edge at the input calls the OB to which it is assigned.

② If a new event occurs that triggers a hardware interrupt while the OB is executing, this event is entered in a queue.

③ The new event that triggers a hardware interrupt starts the hardware interrupt OB assigned to the event.

Assigning the interrupt-triggering event

The interrupt-triggering event is assigned to an OB in the input properties of the device view.

- An interrupt-triggering event can only be assigned to a single OB.
- OBs, however, can be assigned to several interrupt-triggering events.
  This means, for example, that you can assign both the rising and the falling edge to the same interrupt OB in order to trigger the same reaction to any change of the input signal.

- The started OB can interrupt a cycle OB at every instruction. Consistent data access is secured up to dword size.
- You can parameterize module-specific interrupt-triggering events such as a rising and the falling edge at the input.
- Assign the interrupt-triggering event and the OB to be started in the configuration of the interrupt-triggering module. However, within the started hardware interrupt OB you can override this assignment using the DETACH instruction, or assign the same event to a different OB using the ATTACH instruction. This functionality allows a flexible reaction to external process signals.

Setting the operating behavior

Changing properties of the modules

Default settings

When they leave the factory, all hardware components with parameters have default settings suitable for standard applications. These default values allow the hardware components to be used immediately without making any additional settings.

You can, however, modify the behavior and the properties of the hardware components to suit the requirements and circumstances of your application. Hardware components with settable parameters include, for example, communications modules and several analog and digital modules.

Setting and loading parameters

When you have selected a hardware component in the device or network view, you can set the properties in the Inspector window. When you save a device configuration with its parameters, data is generated that needs to be loaded on the CPU. This data is transferred to the relevant modules during startup.

Properties of the CPUs

The properties of the CPUs have special significance for system behavior. For example for a CPU you can set:

- Interfaces
- Inputs and outputs
- High-speed counters
- Pulse generators
- Startup behavior
- Time-of-day
- Protection level
- Bit memory for system and clock

- Cycle time
- Communications load

The entry possibilities specify what is adjustable and in which value ranges. Fields that cannot be edited are disabled or are not shown in the properties window.

Requirement

You have already arranged the hardware components for which you want to change properties on a rack.

Procedure

To change the properties and parameters of the hardware components, follow these steps:

1. In the device or network view, select the hardware component or interface that you want to edit.
2. Edit the settings for the selected object:
   - For example in the device view you can edit addresses and names.
   - In the Inspector window additional setting possibilities are available.

You do not need to confirm your entries, the changed values will be applied immediately.

See also

Editing properties and parameters (Page 574)

Introduction to loading a configuration (Page 1146)

CPU properties

Overview of the CPU properties

Overview

The following table provides you with an overview of the CPU properties:

Group	Properties	Description
General	Project information	General information to describe the inserted CPU. Except for the slot number, you can change this information.
	Catalog information	Read-only information from the hardware catalog for this CPU.

Group	Properties	Description
PROFINET interface	General	Name and comment for this PROFINET interface. The name is limited to 110 characters.
	Ethernet addresses	Select whether the PROFINET interface is networked. If subnets have already been created in the project, they are available for selection in the drop-down list. If not, you can create a new subnet with the "Add new subnet" button. Information on the IP address, subnet mask and IP router usage in the subnet is available in the IP protocol. If an IP router is used, the information about the IP address of the IP router is necessary.
	Advanced options	Name, comment and additional setting options of the Ethernet interface port.
	Time synchronization	Settings for time synchronization in the NTP time format. The NTP (network time protocol) is a general mechanism for synchronizing system clocks in local and global area networks. In NTP mode, the interface of the CPU sends time queries (in client mode) at regular intervals to NTP servers on the subnet (LAN) and the addresses must be set in the parameters here. Based on the replies from the server, the most reliable and most accurate time is calculated and synchronized. The advantage of this mode is that it allows the time to be synchronized across subnets. The accuracy depends on the quality of the NTP server being used.
DI#/DO#	General	Name of and comment on the integrated digital inputs of the CPU.
	Digital inputs	Input delays can be set for digital inputs. The input delays can be set in groups (in each case for 4 inputs). The detection of a positive and a negative edge can be enabled for each digital input. A name and a hardware interrupt can be assigned to this event. Depending on the CPU, pulse catches can be activated at various inputs. When the pulse catch is activated, even pulse edges that are shorter than the cycle time of the program are detected.
	Digital outputs	The reaction to a mode change from RUN to STOP can be set for all digital outputs: The state can either be frozen (corresponds to retain last value) or you set a substitute value ("0" or "1")
	I/O addresses	The address space of the input and output addresses is specified as is the process image.
	Hardware identifier	The hardware identifier of the device is displayed.

Group	Properties	Description
AI#	General	Name of and comment on the integrated analog inputs of the CPU.
	Analog inputs	During noise reduction, the specified integration time suppresses interference frequencies at the specified frequency (in Hz). The channel address, measurement type, voltage range, smoothing and overflow diagnostics must be specified in the "Channel #" group. The measurement type and voltage range are set permanently to voltage, 0 to 10 V. Smoothing analog values provides a stable analog signal for further processing. Smoothing analog values can be useful with slow measured value changes, for example, in temperature measurement. The measured values are smoothed with digital filtering. Smoothing is achieved by the module forming mean values from a specified number of converted (digitalized) analog values. The selected level (slight, medium, strong) decides the number of analog signals used to create the mean value. If overflow diagnostics is enabled, a diagnostics event is generated if an overflow occurs.
	I/O addresses	The address space of the input addresses is specified as is the process image.
	Hardware identifier	The hardware identifier of the device is displayed.
High-speed counter (HSC)	High-speed counter (HSC)#	High-speed counters are typically used to drive counting mechanisms. See: Configuring high-speed counters (Page 1196)
Pulse generators (PTO/PWM)	PTO#/PWM#	A pulse generator is activated and can be initialized with project information. For the configuration of an activated pulse generator, specify the usage as PWM (Pulse Width Modulation) or as PTO (Pulse Train Output). Specify the output source, time base, pulse width format, cycle time and initial pulse width for PWM. A pulse output is specified as the hardware output. The PWM output is controlled by the CTRL_PWM instruction, see CTRL_PWM (Page 3310). Specify the output source for PTO. A pulse output and a direction output are specified as the hardware outputs. A PTO is operated together with a high-speed counter in the "axis of motion" count mode and controlled by the Motion Control technology object (see keyword "Motion Control S7-1200") . The hardware ID is displayed in the I/O-diagnostics addresses and, if the PWM function is selected, the address space of the output addresses and the process image can be selected.

Group	Properties	Description
Startup	Startup after POWER ON	Setting the startup characteristics after cycling power. See: Principles of the STARTUP mode (Page 1165)
	Comparison of preset configuration and actual configuration	Specifies whether modules (SM, SB, CM, CP or even the CPU) can be replaced: • Startup of the CPU only if compatible • Startup of the CPU even if there are differences Example: A signal module with 16 digital inputs and 16 digital outputs (DI16/DQ16) can be a compatible replacement for a signal module with 8 digital outputs (DQ8) or 4 digital inputs (DI4).
	Parameter assignment time for distributed I/O	Specifies a maximum period (standard: 60000 ms) in which the distributed I/O must start up. (The CMs and CPs are supplied with voltage and communication parameters during the CPU startup. This configuration time provides a time period during which I/O modules connected to the CM or CP must start up.) The CPU switches to RUN as soon as the distributed I/O has started and is ready for operation, regardless of the "Parameter assignment time for distributed I/O" parameter. If the distributed I/O has not started up during this period, the CPU switches to RUN without the distributed I/O.
Cycle	Maximum cycle time and minimum cycle time.	Specification of a maximum cycle time or a fixed minimum cycle time. If the cycle time exceeds the maximum cycle time, the CPU goes to STOP mode. See: Cycle time and maximum cycle time (Page 1189)
Communication load	Maximum allocation of the cycle for communication (as a percentage)	Controls the duration of communication processes that always also extend the cycle time, within certain limits. Examples of communication processes include: Transferring data to another CPU or loading blocks (initiated via the PC). See: Cycle loading by communications (Page 1190)
System and clock memory	System memory bits and clock memory bits	You use system memory bits for the following scans: • Is the current cycle the first since cycling power? • Have there been any diagnostics state changes compared with the previous cycle? • Scan for "1" (high) • Scan for "0" (low) Clock memory bits change their values at specified periodic intervals. See: Enabling system memory (Page 1206) See: Using clock memory (Page 1207)
Web server	Automatic update	Sends the requested web page with current CPU data periodically to the web browser. Enter the period duration under "Update interval". Automatic update can only be activated if the web server is enabled. See: Auto-Hotspot
	User-defined web pages	Allows access to freely-designed web pages of the CPU via a web browser. See: Auto-Hotspot

Group	Properties	Description
Time	Local time and daylight saving time	Setting of the time zone in which the CPU is operated and setting of the daylight-saving/standard time changeover.
Protection	Protection and password for read/write access	Setting the read/write protection and the password for access to the CPU. See: Setting options for the protection level (FW V1 to V3) (Page 1208) See: Setting options for the protection (FW as of V4) (Page 1209)
Connection resources	-	Display of available, reserved and previously configured connection resources of the CPU.
Address overview	-	Tabular representation of all addresses used by the CPU for integrated inputs/outputs as well as for the inserted modules. Addresses that are not used by any module are represented as gaps. The view can be filtered according to • Input addresses • Output addresses • Address gaps

See also

Specifying input and output addresses (Page 849)

Assigning parameters to hardware interrupt OBs (Page 1230)

Access to the I/O addresses (Page 1177)

Addressing modules (Page 848)

Special features during startup (Page 1167)

Cycle time and maximum cycle time

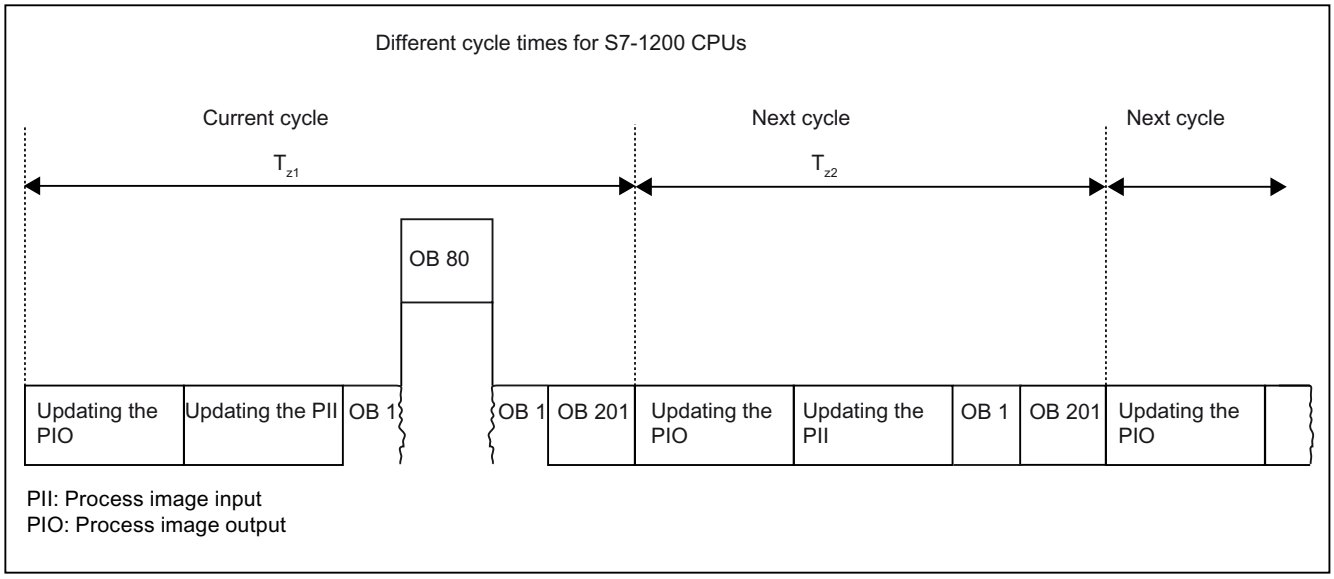
Function

The cycle time is the time that the operating system requires to execute the cyclic program and all the program sections that interrupt this cycle. The program execution can be interrupted by:

- Time errors and 2xMaxCycleTime errors
- System activities, e.g., process image updating

The cycle time (Tcyc) is therefore not the same for every cycle.

The following schematic shows an example of different cycle times (TZ1 ≠ TZ2) for S7-1200 CPUs:



In the current cycle, the cyclic OB used here (e.g. OB 1) will be interrupted by a time error (e.g. OB 80). Following the cyclic OB, the next cycle OB 201 is processed.

Maximum cycle time

The operating system monitors the execution time of the cyclic program for a configurable upper limit known as the maximum cycle time. You can restart this time monitoring at any point in your program by calling the RE_TRIGR instruction.

If the cyclic program exceeds the maximum cycle time, the operating system will attempt to start the time error OB (OB 80). If the OB is not available, the CPU ignores the overshoot of the maximum cycle time.

In addition to monitoring the runtime for overshooting of the maximum cycle time, adherence to a minimum cycle time is guaranteed. To do this, the operating system delays the start of the new cycle until the minimum cycle time has been reached. During this waiting time, new events and operating system services are processed.

If the maximum cycle time is exceeded a second time, for example while the time error OB is being processed (2xMaxCycleTime error), the CPU changes to STOP mode.

Cycle loading by communications

Function

The cycle time of the CPU can be extended due to communications processes. These communications processes include for example:

- Transferring data to another CPU
- Loading of blocks initiated by a programming device

You can control the duration of these communications processes to some extent using the CPU parameter "Cycle load due to communication".

In addition to communications processes, test functions also extend the cycle time. The "Cycle load due to communication" parameter can be used to influence the duration.

How the parameter works

You use the "Cycle load due to communication" parameter to enter the percentage of the overall CPU processing capacity that can be available for communications processes. This CPU processing capacity is now available at all times for communication. This processing capacity can be used for program execution when not required for communication.

Effect on the actual cycle time

The "Cycle load due to communication" parameter can be used to extend the cycle time of the cyclic organization block (e.g., OB 1) by a factor calculated according to the following formula:

$$\frac{100}{100 - \text{"Cycle load due to communication"}}$$

The formula does not take into account the effect of asynchronous events such as hardware interrupts or cyclic interrupts on the cycle time.

If the cycle time is extended due to communication processes, more asynchronous events may occur within the cycle time of the cyclic organization block. This extends the cycle still further. The extension depends on how many events occur and how long it takes to process them.

Example 1 – no additional asynchronous events:

If the "Cycle load due to communication" parameter is set to 50%, this can cause the cycle time of the cyclic organization block to increase by up to a factor of 2.

Example 2 – additional asynchronous events:

For a pure cycle time of 500 ms, a communication load of 50% can result in an actual cycle time of up to 1000 ms, provided that the CPU always has enough communications jobs to process. If, parallel to this, a cyclic interrupt with 20 ms processing time is executed every 100 ms, this cyclic interrupt would extend the cycle by a total of 5*20 ms = 100 ms without communication load. That is, the actual cycle time would be 600 ms. Because a cyclic interrupt also interrupts communications, it affects the cycle time by adding 10 * 20 ms at 50 % communication load. That is, in this case, the actual cycle time amounts to 1200 ms instead of 1000 ms.

Note

Observe the following:

- Check the effects of changing the value of the "Cycle load due to communication" parameter while the system is running.
- You must always consider the communication load when setting the minimum cycle time as time errors will otherwise occur.

Recommendations

- Increase this value only if the CPU is used primarily for communication purposes and the user program is not time critical.
- In all other situations you should only reduce this value.

Time-of-day functions

Basic principles of time of day functions

All S7-1200 CPUs are equipped with an internal clock. The backup supports the display of the correct time for up to 10 hours if the power supply is interrupted.

Time-of-day format

The clock always shows the time of day with a resolution of 1 millisecond and the date including the day of the week. The time adjustment for daylight-saving time is also taken into account.

Setting and reading the time of day

Setting and reading the time with instructions

You can set, start and read the time-of-day and date on the CPU clock with the following instructions in the user program:

- Set the time-of-day: "WR_SYS_T"
- Read time of day "RD_SYS_T"
- Read local time "RD_LOC_T"
- Set time zone "SET_TIMEZONE"

Manual setting

You can also read and set the time-of-day manually in the online and diagnostics view under "Functions > Set time-of-day".

See also

WR_SYS_T: Set time-of-day (Page 2981)

RD_SYS_T: Read time-of-day (Page 2983)

RD_LOC_T: Read local time (Page 2985)

SET_TIMEZONE: Set time zone (Page 2989)

Assigning the clock parameters

Clock parameters

The clock parameters allow you to make the following settings:

- Enable time synchronization via NTP server
  Select this check box if you want the internal clock to be synchronized using the NTP synchronization mode.
- Network time server
  The IP addresses of up to four NTP servers need to be configured.
- Update interval
  The update interval defines the interval between time queries.

High-speed counters

General information on high-speed counters

Introduction

High-speed counters are typically used to drive counting mechanisms in which a shaft turning at a constant speed is equipped with an incremental step encoder. The incremental step encoder ensures a certain number of count values per rotation and a reset pulse once per rotation. The clock memory bit(s) and the reset pulse of the incremental step encoder supply the inputs for the high-speed counter.

The various S7-1200 CPUs have differing numbers of high-speed counters available:

S7-1200 CPU	Number of HSCs	HSC designation
CPU 1211C	3 (with digital signal board 4)*	HSC1…3 (and HSC5)*
CPU 1212C	4 (with digital signal board 5)*	HSC1…4 (and HSC5)*
CPU 1214C CPU 1215C CPU 1217C	6	HSC1…6

* with DI2/DO2 signal board

How it works

The first of several default values is loaded on the high-speed counter. The required outputs are enabled for the time during which the current value of the counter is lower than the default value. The counter is set up so that an interrupt occurs if the current value of the counter is equal to the default value or when the counter is reset.

If the current value is equal to the default value and an interrupt event results, a new default value is loaded and the next signal state is set for the outputs. If an interrupt event occurs because the counter is reset, the first default value and the first signal states of the outputs are set and the cycle repeated.

Since the interrupts occur much less frequently than the high-speed counter counts, a precise control of the fast operations can be implemented with only a slight influence on the overall cycle of the automation system. Since you can assign specific interrupt programs to interrupts, each new default can be loaded in a separate interrupt program allowing simple control of the state.

Note

You can also process all interrupt events in a single interrupt program.

Count algorithms of the various counters

All counters work in the same way, however some high-speed counters do not support all count algorithms. There are four basic count algorithms:

- Single-phase counter with internal direction control
- Single-phase counter with external direction control
- 2-phase counter with 2 clock inputs
- A/B counter

Each high-speed counter can be used with or without a reset input. If the reset input is activated, this resets the current value. The current value remains reset until the reset input is deactivated.

See also

Configuring high-speed counters (Page 1196)

Interdependency of the counter mode and counter inputs (Page 1194)

Interdependency of the counter mode and counter inputs

General information on counter mode and counter inputs

You can assign not only the counter modes and counter inputs to the high-speed counters but also functions such as clock pulse generator, direction control, and reset. The following rules apply:

- An input cannot be used for two different functions.
- If an input is not required by the current counter mode of the defined high-speed counter, it can be used other purposes.

If, for example, you set HSC1 to counter mode 1, in which inputs I0.0 and I0.3 are required, you can use I0.1 for edge interrupts or for HSC2.

If, for example, you set HSC1 and HSC5, inputs I0.0 (HSC1) and I1.0 (HSC5) are always used with the counting and frequency counter modes. As a result, these two inputs are not available for any other functions when counters are operated.

You have additional inputs available if you use a digital signal board.

Overview of the interdependency of counter mode and counter inputs

Counter mode	Description	Inputs		
	HSC1	I0.0 (CPU) I4.0 (signal board)	I0.1 (CPU) I4.1 (signal board)	I0.3 (CPU) I4.3 (signal board)
	HSC2	I0.2 (CPU) I4.2 (signal board)	I0.3 (CPU) I4.3 (signal board)	I0.1 (CPU) I4.1 (signal board)
	HSC3*	I0.4 (CPU)	I0.5 (CPU)	I0.7 (CPU)
	HSC4 (CPU 1212/14/15/17C only)	I0.6 (CPU)	I0.7 (CPU)	I0.5 (CPU)
	HSC5 (CPU 1214/15/17C only)**	I1.0 (CPU) I4.0 (signal board)	I1.1 (CPU) I4.1 (signal board)	I1.2 (CPU) I4.3 (signal board)
	HSC6 (CPU 1214/15/17C only)**	I1.3 (CPU)	I1.4 (CPU)	I1.5 (CPU)
Counting / frequency	Single-phase counter with internal direction control	Clock pulse generator	-	-
Counting				Resetting
Counting / frequency	Single-phase counter with external direction control	Clock pulse generator	Direction	-
Counting				Resetting
Counting / frequency	2-phase counter with 2 clock inputs	Clock pulse generator forwards	Clock pulse generator backwards	-
Counting				Resetting
Counting / frequency	A/B counter	Clock pulse generator A	Clock pulse generator B	-
Counting				Resetting
Motion axis	Pulse generators PWM/PTO	HSC1 and HSC2 support the motion axis count mode for the PTO1 and PTO2 pulse generators: • For PTO1, HSC1 evaluates the Q0.0 outputs for the number of pulses. • For PTO2, HSC2 evaluates the Q0.2 outputs for the number of pulses. Q0.1 is used as the output for the motion direction.		

* HSC3 can only be used for CPU 1211 without a reset input

** HSC5 can be also be used for CPU 1211/12 if a DI2/DO2 signal board is used

See also

General information on high-speed counters (Page 1193)

Configuring high-speed counters (Page 1196)

Configuring high-speed counters

Requirement

An S7-1200 CPU has been inserted in the hardware configuration.

Procedure

To configure a high-speed counter, follow these steps:

1. Select an S7-1200 CPU in the device or network view.
2. Click on the required high-speed counter under "Properties > High-speed counter" in the inspector window:
   - CPU 1211C: HSC1 to HSC3 (also HSC5 with a DI2/DO2 signal board)
   - CPU 1212C: HSC1 to HSC4 (also HSC5 with a DI2/DO2 signal board)
   - CPU 1214C / 1215C / 1217C: HSC1 to HSC6
3. Enable the high-speed counter in the "General" parameter group using the relevant check box.

   Note

   If you use a CPU 1211C or CPU 1212C with a DI2/DO2 signal board, you can also enable the high-speed counter HSC5.

   Note

   If you activate the pulse generators and operate them as PTO1 or PTO2, they use the associated high-speed counter HSC1 or HSC2 with "Motion axis" counting mode to evaluate the hardware outputs. If you configure high-speed counter HSC1 or HSC2 for other counting tasks, these cannot be used by pulse generator PTO1 or PTO2, respectively.

   If required, you can enter a name and a comment for the high-speed counter here.
4. Define the functionality of the high-speed counter in the "Function" parameter group:
   - Count mode: Select what you want to be counted from the drop-down list.
   - Operating phase: Select the count algorithm from the drop-down list.
   - Input source: Select the onboard CPU inputs or the inputs of an optional digital signal board as the input source for the count pulses from the drop-down list.
   - Count direction is specified by: If you have selected a single-phase operating phase, open the drop-down list and select whether the count direction is set internally by an SFB parameter of the user program or externally via a digital input.
   - Initial count direction: If the user program is set as the internal direction control for the count direction, you can select the count direction at the start of counting from the drop-down list.
   - Frequency meter period: If frequency is set as the count mode, you can select the duration of the frequency meter periods in the drop-down list.

5. Specify the initial values and reset condition of the high-speed counter in the "Reset to initial values" parameter group:
   - Initial counter value: Enter a start value for the high-speed counter.
   - Initial reference value: Enter a maximum value for the high-speed counter.

   Here, you can also specify whether the high-speed counter will use a reset input and the set the corresponding signal level for the reset input from the drop-down list.
6. Configure the reaction of the high-speed counter to certain events in the "Event configuration" parameter group. The following events can trigger an interrupt:
   - The counter value matches the reference value.
   - An external reset event was generated.
   - A change of direction was triggered.

   Enable an interrupt reaction using the check box, enter a name and select a hardware interrupt for the interrupt from the drop-down list.
7. Assign the start address for the high-speed counter in the "I/O/Diagnostics addresses" parameter group.

Note

In the "Hardware inputs" parameter group, you can only see which hardware inputs and values are being used for the clock, direction determination, reset pulse, and maximum count speed.

Result

You have now adapted the parameters of the high-speed counter to the requirements of your project.

See also

General information on high-speed counters (Page 1193)

Interdependency of the counter mode and counter inputs (Page 1194)

Point-to-point communication

Overview of point-to-point communication

PtP communication is communication via a serial interface that uses standardized UART data transmission (Universal Asynchronous Receiver/Transmitter). The S7-1200 uses communications modules with an RS-232 or RS-485 interface to establish PtP communication.

Functions of point-to-point communication

Point-to-point communication (PtP) allows numerous applications:

- Direct transmission of information to an external device, for example a printer or a barcode reader
- Reception of information from external devices such as barcode readers, RFID readers, cameras and third-party optical systems as well as many other devices.
- Exchange of information with third-party devices, for example GPS devices, radio modems and many others

The Freeport protocol

The S7-1200 supports the Freeport protocol for character-based serial communication. Using Freeport communication, the data transmission protocol can be configured entirely by the user program.

Siemens provides libraries with Freeport communication functions that you can use in your user program:

- USS Drive protocol
- Modbus RTU Master protocol
- Modbus RTU Slave protocol

See also

Configuring a communications port (Page 1199)

Using RS-232 and RS-485 communications modules

Communications modules with RS-232 and RS-485 interfaces

In an S7-1200 CPU, you can use two different communications modules:

- RS-232 communications module
- RS-485 communications module

The communications modules can be connected to the S7-1200 CPU via the I/O channel on the left. You can plug in up to three different modules.

Properties of the communications modules

The communications modules have the following features:

- Support of the Freeport protocol
- Configuration by the user program with the aid of expanded instructions and library functions

Configuring a communications port

Configuring a communications port

After you have inserted a communications module with an RS-232 or RS-485 interface, you then set the interface parameters. You set the parameters for the interface either in the properties of the interface or you control the interface parameters from the user program using the PORT_CFG instruction. The following description relates to the graphic configuration.

Note

If you use the user program to change the port setting, the settings of the graphic configuration are overwritten.

You should also keep in mind that the settings made by the user program are not retained if there is a power down.

Requirement

- A communications module is already plugged in.
- You are in the device view.

Procedure

To configure the communications port, proceed as follows:

1. Select the interface in the graphic representation in the device view.
   The properties of the interface are displayed in the Inspector window.
2. Select the "Port configuration" group in the area navigation of the Inspector window.
   The settings of the port are displayed.
3. From the "Transmission speed" drop-down list, select the speed for the data transmission.
   With user-programmed communication, remember the influence of the transmission speed on the changeover time.
4. From the "Parity" drop-down list, select the type of detection of bad information words.
5. Using the "Data bits" drop-down list, decide whether a character consists of eight or seven bits.
6. From the "Stop bit" drop-down list, select how many bits will identify the end of a transmitted word.

7. From the "Flow control" drop-down list, select the method for ensuring a trouble-free data stream between sender and receiver. This parameter can only be set for the RS-232 interface.
   - Enter a HEX value in the "XON character" box that will cause the transmission of the message to be continued when it is detected. This parameter can only be set for software-controlled data flow control.
   - Enter a HEX value in the "XOFF character" box that will cause the transmission of the message to be suspended for the set wait time. This parameter can only be set for software-controlled data flow control.
8. In the "Wait time" box, enter a wait time in ms that must be kept to after the end of the message before the next transmission can start.

Note

You can configure the interface in the network view as well. To do so, you must first select the communication module in the tabular network view and then select the interface in the Inspector window. Then you can continue as described above.

See also

Setting data flow control (Page 1200)

Setting data flow control

Data flow control

Data flow control is a method that ensures a balanced send and receive behavior. In the ideal situation, the intelligent control does not allow data to be lost. It ensures that a device does not send more information than the receiving partner can process.

There are two methods of data flow control:

- Hardware-controlled data flow control
- Software-controlled data flow control

With both methods, the DSR signals of the communications partners must be active at the beginning of transmission. If the DSR signals are inactive, the transmission is not started.

The RS-232 communications module can handle both methods. The RS-485 communications module does not support data flow control.

Hardware-controlled data flow control

Hardware-controlled data flow control uses the request to send (RTS) and clear to send (CTS) signals. With the RS-232 communications module, the RTS signal is transmitted via the output by pin 7. The CTS signal is received via pin 8.

If hardware-controlled data flow control is enabled, the RTS signal is then always set to activated when data is sent. At the same time, the CTS signal is monitored to check whether

the receiving device can accept data. If the CTS signal is active, the module can transfer data until the CTS signal becomes inactive. If the CTS signal is inactive, the data transfer must be suspended for the set wait time. If the CTS signal is still inactive after the set wait time, the data transfer is aborted and an error is signaled back to the user program.

Data flow control using hardware handshaking

If data flow control is controlled by hardware handshaking, the sending device sets the RTS signal to active as default. A device such as a modem can then transfer data at any time. It does not wait for the CTS signal of the receiver. The sending device itself monitors it own transmission by sending only a limited number of frames (characters), for example to prevent overflow of the receive buffer. If there is nevertheless an overflow, the transferring device must hold back the message and signal an error back to the user program.

Software-controlled data flow control

Software-controlled data flow control uses certain characters within the messages and these control the transfer. These characters are ASCII characters selected for XON and XOFF.

XOFF indicates when a transmission must be suspended. XON indicates when a transmission can be continued.

If the sending device receives the XOFF character, it must suspend sending for the selected wait time. If the XON character is sent after the selected wait time, the transfer is continued. If no XON character is received after the wait time, an error is signaled back to the user program.

Software data flow control requires full duplex communication because the receiving partner needs to send the XON character during the ongoing transfer.

See also

Configuring a communications port (Page 1199)

Configuration of message transfer

User-programmed communication

You can control the data traffic between a communications module and a device connected externally via the serial interface using your own mechanisms. If you want to do this, you will need to define a communications protocol yourself. In freely programmable communication, ASCII and binary protocols are supported for message transfer.

Within the communications protocol, you will need to specify the criteria by which the start and end of a transferred message can be recognized in the data stream.

User-programmed communication can only be activated in RUN mode. If there is a change to STOP mode, the user-programmed communication is stopped.

Specifying the communications protocol

You can specify the communications protocol as follows:

- With the user program
  - The behavior when sending data is controlled by the SEND_CFG instruction.
  - The behavior when receiving data is controlled by the RCV_CFG instruction.
- Using parameter settings set graphically in the Inspector window

Note

If you change the communications protocol from the user program, the settings of the graphic configuration are overwritten.

You should keep in mind that the settings made by the user program are not retained if there is a power down.

See also

User-programmed communication with RS-232 devices (Page 1202)

Making the settings for sending (Page 1204)

Specifying the start of the message (Page 1204)

Specifying the end of the message (Page 1205)

User-programmed communication with RS-232 devices

RS-232/PIP multi-master cable and user-programmed communication with RS-232 devices

Using the RS-232/PIP multi-master cable and user-programmed communication, you can connect a wide variety of RS-232-compliant devices to the communications modules of the S7-1200. The cable must, however, be set to the "PIP/user-programmed communication" mode.

Settings on the cable

The switches on the cable must be set as follows:

- Switch 5 must be set to 0
- Switch 6 sets either the local mode (DCE) or the remote mode (DTE):
  - Switch set to 0 for the local mode
  - Switch set to 1 for the remote mode

Changing over between send and receive mode

The RS-232/PIP multi-master cable is in send mode when data is sent from the RS-232 interface to the RS-485 interface. The cable is in receive mode when it is idle or when data is sent from the RS-485 interface to the RS-232 interface. The cable changes from receive to send mode immediately when it detects characters on the RS-232 send line.

Supported transmission speeds

The RS-232/PIP multi-master cable supports transmission rates between 1200 baud and 115.2 kbaud. The RS-232/PIP multi-master cable can be set to the required transmission rate using the DIP switch on the PC/PIP cable.

The following table shows the switch settings for the various transmission speeds:

Transmission speed	Switchover time	Settings (1 = up)
115200 bps	0.15 ms	110
57600 bps	0.3 ms	111
38400 bps	0.5 ms	000
19200 bps	1.0 ms	001
9600 bps	2.0 ms	010
4800 bps	4.0 ms	011
2400 bps	7.0 ms	100
1200 bps	14.0 ms	101

The cable returns to receive mode when the RS-232 send line is idle for a certain time that is defined as the changeover time of the cable. The set transmission speed influences the changeover time as shown in the table.

Influence of the changeover time

When working with an RS-232/PIP multi-master cable in a system in which user-programmed communication is used, the program must take into account the changeover time for the following reasons:

- The communications module reacts to messages sent by the RS-232 device.
  Once the communications module has received a request from the RS-232 device, it must delay the reaction message for a period that is equal to or longer than the changeover time of the cable.
- The RS-232 device reacts to messages sent by the communications module.
  Once the communications module has received a reaction message from the RS-232 device, it must delay the next request message for a period that is equal to or longer than the changeover time of the cable.

In both situations, the RS-232-PIP multi-master cable has enough time to change from send to receive mode so that the data can be sent from the RS-485 interface to the RS-232 interface.

See also

Configuration of message transfer (Page 1201)

Making the settings for sending (Page 1204)

Specifying the start of the message (Page 1204)

Specifying the end of the message (Page 1205)

Making the settings for sending

Sending messages

You can program pauses between individual messages.

The following table shows which pauses can be set:

Parameter	Definition
RTS ON delay	You can set the time that must elapse after the send request RTS (request to send) before the actual data transfer starts.
RTS OFF delay	You can set the time that must elapse after the complete transfer before RTS signal is deactivated.
Send pause at the start of the message	You can specify that a pause is sent at the start of every message transfer when the RTS ON delay has elapsed. The pause is specified in bit times.
Send Idle Line after a pause	You can make a setting so that following a selected pause at the start of the message, the "Idle Line" signal is output to signal that the line is not in use. To enable the parameter, "Send pause at message start" must be set. The duration of the "Idle Line" signal is specified in bit times.

See also

Specifying the start of the message (Page 1204)

Specifying the end of the message (Page 1205)

User-programmed communication with RS-232 devices (Page 1202)

Specifying the start of the message

Recognizing the start of the message

To signal to the receiver when the transfer of a message is completed and when the next message transfer starts, criteria must be specified in the transmission protocol to identify the end and start of a message.

If a criterion is met that indicates the start of a message, the receiver starts searching the data stream for criteria that mean the end of the message.

There are two different methods for identifying the start of a message:

- Starting with any character:
  Any character can defined the start of a message. This is the default method.
- Starting with a specific condition:
  The start of a message is identified based on selected conditions.

Conditions for detecting the start of a message

The following table shows the various options for defining the start of a message:

Parameter	Definition
Recognize start of message by line break	The receiver recognizes a line break when the received data stream is interrupted for longer than one character. If this is the case, the start of the message is identified by the line break.
Recognize start of message by idle line	The start of a message is recognized when the send transmission line is in the idle state for a certain time (specified in bit times) followed by an event such as reception of a character.
Recognize start of message with individual characters	The start of a message is recognized when a certain character occurs. You can enter the character as a HEX value.
Recognize start of message by a character string	The start of a message is detected when one of the specified character sequences arrives in the data stream. You can specify up to four character sequences each with up to five characters.

The individual conditions can be logically linked in any way.

See also

Making the settings for sending (Page 1204)

User-programmed communication with RS-232 devices (Page 1202)

Specifying the end of the message

Recognizing the end of the message

To signal to the receiver when the transfer of a message is completed and when the next message transfer starts, criteria must be specified in the transmission protocol to identify the end and start of a message.

In total, there are six different methods of recognizing the end of a message and these can all be logically linked in any way. The following table shows the various possible setting options:

Parameter	Definition
Recognize end of message by message timeout	The end of a message is recognized automatically when a selected maximum duration for a message is exceeded. Values from 0 to 65535 ms can be set.
Recognize end of message by reply timeout	The end of a message is recognized when there is no reply within a set time after transferring data. Values from 0 to 65535 ms can be set.

Parameter	Definition
Recognize end of message by timeout between characters	The end of a message is detected when the time between two characters specified in bit times is exceeded. Values from 0 to 2500 bit times can be set. The S7-1200 CPU only accepts a maximum time of eight seconds even if the value that is set results in a duration of more than eight seconds.
Recognize end of message by maximum length	The end of a message is recognized when the maximum length of a message is exceeded. Values from 1 to 1023 characters can be set.
Read message length from message	The message itself contains information about the length of the message. The end of a message is reached when the value taken from the message is reached. Which characters are used for the evaluation of the message length is specified with the following parameters: • Offset of the length field in the message The value decides the position of the character in the message that will be used to indicate the message length. Values from 1 to 1022 characters can be set. • Size of the length field This value specifies how many characters starting at the first evaluation position will be used to indicate the message length. Values of 0, 1, 2 and 4 characters can be set. • The data following the length field (does not belong to the message length) The value specifies the number of bytes after the length field that must be ignored in the evaluation of the message length. Values from 0 to 255 characters can be set.
Recognize message end with a character sequence	The end of a message is detected when the specified character sequence arrives in the data stream. You can define up to five characters in the character string that are to be checked. If the specified characters appear at the correct location in the message, the message end is recognized. To recognize the message end when character 1 and character 3 have a certain value, for example, you have to activate the check box for character 1 and character 3 and enter a character value.

See also

Making the settings for sending (Page 1204)

User-programmed communication with RS-232 devices (Page 1202)

Enabling system memory

System memory

A system memory is a bit memory with defined values.

You decide which memory byte of the CPU will become the system memory byte when assigning the system memory parameters.

Benefits

You can use system memory in the user program, for example to run program segments in only the first program cycle after start-up. Two system memory bits are constant 1 or constant 0.

Bits of the system memory bytes

The following table shows the meaning of the system memory:

Bit of the system memory bytes	7	6	5	4	3	2	1	0
Meaning	Reserved (=0)	Reserved (=0)	Reserved (=0)	Reserved (=0)	=0	=1	=1 with change to the diagnosis status	=1 in first program cycle after startup, otherwise 0

Note

The selected memory byte cannot be used for intermediate storage of data.

Using clock memory

Clock memory

A clock memory is a bit memory that changes its binary status periodically in the pulse-no-pulse ratio of 1:1.

You decide which memory byte of the CPU will become the clock memory byte when assigning the clock memory parameters.

Benefits

You can use clock memory, for example, to activate flashing indicator lamps or to initiate periodically recurring operations such as recording of actual values.

Available frequencies

Each bit of the clock bit memory byte is assigned a frequency. The following table shows the assignment:

Bit of the clock memory byte	7	6	5	4	3	2	1	0
Period (s)	2.0	1.6	1.0	0.8	0.5	0.4	0.2	0.1
Frequency (Hz)	0.5	0.625	1	1.25	2	2.5	5	10

Note

Clock memory runs asynchronously to the CPU cycle, i.e. the status of the clock memory can change several times during a long cycle.

The selected memory byte cannot be used for intermediate storage of data.

Protection

Setting options for the protection level (FW V1 to V3)

Protection level

The following section describes how to use the various protection levels of the S7-1200 CPUs V1 to V3.

Effects of the protection level setting

You can choose between the following protection levels:

- No protection: This corresponds to the default behavior. You cannot enter a password. Read and write access is always permitted.
- Write protection: Only read-only access is possible. You cannot change any data on the CPU and cannot load any blocks or a configuration. HMI access and communication between CPUs are excluded from the write protection. Assignment of a password is required to select this protection level.
- Write/read protection: No write or read access is possible in the "Accessible devices" area or in the project for devices that are switched online. Only the CPU type and the identification data can be displayed in the project tree under "Accessible devices". Display of online information or blocks under "Accessible devices", or in the project for devices interconnected online, is possible.
  HMI access and communication between CPUs are excluded from the write protection.
  Assignment of a password is required to select this protection level.

Behavior of a password-protected CPU during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Validity is checked before the online function is executed. If password protection is in place, you are prompted to enter a password.

Example: The module was assigned write protection and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on with a password.

Access authorization to the protected data is in effect for the duration of the online connection or until the access authorization is manually rescinded with "Online > Delete access rights". Access authorization will also expire when the project is closed.

Note

You can not restrict functions for process control, monitoring, and communications.

Some functions are still protected due to their use as online data. RUN/STOP in the "Online Tools" task card or "Set the time" in the diagnostics and online editor is therefore write-protected.

Setting options for the protection (FW as of V4)

Protection level

The following section describes how to use the various access levels of the S7-1200 CPUs as of V4.

S7-1200 CPUs provide various access levels to limit the access to specific functions.

The parameters for the access levels are assigned in a table. The green checkmarks in the columns to the right of the respective access level specify which operations are possible without knowing the password of this access level. If you want to use the functions of check boxes that are not selected, a password has to be entered.

Notice

Configuring an access level does not replace know-how protection

Configuring access levels prevents unauthorized changes to the CPU by restricting download privileges. However, blocks on the memory card are not write- or read-protected. Use know-how protection to protect the code of blocks on the memory card.

Default characteristics

The default access level is "Full access (no protection)". Every user can read and change the hardware configuration and the blocks. A password is not set and is also not required for online access.

The access levels in detail

With an S7-1200 CPU, you can configure the following access levels:

- Full access (no protection): The hardware configuration and the blocks can be read and changed by all users.
- Read access: With this access level, only read access to the hardware configuration and the blocks is possible without entering a password - meaning that you can load the hardware configuration and blocks into the programming device. In addition, HMI access and access to diagnostics data is possible.
  You cannot load blocks or a hardware configuration into the CPU without entering the password. Moreover, writing test functions and firmware updates are **not** possible without a password.
- HMI access: With this access level, only HMI access and access to diagnostics data is possible without entering the password.
  Without entering the password, you can neither load blocks and hardware configuration into the CPU, nor load blocks and hardware configuration from the CPU into the programming device. In addition, the following is **not** possible without a password: Writing test functions, changing the operating state (RUN/STOP) and firmware updates.
- No access (complete protection): When the CPU is completely protected, no read or write access to the hardware configuration and the blocks is possible. HMI access is also not possible. The server function for PUT/GET communication is disabled in this access level (cannot be changed).
  Authorization with the password again provides you full access to the CPU.

Behavior of a password-protected module during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Validity is checked before the online function is executed. If password protection is in place, you are prompted to enter a password.

Example: The module was configured with read access and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on.

Access authorization to the protected data is in effect for the duration of the online connection or until the access authorization is manually rescinded with "Online > Delete access rights".

Each access level allows unrestricted access to certain functions without entering a password, for example, identification using the "Accessible devices" function.

Configuring access levels

The following section describes how to configure an access level and enter passwords for an S7-1200 CPU as of V4.

For an S7-1200 CPU, you can enter multiple passwords and thereby set up different access rights for individual user groups.

The passwords are entered in a table in such a way that exactly one access level is assigned to each password.

The effect of the password is visualized in the "Access" column and explained in the text below the table.

Example

You select the "No access (complete protection)" access level for a standard CPU (in other words, not an F-CPU) and enter a separate password for each of the access levels that lie above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the set passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (Full access (no protection)) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (read access) allows access as if the CPU were write-protected. Users who know this password have read-only access to the CPU.
- The password in row 3 (HMI access) allows access as if the CPU were write-protected and read-protected so that only HMI access is possible for users who know this password.

Procedure

To configure the access levels of an S7-1200 CPU, follow these steps:

1. Open the properties of the module in the inspector window.
2. Open the "Protection" entry in the area navigation.
   A table with the possible access levels appears in the inspector window.

Protection level	Access			Access permission	
Protection	HMI	Read	Write	Password	Confirmation
Full access (no protection)	✓	✓	✓		
Read access	✓	✓			
HMI access	✓				
No access (complete protection)					

Figure 10-4 Access protection

3. Activate the required access level in the first column of the table. The green checkmarks in the columns to the right of the respective protection level show you which operations are still available without entering the password.

4. In the "Password" column, specify a password for full access in the first row. In the "Confirmation" column, enter the selected password again to protect against incorrect entries.
   Ensure that the password is sufficiently secure, in other words, that is does not follow a machine-recognizable pattern!
   You must enter a password in the first row "Full access (no protection)". This enables unrestricted access to the CPU for those who know the password, regardless of the selected protection level.
5. Assign additional passwords as needed to other access levels if the selected protection level allows you to do so.
6. Download the hardware configuration so that the access level takes effect.

Result

The hardware configuration and the blocks are protected against unauthorized access according to the set access level. If an operation cannot be executed without a password due to the set access level, a dialog for entering a password is displayed.

Restriction of communication services

Introduction

The CPU can be the server for a number of communication services. This means that other communication participants can access CPU data even if you have not configured and programmed connections for the CPU.

The local CPU as a server thus does not have the possibility to control communication to the clients.

The parameter "Connection mechanisms" in the "Protection" area of the CPU parameters is used to specify whether this type of communication is permitted or not for the local CPU during operation.

Permit access with PUT/GET communication from remote partners

By default, the "Permit access with PUT/GET communication from remote partners (...)" option is disabled. In this case, read and write access to CPU data is only possible for communication connections that require configuration or programming both for the local CPU and for the communication partner. Access through BSEND/BRCV instructions is possible, for example.

Connections for which the local CPU is only a server (meaning that no configuration/ programming of the communication with the communication partner exists at the local CPU), are therefore not possible during operation of the CPU, for example,

- For PUT/GET, FETCH/WRITE or FTP access via communication modules
- For PUT/GET access from other S7 CPUs
- For HMI access that is realized via PUT/GET communication

If you want to allow access to CPU data from the client side, meaning that you do not want to restrict the communication services of the CPU, activate the "Permit access with PUT/GET communication from remote partners" option.

Organization blocks

Organization blocks for startup

Description

You can determine the boundary conditions for the startup characteristics of your CPU, for example, the initialization values for "RUN". To do this, write a startup program. The startup program consists of one or more startup OBs (OB numbers 100 or >= 123).

The startup program is executed once during the transition from "STOP" mode to "RUN" mode. Current values from the process image of the inputs are not available for startup program, nor can these values be set.

After the complete execution of the startup OBs, the process image of the inputs is read in and the cyclic program is started.

There is no time limit for executing the startup routine. Therefore the scan cycle monitoring time is not active. Time-driven or interrupt-driven organization blocks cannot be used.

Start information

A startup OB has the following start information:

Tag	Data type	Description
LostRetentive	BOOL	= 1, if retentive data storage areas have been lost
LostRTC	BOOL	= 1, if realtime clock has been lost

See also

Events and OBs (Page 1177)

Organization blocks for cyclic program execution

Introduction

For the program execution to start, at least one program cycle OB must be present in the project. The operating system calls this program cycle OB once in each cycle and thereby starts the execution of the user program. You can use multiple OBs (OB numbers >= 123). When multiple program cycle OBs are used, these are called in one after the other in the order of their OB numbers. The program cycle OB with the lowest OB number is called first.

The program cycle OBs have the priority class 1. This corresponds to the lowest priority of all OBs. The cyclic program can be interrupted by events of any other event class.

Programming cyclic program execution

You program cyclic program execution by writing your user program in the cycle OBs and the blocks that they call.

The first cyclic program execution begins as soon as the startup program has ended without errors. The cycle restarts after the end of each cyclic program execution.

Sequence of cyclic program execution

One cycle of the program execution encompasses the following steps:

1. The operating system starts the maximum cycle time.
2. The operating system writes the values from the process image output to the output modules.
3. The operating system reads out the state of the inputs of the input modules and updates the process image input.
4. The operating system processes the user program and executes the operations contained in the program.
5. At the end of a cycle, the operating system executes any tasks that are pending, for example, loading and deleting blocks or calling other cycle OBs.
6. Finally, the CPU returns to the start of the cycle and restarts the scan cycle monitoring time.

See also: Auto-Hotspot

Options for interrupting

Cyclic program execution can be interrupted by the following events:

- Interrupt
- A STOP command, triggered by
  - Operation of the programming device
  - "STP" instruction
- Supply voltage failure
- Occurrence of a device fault or program error

Start information

- None
- Optimized start information:

Name	Data type	Meaning
first_scan	BOOL	= TRUE in the first call of this OB: • Transition from STOP or HOLD to RUN • After reloading
retentivity	BOOL	= TRUE, if retentive data are available

See also

Events and OBs (Page 1177)

Organization blocks for interrupt-driven program execution

Organization blocks for time-of-day interrupt

Function

Organization blocks for time-of-day interrupt (OB number >=123) can be processed as follows:

- One time to a preset time (date with time of day)
- Periodically with preset start time and the following intervals:
  - Every minute
  - Hourly
  - Daily
  - Weekly
  - Monthly
  - Yearly
  - End of month

Time-of-day interrupt-OBs are therefore used to run parts of the user program on a time-controlled basis.

Status for time-of-day interrupt

The following table contains the possible status of a time-of-day interrupt as well as it's meaning.

Status	Meaning
Cancelled	The one-time processing has already taken place, or the start event of an not yet processed time-of-day interrupt has been deleted with the extended instruction CAN_TINT.
Set	You have scheduled the time or start time for processing.
Activated	You have scheduled whether processing takes place one time or periodically, and in the case of periodic processing, you have scheduled the interval.

Rules for time-of-day interrupts

The following rules apply to the use of time-of-day interrupts:

- A time-of-day interrupt can only be processed if it is set and activated, and a corresponding organization block exists in the user program.
- The start times of periodic time-of-day interrupts must correspond to a real date. For example, it is not possible to repeat an organization block monthly which first occurs on January 31st. In this case, on OB will only be started in the months that have 31 days.

- A time-of-day interrupt activated during startup by extended instruction call ACT_TINTwill not be executed until the startup is complete.
- After each CPU startup, you must reactivate previously set time-of-day interrupts

Setting and activating a time-of-day interrupt OB

Before a time-of-day interrupt can be deleted and the corresponding time-of-day interrupt OB can be processed from the operating system, you must set and activate the interrupt. You have the following options:

Set time-of-day interrupt	Activate time-of-day interrupt
Means of configuring	Means of configuring
Means of configuring	By extended instruction call ACT_TINT
By extended instruction call SET_TINTL	By extended instruction call ACT_TINT

Note

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed once, the start time must not be in the past (relative to the real-time clock of the CPU).

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed periodically, but the start time is in the past, then the time-of-day interrupt will be processed the next time it is due.

Time-of-day interrupt status query

In order to query the status of the time-of-day interrupt, call the extended instruction QRY_TINT.

Cancelling a time-of-day interrupt

You can cancel a time-of-day interrupt which has not yet been processed with the extended instruction CAN_TINT.

You can reset cancelled time-of-day interrupts with the extended instruction SET_TINTL and activate them with the extended instruction ACT_TINT.

Conditions that effect the time-of-day interrupt OBs

Since a time-of-day interrupt occurs only at specific intervals, certain conditions can affect the function of the associated OBs during execution of your program. The following table shows some of these conditions and describes how these affect the processing of a time-of-day interrupt OB.

Condition	Result
The extended instruction CAN_TINT is called in the user program.	The operating system deletes the start event (date and time) of the time-of-day interrupt. You must reset and reactivate the time-of-day interrupt if you are going to call the corresponding time-of-day interrupt OB again.
By synchronizing or correcting the CPU system clock, the time of day will be set ahead. With this, the start time for a time-of-day interrupt OB will be bypassed.	The operating system calls the time error interrupt OB (OB 80) and records the start event, the number, and the priority of the first bypassed time-of-day interrupt OB in its start information. After completion of the OB 80, the operating system processes the bypassed time-of-day interrupt OB only once.
By synchronizing or correcting the CPU system clock, the time of day will be set back. The corrected clock time is before the start time of an already processed time-of-day interrupt OB.	The time-of-day interrupt OB is repeated.
A time-of-day interrupt OB is still being processed when the start event for its next execution occurs.	The operating system then calls time error interrupt OB 80. The requested OB is processed only after the processing and further execution of the current time-of-day interrupt OB has completed.

Start information

A time-of-day interrupt OB has the following start information:

Tag	Data type	Description
CaughtUp	BOOL	=1, if the OB call is executed because clock time is turned ahead.
SecondTime	BOOL	=1 if the OB is called a second time because clock time is turned back (more exactly, if the planned time of the current OB processing is earlier than or the same time as the planned time for the previous OB processing). Note: SecondTime is only set once.

Organization block for status interrupts

Description

When it receives a status interrupt, the operating system of the S7-1200 CPU calls the status interrupt OB from a DP master or IO controller. This may be the case if a module of a slave changes its operating mode, for example, from "RUN" to "STOP". For more detailed information on events that trigger a status interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

The status-interrupt OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware address of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame

See also

Events and OBs (Page 1177)

Organization block for update interrupts

Description

When it receives a status interrupt, the operating system of the S7-1200 CPU calls the update interrupt OB from a DP master or IO controller. This may be the case if you changed a parameter on a slot of a slave or device. For more detailed information on events that trigger an update interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

The update interrupt OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware address of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame

See also

Events and OBs (Page 1177)

Organization block for manufacturer-specific or profile-specific interrupts

Description

When it receives a manufacturer-specific or profile-specific interrupt from a DP master or IO controller, the operating system of the S7-1200 CPU calls OB 57. For more detailed information on events that trigger this type of interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

The OB for manufacturer-specific and profile-specific interrupts has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware address of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame

See also

Events and OBs (Page 1177)

Organization blocks for time-delay interrupts

Description

A time-delay interrupt OB is started after a configurable time delay of the operating system. The delay time starts after the SRT_DINT instruction is called.

You can use up to four time-delay interrupt OBs or cyclic OBs (OB numbers >= 123) in your program. If, for example, you are already using two cyclic interrupt OBs, you can insert a maximum of two further time-delay interrupt OBs in your program.

You can use the CAN_DINT instruction to prevent the execution of a time-delay interrupt that has not yet started.

Function of time-delay interrupt OBs

The operating system starts the corresponding OB after the delay time, which you have transferred with an OB number and an identifier to the SRT_DINT instruction.

To use a time-delay interrupt OB, you must execute the following tasks:

- You must call the instruction SRT_DINT.
- You must download the time-delay interrupt OB to the CPU as part of your program.

The delay time is measured with a precision of 1 ms. A delay time can immediately start again after it expires.

Time delay interrupt OBs are executed only when the CPU is in the "RUN" mode. A warm restart clears all start events of time-delay interrupt OBs.

The operating system calls the time-delay interrupt OB if one of the following events occurs:

- If the operating system attempts to start an OB that is not loaded and you specified its number when calling the SRT_DINT instruction.
- If the next start event for a time-delay interrupt occurs before the time delay OB has completely executed.

You can disable and re-enable time-delay interrupts using the DIS_AIRT and EN_AIRT instructions.

Note

If you disable an interrupt with DIS_AIRT after executing SRT_DINT, this interrupt executes only after it has been enabled with EN_AIRT. The delay time is extended accordingly.

Start information

- None
- Optimized start information:

Name	Data type	Meaning
sign	WORD	User ID: Input parameter SIGN from the call of the "SRT_DINT" instruction

See also

SRT_DINT: Start time-delay interrupt (Page 3230)

CAN_DINT: Cancel time-delay interrupt (Page 3231)

Events and OBs (Page 1177)

Organization blocks for cyclic interrupts

Description

Cyclic interrupt OBs serve to start program in periodic time intervals independently of the cyclic program execution. The start times of a cyclic interrupt OB are specified using the time base and the phase offset.

The time base defines the intervals at which the cyclic interrupt OB is started and is an integer multiple of the basic clock cycle of 1 ms. The phase offset is the time by which the start time is offset compared with the basic clock cycle. If several cyclic interrupt OBs are being used, you can use this offset to prevent a simultaneous start time if the time bases of the cyclic interrupt OBs have common multiples.

You can specify a time period between 1 ms and 60000 ms as the time base.

You can use up to four cyclic interrupt OBs or time-delay OBs (OB numbers >= 123) in your program. If, for example, you are already using two time-delay interrupt OBs, you can insert a maximum of two further cyclic interrupt OBs in your program.

Note

The execution time of each cyclic interrupt OB must be noticeably smaller than its time base. If a cyclic interrupt OB has not been completely executed but execution is again pending because the cycle clock has expired, the time error OB is started. The cyclic interrupt that caused the error is executed later or discarded.

Example of the use of phase offset

You have inserted two cyclic interrupt OBs in your program:

- Cyclic interrupt OB1
- Cyclic interrupt OB2

For cyclic interrupt OB1, you have set a time base of 20 ms and for cyclic interrupt OB2 a time base of 100 ms. After expiration of the time base of 100 ms, the cyclic interrupt OB1 reaches the start time for the fifth time, cyclic interrupt OB2 for the first time. To nevertheless execute the cyclic interrupt OBs offset, enter a phase offset for one of the two cyclic interrupt OBs.

Start information

- None
- Optimized start information:

Name	Data type	Meaning
first_scan	BOOL	= TRUE in the first call of this OB • At the transition from STOP or HOLD to RUN • After reloading
event_count	INT	Number of lost start events since the last start of this OB

See also

Assigning parameters to cyclic interrupt OBs (Page 1229)

Events and OBs (Page 1177)

Organization blocks for hardware interrupts

Description

You can use hardware interrupt OBs to react to specific events. You can assign an event that triggers an alarm to precisely one hardware interrupt OB. A hardware interrupt OB in contrast can be assigned to several events.

Hardware interrupts can be triggered by high-speed counters and input channels. For each high-speed counter and input channel that should trigger a hardware interrupt, the following properties need to be configured:

- The process event that should trigger the hardware interrupt (for example, the change of a count direction of a high-speed counter)
- The number of the hardware interrupt OB which is assigned to this process event

You can use up to 50 hardware interrupt OBs (OB numbers >= 123) that are independent of each other in your program.

Functionality of a hardware interrupt OB

After triggering a hardware interrupt, the operating system identifies the channel of the input or the high-speed counter and determines the assigned hardware interrupt OB.

If no other interrupt OB is active, the determined hardware interrupt OB is called. If a different interrupt OB is already being executed, the hardware interrupt is placed in the queue of its priority class. The hardware interrupt is acknowledged after the completion of the assigned hardware interrupt OB.

If another event that triggers a hardware interrupt occurs on the same module during the time between identification and acknowledgement of a hardware interrupt, the following applies:

- If the event occurs on the channel that previously triggered the hardware interrupt, then no additional hardware interrupt is triggered. Another hardware interrupt can only be triggered if the current hardware interrupt is acknowledged.
- If the event occurs on a different channel, a hardware interrupt is triggered.

Hardware interrupt OBs are called only in the CPU's "RUN" mode.

Start information

- None
- Optimized start information:

Name	Data type	Meaning
Laddr	HW_IO	Hardware identifier of the module that triggers the hardware interrupt
USI	WORD	Identifier for future extensions (not user-relevant)
IChannel	USINT	Number of the channel that triggered the hardware interrupt
EventType	BYTE	Identifier for the event type associated with the event triggering the interrupt (e.g., positive edge) This identifier can be found in the description of the respective module.

See also

Assigning parameters to hardware interrupt OBs (Page 1230)

Events and OBs (Page 1177)

Organization block for time error

Description

The operating system calls the time error OB (OB 80) if one of the following events occurs:

- The cyclic program exceeds the maximum cycle time.
- The OB called is currently still being executed (possible for time-delay interrupt OBs and cyclic interrupt OBs).
- A time-of-day interrupt was missed because the clock time was set forward by more than 20 seconds.
- A time-of-day interrupt was missed during a STOP.
- An overflow has occurred in an interrupt OB queue.
- An interrupt was lost due to the excessive interrupt load.

If you have programmed no time error OB, the S7-1200 CPU reacts as follows:

- CPUs with firmware version V1.0: The CPU remains in RUN mode.
- CPUs with firmware version V2.0:
  - The CPUs goes to STOP mode when the maximum cycle time is exceeded.
  - With all other start events of the time error OB, the CPU remains in RUN mode.

With CPUs with firmware version V1.0, the two-time overshooting of the maximum cycle time does not lead to the calling of an OB, but to the STOP of the CPU. You can avoid the second violation by restarting the cycle monitoring of the CPU with the RE_TRIGR instruction.

You can use only one time error OB in your program.

Start information

The time error OB has the following start information:

Tag	Data type	Description
fault_id	BYTE	• 0x01: Maximum cycle time exceeded • 0x02: Called OB is still being executed • 0x05: Expired time-of-day interrupt due to time jump • 0x06: Expired time-of-day interrupt on return to RUN mode • 0x07: Queue overflow • 0x09: Interrupt loss due to high interrupt load
csg_OBnr	OB_ANY	Number of the OB being executed at the time of the error
csg_prio	UINT	Priority of the OB being executed at the time of the error

See also

Events and OBs (Page 1177)

Organization block for diagnostic interrupts

Description

You can enable the diagnostic error interrupt for diagnostics-capable modules so that the module detects changes in the I/O status. As a result, the module triggers a diagnostic error interrupt in the following cases:

- A fault is present (incoming event)
- A fault is no longer present (outgoing event)

If no other interrupt OB is active, then the diagnostic interrupt OB (OB 82) is called. If another interrupt OB is already being executed, the diagnostic error interrupt is placed in the queue of its priority group.

You can use only one diagnostic interrupt OB in your program.

Start information

The diagnostic interrupt OB has the following start information:

Tag	Data type	Description
IO_state	WORD	Contains the I/O status of the diagnostics-capable module.
laddr	HW_ANY	HW-ID
Channel	UINT	Channel number
multi_error	BOOL	= 1, if there is more than one error

IO_state tag

The following table shows the possible I/O states that the IO_state tag can contain:

IO_state	Description
Bit 0	Configuration correct: = 1, if the configuration is correct = 0, if the configuration is no longer correct
Bit 4	Error: = 1, if an error is present, e.g., a wire break = 0, if the error is no longer present
Bit 5	Configuration not correct: = 1, if the configuration is not correct = 0, if the configuration is correct again
Bit 6	I/O cannot be accessed: = 1, if an I/O access error has occurred In this case, laddr contains the hardware identifier of the I/O with the access error. = 0, if the I/O can be accessed again

See also

Events and OBs (Page 1177)

Organization block for pulling and plugging

Description

The S7-1200 CPU operating system calls the pull/plug interrupt OB (OB 83) if a configured and non-disabled distributed I/O module or submodule (PROFIBUS, PROFINET, AS-i) is removed or inserted.

Note

The removal or insertion of a central module leads to the STOP of the CPU.

Start information

The pull/plug interrupt OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the affected module or submodule
Event_Class	BYTE	• B#16#38: (Sub)module plugged • B#16#39: (Sub)module pulled or not responding
Fault_ID	BYTE	Error code (possible values: B#16#51, B#16#54, B#16#55, B#16#56, B#16#57, B#16#58)

The following table shows which event caused the start of the pull/plug interrupt OB.

ev_class (B#16# ...)	fault_id (B#16# ...)	Meaning
39	51	Module pulled
39	54	Submodule pulled
38	54	Submodule plugged, which conforms to the parameterized submodule
38	55	Submodule inserted, which does not conform to the submodule parameter assignment
38	56	Submodule inserted, but error in module parameter assignment
38	57	Submodule or module inserted, but with a fault or maintenance
38	58	Submodule access error remedied

See also

Events and OBs (Page 1177)

Organization block for rack fault

Description

The S7-1200 CPU operating system calls the OB 86 in the following cases:

- The failure of a DP master system or of a PROFINET IO system is detected (in the case of either an incoming or an outgoing event).
- The failure of a DP slave or of an IO device is detected (in the case of either an incoming or an outgoing event).
- Failure of some of the submodules of a PROFINET I-device is detected.

Structure of the start information

The rack fault OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the defective hardware object
Event_Class	BYTE	• B#16#32: Activation of a DP slaves or an IO device • B#16#33: Deactivation of a DP slaves or an IO device • B#16#38: outgoing event • B#16#39: incoming event
Fault_ID	BYTE	Error code (possible values: B#16#C3, B#16#C4, B#16#C5, B#16#C6, B#16#C7, B#16#C8, B#16#C9, B#16#CA, B#16#CB, B#16#CC, B#16#CD, B#16#CE, B#16#CF, B#16#F8, B#16#F9)

The following table shows which event caused the start of OB 86.

Ev_class B#16# ...	Fault_id B#16# ...	Meaning
39	C3	Failure of a DP master system
39/38	C4	Failure/return of a DP slave
38	C5	Return of a DP slave; however, slave is faulty
38	C6	Expansion unit return, but error in module parameter assignment
38	C7	Return of a DP device, but there is error in module configuration
38	C8	Return of a DP device, but discrepancy between preset/actual configuration
32/33	C9	Activation/deactivation of a DP slave with the "D_ACT_DP" instruction
39	CA	Failure of a PROFINET IO system
39/38	CB	Failure/return of a PROFINET IO device
38	CC	Return of a PROFINET IO device with fault or maintenance
38	CD	Return of a PROFINET IO device, deviation between preset/actual configuration
38	CE	Return of a PROFINET IO device, error in module configuration
32/33	CF	Activation/deactivation of an IO device with the "D_ACT_DP" instruction
39/38	F8	Failure/return of some of the submodules of a PROFINET I-device
38	F9	Return of some of the submodules of a PROFINET I-device with a device configuration difference

See also

Events and OBs (Page 1177)

Block parameters of organization blocks

Basics of block parameters

Introduction

Several organization blocks (OBs) have properties with which you can control their behavior or their assignment to specific events. You can influence these properties by assigning parameters.

Overview

You can assign parameters to the properties for the following organization blocks:

- Time-of-day interrupt OBs
- Cyclic interrupt OBs
- Hardware interrupt OBs

See also

Assigning parameters to hardware interrupt OBs (Page 1230)

Assigning parameters to cyclic interrupt OBs (Page 1229)

Parameter assignment for time-of-day interrupt OBs

Procedure for setting the parameters

To set the parameters of a time-of-day interrupt OB, proceed as follows:

1. Open the "Properties" dialog belonging to the time-of-day interrupt OB in question.
2. Click the "Time-of-day interrupt" group in the area navigation.

Overview of the parameters that can be set

You can set the following parameters:

- Execution
- Start date and time-of-day
- Option buttons "Local time" and "System time"

"Execution" parameter

Use the drop-down list "Execution" to define the periods at which the time-of-day interrupt OB is to be executed. The time intervals relate to the settings for "Start date" and "Time-of-day".

The following values are possible for "Execution":

- Never
- Once
- Every minute
- Hourly
- Daily
- Weekly
- Monthly
- Yearly
- End of month

Note

With the value "End of month", the value specified under "Start date" is irrelevant.

"Start date" and "Time-of-day" parameter

Here, you specify the time at which the time-of-day interrupt is to be executed for the first time.

Example: Start date = 07/05/2013, time =11:16

Depending on the value of the "Execution" parameter, the CPU generates additional time-of-day interrupts periodically. Depending on the setting, the start time relates either to the local time or to the coordinated universal timeUTC.

Note

If you set the "Execution" parameter to "Monthly", the start date cannot be set to the 29th, the 30th, or the 31st. If you want the time-of-day interrupt OB to start at the end of the month, you should set the parameter "Execution" to "End of month".

"Local time" or "System time"

Here, you decide which time the start time of the time-of-day interrupt OB relates to:

- "Local time": The start time relates to the time zone set for the CPU.
- "System time": The start time relates to the coordinated universal time UTC (Universal Time Coordinated).

Assigning parameters to cyclic interrupt OBs

Introduction

You can use cyclic interrupt OBs to start programs at regular time intervals. To do so you must enter a scan time and a phase shift for each cyclic interrupt OB used.

You can use up to four cyclic interrupt OBs or time-delay OBs (OB numbers >= 200) in your program. If, for example, you are already using two time-delay interrupt OBs, you can insert a maximum of two further cyclic interrupt OBs in your program.

Note

If you assign multiple cyclic OBs, make sure that you assign a different cycle time or phase offset to each cyclic interrupt OB to avoid them executing at the same time or having to queue. When you create a cyclic interrupt OB, the cycle time 100 and the phase offset 0 are entered as start value.

Procedure

To enter a scan time and a phase shift for a cyclic interrupt OB, proceed as follows:

1. Open the "Program blocks" folder in the project tree.
2. Right-click on an existing cyclic interrupt OB.
3. Select the "Properties" command in the shortcut menu.
   The "<Name of the cyclic interrupt OB>" dialog box opens.
4. Click the "Cyclic interrupt" group in the area navigation.
   The text boxes for the scan time and the phase shift are displayed.
5. Enter the scan time and the phase shift.
6. Confirm your entries with "OK".

See also

Basics of block parameters (Page 1227)

Organization blocks for cyclic interrupts (Page 1220)

Assigning parameters to hardware interrupt OBs

Introduction

You must select the corresponding event and assign the following parameters for every input channel and high-speed counter that should trigger a hardware interrupt:

- Event name
- Number of the hardware interrupt OB that is assigned to this process event

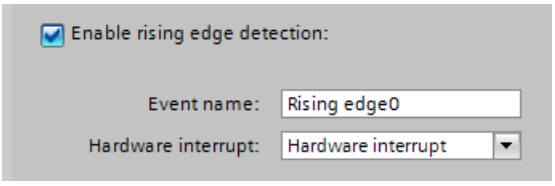
The parameters of the hardware interrupt are assigned in the properties of the corresponding device. You can assign parameters for up to 50 hardware interrupt OBs.

You can create the hardware interrupt OB to be assigned parameters either before or during activation of an event.

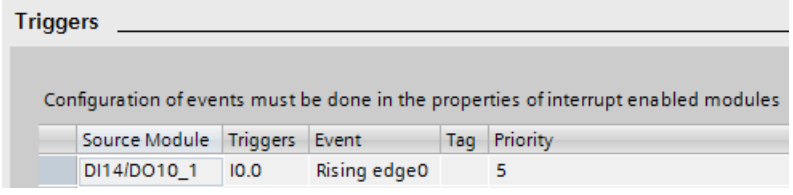
Procedure

To configure a hardware interrupt event, follow these steps:

1. Double-click the "Devices & Networks" command in the project tree.
   The hardware and network editor opens in the network view.
2. Change to the device view.
3. If the Inspector window closed in the device view, select the "Inspector window" check box in the "View" menu.
   The Inspector window opens.
4. Click the "Properties" tab.
5. In the device view, select the module for which you want to a assign a hardware interrupt.
6. Select the corresponding event that will trigger a hardware interrupt, e.g., a positive edge.



7. Enter an event name.
8. Select an existing hardware interrupt OB from the "Hardware interrupt" drop-down list or create a new hardware interrupt OB. If you have not previously created any hardware interrupt OBs, you can click the "Add new block" button in the drop-down list.
   The start information of the corresponding hardware interrupt OB, including all specifications for the interrupt-triggering event, is updated.



9. If you want to assign further hardware interrupts, repeat steps 5 to 8.

A system constant of data type Event_HwInt is created automatically for the event identified by the explicit event name. The system constants are displayed in the standard tag table.

See also

Basics of block parameters (Page 1227)

Organization blocks for hardware interrupts (Page 1221)

Events and OBs (Page 1177)

Symbolic and numerical names of instructions

Description

The instructions from the task card are comprised of functions (FC), function blocks (FB), system functions (SFC) and system function blocks (SFB) that are identified internally by numbers.

The following tables show the assignment of numerical and symbolic names.

Function blocks (FBs)

Numerical name	Symbolic name
FB 105	TC_CONFIG
FB 110	Port_Config
FB 111	Send_Config
FB 112	Receive_Config
FB 113	Send_P2P
FB 114	Receive_P2P
FB 115	Receive_Reset
FB 116	Signal_Get
FB 117	Get_Features
FB 118	Set_Features
FB 163	TC_SEND
FB 164	TC_RECV
FB 165	TC_CON
FB 166	TC_DISCON
FB 804	SET_TIMEZONE
FB 1030	TSEND_C
FB 1031	TRCV_S
FB 1071	USS_DRIVE
FB 1080	MB_COMM_LOAD
FB 1081	MB_MASTER
FB 1082	MB_SLAVE

Numerical name	Symbolic name
FB 1084	MB_CLIENT
FB 1085	MB_SERVER
FB 1100	MB_Halt
FB 1101	MC_Home
FB 1102	MC_MoveAbsolute
FB 1103	MC_MoveJog
FB 1104	MC_MoveRelative
FB 1105	MC_MoveVelocity
FB 1107	MC_Power
FB 1108	MC_Reset
FB 1110	MC_MoveInterrupt
FB 1111	MC_ChangeDynamik
FB 1112	MC_CommandTable
FB 1113	MC_MoveLinearAbs_2D
FB 1114	MC_MoveLinearRel_2D
FB 1115	MC_MoveCircular_2D
FB 1130	PID_Compact
FB 1134	PID_3Step
FB 1140	HSC
FB 2040	RecipeCreate
FB 2041	RecipeOpen
FB 2042	RecipeRead
FB 2043	RecipeWrite
FB 2044	RecipeAppend
FB 2045	RecipeClose

Functions (FCs)

Numerical name	Symbolic name
FC 2 (1)	CONCAT
FC 4 (1)	DELETE
FC 11 (1)	FIND
FC 17 (1)	INSERT
FC 20 (1)	LEFT
FC 21 (1)	LEN
FC 22 (1)	LIMIT
FC 25 (1)	MAX
FC 26 (1)	MID
FC 27 (1)	MIN
FC 31 (1)	REPLACE
FC 32 (1)	RIGHT
FC 36 (1)	ENCO

Numerical name	Symbolic name
FC 36 (1)	SEL
FC 37	DECO
FC 800	LED
FC 801	IM_DATA
FC 802	DeviceStates
FC 803	ModuleStates
FC 1070	USS_PORT
FC 1072	USS_RPM
FC 1073	USS_WPM
(1) MC7+ instruction	

System data types (SDTs)

Numerical name	Symbolic name
SDT 99	WWW_CDB
SDT 513	CONDITIONS
SDT 581	Send_Conditions
SDT 582	Receive_Conditions

System function blocks (SFBs)

Numerical name	Symbolic name
SFB 0 (1)	CTU
SFB 1 (1)	CTD
SFB 2 (1)	CTUD
SFB 3 (1)	TP
SFB 4 (1)	TON
SFB 5 (1)	TOF
SFB 27	START_OB
SFB 52	RDREC
SFB 53	WRREC
SFB 54	RALRM
SFB 105	T_CONFIG
SFB 106	TDIAG
SFB 107	TRESET
SFB 110	PORT_CFG
SFB 111	SEND_CFG
SFB 112	RCV_CFG
SFB 113	SEND_PTP
SFB 114	RCV_PTP
SFB 115	SGN_GET
SFB 116	SGN_SET

Numerical name	Symbolic name
SFB 117	RCV_RST
SFB 120	CTRL_HSC
SFB 122	CTRL_PWM
SFB 124	CTRL_HSC_EXT
SFB 140	DataLogCreate
SFB 141	DataLogOpen
SFB 142	DateLogWrite
SFB 143	DataLogClear
SFB 144	DataLogClose
SFB 145	DataLogDelete
SFB 146	DataLogNewFile

System functions (SFCs)

Numerical name	Symbolic name
SFC 7	DP_PRAL
SFC 11	DPSYC_FR
SFC 13	DPNRM_DG
SFC 14	DPRD_DAT
SFC 16	RD_OBINF
SFC 23	DEL_DB
SFC 28	SET_TINT
SFC 29	CAN_TINT
SFC 30	ACT_TINT
SFC 31	QRY_TINT
SFC 32	SRT_DINT
SFC 33	CAN_DINT
SFC 34	QRY_DINT
SFC 41	DIS_AIRT
SFC 42	EN_AIRT
SFC 43	RE_TRIGR
SFC 45	D_ACT_DP
SFC 46	STP
SFC 82	CREA_DBL
SFC 83	READ_DBL
SFC 84	WRIT_DBL
SFC 86	CREATE_DB
SFC 89	RST_EVOV
SFC 99	WWW
SFC 101	RTM
SFC 117	GET_DIAG
SFC 124	ATTR_DB

Numerical name	Symbolic name
SFC 140	IO2MOD
SFC 143	RD_ADDR
SFC 154	RD_LOC_T
SFC 154	DPWR_DAT
SFC 161	WR_LOC_T
SFC 180	ID2LOG
SFC 181	LOG2ID
SFC 182	ID2GEO
SFC 190	SET_CINT
SFC 191	QRY_CINT
SFC 192	ATTACH
SFC 193	DETACH
MC7+ Anweisung	GET_ERROR
MC7+ Anweisung	GET_ERR_ID

Useful information on CPU firmware versions and STEP 7 versions

CPUs and engineering software for configuring the CPUs is constantly developed further to improve performance and security. New versions that have some special features with respect to the interaction of the components are created in this way. The sections below describe the special features of the S7-1200 CPUs with firmware version V4 as compared to firmware versions V1 to V3.

For a detailed comparison of the range of functions, read the S7-1200 System Manual (description of new instructions, new organization blocks and advanced configuration options).

Required engineering software

S7-1200 CPUs V4 can be configured with STEP 7 as of V13.

Compatibility between memory card content and firmware version of the CPU

Memory cards (transfer cards or program cards) with configuration and program for an S7-1200 CPU V1, V2 or V3 do not work with an S7-1200 CPU V4.
Memory cards with configuration and program for an S7-1200 CPU V4 do not work with an S7-1200 CPU V1, V2 or V3.

You must change an S7-1200 CPU V1 to V3 configuration to an S7-1200 CPU V4 configuration (device replacement) and then download it to the CPU. A gradual device replacement is required for an S7-1200 CPU V1-V2 (see below).

When you insert the memory card into a CPU with an incompatible firmware version, the CPU does not start up. When you insert a memory card for a CPU V1, V2 or V3 into an S7-1200 CPU V4, this CPU outputs a version error.

Going online and loading

When you have configured an S7-1200 CPU with firmware version V1, V2 or V3 with STEP 7, the CPU to which you want to go online or which you want to download must have one of these firmware versions. You cannot go online to an S7-1200 CPU V4 with a configured S7-1200 CPU V1, V2 or V3.

On the other hand, you cannot go online to an S7-1200 CPU V1, V2 or V3 with a configured S7-1200 CPU V4 or download this CPU.

Replacing an existing CPU

You can replace a configured S7-1200-CPU V1, V2 or V3 with a new CPU with a firmware version greater than or equal to 4. In case of an existing S7-1200 CPU V1 or V2, you must start by replacing the device with an S7-1200 CPU V3: It is not possible to directly replace it with an S7-1200 CPU V4.

1. S7-1200 CPU V1 (V2) > S7-1200 CPU V3
2. S7-1200 CPU V3 > S7-1200 CPU V4

As long as you do not download the configuration, you can undo the device replacement ("Undo" command in the "Edit" menu). When you download the configuration for the new CPU firmware version ("V4 configuration") to the CPU, you can no longer return to version V3.

This means you should save the existing project, for example, with a V3 configuration as project archive so that you can access it later.

Special notes for device replacement (V3 > V4):

- The interrupt behavior of interrupt OBs remains the same; they are not configured as interruptible. This is also the default behavior of S7-1200 CPUs V1-V3. The interrupt behavior of the interrupt OB can be configured for S7-1200 V4 CPUs. If you drag an S7-1200 CPU V4 directly from the hardware catalog into the network view, this option is activated (interrupt OBs are interruptible).
- The behavior for PUT/GET access of remote partners remains the same; access is permitted. This is also the default behavior of S7-1200 CPUs V1-V3. The access via PUT/GET communication by means of remote partners can be configured for S7-1200 V4 CPUs ("Protection" area of CPU parameters). If you drag an S7-1200 CPU V4 directly from the hardware catalog into the network view, access is not permitted and must be explicitly enabled.
- The wording for the protection levels changes but the effect of the settings remains the same. You can also select the access level "No access (complete protection)".
- The web server settings for activation of the web server and HTTP/HTTPS settings are applied. You also have the option to create users and assign them specific rights (Web server area > User management of CPU parameters). Web server users only have access to standard websites when you do not configure users. An S7-1200 CPU V4 no longer supports the user "admin" and his/her password.

Communication with HMI devices

When you connect an HMI device to an S7-1200 CPU V4, make sure that you use the appropriate Runtime software version of the HMI device.

You may have to transfer the latest HMI Runtime version by means of the WinCC engineering software.

You must compile the HMI configuration once again and download it to the HMI device for the CPU-HMI communication to work.

See also

Useful information on memory cards (Page 1169)

10.1.6.2 Identification systems

RFID systems

Ident profile and Ident blocks

You will find detailed information on the Ident profile in the manual "Ident Profile and Ident Blocks, Standard Function for Ident Systems" on the pages of "Product Support (http://support.automation.siemens.com/WW/view/en/10805817)".

Communications module RF120C

Reader parameter group

The "Reader" parameter group contains the following:

- Diagnostics: Setting to decide whether or not hardware diagnostics messages are output.
- Reader System: Selection of the connected RFID system. Depending on the selection you make, the "Reader System" parameter group is adapted.

"Diagnostics" parameter

Parameter assignment options:

- None
  Apart from standard diagnostics messages, no other diagnostics messages are generated.
- Hard Errors
  Extended diagnostics messages are generated if the following events occur.
  - Hardware error (memory test)
  - Firmware error (checksum)
  - Connection to reader lost
  - Short-circuit fault/interruption if supported by the hardware

Further information

For further information on diagnostics, refer the documentation for the RF120C communications module with application blocks for S7-1200 and S7-1500.

Reader System parameter group

The parameters for the selected identification system are set in the "Reader system" parameter group.

The table shows the parameters that exist for all identification systems (RFID and code reader systems).

Table 10-76 Standard parameters for all identification systems

Parameter	Parameter value	Default value	Description
Baud rate reader	19.2 kBd 57.6 kBd 115.2 kBd	115.2 kBd	After changing the baud rate, the reader must be turned off and on again.
Presence Check	On Off (RF field off) Off (RF field on)	On	On = presence is reported as soon as there is a transponder in the antenna field of the reader Off (RF field on) = the presence check in the FB is suppressed. The antenna on the reader is nevertheless turned on as long as it has not been turned off by a command. Off (RF field off) = the antenna is turned on only when a command is sent and it then turns itself off again (RF300 only)
Reset ERR-LED	On Off	Off	On = the flashing of the error LED on the RF120C is reset by each FB reset. Off = the error LED always indicates the last error. The display can only be reset by turning off the RF120C.

The following parameters are system specific according to the selection you made in the "Reader" parameter group.

RF200

Table 10-77 RF200 general

Parameter	Parameter value	Default value	Description
-	-	-	Only the standard parameters are available.

Table 10-78 RF290R

Parameter	Parameter value	Default value	Description
RF power	0.50 - 5.00 W	1.00 W	Setting for the output power of the reader.

RF300

Table 10-79 RF300 general

Parameter	Parameter value	Default value	Description
Transponder type	RF300 ISO 15693	RF300	Selection of the transponders used.

Table 10-80 RF380R

Parameter	Parameter value	Default value	Description
RF power	0.50 - 2.00 W	1.25 W	Setting for the output power of the reader.
Transponder type	RF300 ISO 15693	RF300	Selection of the transponders used.

RF600

Table 10-81 Reader System: RF600

Parameter	Parameter value	Default value	Description
Max. no. of transponders	1 - 80	1	Number of transponders expected in the antenna field. Permitted values: • 1 - 40 for RF620R • 1 - 80 for RF630R with 2 antennas • 1 - 40 for RF630R with 1 antenna The value stored in "multitag" defines the expected number of transponders to be read (EPC-ID) in the inventory. The value does not restrict the number of transponders to be processed in the antenna field. To ensure an efficient inventory of transponders in the antenna field, make sure that the values specified here do not deviate by more than approximately 10% from the maximum number of transponders to be expected in the antenna field.
Reader type	RF620R ETSI RF620R FCC RF620R CMIIT RF630R ETSI RF630R FCC RF630R CMIIT	RF620R ETSI	Selection of the reader used. By selecting a reader, you open the "Reader type" parameter group. The parameters are described in the following table.

Reader type (RF600 only)

In the "Reader type" parameter group further specific parameters are set for the reader type selected in the "Reader system" parameter group.

Table 10-82 RF600 menu: Reader type

Parameter	Parameter value	Default value	Description
Wireless profile	-	-	Selection of the relevant wireless profile for ETSI, FCC or CMIIT.
Multitag mode	UID = EPC-ID (8 bytes) UID = Handle ID (4 bytes)	UID = EPC-ID (8 bytes)	• UID = EPC-ID (8 bytes) 8 byte UID of the bytes 5-12 of the 12 byte long EPC-ID • UID = handle ID (4 bytes) 4 byte UID as handle ID for access to transponders with an EPC-ID of any length
Intelligent Single Tag Mode (ISTM)	On Off	Off	Enable/disable the "Intelligent Single Tag Mode ISTM" algorithm [1]
Black List	On Off	Off	Enable/disable the "Black list" [1]
Radiated power internal antenna (RF620R)	0 - B	4	Setting the radiated power for the internal antenna [1] [2]
Internal antenna (RF620R)	-	-	Enable/disable the internal antenna. With the RF620R, either only the internal or only the external antenna can be set.
Transmit power (external antenna RF620R)	0 - 9	4	Setting the transmit power for the external antenna [1] [2]
External antenna (SetAnt block required) (RF620R)	-	-	Enable/disable the external antenna. With the RF620R, either only the internal or only the external antenna can be set.
Transmit power ANT 1 (RF630R)	0 - 9	4	Setting the transmit power for antenna 1 [1] [2]
Transmit power ANT 2 (RF630R)	0 - 9	4	Setting the transmit power for the antenna 2 [1] [2]
Communications speed	Reliable detection Fast detection	Reliable detection	[1]
Tag hold	On Off	Off	Enable/disable "Tag hold" [1]
Scanning mode	On Off	Off	Enable/disable the "Scanning mode" [1]
Channel assignment (only with wireless profile ETSI)	-	-	Selection of the wireless channels to be used [1]

[1] You will find further information in the "Configuration manual RF620R/RF630R". *****

[2] The values for the transmit/radiated power of the antennas can be found in the following table.

***** You will find the configuration manual RF620R/RF630R here (http://support.automation.siemens.com/WW/view/en/10805817).

Table 10-83 Transmit / radiated power of the antennas

Hex value	RF630R transmit power	RF620R radiated power (internal antenna)			RF620R transmit power
	dBm / (mW)	ETSI dBm / (mW) ERP	FCC dBm / (mW) EIRP	CMIIT dBm / (mW) ERP	dBm / (mW)
0	18 / (63)	18 / (63)	20 / (100)	18 / (65)	18 / (63)
1	19 / (79)	19 / (79)	21 / (126)	19 / (79)	19 / (79)
...	...	...	...	...	...
4	22 / (158)	22 / (158)	24 / (251)	22 / (158)	22 / (158)
...	...	...	...	...	...
9	27 / (501)	27 / (501)	29 / (794)	27 / (501)	27 / (501)
A	27 / (501)	28 / (631)	30 / (1000)	28 / (631)	27 / (501)
B (...F)	27 / (501)	29 / (794)	31 / (1259)	29 / (794)	27 / (501)

SLG D10S

Table 10-84 SLG D10S

Parameter	Parameter value	Default value	Description
RF power	0.50 - 10.00 W	1.00 W	Setting for the output power of the reader.
Transponder type	ISO I-Code (e.g. MDS D139)	ISO	Selection of the transponders used.

SLG D11S/D12S

Table 10-85 SLG D11S/D12S

Parameter	Parameter value	Default value	Description
Transponder type	ISO I-Code (e.g. MDS D139)	ISO	Selection of the transponders used.

MOBY U

Table 10-86 MOBY U

Parameter	Parameter value	Default value	Description
Standby time	0 - 1400 ms	0 ms	Standby time (scanning_time) for the transponder. If the transponder receives a further command before the standby time has elapsed, this command can be executed immediately. If the transponder receives a command after standby time has elapsed, command execution is delayed by the "sleep_time" of the transponder.
Range limitation	0.2 m 0.5 m 1.0 m 1.5 m 2.0 m 2.5 m 3.0 m 3.5 m	1.5 m	-
Max. no. of transponders	1 - 12	1	Maximum number of transponders that can be processed at the same time in the antenna field.
BERO mode	Without BEROs 1 or 2 BEROs 1st BERO on, 2nd BERO off Synchronization by cable connection	Without BEROs	• Without BEROs No reader synchronization • 1 or 2 BEROs The BEROs are logically ORed. The antenna field is turned on during the actuation of a BERO. • 1st BERO on, 2nd BERO off 1st BERO turns on the antenna field and the 2nd BERO turns the antenna field off. If there are two BEROs present and "BERO time in s" is set, the antenna field is turned off automatically if the 2nd BERO does not switch within this BERO time. If no "BERO time in s" is set, the antenna field remains turned on until the 2nd BERO is activated. • Synchronization by cable connection Enable reader synchronization via cable connection (see manual on configuration, installation and service for MOBY U).
BERO time in s	0 - 255 s	0 s	Can only be set if the BERO mode is set to "1st BERO on, 2nd BERO off". • 0 The time monitoring is disabled. To turn off the field, the 2nd BERO is required. • 1 ... 255 s Operating time for the reader field

General Reader

Table 10-87 General Reader

Parameter	Parameter value	Default value	Description
Input box for byte sequence (hexadecimal)	00000000	00000000	Expert mode With this function, you can specify the reset parameters directly in hexadecimal notation. This setting may only be selected if you have previously received the hexadecimal string for the setting from a member of the SIEMENS staff.

Parameters via FB / code readers

Table 10-88 Parameters via FB / code readers

Parameter	Parameter value	Default value	Description
MOBY mode	RF200, RF300, RF600, MOBY D/U, MV	RF200, RF300, RF600, MOBY D/U, MV	Only the mode "RF200, RF300, RF600, MOBY D/U, MV" is supported. For further information on resetting a function block, refer to the documentation for the RF120C communications module with application blocks for S7-1200 and S7-1500. *

* You will find information on the documentation here (http://support.automation.siemens.com/WW/view/en/10805817).

ASM 475

What you should know about the ASM 475

Introduction

Up to eight ASM 475 interface modules can be plugged in and operated in a rack of the SIMATIC S7-300. If you have a configuration with several racks (max. four racks), the ASM 475 can be plugged in and operated in each rack. This means that with the maximum configuration of a SIMATIC S7-300, 32 ASM 475 modules can be operated.

A maximum of two SLGs (write/read devices) can be connected to the ASM 475. The processing of the connected SLGs is parallel. FC45 allows simple programming using the SIMATIC S7 tools.

FC45 can be used both in the S7-300 and S7-400.

With S7-400, the ASM 475 is connected via an ET 200M.

Note that the

- IM 153-1 must have at least order number 6ES7 153-1AA03-0XB0 or 6ES7 153-1AA83-0XB0 and
- IM 153-2 at least order number 6ES7 153-2AA02-0XB0 or 6ES7 153-2AB01-0XB0.

Access to the MDS data is with normal addressing.

Physical addressing of the MDS (mobile data storage)

The physical addressing of an MDS is also known as **normal addressing**. Users set up the structure of the MDS themselves. They know the physical MDS address to which data is written. The addressing of the MDS storage generally begins at 0000 hex and ends at an end address corresponding to the size of the MDS storage.

File handler addressing of the MDS

For file handler addressing, the user specifies a file name for access to the data. The file name consists of eight ASCII characters. The file handler manages the user data independently on the MDS. The user does not need to configure data structures on the MDS.

Prior to use, the MDS needs to be formatted.

10.1.6.3 Distributed I/O

Distributed I/O systems

SIMATIC ET 200 - The right solution for all applications

SIMATIC ET 200 provides the most varied range of distributed I/O systems.

- Solutions for use in the control cabinet
- Solutions without control cabinet directly at the machine

Additionally, there are also components that can be used in explosive areas. SIMATIC ET 200 systems for construction without a control cabinet are contained in robust, glass-fibre reinforced plastic casing and are therefore shock-resistant, resistant to dirt and watertight.

Their modular design allows the ET 200 systems to be easily scaled and expanded in small steps. Fully-integrated auxiliary modules lower costs and also provide a wide range of possible applications. There are several combination possibilities available:

- Digital and analog I/OS
- Intelligent modules with CPU functions,
- Safety technology,
- Pneumatics,
- Frequency converters
- Various technology modules.

Communication via PROFIBUS and PROFINET, uniform engineering, clear diagnostic possibilities as well as optimal connection to SIMATIC controller and HMI devices vouch for the unique consistency provided by Totally Integrated Automation.

The following table provides an overview of I/O devices for use in the control cabinet:

I/O device	Properties
ET 200S	• Highly modular design with multiple conductor connections • Multifunctional due to a wide range of modules • Use in explosive areas (Zone 2)
ET 200S COMPACT	• Highly modular design with multiple conductor connections • Multifunctional due to a wide range of modules • Use in explosive areas (Zone 2) • Integrated DE/DA
ET 200L	• Cost-effective digital block I/OS • Digital electronic blocks up to 32 channels
ET 200M	• Modular design with standard modules from SIMATIC-S7-300 • Failsafe I/O modules • Use in explosive areas up to Zone 2, sensors and actuators up to Zone 1 • High level of plant availability, for example by plugging and unplugging when in operation
ET 200iSP	• Modular design, also possible with redundancy • Robust and intrinsically safe design • Use in explosive areas up to Zone 1/21; sensors and actuators even up to Zone 0/20 • High level of plant availability, for example by plugging and unplugging when in operation

The following table provides an overview of I/O devices for use without a control cabinet:

I/O device	Properties
ET 200pro	• Modular design with compact housing • Easy assembly • Multifunctional due to a wide range of modules • High level of availability due to plugging and unplugging in operation and permanent wiring • Comprehensive diagnostics
ET 200eco PN	• Cost-efficient, space-saving block I/OS • Digital modules up to 16 channels (also configurable) • Analog modules, IO-link master and load voltage distributor • PROFINET connection with 2-port switch in each module • Can be flexibly distributed via PROFINET in line or star shape directly within the plant
ET 200eco	• Cost-effective digital block I/OS • Flexible connection possibilities • Failsafe modules • High level of plant availability
ET 200R	• Specially for use on robots • Assembled directly on the chassis • Resistant to weld spatter due to robust metal housing

See also

Documentation on ET 200L (http://support.automation.siemens.com/WW/view/de/1142908/0/en)

Documentation on ET 200S (http://support.automation.siemens.com/WW/view/en/1144348)

Documentation on ET 200M (http://support.automation.siemens.com/WW/view/de/1142798/0/en)

Documentation on ET 200pro (http://support.automation.siemens.com/WW/view/de/21210852/0/en)

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Documentation on ET 200R (http://support.automation.siemens.com/WW/view/de/11966255/0/en)

Documentation on ET 200eco PN (http://support.automation.siemens.com/WW/view/de/29999018/0/en)

Documentation on ET 200eco (http://support.automation.siemens.com/WW/view/de/12403834/0/en)

Configuring HART variables

Introduction

Numerous HART field devices make available additional measured quantities (e.g. sensor temperature). These can be read if they are set accordingly in the field device configuration. Using the HART variables. it is possible to apply the set measured values directly from the field device into the I/O area of your automation system.

Regardless of the number of configured channels, a maximum of 8 HART variables can be assigned for HART modules and no more than 4 HART variables per channel. You assign the HART variables to a channel in the properties for the module ("HART variable settings" area). To do this, check the manual of the corresponding module.

Address assignment

By default, the HART modules occupy 16 input/output bytes (user data). If you configure HART variables, the modules occupy an additional 5 bytes for each HART variable.

If you use all 8 HART variables, the HART input module occupies a total of 56 input/output bytes (16 bytes + 8 x 5 bytes = 56 bytes).

The "None" configuration occupies no additional input bytes.

Configuration of HART variables

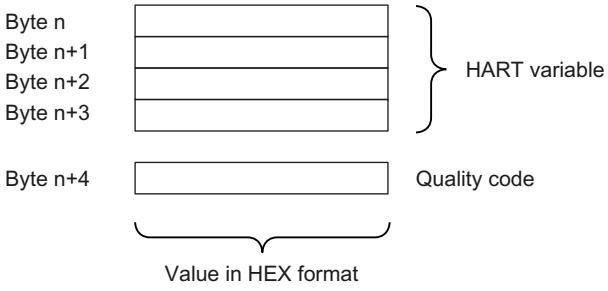
You can configure up to 4 HART variables for a channel

- PV (Primary Variable)
- SV (Secondary Variable)
- TV (Tertiary Variable)
- QV (Quaternary Variable)

CiR is a placeholder that reserves the address space for a HART variable. You must configure the HART variables you are not using with the "None" parameter.

Configuration of HART variables

Each HART variable occupies 5 bytes of input data and is structured as follows:



The quality code can accept different values depending on the module. To do this, check the manual of the corresponding module.

See also

Documentation for HART analog modules (http://support.automation.siemens.com/WW/view/en/22063748)

ET 200iSP

ET 200iSP Distributed I/O Station

Definition

The ET 200iSP distributed I/O station is a highly modular and intrinsically safe DP slave with degree of protection IP 30.

Area of application

The ET 200iSP distributed I/O station can be operated in potentially explosive atmospheres characterized by gas and dust:

Approval	ET 200iSP Station*	Inputs and outputs
ATEX	Zone 1, Zone 21	up to Zone 0, Zone 20 **
IECEx	Zone 2, Zone 22	up to Zone 0, Zone 20 **
* In combination with an appropriate enclosure ** for electronic module 2 DO Relay UC60V/2A: up to Zone 1, Zone 21		

The ET 200iSP distributed I/O station can, of course, also be used in the safety area.

You can insert almost any combination of ET 200iSP I/O modules directly next to the interface module that transfers the data to the DP master. This means you can adapt the configuration to suit your on-site requirements.

Every ET 200iSP consists of a power supply module, an interface module and a maximum of 32 electronic modules (for example digital electronics modules). Remember not to exceed the maximum current consumption.

Terminal modules and electronic modules

In principle, the ET 200iSP distributed I/O station consists of various passive terminal modules onto which you plug the power supply and the electronic modules.

The ET 200iSP is connected to PROFIBUS RS 485-IS by means of a connector on terminal module TM-IM/EM. Every ET 200iSP is a DP slave on the PROFIBUS RS 485-IS.

DP master

All ET 200iSP modules support communication with DP masters that are compliant with *IEC 61784-1:2002 Ed1 CP 3/1* and operate with "DP" transmission protocol (DP stands for distributed peripherals or distributed I/O).

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Assigning the channel and IEEE tag

Properties

Analog electronic modules 4 AI I 2WIRE/HART, 4 AI I 4WIRE/ HART and 4 AO I HART support up to four IEEE tags.

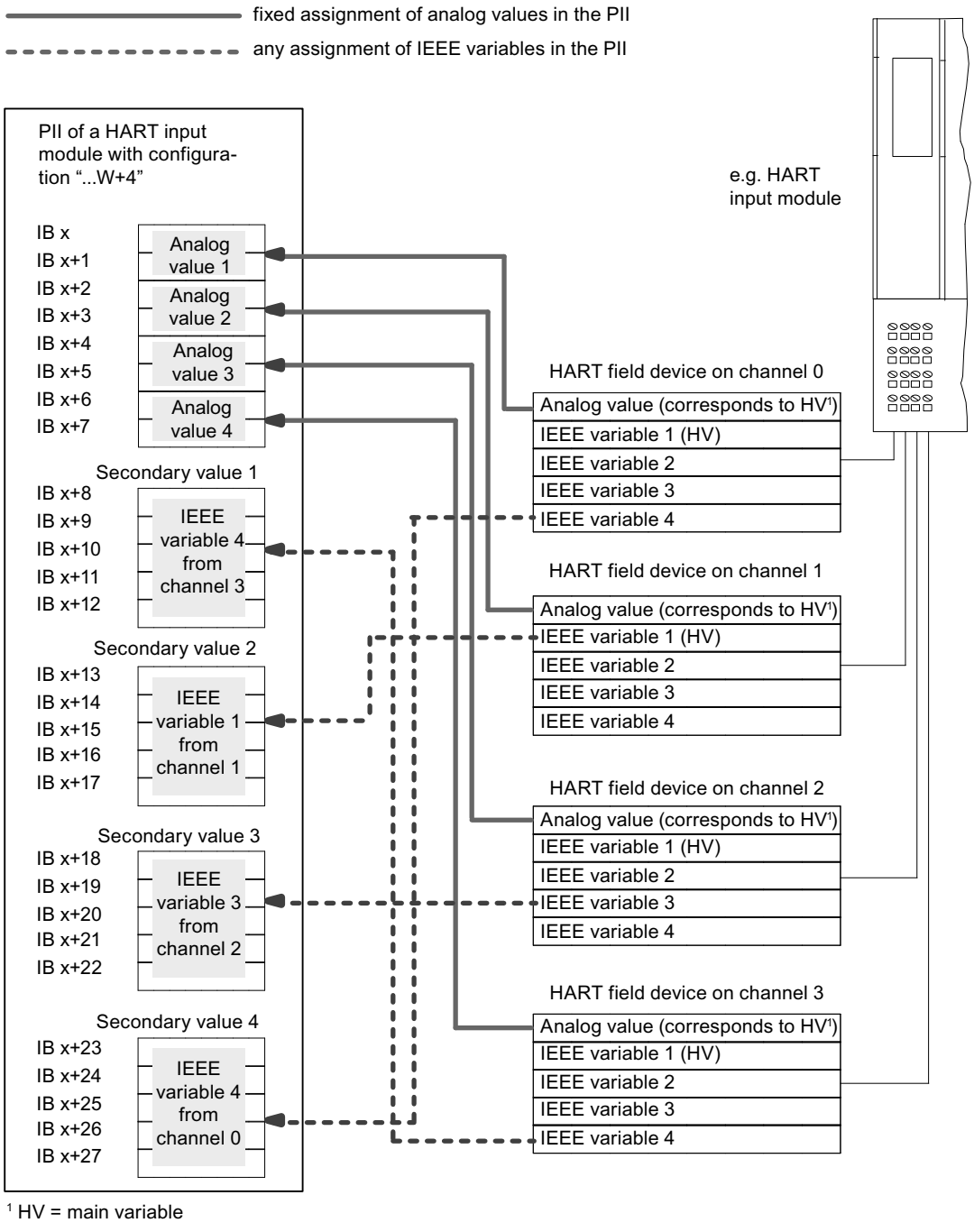
The process input image (PII) provides up to 20 bytes per module for the IEEE tags. Thus, four blocks of 5 bytes each are available for the four IEEE tags within the PII.

Requirements

The HART field device must support the assigned number of IEEE tags.

Assigning IEEE tags

You assign the IEEE tags of the field devices to any one of the four blocks in the PII.



[1] HV = main variable

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Assigning parameters to reference junctions for thermocouples

Compensation of the reference junction temperature

There are various ways of obtaining the reference junction temperature in order to get an absolute temperature value from the temperature difference between the reference junction and the measuring point.

Table 10-89 Compensation of the reference junction temperature

Option	Explanation	Reference junction parameters
No compensation	You record not only the temperature of the measurement point. The temperature of the reference junction (transition from Cu line to thermocouple line) also affects the thermo-electromotive force. The measured value then includes an error.	None
Use of a Pt100 Climatic Range resistance thermometer to record the reference junction temperature (best method)	You can record the reference junction temperature using a resistance thermometer (Pt100 Climatic Range). If parameterized accordingly, this temperature value is distributed to the 4 AI TC modules in the ET 200iSP where it is offset against the temperature value obtained at the measuring location. Number of reference junctions: 2	The parameter assignment of the IM 152 and the 4 AI TC must be coordinated: • 4 AI RTD assigned parameters for Pt100 climatic range in correct slot; • 4 AI TC: Reference junction : "yes"; select reference junction number "1" or "2" • IM 152-1:Assignment of the reference junction to a slot with 4 AI RTD; channel selection;
Internal compensation 4 AI TC	The TC sensor module (temperature sensor) is mounted onto the terminals of terminal module EM 4 AI TC. The temperature sensor reports the temperature of the terminals to the 4 AI TC. This value is then calculated together with the measured value from the channel of the electronic module.	• 4 AI TC: Reference junction number "internal"

Compensation by means of a resistance thermometer at the 4 AI RTD

If thermocouples that are connected to the inputs of the 4 AI RTD have the same reference junction, compensate by means of a 4 AI RTD.

For both channels of the 4 AI TC module, you can select "1", "2" or "internal" as the reference junction number. If you select "1" or "2", the same reference junction (RTD channel) is always used for all four channels.

Setting parameters for the reference junction

You set the reference junctions for the 4 AI TC electronic modules by means of the following parameters:

Table 10-90 Reference junction parameters

Parameter	Module	Range of values	Explanation
Slot reference junction 1 to slot 2	IM 152	none, 4 to 35	With this parameter, you can assign up to 2 slots (none, 4 to 35), on which the channels for reference temperature measurement (calculating the compensation value) are located.
Input reference junction 1 to 4 input reference junction	IM 152	RTD on channel 0 RTD on channel 1 RTD on channel 2 RTD on channel 3	This parameter allows you to set the channel (0/1/2/3) for measuring the reference temperature (calculation of the compensation value) for the assigned slot.
Reference junction E0 to reference junction E3	4 AI TC	None yes	This parameter allows you to enable the use of the reference junction.
Reference junction number	4 AI TC	1 2 Internal	This parameter allows you to assign the reference junction (1, 2) that contains the reference temperature (compensation value).

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Fundamentals of Time Stamping

Properties

Time stamping is possible with the IM 152 in customer applications using FB 62 (FB TIMESTMP).

Principle of operation

A modified input signal is assigned a time stamp and stored in a buffer (data record). If time stamped signals exists or a data record is full, a hardware interrupt is generated to the DP master. The buffer is evaluated with "Read data record". Special messages are generated for events that influence the time stamping (communication with the DP master interrupted, frame failure of time master, ...).

Parameter Assignment

With the parameter assignment you define which IM 152 user data will be monitored. For the time stamping these are digital inputs that are monitoring for signal changes.

Parameter	Setting	Description
Time stamping	• disabled • enabled	Activate the time staming for the channels of the electronics module 8 DI NAMUR.
Edge evaluation incoming event	• rising edge • falling edge	Determine the type of signal change that will be time-stamped.

Counting

Count properties

Counting functions

The 8 DI NAMUR electronics module has configurable counting functions:

- 2 x 16-bit up counters (standard counting function) or
- 2 x 16-bit down counters (standard counting function) or
- 1 x 32-bit down counter (cascading counter function)
- Setting a setpoint with the PIQ
- GATE function
- You can configure the control signals of the counters:
  - Configuration channel 0..1: "Counter", channel 2..7: "DI": Two counters are configured. The control signals of the counters are stored in the PIQ (process image output).
  - Configuration channel 0..1: "Counter", channel 2..7: "Control": Two counters are configured. The control signals of the counters are stored in the PIQ (process image output). They are also controlled by the digital inputs of the 8 DI NAMUR.

See also

Principle of operation (Page 1253)

Configuring counters (Page 1256)

Assigning parameters to counters (Page 1258)

Principle of operation

16-bit up counters (standard counting function)

The counting range is 0 to 65,535.

With each count pulse at the digital input, the count is incremented by 1. Once the count limit is reached, the counter is reset to 0 and it counts up again from this value.

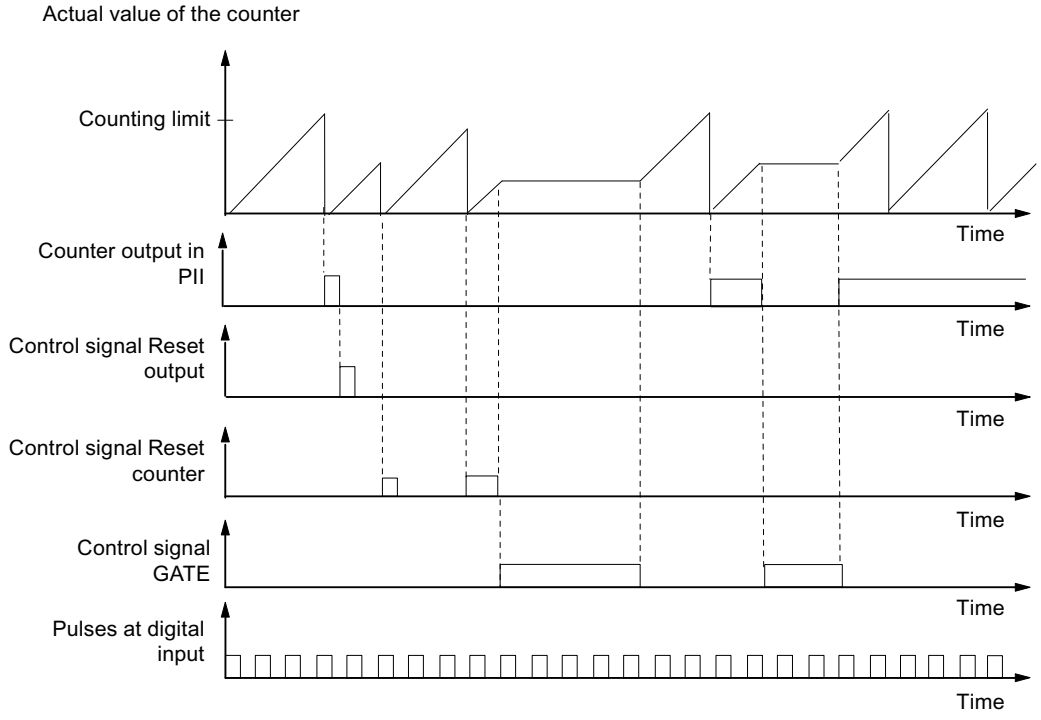
If there is counter overflow, the corresponding output is set in the PII.

A positive edge of the *Reset output* control signal resets the output in the PII. This does not affect the current count value.

In 16-bit up counting operations, the system does not set any outputs in the PIQ. These are always reset.

The positive edge of the *Reset counter* control signal sets the counter to 0 and resets the set counter output.

The *GATE* control signal pauses the counting on a positive edge. Count pulses are processed at the digital input again, but only at the negative edge. The *Reset counter* control signal is also effective when *GATE* is active.



16-bit down counters (periodic counting function)

The maximum counting range is always 65,535 to 0.

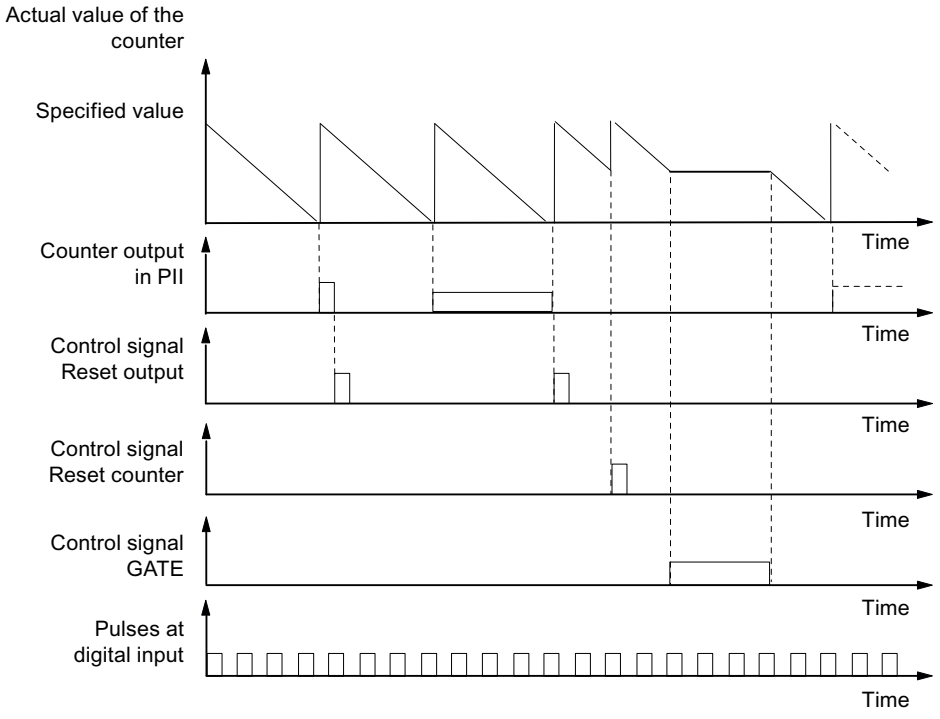
When the counter is started, the actual value is set to the selected setpoint. Each counted pulse reduced the actual value by 1. Once the actual value reaches 0, the corresponding output in the PII is turned on and the actual value is set to the selected setpoint. The counter then counts down from this value.

The positive edge of the *Reset counter* control signal resets the selected setpoint and the corresponding output in the PII.

A positive edge of the *Reset output* control signal resets the output in the PII. This does not affect the current count value.

The *GATE* control signal pauses the counting on a positive edge. At the same time, the assigned output in the PII is reset. Count pulses are processed at the digital input again, but only at the negative edge. The *Reset output* and *Reset counter* control signals are also effective when *GATE* is active.

The setpoint of the counter is set and changed using the PIQ. The setpoint is adopted on a positive edge of the *Reset counter* control signal or when the counter has reached zero.



32-bit down counter (cascading counter function)

The maximum counting range is always 4294967295 to 0.

The principle of operation is identical to that of the 16-bit down counter. Channel 1 has no function.

See also

Count properties (Page 1253)

Configuring counters

Procedure

1. Using the mouse, pull module 8 DI Namur from the hardware catalog into distributed I/O station ET 200iSP.
2. Select the required configuration (channel 0..1: "Counter", channel 2..7: "DI" or "Control"). In the module properties (inspector window), you can find this setting under "Parameters > Inputs > Configuration".

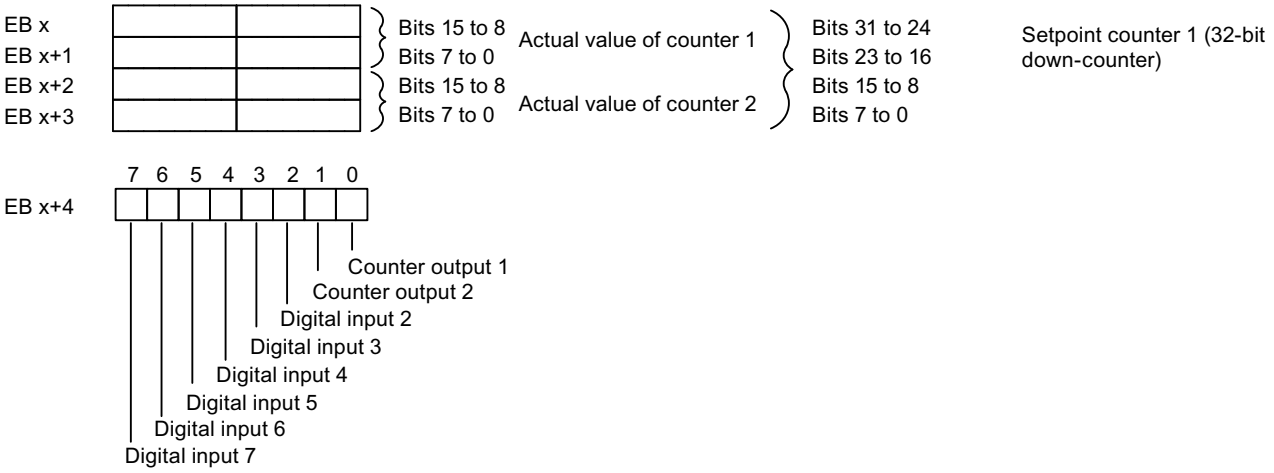
Configuration channel 0..1: "Counter", channel 2..7: "DI"

- Assignment of the digital inputs on the electronic module 8 DI NAMUR

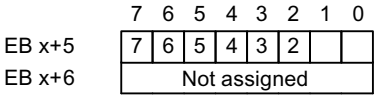
Table 10-91 Assignment of digital inputs for channel 0..1: "Counter", channel 2..7: "DI":

Digital input	Terminal	Assignment
Channel 0	1, 2	Counter 1
Channel 1	5, 6	Counter 2 (does not apply to 32-bit down counters)
Channel 2	9, 10	Digital input 2
Channel 3	13, 14	Digital input 3
Channel 4	3, 4	Digital input 4
Channel 5	7, 8	Digital input 5
Channel 6	11, 12	Digital input 6
Channel 7	15, 16	Digital input 7

- Assignment of the process image input (PII)

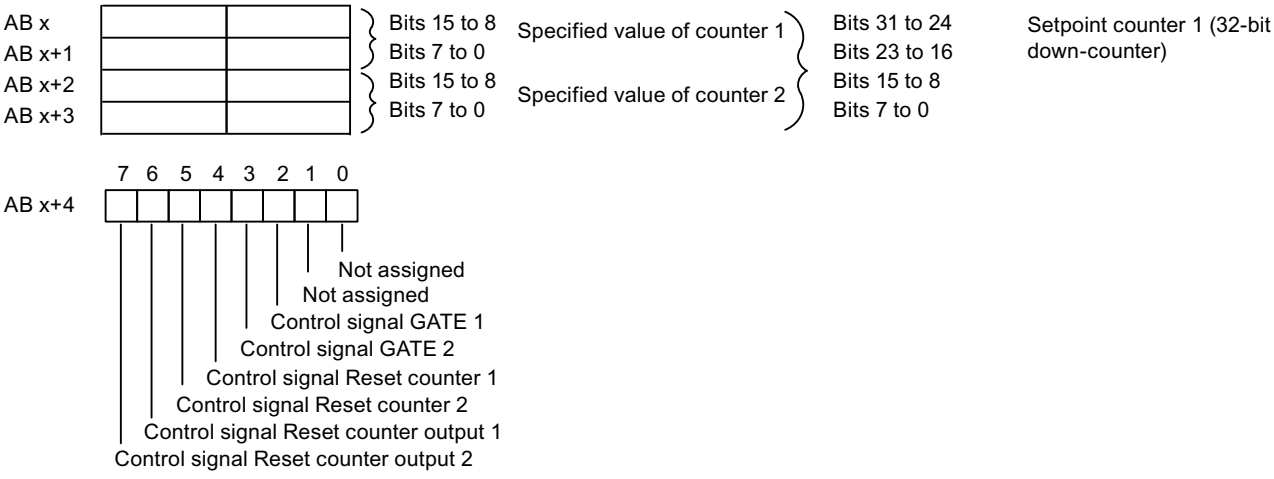


S7 format



Value status for channels 2 to 7:
1_B: Input signal is valid
0_B: Input signal is invalid

- Assignment of the process image output (PIQ)



Configuration channel 0..1: "Counter", channel 2..7: "CONTROL"

With this configuration, you can also control the counters over the digital inputs.

- Assignment of the digital inputs on electronic module 8 DI NAMUR
  For further information on input assignments, refer to the technical data for electronic module 8 DI NAMUR.

Table 10-92 Assignment of the digital inputs for 2 Count/ 6 Control

Digital input	Terminal	Assignment
Channel 0	1, 2	Counter 1
Channel 1	5, 6	Counter 2 (does not apply to 32-bit down counters)
Channel 2	9, 10	control signal *GATE 1*
Channel 3	13, 14	control signal *GATE 2*
Channel 4	3, 4	control signal *Reset counter 1*
Channel 5	7, 8	control signal *Reset counter 2*
Channel 6	11, 12	control signal *Reset counter output 1*
Channel 7	15, 16	control signal *Reset counter output 2*

- Assignment of the process image input (PII)
  Assignment is identical to configuration 0..1: "Counter", channel 2..7: "DI".
- Assignment of the process image output (PIQ)
  Assignment is identical to configuration 0..1: "Counter", channel 2..7: "DI".

See also

Count properties (Page 1253)

Assigning parameters to counters

Parameters for the counting function

Only those parameters that are relevant for the counters are explained below. These belong to the parameters of electronic module 8 DI NAMUR and depend on the selected configuration:

Table 10-93 Parameters for the counters

Parameter	Setting	Description
Sensor type counter inputs	• Channel disabled • NAMUR sensor • Single contact, no load resistance	Select the sensor for the respective counter of channels 0 or 1.
Mode for counter 1	• Standard counting function • Periodic counting function • Cascaded counting function	Select the mode for counter 1.
Mode for counter 2	• Standard counting function • Periodic counting function • Cascaded counting function	Select the mode for counter 2. This parameter is not relevant if you have set the "Mode for counter 1" parameter to "Cascaded counter function".

See also

Count properties (Page 1253)

Frequency measurement

Frequency measurement properties

Properties

The electronic module 8 DI NAMUR allows the frequencies to be measured on channel 0 and 1:

- 2 frequency meters from 1 Hz to 5 kHz
- Configurable metering window (GATE)
- The signals of the frequency meter are read in by means of the digital inputs of the electronic module.

See also

Principle of operation (Page 1259)

Configuring frequency meters (Page 1260)

Assigning parameters for the frequency meters (Page 1262)

Principle of operation

Frequency measurement

The signal frequencies are identified from the input signals of channel 0 or 1 of the electronic module. To calculate the frequency the signals are measured within a configurable gate.

The frequency is displayed as 16-bit value in fixed-point format and transferred to the PII.

The frequency meter calculates the frequency according to the follow formula:

$$\text{Frequency [Hz]} = \frac{\text{Number of rising edges at digital input}}{\text{Measuring window [s]}}$$

Exceeding the input frequency

If the input frequency exceeds 5kHz, $7FFF_H$ is reported as actual value. If the input frequency is above approx. 8 kHz it is no longer possible to display correct actual values.

See also

Frequency measurement properties (Page 1259)

Configuring frequency meters

Procedure

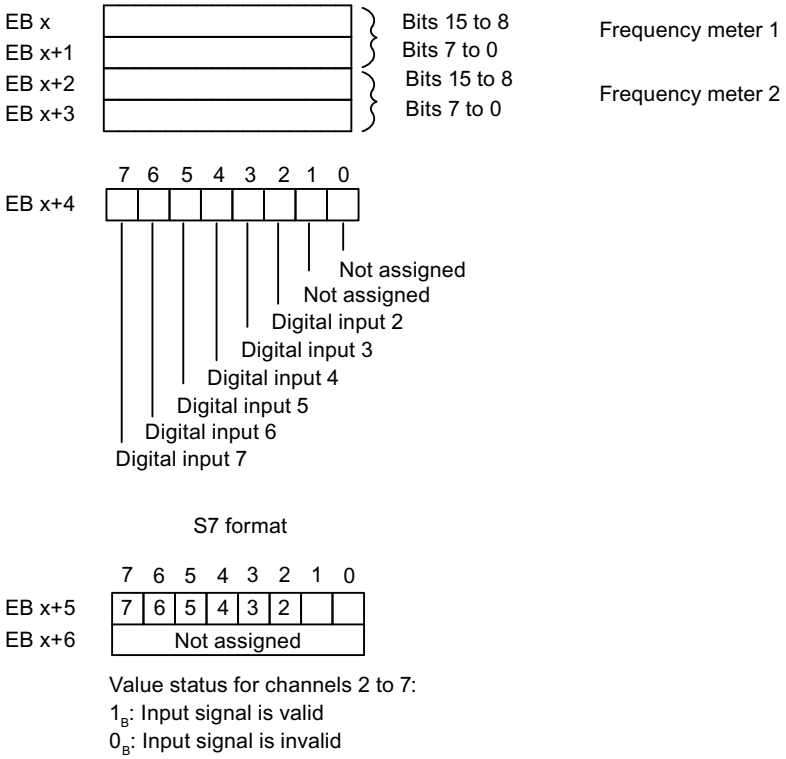
1. Using the mouse, pull module 8 DI Namur from the hardware catalog into distributed I/O station ET 200iSP.
2. Select the required configuration (channel 0..1: "Trace", channel 2..7: "DI"). In the module properties (inspector window), you can find this setting under "Parameters > Inputs > Configuration".

Configuration 0..1: "Trace", channel 2..7: "DI"

Assignment of the digital inputs on electronic module 8 DI NAMUR

Digital input	Terminal	Assignment
Channel 0	1, 2	Frequency counter 1
Channel 1	5, 6	Frequency counter 2
Channel 2	9, 10	Digital input 2
Channel 3	13, 14	Digital input 3
Channel 4	3, 4	Digital input 4
Channel 5	7, 8	Digital input 5
Channel 6	11, 12	Digital input 6
Channel 7	15, 16	Digital input 7

Assignment of process image input (PII) for configuration of channel 0..1: "Trace", channel 2..7: "DI"



Assignment of the process image output (PIQ): The PIQ is not assigned.

See also

Frequency measurement properties (Page 1259)

Assigning parameters for the frequency meters

Parameters for frequency meter

Only those parameters that are relevant for the frequency meters are explained below. These are part of the parameters of electronic module 8 DI NAMUR.

Table 10-94 Parameters for the frequency meters

Parameter	Setting	Description
Sensor type frequency inputs	• Channel disabled • NAMUR sensor • Single contact, no load resistance	Select the sensor for the relevant frequency meter for channel 0 or 1.
Measuring window (GATE)	• 50 ms • 200 ms • 1 s	Select the required measuring window for channel 0 or 1. To achieve the highest possible accuracy when metering frequencies, remember the following rules: • High frequencies (> 4 kHz): Set a low measuring window (50 ms) • Variable/medium frequencies: set medium measuring window (200 ms) • Low frequencies (< 1 kHz): Set a high measuring window (1 s)

See also

Frequency measurement properties (Page 1259)

ET 200eco PN

ET 200eco PN Distributed I/O Device

Definition

The ET 200eco PN distributed I/O device is a compact PROFINET IO device in degree of protection IP 65/66 or IP 67 and UL Enclosure Type 4x, Indoor use only.

Field of application

The fields of application of the ET 200eco PN are derived from its special properties.

- A robust design and degree of protection IP 65/66 or IP 67 make the ET 200eco PN distributed I/O device suitable in particular for use in rugged industrial environments.
- The compact design of the ET 200eco PN is particularly favorable for applications in confined areas.
- The easy handling of ET 200eco PN facilitates efficient commissioning and maintenance.

Properties

The ET 200eco PN has the following properties:

- Integrated switch with 2 ports
- Supported Ethernet services:
  - ping
  - arp
  - Network diagnostics (SNMP)
  - LLDP
- Interrupts
  - Diagnostics interrupts
  - Maintenance interrupts
- Port diagnostics
- Isochronous real-time communication
- Prioritized startup
- Device replacement without programming device
- Media redundancy
- Connection to intelligent sensors/actuators via IO link master interface module.

IO Controller

The ET 200eco PN can communicate with all IO Controllers that conform to IEC 61158.

ET 200eco PN can be configured on a CPU with advanced diagnostics.

See also

Documentation on ET 200eco PN (http://support.automation.siemens.com/WW/view/en/29999018)

Parameter description analog input

Group diagnostics

You can generally enable and disable the diagnostics function of the device with this parameter.

The "Fault" and "Parameter assignment error" diagnostics functions are always independent of the group diagnostics.

Diagnostics, missing 1L+

If you enable this parameter, the check for missing supply voltage is enabled.

Diagnostics, sensor supply short circuit

If you enable this parameter, a diagnostics event is generated if a short-circuit of the sensor supply to ground is detected and the channel is enabled. The sensor supply is monitored for connectors X1, X3, X5 and X7. It is not possible to differentiate which connector has experienced the sensor short circuit.

Interference frequency suppression

With this parameter, you set the integration time of the device, based on the selected interference frequency. Select the frequency of the supply voltage used. Interference frequency suppression **Off** means 500 Hz, which corresponds to an integration time of 2 ms for a measurement channel.

Temperature unit

Specify the unit of the temperature measurement here.

Measurement type (channel-wise)

With this parameter, you set the measurement type, for example, voltage. For any unused channels, you must select the **disabled** setting. For a disabled channel, the conversion time and integration time of the channel = 0 s and the cycle time is optimized.

Measuring range

With this parameter, you set the measuring range of the selected measurement type.

Temperature coefficient (for RTD, thermoresistor)

The correction factor for the temperature coefficients (α-value) indicates by what extent the resistance of specific material changes relatively if the temperature increases by 1 °C.

The α-values conform to EN 60751, GOST 6651, JIS C 1604 and ASTM E-1137.

The temperature coefficient depends on the chemical composition of the material.

Smoothing

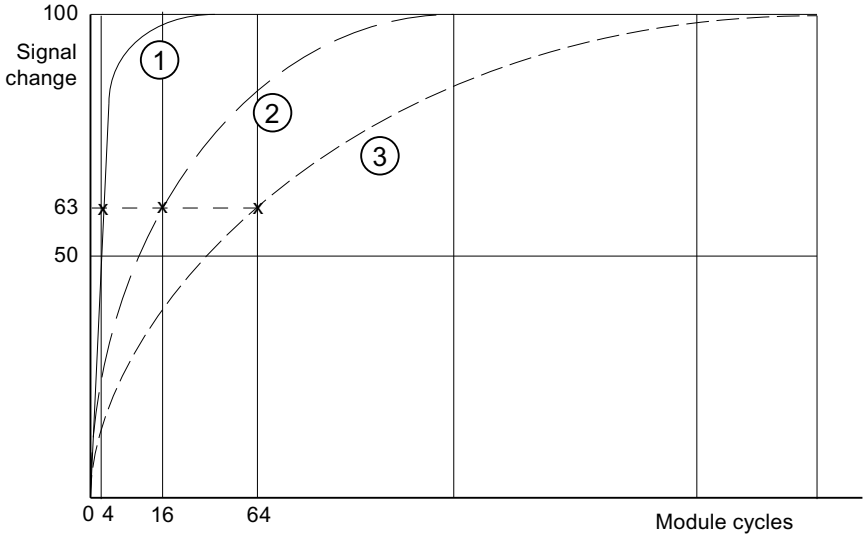
Smoothing of the analog values produces a stable analog signal for further processing. The smoothing of analog values is useful when handling wanted signals (measured values) with a slow rate of change, for example, temperature measurements.

The measured values are smoothed by digital filtering. To achieve smoothing, the device generates a mean value from a specified number of converted (digitized) analog values.

You assign a maximum of four levels for the smoothing (none, weak, medium, strong). The level determines the number of module cycles, from which the mean value is generated.

The stronger the smoothing, the more stable the smoothed analog value and the longer it takes until the smoothed analog value is applied following a signal change (see the example below).

The figure below shows the number of cycles a module requires to apply the smoothed analog value at almost 100% after a step response, based on the smoothing function settings. The figure applies to all signal changes at the analog input. The smoothing value defines the number of cycles a module requires to reach 63% of the end value of the changed signal.



① Smoothing, weak
② Smoothing, medium
③ Smoothing, strong

Diagnostics, wire break

When this parameter is enabled, the **Wire break** diagnostics event is generated when a wire break is detected.

Observe the rules outlined below to handle a wire break in the 1 to 5 V and 4 to 20 mA measuring ranges:

Parameter	Event	Measured value	Explanation
Enable wire break[1]	Wire break	$7FFF_H$	Diagnostics, **wire break**
Wire break disabled[1] Underflow enabled	Wire break	8000_H	Measured value after leaving the underrange Diagnostic message **Lower limit value undershot**
Wire break disabled[1] Underflow disabled	Wire break	8000_H	Measured value after leaving the underrange
[1] Measuring range limits for wire break detection and measuring range undershoot detection: • 1 to 5 V: At 0.296 V • 4 to 20 mA: At 1.185 mA			

Diagnostics, underflow

If you enable this parameter, the **Underflow** diagnostics event is generated when the measured value reaches the underflow range.

Diagnostics, overflow

If you enable this parameter, the **Overflow** diagnostics event is generated when the measured value reaches the overflow range.

Reference junction for thermoresistor (TC)

A difference in temperature between the measuring point and the free ends of the thermocouple (terminal point) generates a voltage between the free ends, namely the thermoelectric voltage. The value of this thermoelectric voltage is determined by the temperature difference between the measuring point and the free ends and by the type of material combination of the thermocouple. Since a thermocouple always measures a temperature difference, the free ends at the reference junction must be maintained at a known temperature in order to determine the temperature of the measuring point.

If you specify **Internal compensation**, the temperature of the measuring point in the housing of the I/O device is measured. With the **External compensation** setting, you can connect a compensation box in series in order to increase the accuracy of the temperature measurement.

Parameter description analog output

Group diagnostics

You can generally enable and disable the diagnostics function of the device with this parameter.

The "Fault" and "Parameter assignment error" diagnostics functions are always independent of the group diagnostics.

Diagnostics, missing 1L+

If you enable this parameter, the check for missing supply voltage is enabled.

Diagnostics, sensor supply short circuit

When this parameter is enabled, the system generates a diagnostics event if it detects a short-circuit of the sensor supply to ground. This diagnostics function is activated when the group diagnostics function is enabled.

Response to CPU/Master STOP

Select how the module's outputs will respond to a CPU STOP:

- Shut down
  The I/O device goes to the safe state. The process image output is deleted (=0).
- Keep last value
  The I/O device retains the last value to be output before STOP.
- Substitute value
  The I/O device outputs the value for the channel set beforehand.

Note

Make sure that the plant is always in a safe state if "Keep last value" is selected.

Type of output

With this parameter, you set the output type, for example, voltage. For any unused channels, select the **disabled** setting. For a disabled channel, the conversion time and integration time of the channel = 0 s, and the cycle time is optimized.

Output range

With this parameter, you set the output range of the selected output type.

Diagnostics, wire break (in current mode)

When this parameter is enabled, the **Wire break** diagnostics event is generated when a wire break is detected. This diagnostics event cannot be detected in the zero range.

Diagnostics, short circuit (in voltage mode)

If you enable this parameter, a diagnostics event is generated in the event of a short circuit in the output line. This diagnostics event cannot be detected in the zero range.

Diagnostics, overload

If you enable this parameter, the diagnostics event is generated in the event of an overload.

Substitute values

With this parameter, you enter a substitute value that the module is to output in CPU-STOP mode. The substitute value must be in the nominal range, overrange, or underrange.

ET 200SP

ET 200SP distributed I/O system

Definition

The ET 200SP distributed I/O system is a scalable, highly flexible distributed I/O system for connection of process signals to a central controller via a field bus.

Application area

The ET 200SP is a multi-functional distributed I/O system for various fields of application. The scalable design allows you to configure the system exactly to the specific requirements on location.

The ET 200SP is approved for degree of protection IP 20 and for installation in a control cabinet.

Structure

The ET 200SP is mounted on a mounting rail and comprises:

- An interface module which can communicate with all IO controllers that conform to the PROFINET standard IEC 61158
- Up to 32 I/O modules which can be inserted on passive BaseUnits in any combination
- A server module that completes the design of the ET 200SP.

Expanding ET 200SP with ET 200AL modules

Introduction

The ET 200SP is a distributed I/O system for installation in a control cabinet.

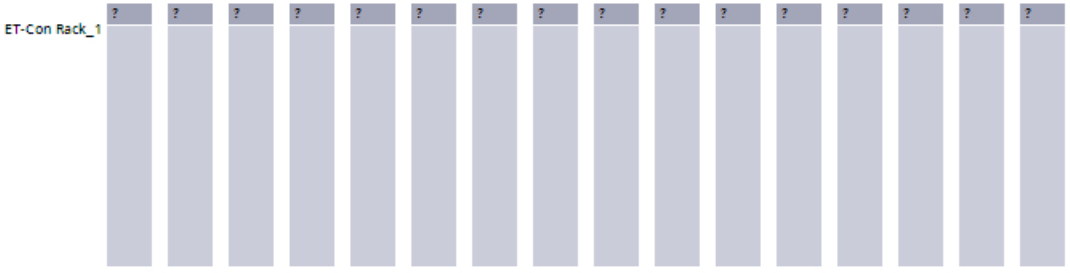
The system can be expanded with modules of the ET 200AL series with IP65/67 degree of protection. ET 200AL modules can be mounted on site, for example on a machine.

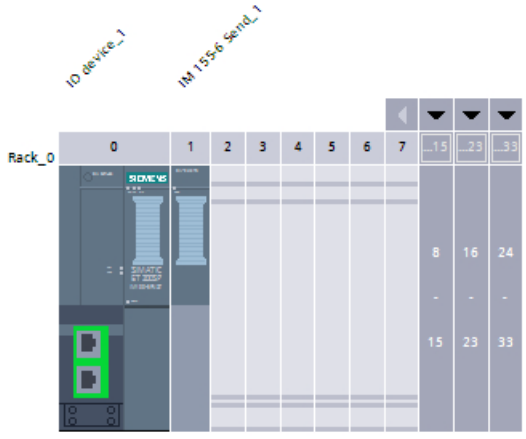
The following section describes how you expand an ET 200SP station with ET 200AL modules in STEP 7.

Procedure

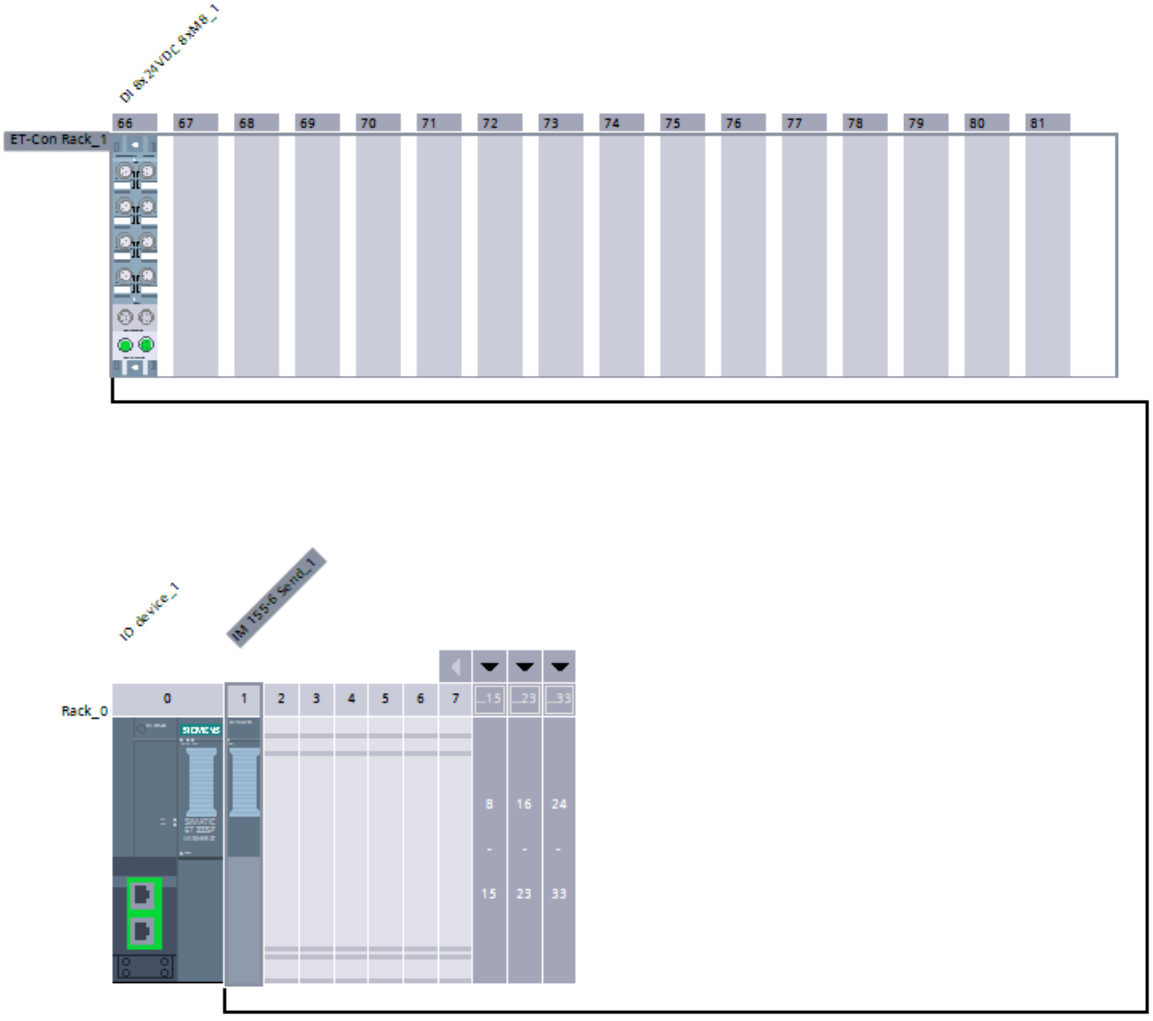
Follow the steps below to configure an ET 200SP with ET 200AL modules:

1. Drag an interface module (PROFINET or PROFIBUS) from the ET 200SP series to the network view.
2. Go to the device view. You do this by clicking twice on the icon of the module you have just inserted.
3. Insert the module "BA Send 1xFC" into slot 1 of the ET 200SP.
   STEP 7 now generates an ET-Connection rack with 16 slots for ET 200AL modules (figure below).
   An ET-Connection rack is a virtual rack that sets the order of the connected ET 200AL modules.
   Question marks are displayed above the slots as an ET 200AL module has yet to be connected to BA-Send.

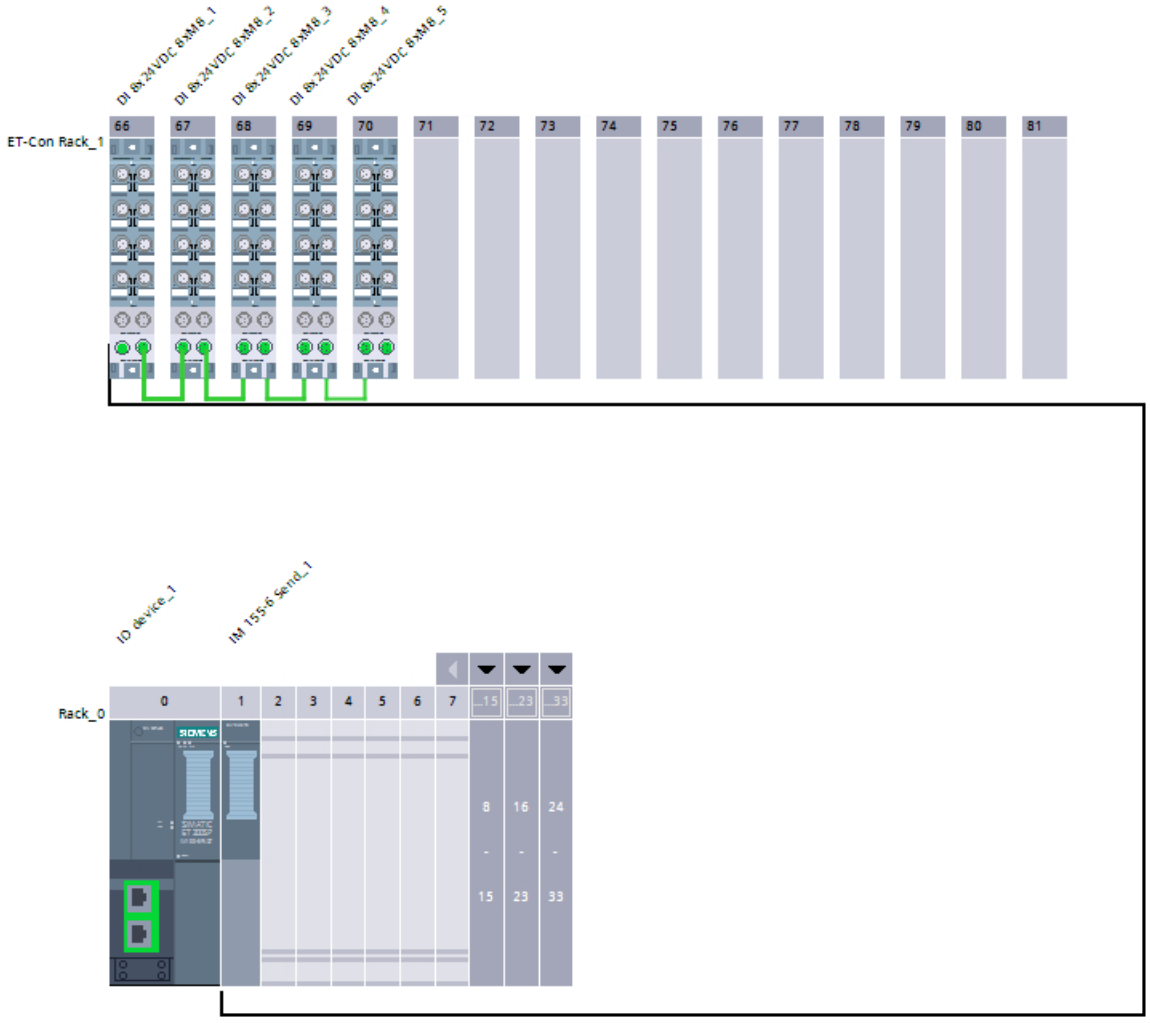




4. From the hardware catalog (subfolder ET 200AL in the ET 200SP folder), select the first ET 200AL module to be connected to the ET 200SP: Drag this module to the 1st slot in the ET-Connection rack.
   From this module, STEP 7 generates a line to the "BA-Send 1xFC" module and allocates the slot numbers 66 to 81 (figure below).
   If you configure the ET 200SP with a DP interface module, STEP 7 assigns the slot numbers 34 to 49 for the ET 200AL modules.



5. Now drag all other ET 200AL modules into the free slots in the ET-Connection rack.
   STEP 7 automatically inserts the ET-Connection between the individual ET 200AL modules (green loops).
   In the configuration below, five ET 200AL modules are connected in series.



6. Complete the configuration of the ET 200SP: Drag all ET 200SP modules from the hardware catalog to the slots in the ET 200SP.
   Five ET 200SP modules are plugged in the configuration example below: The PN interface module in slot 0, the module "BA-Send 1xFC" in slot 1, one input module each in slots 2 and 3, and a server module in slot 4:

Rules

- The module "BA-Send 1xFC" must be plugged in slot 1 of the ET 200SP if the ET 200SP is to be expanded with ET 200AL modules
- The ET 200AL modules must be configured with no gaps.

See also

Configuration control with ET 200AL (Page 1316)

Interface module parameters

Status bytes

Status bytes

If you enable the "Status bytes" option, 4 bytes of input data are reserved for the status of the supply voltage of each I/O module.

	7	6	5	4	3	2	1	0
Byte 0	8	7	6	5	4	3	2	1
Byte 1	16	15	14	13	12	11	10	9
Byte 2	24	23	22	21	20	19	18	17
Byte 3	32	31	30	29	28	27	26	25

I/O module slots

Bit = 0: Load voltage missing or I/O module not present

Bit = 1: Load voltage and I/O module present

Note

An inserted or missing server module always reports for the slot bit = 0.

Group diagnostics, missing supply voltage L+

Group diagnostics, missing supply voltage L+

This diagnostics is a group diagnostics that covers the supply voltage status of all I/O modules of a potential group which are defined by BaseUnits with incoming power supply (light-colored BaseUnit BU...D).

The group diagnostics is formed from the states of the supply voltage of the inserted I/O modules within the potential group.

The group diagnostics does not depend on the "Missing supply voltage L+" parameter of the I/O modules being enabled.

The server module does not influence the missing supply voltage L+ group diagnostics.

Requirements for the correct operation of the group diagnostics for missing supply voltage L+:

- I/O modules or BU covers must be inserted on the light-colored and dark-colored BaseUnits. If no I/O module is inserted on a light-colored BaseUnit, the start of this potential group will not be detected by the interface module; the I/O modules of this potential group will thus belong to the previous potential group. A supply voltage L+ group error will then be assigned to the wrong potential group.
  When an I/O module is inserted on the light-colored BaseUnit, the interface module detects the new potential group, re-evaluates the status, and reports a new group diagnostics in the case of an error.
- The server module must be inserted.
  The server module itself does not influence the missing supply voltage L+ group diagnostics.

Configuration control with ET 200SP

Operating principle

Configuration control allows you to operate various real configurations (options) with a single configuration of the distributed I/O device ET 200SP.

Configuration control provides you with the option of configuring the ET 200SP distributed I/O device with its maximum configuration and nevertheless operating it with missing modules. If missing modules are retrofitted later, no new configuration is required and the hardware configuration does not have to be reloaded either.

Using control data record 196, which is transferred to the interface module in the user program, you define a current preset configuration.

- The configured module is not present on a slot.
  - A BU cover can be inserted in this slot instead of the configured I/O module. As there is no configured module on the slot, the term "Configuration control with empty slots" is also used.
  - The module that is configured to the right of the missing module can be inserted on this slot instead of the configured module. The missing module makes the actual configuration appear pushed together. As the configured module is missing but no gap arises in the configuration, this is also referred to as a "Configuration without empty slots".
- The configuration is extended by an already configured module.
  - In the case of configuration control with empty spaces, you extend the configuration by inserting the configured module on the corresponding empty slot.
  - In the case of configuration control without empty spaces, insert the configured module on the right-hand side next to the last module of the ET 200SP.

Requirement

- STEP 7 Professional version V13 SP1 or higher
- The CPU startup parameter "Compare preset to actual configuration" is set to Startup even if mismatch (default setting). This setting is also selected for the startup parameters of the individual modules of the ET 200SP.

Enabling configuration control

In the properties of the interface module under Module parameters > General > Configuration control, select "Enable reconfiguration of device via user program". This activates configuration control.

Control data record 196 for ET 200SP

The figure below shows the data block 196 for the configuration control of an ET 200SP with four modules.

The value "12" is in the "block_length" element.

If you configure an ET 200SP in STEP 7 with more modules, the data block will be longer. The data record for the maximum configuration with 65 modules is 134 bytes long (configuration with PN interface module).

There are two bytes in the data record for each module.

The positions of these two bytes in the data record each code a module in the original configuration with STEP 7:

- "slot_1" and "info_slot_1" (bytes 4 and 5 in the data record, see figure below) correspond to the module in slot 1 in configuration with STEP 7.
- "slot_2" and "info_slot_2" (bytes 6 and 7) correspond to the module in slot 2 in configuration with STEP 7.
- "slot_3" and "info_slot_3" (bytes 8 and 9) correspond to the module in slot 3 in configuration with STEP 7.
- etc.

"slot_x" byte

The current slot is coded by the figure that is assigned to "slot_x" (by its value). Examples:

- The value "1" in byte 4 means you are assigning the module originally inserted in slot 1 to slot 1 in the current configuration (slot_1 = 1).
- The value "2" in byte 4 means you are assigning the module originally inserted in slot 1 to slot 2 in the current configuration (slot_1 = 2).
- The value "3" in byte 4 means you are assigning the module originally inserted in slot 1 to slot 3 in the current configuration (slot_1 = 3).
- etc.
- The value "1" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 1 in the current configuration (slot_2 = 1).
- The value "2" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 2 in the current configuration (slot_2 = 2).
- The value "3" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 3 in the current configuration (slot_2 = 3).
- etc.

If a BU cover can be inserted instead of a module, code this by adding 128 to the slot (bit 7 of the "slot_x" byte is set). Examples:

- The value "129" in slot_1 means you are also assigning the module originally inserted in slot 1 to slot 1 in the current configuration. A BU cover can also be used instead of this module. In the real plant configuration, either the module or a BU cover is inserted.
- The value "130" in slot_1 means you are assigning the module originally inserted in slot 1 to slot 2 in the current configuration. A BU cover can also be used instead of this module. In the real plant configuration, either the module or a BU cover is inserted.
- The value "131" in slot_1 means you are assigning the module originally inserted in slot 1 to slot 3 in the current configuration. A BU cover can also be used instead of this module. In the real plant configuration, either the module or a BU cover is inserted.

"info_slot_x" byte

If a new potential group is opened with the module, assign the "info_slot_x" byte the value 1 (bit 0 of the byte is set). Examples:

- The value "1" in the "info_slot_2" byte means that a new potential group is opened with module 2.
- The value "1" in the "info_slot_3" byte means that a new potential group is opened with module 3.
- The value "1" in the "info_slot_4" byte means that a new potential group is opened with module 4.

Exception: A new potential group is automatically assigned to the first module in the original configuration in STEP 7. This is not coded in the data record. You can enter any value in "info_slot_1".

You can choose any name for the components of the control data record (for example "slot_1").

Example of control data record 196 for ET 200SP

The figure below shows control data record 196 for an ET 200SP with four modules.

The module inserted in slot 2 in the configuration in STEP 7 can also be inserted in slot 2 in this configuration. It can also be inserted in slot 2 of a BU cover. Otherwise, nothing has changed compared to the original configuration.

Record_196_ET_200SP

	Name	Data type	Default value
1	block_length	USInt	12
2	block_ID	USInt	196
3	version	USInt	2
4	subversion	USInt	0
5	slot_1	USInt	1
6	info_slot_1	USInt	0
7	slot_2	USInt	130
8	info_slot_2	USInt	0
9	slot_3	USInt	3
10	info_slot_3	USInt	0
11	slot_4	USInt	4
12	info_slot_4	USInt	0

Addressing the interface module using the HW identifier

To transfer data record 196 with the instruction WRREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Readback data record 197 for ET 200SP

Readback data record 197 is used to read the actual configuration of a station (in this case, of an ET 200SP).

This readback data record allows you to check the real configuration of the ET 200SP (actual configuration). The readback data record for each configured module specifies whether or not it is actually available.

- The value "1" means that the correct module is inserted in the correct slot.
- The value "0" codes all other options (wrong module, empty slot, BU cover).

Configuration details:

The configuration of the data block corresponds to the original configuration of the ET 200SP with STEP 7.

There are two bytes in the data record for each module. The position of these two bytes in the data record corresponds to the position of a module in the original configuration with STEP 7.

Sequence of bytes:

- "status_slot_1" and "reserve_1" (bytes 4 and 5 in the data record) correspond to the module in slot 1 in the configuration,
- "status_slot_2" and "reserve_2" (bytes 6 and 7) correspond to the module in slot 2 in the configuration
- "status_slot_3" and "reserve_3" (bytes 8 and 9) correspond to the module in slot 3 in the configuration,
- etc.

Example

The original configuration in STEP 7 has been changed by a control data record 196 (see example above): In the modified configuration, module 2 can either be inserted in slot 2 or be replaced by a BU cover.

The figure below shows readback data record 197 which ET 200SP outputs to indicate that there is a module in slot 2: The "status_slot_2" byte has the value "1".

The other modules are also available and are inserted in the correct slots.

Record_197_ET_200SP

	Name	Data type	Default value
1	block_length	USInt	12
2	block_ID	USInt	197
3	version	USInt	2
4	subversion	USInt	0
5	status_slot_1	USInt	1
6	reserve_1	USInt	0
7	status_slot_2	USInt	1
8	reserve_2	USInt	0
9	status_slot_3	USInt	1
10	reserve_3	USInt	0
11	status_slot_4	USInt	1
12	reserve_4	USInt	0

The figure below displays the readback data record 197 which ET 200SP outputs to indicate that a BU cover is being used in slot 2: The "status_slot_2" byte has the value "0".

The other modules are available and are inserted in the correct slots.

Record_197_ET_200SP			
	Name	Data type	Default value
1	block_length	USInt	12
2	block_ID	USInt	197
3	version	USInt	2
4	subversion	USInt	0
5	status_slot_1	USInt	1
6	reserve_1	USInt	0
7	status_slot_2	USInt	0
8	reserve_2	USInt	0
9	status_slot_3	USInt	1
10	reserve_3	USInt	0
11	status_slot_4	USInt	1
12	reserve_4	USInt	0

Reading readback data record 197

You can read readback data record 197 from the ET 200SP with the instruction RDREC. RDREC operates asynchronously. If you call RDREC in the startup OB, you must call the instruction multiple times using a loop until the "BUSY" or "DONE" output parameter indicates that the data record has been read.

To read data record 197 with the instruction RDREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Further information and examples

Specific examples of configuration control can be found in this application description (http://support.automation.siemens.com/WW/view/en/29430270).

Further information on ET 200SP can be found in the manuals IM 155-6 PN (http://support.automation.siemens.com/WW/view/en/73184046) and IM 155-6 DP (http://support.automation.siemens.com/WW/view/en/73098660)

See also

Configuration control with ET 200AL (Page 1316)

Configuration control for ET 200SP with integrated ET 200AL modules (Page 1278)

Configuration control for ET 200SP with integrated ET 200AL modules

ET 200SP with integrated ET 200AL modules

Configuration control for ET 200SP and ET 200AL is described in separate texts. See the links under "See also".

The steps outlined also apply to configuration control of an ET 200SP with integrated AL modules. The procedure is different for control data record 196 and readback data record 197.

The following help text describes control data record 196 and the readback data record 197 for an ET 200SP expanded with ET 200AL modules.

Control data record 196

The two figures below show parts of control data record 196 for a configuration of an ET 200SP expanded with ET 200AL modules.

This configuration is given as an example:

- The module "BA Send 1xFC" is in slot 1 "slot_1". This module allows you to integrate ET 200AL modules into an ET 200SP. In our configuration example, 16 AL modules are connected to the BA-Send (maximum configuration). If a BA-Send is being used, this module must be inserted in slot 1.
- All slots from 2 to 64 are assigned ET 200SP modules.
- A server module is inserted in slot 65.
- There are 16 AL modules in slots 66 to 81.

The ET 200SP with integrated AL modules, originally configured with STEP 7, is now to be reconfigured from the user program.

The new configuration has the following properties:

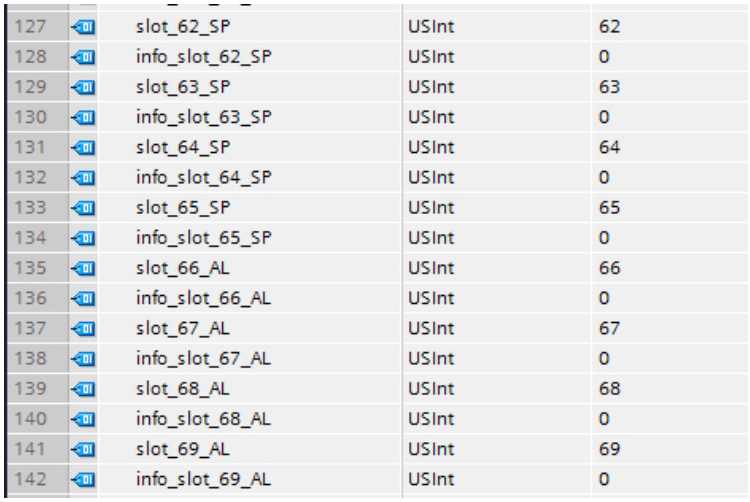
- The "BA Send 1xFC" module is inserted in "slot_1_BA-Send" (fixed setting).
- Module 2 "slot_2" is not available in the modified configuration (value "0").
- Module 3 "slot_3" is in slot 2 in the modified configuration (value "2").
- Module 4 "slot_4" is in slot 3 in the modified configuration (value "3").
- All modules from slot 5 to slot 81 are operated in the original configuration with STEP 7.

Configuration_ET200SP_with_AL-Modules

		Name	Data type	Start value
1		Static		
2		block_length	USInt	166
3		block_ID	USInt	196
4		version	USInt	2
5		subversion	USInt	0
6		slot_1_BA-Send	USInt	1
7		info_slot_1_BA-Send	USInt	0
8		slot_2	USInt	0
9		info_slot_2	USInt	0
10		slot_3	USInt	2
11		info_slot_3	USInt	0
12		slot_4	USInt	3
13		info_slot_4	USInt	0
14		slot_5	USInt	5
15		info_slot_5	USInt	0
16		slot_6	USInt	6
17		info_slot_6	USInt	0
18		slot_7	USInt	7
19		info_slot_7	USInt	0

The components of control data record 196 (figure above):

- block_length: Note the length of the control data record here; in the example: 166 (bytes). The length of the control data block is calculated using the formula "2 x number of slots + 4".
- block_ID: Enter the figure 196 here.
- version: The ET 200SP uses version 2 of control data record 196.
- subversion: The ET 200SP uses subversion 0 of control data record 196.



127	slot_62_SP	USInt	62
128	info_slot_62_SP	USInt	0
129	slot_63_SP	USInt	63
130	info_slot_63_SP	USInt	0
131	slot_64_SP	USInt	64
132	info_slot_64_SP	USInt	0
133	slot_65_SP	USInt	65
134	info_slot_65_SP	USInt	0
135	slot_66_AL	USInt	66
136	info_slot_66_AL	USInt	0
137	slot_67_AL	USInt	67
138	info_slot_67_AL	USInt	0
139	slot_68_AL	USInt	68
140	info_slot_68_AL	USInt	0
141	slot_69_AL	USInt	69
142	info_slot_69_AL	USInt	0

The components of control data record 196 (figure above):

- slot_65_SP: This byte relates to the server module in the ET 200SP rack. It ends the backplane bus of the ET 200SP.
- From "slot_66_AL" come the 16 configured ET 200AL modules: Our configuration example does not change the configuration with STEP 7. The byte "slot_66_AL" has the value "66", the byte "slot_67_AL" has the value "67", the byte "slot_68_AL" has the value "68", etc.

Definition of control data record 196

A control data record 196 containing a slot assignment is defined for configuration control.

Byte	Component	Value	Description	
0	block_length	e.g. 166 for maximum configuration with 65 ET-200SP modules and 16 ET 200AL modules (with DP interface module, maximum of 33 ET 200SP modules and 16 ET 200AL modules)	The length of the data record is calculated using the formula: 4 + "number of modules" x 2	Header
1	Block ID	196	ID for control data record 196	
2	version	2	Version 2 of control data record 196	
3	subversion	0	Subversion 0 of control data record 196	
4	slot_1_BA-Send	Real slot for SP module 1 Possible value: 1	When AL modules are integrated in ET 200SP, the module "BA Send 1xFC" must always be inserted in slot 1.	1. Slot for SP modules Assignment for the configured SP module 1 to a real slot
5	info_slot_1_BA-Send	0 or 1	The value "1" means that a new potential group is opened with this module. (Not evaluated in this byte)	
6	slot_2	Real slot for SP module 2 Possible values: 2 to 65 (not 66 to 81, reserved for AL modules) 0 (if the configured module 2 is not available)	The configured SP module 2 can be inserted in any real slot from 2 to slot 65 (2 to 33 for a DP interface module). Slots 66 to 81 are for AL modules (34 to 49 with DP interface module).	2. Slot for SP modules Assignment for the configured SP module 2 to a real slot
7	info_slot_2	1	The value "1" means that a new potential group is opened with this module. A new potential group must always be opened in this byte as BA-Send cannot open a new potential group.	

8	slot_3	Real slot for SP module 3 Possible values: 2 to 65 (not 66 to 81, reserved for AL modules) 0 (if the configured module 3 is not available)	The configured SP module 3 can be inserted in any real slot from 2 to slot 65 (2 to 33 for a DP interface module). Slots 66 to 81 are for AL modules (34 to 49 with DP interface module).	3. Slot for SP modules Assignment for the configured SP module 3 to a real slot
9	info_slot_3	1	The value "1" means that a new potential group is opened with this module. .	
:	:	:	:	:
132	slot_65	Real slot for SP module 65 Possible values: 2 to 65 (not 66 to 81, reserved for AL modules) 0 (if the configured module 65 is not available)	The configured SP module 65 can be inserted in any real slot from 2 to slot 65 (2 to 33 for a DP interface module). Slots 66 to 81 are for AL modules (34 to 49 with DP interface module)	65. Slot for SP modules Assignment for the configured SP module 65 to a real slot
133	info_slot_65	0 or 1	The value "1" means that a new potential group is opened with this module (the value is not evaluated in this slot).	
134	slot_66	Real slot for AL module 1 Possible values: 66 to 81 (not 1 to 65, reserved for SP modules) 0 (if the configured AL module 1 is not available)	The configured AL module 1 can be inserted in any slot from 66 to slot 81 (34 to 49 for PROFIBUS).	1. Slot for AL modules Assignment for the configured AL module 1 to a real slot
135	info_slot_66	-	Reserve	
:	:	:	:	:

164	slot_81	Real slot for AL module 16 Possible values: 66 to 81 (not 1 to 65, reserved for SP modules) 0 (if the configured AL module 16 is not available)	The configured AL module 16 can be inserted in any slot from 66 to slot 81 (34 to 49 for DP interface module).	16. Slot for AL modules Assignment for the configured AL module 16 to a real slot
165	info_slot_81	-	Reserve	

Rules

- If the "BA Send 1xFC" module is being used, it must be inserted in slot 1.
- ET 200SP modules are inserted in slots 2 to 65 (slots 2 to 33 for DP interface module).
- AL modules are inserted in slots 66 to 81 (slots 34 to 49 for DP interface module).
- If you expand an ET 200SP with ET 200 AL modules, the 1st AL module is always coded in bytes 134 and 135 of the control data record, the 2nd AL module in bytes 136 and 137, etc., even if not all SP slots are to be assigned SP modules. Non-assigned SP slots are coded with the value "0".

Error messages

The following error messages are returned if an error occurs when writing control data record 196:

Table 10-95 Error messages

Error code	Meaning
16#80A2	DP protocol error on layer 2. Indicates that a data record has not been acknowledged due to the system.
16#80B1	Invalid length; the length information in data record 196 is not correct.
16#80B5	Configuration control parameters not assigned.
16#80B2	Invalid slot: The configured slot is not assigned.
16#80B8	Parameter error; module signals invalid parameters.
16#80C5	DP slave or module not available. Indicates that a data record has not been acknowledged due to the system.

Readback data record 197 for ET 200SP with AL modules

The actual configuration of an ET 200SP with AL modules can be checked with readback data record 197.

Data record 197 largely corresponds to readback data record 197 for ET 200SP without AL modules; however, it is longer as the additional AL modules also need to be coded.

There are two bytes in the data record for each module. The position of these two bytes in the data record codes a module in the original configuration with STEP 7.

In the figure below:

- The components "status_slot_1" and "reserve_1" (bytes 4 and 5 in the data record) correspond to the module in slot 1 in the configuration with STEP 7,
- "status_slot_2" and "reserve_slot_2" (bytes 6 and 7) correspond to the module in slot 2,
- "status_slot_3" and "reserve_slot_3" (bytes 8 and 9) correspond to the module in slot 3,
- etc.

The following data record is structured for configuration with 65 SP modules and 16 AL modules. The value "166" therefore appears in the "block_length" element of the data record.

If you configure an ET 200SP in STEP 7 with fewer AL modules, the data block will be shorter.

If you use fewer SP modules in a configuration, this has no effect on the length of data record 197 (with an expansion of ET 200SP with ET 200AL modules).

The "reserve_x" component of readback data record 197 is reserved for future applications.

You can choose any name for the components of the readback data record (for example "status_slot_1").

The figure below shows the start of readback data record 197 for reading the actual configuration of an ET 200SP with AL modules.

DB_read_back_ET_200SP_with_AL

	Name	Data type	Start value
1	Static		
2	block_length	USInt	166
3	block_ID	USInt	197
4	version	USInt	2
5	subversion	USInt	0
6	status_slot_1	USInt	1
7	reserve_slot_1	USInt	0
8	status_slot_2	USInt	1
9	reserve_slot_2	USInt	0
10	status_slot_3	USInt	1
11	reserve_slot_3	USInt	0
12	status_slot_4	USInt	1
13	reserve_slot_4	USInt	0
14	status_slot_5	USInt	1
15	reserve_slot_5	USInt	0
16	status_slot_6	USInt	1
17	reserve_slot_6	USInt	0

It does not show the components "status_slot_7" to "reserve_81" (maximum configuration of an ET 200SP with PN interface module) or "status_slot_7" to "reserve_slot_49" (maximum configuration of an ET 200SP with DP interface module).

Meaning of "status_slot_x":

- The value "1" in status_slot_x means that module x is inserted in the correct slot
- The value "0" codes all other options (wrong module, empty slot, BU cover).

The figure below shows part of readback data record 197 for reading the actual configuration of an ET 200SP with AL modules (and PN interface module). ET 200SP modules are inserted up to slot 65, and then the AL modules. For example, a value of "1" in the byte "status_slot_66_AL" means that the 1st AL module is actually available in the plant and inserted in the correct slot.

128	status_slot_62_SP	USInt	1
129	reserve_slot_62_SP	USInt	0
130	status_slot_63_SP	USInt	1
131	reserve_slot_63_SP	USInt	0
132	status_slot_64_SP	USInt	1
133	reserve_slot_64_SP	USInt	0
134	status_slot_65_SP	USInt	1
135	reserve_slot_65_SP	USInt	0
136	status_slot_66_AL	USInt	1
137	reserve_slot_66_AL	USInt	0
138	status_slot_67_AL	USInt	1
139	reserve_slot_67_AL	USInt	0
140	status_slot_68_AL	USInt	1
141	reserve_slot_68_AL	USInt	0
142	status_slot_69_AL	USInt	1
143	reserve_slot_69_AL	USInt	0

Reading readback data record 197

You can read readback data record 197 from the ET 200SP with the instruction RDREC. RDREC operates asynchronously. If you call RDREC in the startup OB, you must call the instruction multiple times using a loop until the "BUSY" or "DONE" output parameter indicates that the data record has been read.

Further information and examples

Specific examples of configuration control can be found here in this application description (http://support.automation.siemens.com/WW/view/en/29430270).

Further information on ET 200SP can be found in the manuals for IM 155-6 PN (http://support.automation.siemens.com/WW/view/en/73184046) and IM 155-6 DP (http://support.automation.siemens.com/WW/view/en/73098660)

Further information on ET 200AL is available here (http://support.automation.siemens.com/WW/view/en/89254863).

See also

Configuration control with ET 200SP (Page 1274)

Configuration control with ET 200AL (Page 1316)

Output module parameters

Substitute value reaction

Substitute value reaction

In the ET 200SP, the substitute value reaction is executed by the IO controller per slot.

The respective output reacts according to its set substitute value reaction:

- "Turn off"
- "Output substitute value"
- "Keep last value"

This substitute value reaction is triggered in the following cases:

- IO controller in STOP
- Controller failure (connection interruption)
- Firmware update
- Reset to factory settings
- More than one I/O module withdrawn simultaneously
- Disable the IO device
- Station stop
  - Missing server module
  - More than one I/O module withdrawn simultaneously
  - At least one I/O module is inserted on the wrong BaseUnit

Note

Reducing a configuration

If you reduce the configuration of the ET 200SP and download the configuration to the CPU, the modules which are no longer configured but still present retain their original substitute value reaction. This applies until the supply voltage on the BaseUnit BU...D or on the interface module is turned off.

Input module parameters

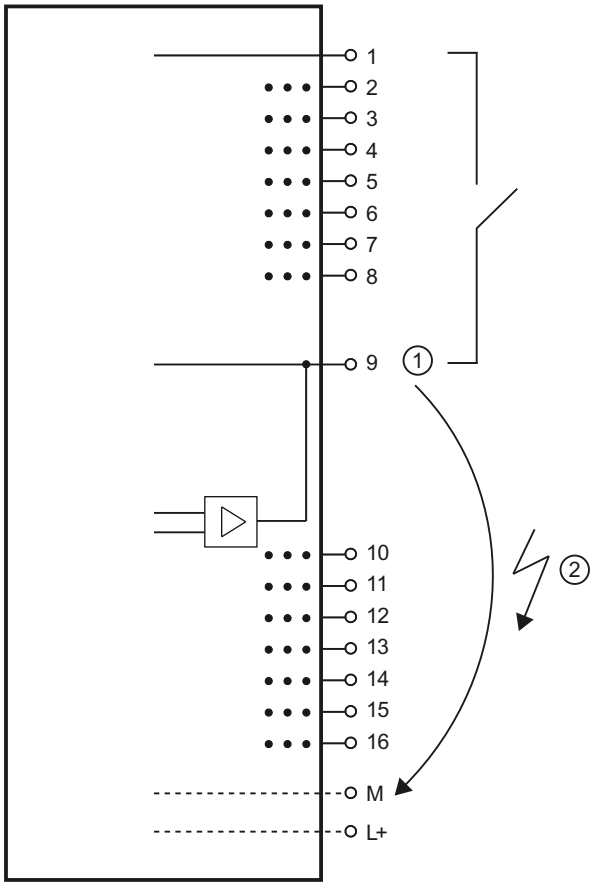
Parameters of the digital input modules

Diagnostics missing supply voltage L+

Enabling of the diagnostics for missing or insufficient supply voltage L+.

Diagnostics short-circuit to ground

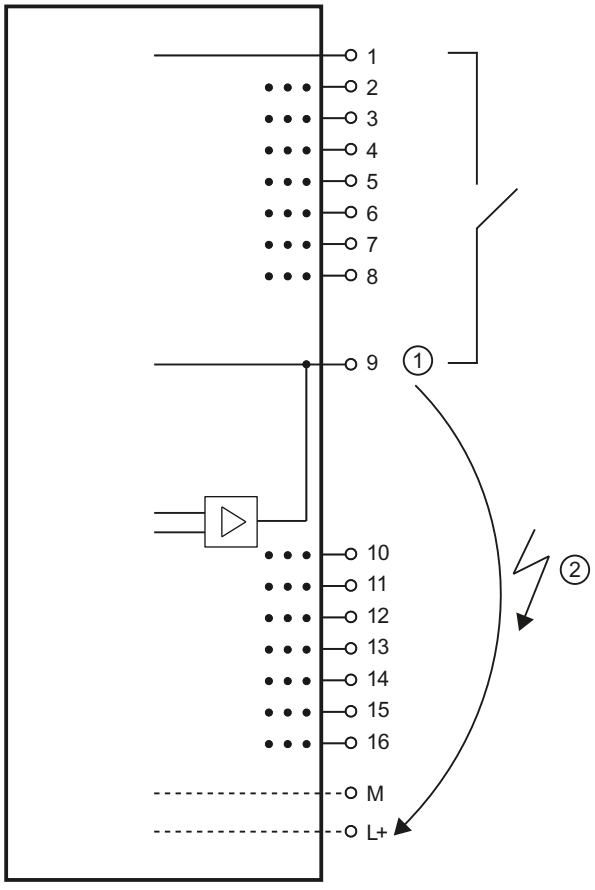
Enabling of the diagnostics if a short-circuit of the actuator supply to ground occurs.



① Encoder supply
② Short-circuit

Diagnostics short-circuit to L+

Enabling of the diagnostics if a short-circuit of the encoder supply to L+ occurs.



① Encoder supply
② Short-circuit

Diagnostics wire break

Enabling diagnostics if the line to the encoder is interrupted.

Operating mode

Determines whether a channel is enabled or disabled.

Pulse extension (only High Feature modules)

Pulse extension is a function for changing a digital input signal. A pulse at a digital input is extended to at least the configured length. If the input pulse is already longer than the configured length, the pulse is not changed.

Pulse extension is started whenever the state of the input signal changes and no pulse extension is active for this channel.

Potential group of the left module / new potential group

Specifies whether the I/O module is located on a base unit with supply voltage feed (new potential group) or whether it is it located on a base unit without supply voltage feed (in which case it belongs to the potential group of the left module).

Parameters of the analog input modules

Missing supply voltage L+

Enabling of the diagnostics, with missing or too little supply voltage L+.

Reference junction (AI 4xRTD/TC 2-/3-/4-wire HF)

A BaseUnit with internal temperature sensor (BU..T) or the channel 0 of the I/O module can be used as reference junction, provided this has been configured as "Thermal resistor Pt100 climatic range Celsius".

A possible parameter assignment is shown below (see also Information about reference channel mode (Page 1292)):

Table 10-96 RTD channel

Setting	Description
No reference channel operation	Temperature value at channel 0 can be used as module-wide reference value if the parameters of the other channels are assigned accordingly.
Reference channel of Group x	The channel acts as sender for the reference junction temperature of Group x. Distribution is performed via the interface module.

Table 10-97 TC channel

Setting	Description
Reference channel of the module	The corresponding TC channel uses the channel 0 of the same module as reference junction temperature. This must be set as "Thermal resistor Pt 100 climatic range Celsius" and "No reference channel operation"; otherwise, reference junction diagnostics is triggered.
Internal reference junction	The reference junction temperature is read by an internal temperature sensor on the BaseUnit. Reference junction diagnostics is triggered if there is a wrong BaseUnit type.
Reference channel of Group x	With the setting "TC" (thermocouple...), the channel acts as receiver for the reference junction temperature of Group x.
Fixed reference temperature	No temperature compensation occurs. The linearization is executed with an assumed reference junction temperature of 0 °C.

Overflow

Enabling of the diagnostics if the measured value exceeds the overflow range.

Underflow

Enabling of the diagnostics if the measured value falls below the underflow range.

Wire break

Enabling of the diagnostics if the module has no current flow or too low a current flow for the measurement on the corresponding configured input.

Smoothing

The individual measurements are smoothed using digital filtering. The smoothing can be set in 4 stages, whereby the smoothing factor k multiplied by the cycle time of the I/O module corresponds to the time constant of the smoothing filter. The larger the smoothing, the larger the time constant of the filter.

The following figure shows the step response for the various smoothing factors depending on the number of module cycles.

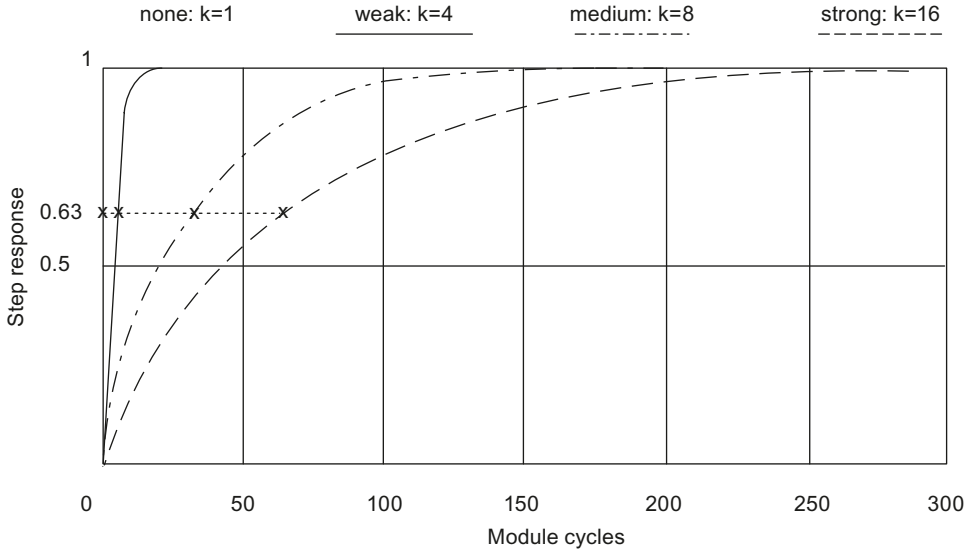


Figure 10-5 Smoothing with AI 4×RTD/TC 2-/3-/4-wire HF

Interference frequency suppression

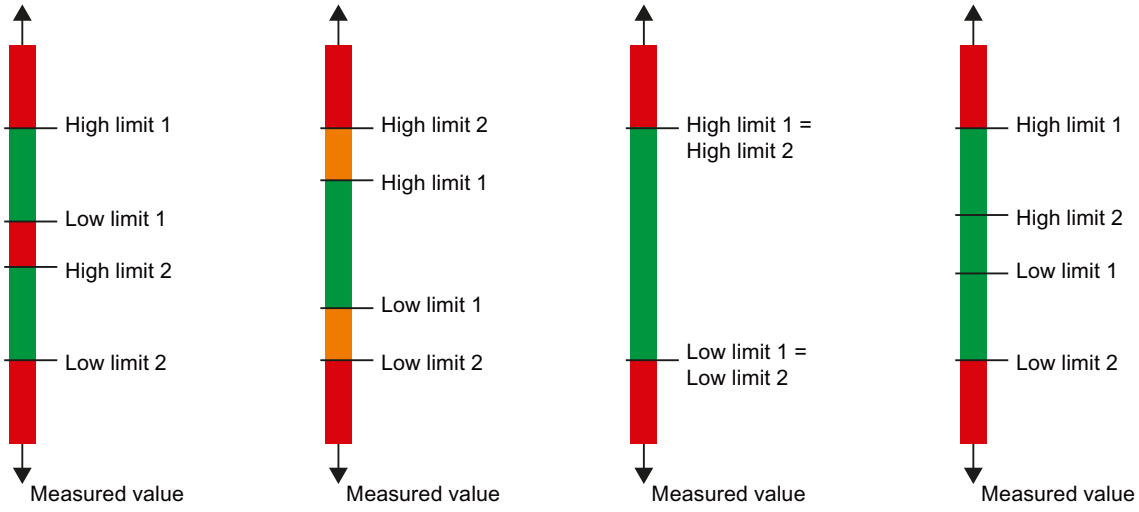
With analog input modules, suppresses the disturbance caused by the frequency of the AC network used.

The frequency of the AC network may interfere with measured values, particularly for measurements within low voltage ranges and when thermocouples are being used. This parameter is used to define the predominant power frequency of the system.

Hardware interrupt limits

If the high limit 1/2 or the low limit 1/2 is violated, the module triggers a hardware interrupt.

Below are some examples for the selection of the limits 1 and 2.



Low limit 1/2

Specify a threshold whose undershoot triggers a hardware interrupt.

High limit 1/2

Specify a threshold whose overrange triggers a hardware interrupt.

Potential group of the left module/New potential group

Specifies whether the I/O module is located on a BaseUnit with supply voltage infeed (new potential group) or on a BaseUnit without supply voltage feed (in which case, it belongs to the potential group of the left module).

Temperature coefficient (measurement type thermoresistor)

The correction factor for the temperature coefficient (α value) defines the relative rate of change of the resistance of a specific material at a temperature rise of 1 °C.

The temperature coefficient depends on the chemical composition of the material. In Europe, only one value is used per sensor type (default value).

The further values facilitate a sensor-specific setting of the temperature coefficient and enhance accuracy.

See also

Special features of AI 4xRTD/TC 2-/3-/4-wire HF (Page 1300)

Information about reference channel mode

An RTD/TC module of the ET 200SP works in reference channel mode when a channel sends the reference temperature to other channels of the station. The receiving channels use the reference temperature for temperature compensation when measuring with thermocouples.

Structure and use of thermocouples

The thermocouple consists of two wires with different metals or alloys that are welded together at the end. The weld is called measuring point.

The other end of the two wires is open. This end is called reference junction.

A thermal voltage, which depends on the temperature at the measuring point, occurs between the two metals/alloys at the measuring point. Other thermal voltages also occur at the reference junction - with transition from thermocouple to copper lines, for example - that falsify the actual measured value and need to be compensated. No compensation is required for a reference junction temperature of 0 °C.

Different methods are used to compensate the reference junction temperature:

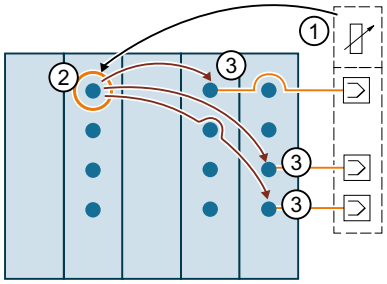
- Fixed reference temperature: The reference junction is permanently set to a specific temperature, for example, to 0 °C by an icewater bath (no compensation required).
- Internal reference junction: The reference junction is the terminal of the BaseUnit on which the analog module is plugged. If you select the "Internal reference junction" compensation type, you have to use BaseUnits with integrated temperature measurement to compensate the reference junction temperature. These BaseUnits have the designation "BU..T". The module records the temperature at the reference junction and uses this to determine the actual temperature at the measuring point.
- Reference channel of Group x: An external thermal resistor records the temperature at the reference junction for the Group x (group of channels within a station). The actual temperature at the respective measuring point can be determined as a result. One external thermal resistor is required for each group. The thermal resistors are each connected to one channel of an analog module. These channels are called senders (for the temperature at the reference junction).
  The mode of operation and settings are described in the section "Station-wide distribution of reference temperature".
- Reference channel of the module: The mode of operation is comparable with "Reference channel of Group x". You connect an external thermal resistor to channel 0 of the module to measure the temperature at the reference junction. Other channels of the same module use this reference temperature for temperature compensation.
  The mode of operation and settings are described in the section "Module-wide distribution of reference temperature".

You can find information about the structure and mode of operation of a thermocouple in the Analog value processing (http://support.automation.siemens.com/WW/view/en/67989094) manual.

Station-wide distribution of the reference temperature

You can record the temperature at the reference junction x using a thermal resistor on a channel (sender of the reference temperature) and send it to other channels within a station (receivers). All channels that receive the temperature at a reference junction x form the Group x.

For each group, you assign parameters to exactly one channel as sender of the reference temperature.



① Thermal resistor at the reference junction

② Channel records temperature at the reference junction and sends it to the other channels within a station (sender of Group x). The temperature value is used for the compensation of the reference junction temperature.

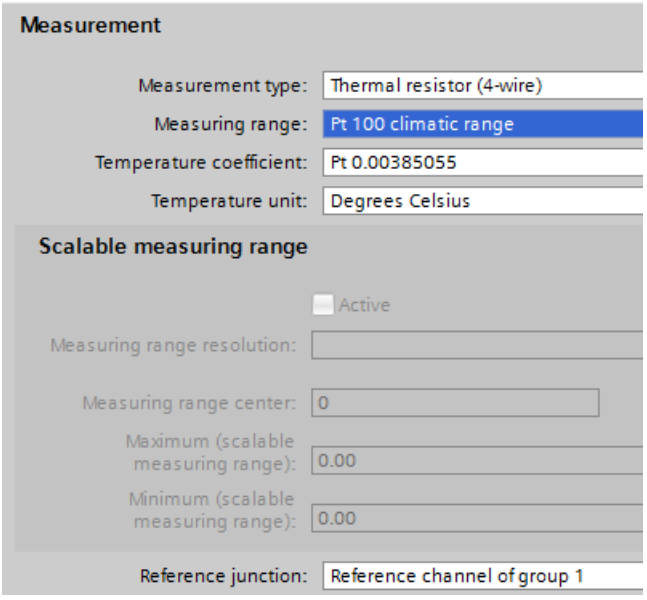
③ Channels of the Group x receive the temperature at the reference junction (receivers)

Parameter assignment of a channel as reference channel (sender for Group 1)

Parameter assignment is described below using Group 1 as example:

1. Open the project in STEP 7
2. Select the required analog module (RTD/TC) in the device view.
3. Then select a channel that is to work as sender of the reference junction temperature.
   The following settings are required:
   "Measurement type": "Thermal resistor", for example, "Thermal resistor (4-wire connection)"
   "Measuring range": "Pt 100 climatic range"
   "Temperature unit": "Degrees Celsius"
   "Reference junction": "Reference channel of Group 1"

The following figure shows the parameter assignment.

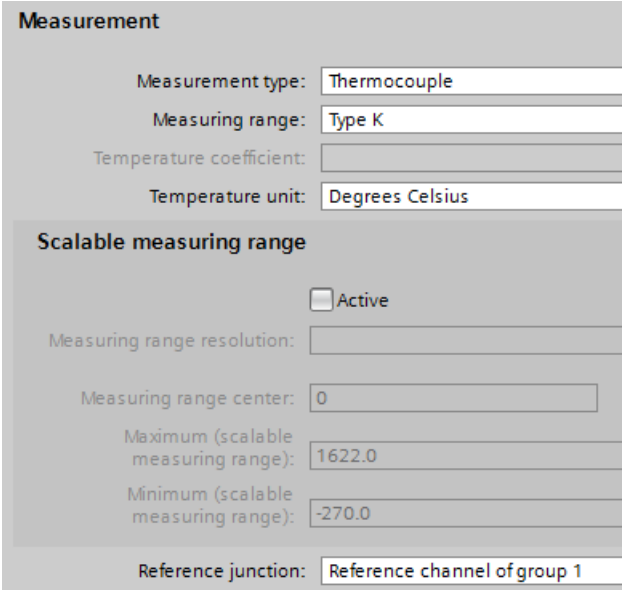


The channel configured in this way (thermal resistor measurement type) works as reference channel of Group 1 and sends the measured temperature to all channels (thermocouple measurement type) that are configured as receivers of Group 1.

In the next section, you will learn how to assign parameters for channels that are receivers of Group 1.

Parameter assignment of a channel as receiver of Group 1

The figure below shows the parameter assignment of a channel that receives the temperature at the reference junction of Group 1.

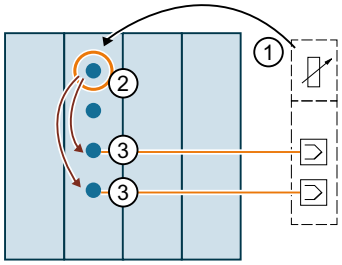


The following settings are required:

- "Measurement type": "Thermocouple"
- "Reference junction": "Reference channel of Group 1"

Module-wide distribution of the reference temperature

You can record the temperature on a reference junction using the channel 0 of the module and use the temperature value for the channels 1, 2, 3... of this module. The recorded temperature value is not sent to the channels of the other modules within the station (no reference channel mode).



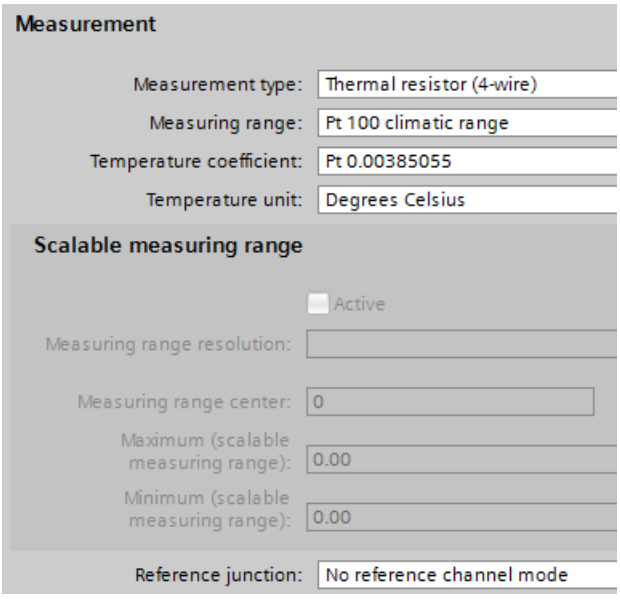
① Thermal resistor at the reference junction

② Channel records temperature at the reference junction, "Reference junction" parameter: "No reference channel mode"

③ Channels of the same module use the temperature value for the compensation of the reference junction temperature for the measurement with thermocouples, "Reference junction" parameter: "Reference channel of the module"

Parameter assignment of channel 0 as reference channel of the module

The figure below shows the parameter assignment of channel 0 of a module that is to be used for recording the temperature at the reference junction:

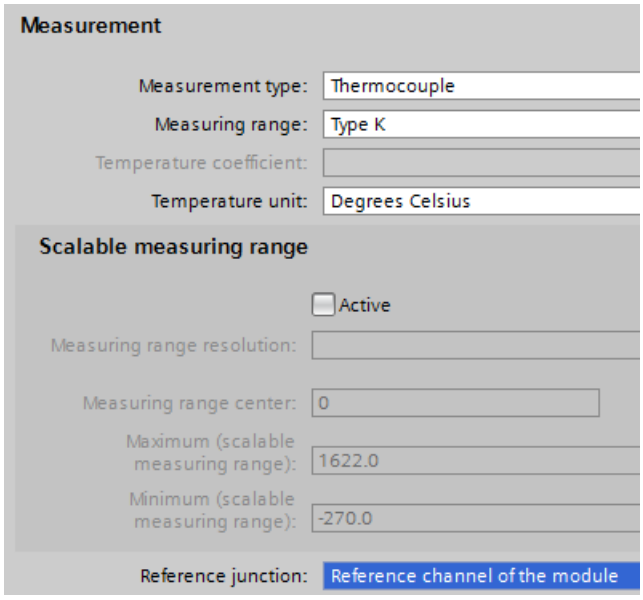


The following settings are required:

- "Measurement type": "Thermal resistor", for example, "Thermal resistor (4-wire connection)"
- "Measuring range": "Pt 100 climatic range"
- "Temperature unit": "Degrees Celsius"
- "Reference junction": "No reference channel mode"

Parameter assignment of a channel that uses channel 0 as reference channel

The figure below shows how parameters have to be assigned for a channel that uses the channel 0 of this module as reference channel for temperature compensation.



The settings below are required for the channels of the module that compensate the temperature of the reference junction using channel 0:

- "Measurement type": "Thermocouple"
- "Reference junction": "Reference channel of the module"

See also

Parameters of the analog input modules (Page 1289)

Information about the Oversampling function

High-speed analog modules (HS) are available to meet high performance and speed requirements. The main characteristics of these HS analog modules compared to Standard analog modules (ST) is their shorter cycle times. To achieve this goal, the input and output modules are equipped with components with extremely short throughput and conversion times. In addition, the entire architecture of the modules is designed for faster signal processing.

HS analog modules convert the output of measured values and output values at the same time. Each channel within the module has its own A/D or D/A converter. This means the cycle time is basically the conversion time and independent of the number of activated channels. This is true for analog input modules as well as analog output modules. This means HS modules can be used in fast isochronous mode.

Apart from isochronous mode, the HS analog modules also provide benefits in non-isochronous (free-running) mode. Due to the fast processing of the process signals, HS analog modules are able to detect changes in the process values more quickly and to respond to

these events with the appropriate program blocks (for example, hardware interrupt or cyclic interrupt organization blocks).

Isochronous mode

Isochronous mode refers to synchronous coupling

- Of signal acquisition and output via the distributed I/O
- Of signal transmission via PROFIBUS or PROFINET
- Of program processing with the constant bus cycle time of the PROFIBUS or PROFINET.

The result is a system that acquires its input signals in constant time intervals, processes them and outputs the output signals. Isochronous mode guarantees reproducible and defined process reaction times as well as equidistant and synchronous signal processing with distributed I/O.

The bus system and the I/O modules work synchronously with configured isochronous mode. The transmitted input and output data are linked to an "isochronous task" in the CPU. As a result, the data of a cycle are always consistent. All data of a process image belong together logically and in time. Jitter in the user program caused by the acquisition of outdated values is therefore almost impossible.

Even fast processes can be perfectly controlled by the exact timely reproducibility of all processes. Isochronous mode thus contributes to high control quality and hence to greater manufacturing precision. While possible fluctuations of the process reaction times are drastically reduced. The time-assured processing can be utilized to improve machine cycle times. Shorter cycle times increase the processing speed and help to lower production costs.

Oversampling

The use of the oversampling function in analog input or analog output modules requires an isochronous configuration.

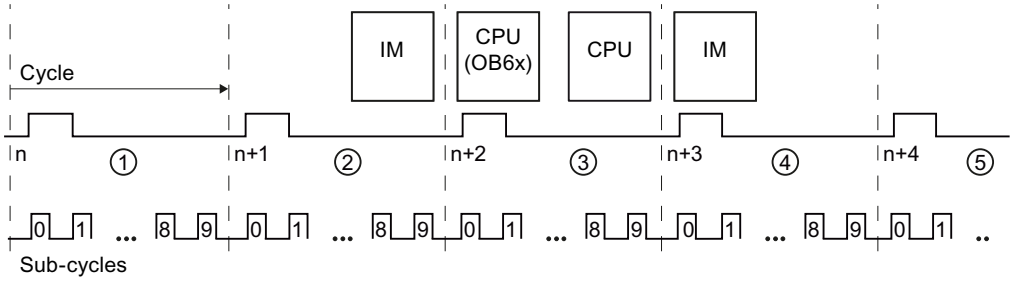
With analog input modules, the set send clock is divided into time-equidistant sub-cycles. The send clocks can be subdivided into 2 to 16 sub-cycles. Each sub-cycle reads in a measured value. The read-in measured values of a data cycle are copied to the interface module (IM) in the next send clock and are then available to the processing CPU one clock later.

With analog output modules, the set send clock is also divided into time-equidistant sub-cycles. The send clocks can be subdivided into 2 to 16 sub-cycles. Each sub-cycle returns an output value. The output values are copied to the interface module by the CPU within the same send clock and are written to the process one send clock later.

The read-in and output values are transmitted in the user data of the analog module. In this way, the address space of the module is extended from 2 bytes of user data per channel to 16 x 2 bytes of user data per channel (with 16 sub-cycles). If you subdivide the send clock into fewer than 16 sub-cycles, the unused addresses are assigned the error value 0x7FFF during input. For output, the values of the unused addresses are ignored.

Because the sub-cycles have to be within a send clock, oversampling needs an additional clock to copy the data to the interface module, unlike the 3-cycle model of isochronous mode. The result is a 5-cycle model.

Isochronous mode



① Send clock is divided into sub-cycles which record the measured value (in this case: 10 sub-cycles)

② Measured values are copied to the IM

③ Measured values are processed and output values determined

④ Output values are copied by the CPU to the IM

⑤ Output values are written to the process

Higher sampling rates through oversampling

Due to the configured modules, the IO device has a minimum possible update time. Within this update period, the IO device/IO controller of the PROFINET IO system is supplied once with new data.

The following applies with respect to the channels of a single standard module in the I/O device: The shortest update time ("sampling rate") is exactly one send clock.

If you want to shorten the update time for the channels of a standard module, you have to shorten the send clock. Because of the properties of the involved components and the structure of the I/O system, this is only possible to a certain extent (e.g., down to 0.25 ms).

Modules with oversampling, however, offer the option of further reducing the update time ("sampling rate") for their channels without having to shorten the send clock for the entire IO device at the same time.

The subdivision of the send clock into time-equidistant sub-cycles enables the processing of faster processes through higher sampling rates.

Example

In practice, the use of oversampling makes sense when the isochronous system works with only one specific send clock (for example, 1 ms) due to the modules used and when faster sampling of the process values is required. By using oversampling and subdividing the send clock into 4 sub-cycles, for example, you can sample the process values in intervals of 250 µs.

Configuring oversampling

Enable the option "Isochronous mode" in the IO device used, and set the corresponding parameters ("Send clock", etc.).

With the distributed analog input modules (e.g. AI 2xU/I 2,4-wire HS), you specify the number of sub-cycles using the "Sampling rate" parameter.

With the distributed analog output modules (e.g. AQ 2xU/I HS), you specify the number of sub-cycles using the "Sampling rate" parameter.

If, for example, you configure a "Sampling rate" of 4 "Values/cycle" for a send clock of 1 ms, the send clock is subdivided into 4 sub-cycles and the process values are sampled at intervals of 250 µs.

Reference

You can find additional information in the manuals on the high-speed analog modules and in the Analog value processing (http://support.automation.siemens.com/WW/view/en/67989094) function manual.

Special features of AI 4xRTD/TC 2-/3-/4-wire HF

Use of Cu10 sensors

- Select "3-wire thermal resistor" and "Cu10" in the parameter assignment.
- Wire the Cu10 sensor using 3-wire connection technology.
- An automatic, internal compensation of the line resistance of the missing measuring line takes place during operation.

Note

To ensure optimum line compensation for Cu10, please note the following:

- The sum of cable resistance and measuring resistance must not exceed 31 Ω.
- Cable resistance should not exceed 8 Ω if you want to use the temperature range up to over 312 °C.
  Example: A 200 m long copper cable with 0.5 mm^2 core cross-section has approx. 7 Ω.
  A lower cross-section reduces the permissible cable length accordingly.

Use of PTC resistors

PTCs are suitable for temperature monitoring of or as thermal protective equipment for complex drives or transformer windings.

- Select "2-wire resistor" and "PTC" in the parameter assignment.
- Connect the PTC using 2-wire technology.
- Use type A PTC resistors (PTC thermistors) in accordance with DIN/VDE 0660, Part 302.
- If the "Over-/underflow" diagnostics is enabled, a "low limit violation" diagnostics which shows a short circuit is generated for resistance values < 18 Ω.
- Sensor data on PTC resistance:

Table 10-98 Use of PTC resistors

Property	Technical specifications	Note
Switching points	**Reaction to rising temperature**	
	< 550 Ω	**Normal range:** • SIMATIC S7: bit 0 = "0", bit 2 = "0" (in the PII)
	550 Ω to 1650 Ω	**Prewarning range:** • SIMATIC S7: bit 0 = "0", bit 2 = "1" (in the PII)
	< 1650 Ω	**Response range:** • SIMATIC S7: bit 0 = "1", bit 2 = "0" (in the PII)
	Reaction to falling temperature	
	< 750 Ω	**Response range:** • SIMATIC S7: bit 0 = "1", bit 2 = "0" (in the PII)
	750 Ω to 540 Ω	**Prewarning range:** • SIMATIC S7: bit 0 = "0", bit 2 = "1" (in the PII)
	< 540 Ω	**Normal range:** • SIMATIC S7: bit 0 = "0", bit 2 = "0" (in the PII)
(TNF-5) °C (TNF+5) °C (TNF+15) °C Measuring voltage Voltage on the PTC	max. 550 Ω min. 1330 Ω min. 4000 Ω max. 7.5 V	RRT= rated response temperature

- Assignment in the process image inputs (PII) with SIMATIC S7

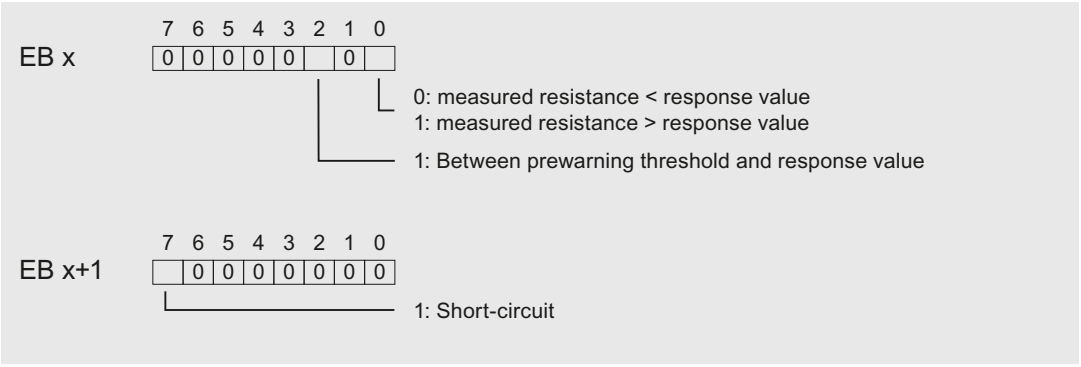


Figure 10-6 Assignment in the process image inputs (PII)

- Notes on programming

Note

Only the bits 0+2 are relevant for the evaluation in the process image inputs. You can use the bits 0+2 to monitor the temperature, for example, of a motor.

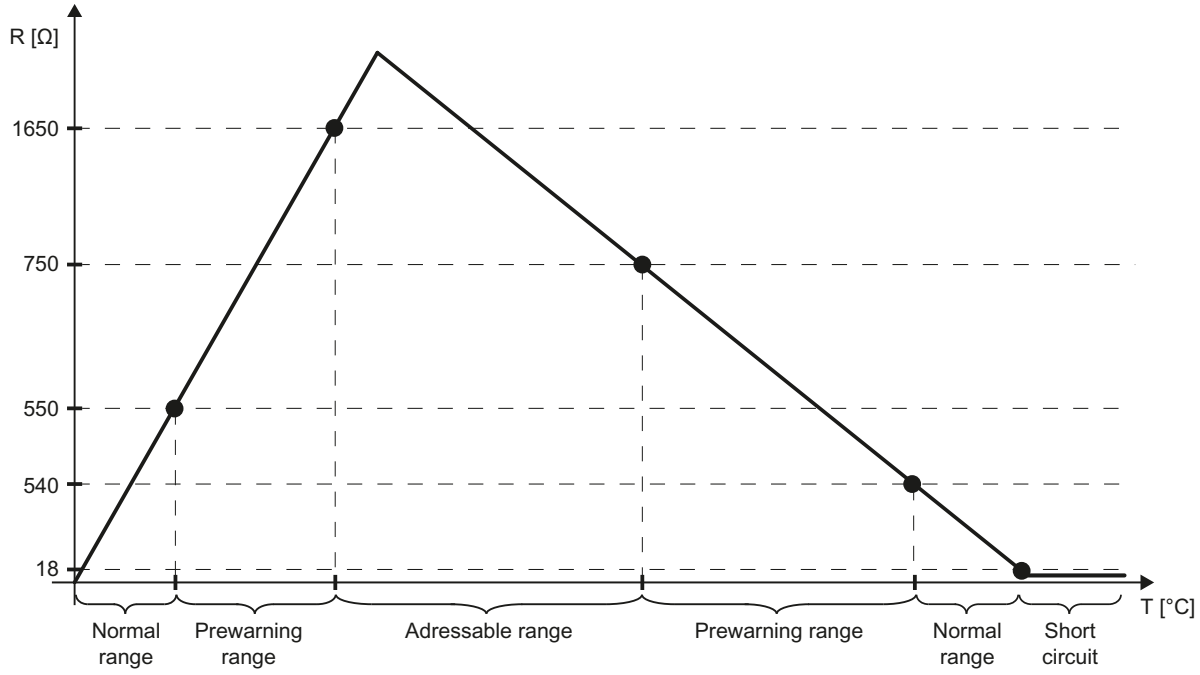
The bits 0+2 in the process image inputs have no latching function. When you are assigning parameters, take into consideration that a motor, for example, starts up in a controlled manner (via an acknowledgment).

Bits 0+2 can never be set simultaneously, but are instead set consecutively.

For safety reasons, always evaluate the diagnostic entries of the AI 4×RTD/TC 2-/3-/4-wire HF, as no measurement is possible if I/O modules are unplugged, if the supply voltage of the I/O module has failed, or if there is a wire break or short circuit of the measuring lines.

Example

The following diagram shows the temperature variation and the associated switching points.



See also

Parameters of the analog input modules (Page 1289)

What you should know about the scalable measuring range (Page 1303)

What you should know about the scalable measuring range

The scalable measuring range

The scalable measuring range is a section from the temperature measuring range of an analog input module (for example, the ET 200SP module "AI 8xRTD/TC 2-wire HF").

In this section, a higher resolution of the measured values is possible, comparable with a magnifier that allows a subsection to be viewed in greater detail.

The scalable measuring range is supported for the following measurement types:

- Thermal resistor (RTD) standard
- Thermocouple

The scalable measuring range is not available with the following measurement types:

- Voltage
- Resistance
- Thermal resistor climatic

Position and resolution of the scalable measuring range

The position and resolution of the scalable measuring range can be set (scaled):

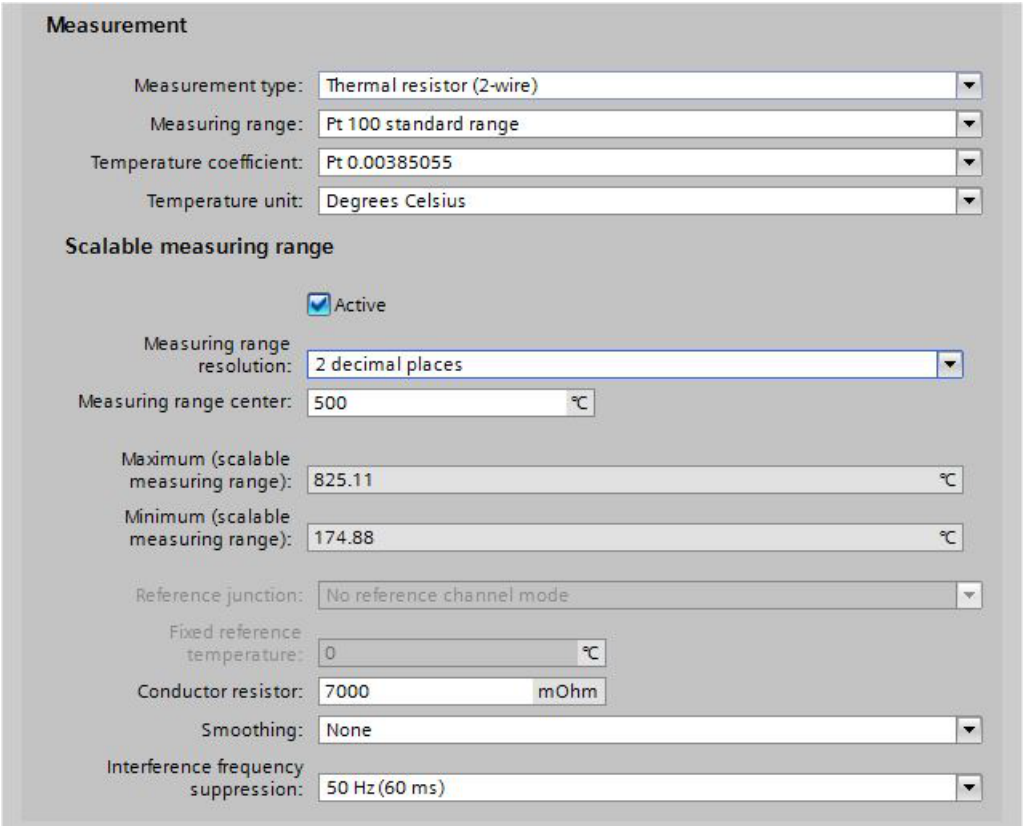
- **Position:** The scalable measuring range can be moved over the entire standard measuring range. This allows you to select the temperature range for which your application requires a higher resolution.
  Exception: The scalable measuring range cannot be moved to the extent that it enters the overflow or underflow of the standard measuring range (Clipping).
  You select the position of the scalable measuring range with the "Measuring range center" parameter (see figure below).
- **Resolution:** The following values can be set:
  - 2 decimal places (0.01 °C)
  - 3 decimal places (0.001 °C)

  You set the resolution with the "Measuring range resolution" parameter (figure below).

Example of a parameter assignment

The following figure shows a parameter assignment for the ET 200SP module "AI 8xRTD/TC 2-,3-,4-wire HF".

In STEP 7, you can find parameters in the properties box via General > AI 4 > Inputs > Channel 0 to channel 3.



Conductor resistor:

The "Conductor resistor" parameter in the parameter assignment above is enabled only if the "Thermal resistor (2-wire terminal)" measurement type was selected.

Here, enter the value for the resistance of the connecting cable of the thermal resistor: A 200 m long copper cable with 0.5 mm^2 wire cross-section, for example, has a resistance value of seven ohms.

Measuring range resolution:

In the parameter assignment above, a resolution of 0.01 °C was selected (measuring range resolution "2 decimal places").

Measuring range center:

The measuring range center is set to 500 °C.

With a resolution of 0.01 °C, this results in a scalable measuring range from 174.88 °C to 825.11 °C.

At a resolution of 0.01 °C, the scalable measuring range covers 650.23 °C.

Maximum (scalable measuring range):

This value represents the high limit of the scalable measuring range. In the example above, 825.11 °C.

The value is calculated by STEP 7 (at a resolution of 0.001 °C, the high limit is at 532.511 °C, see figure below).

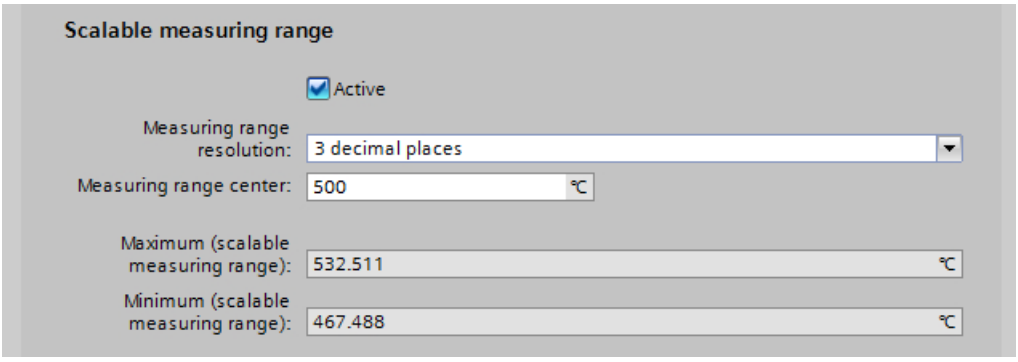
Minimum (scalable measuring range):

This value represents the low limit of the scalable measuring range. In the example above, 174.88 °C.

The value is calculated by STEP 7 (at a resolution of 0.001 °C, the low limit is at 467.488 °C; see figure below).

Higher resolution:

The following figure shows a parameter assignment with a resolution of 0.001 °C (otherwise the same example as in the figure above):



At a resolution of 0.001 °C, the scalable measuring range is between 467.488 and 532.511 °C and covers 65.023 °C (a tenth of the measuring range at a resolution of 0.01 °C).

Standard measuring range with 0.1 °C resolution

The following table shows the standard measuring range for thermal resistors of the type "Pt 100", values in degrees Celsius.

Pt 100 standard in °C (1 digit = 0.1 °C)	Decimal values	Hexadecimal values	Ranges
> 1000.0	32767	7FFF	Overflow
1000.0 : 850.1	10000 : 8501	2710 : 2135	Over range
850.0 : -200.0	8500 : -2000	2134 : F830	Nominal range
-200.1 : -243.0	-2001 : -2430	F82F : F682	Under range
< -243.0	-32768	8000	Underflow

The standard measuring range is the basis for the scalable measuring range.

You can set the measuring range center within the nominal range (-200 °C to 850 °C, see table above).

For temperatures below and above the set measuring range center, you then obtain measured values with a higher resolution.

The width of this range around the measured value center depends on the selected resolution.

Scalable measuring range 0.01 °C and 0.001 °C resolution

The scalable measuring range is identified by the following value ranges:

Scalable measuring range	Measuring range resolution (values in °C)		Hexadecimal values
	2 decimal places	3 decimal places	
Overflow	> 325.11	> 32.511	7FFF
High limit	325.11	32.511	7EFF
Measuring range center	0	0	0
Low limit	325.11	-32.511	8100
Underflow	< -325.11	< -32.511	8000

The maximum and minimum of the scalable measuring range depend on the selected resolution:

- 2 decimal places, resolution of 0.01 °C:
  The high limit is 325.11 °C above the measuring range center you have set.
  The low limit is 325.11 °C below the measuring range center you have set.
  This means that the scalable measuring range is 650.22 °C around the measuring range center.
- 3 decimal places, resolution of 0.001 °C:
  The high limit is 32.511 °C above the measuring range center you have set.
  The low limit is 32.511 °C below the measuring range center you have set.
  This means that the scalable measuring range is 65.022 °C around the measuring range center.

Calculation of the temperature

You calculate the temperature value by adding the value you receive from the module to the measuring range center.

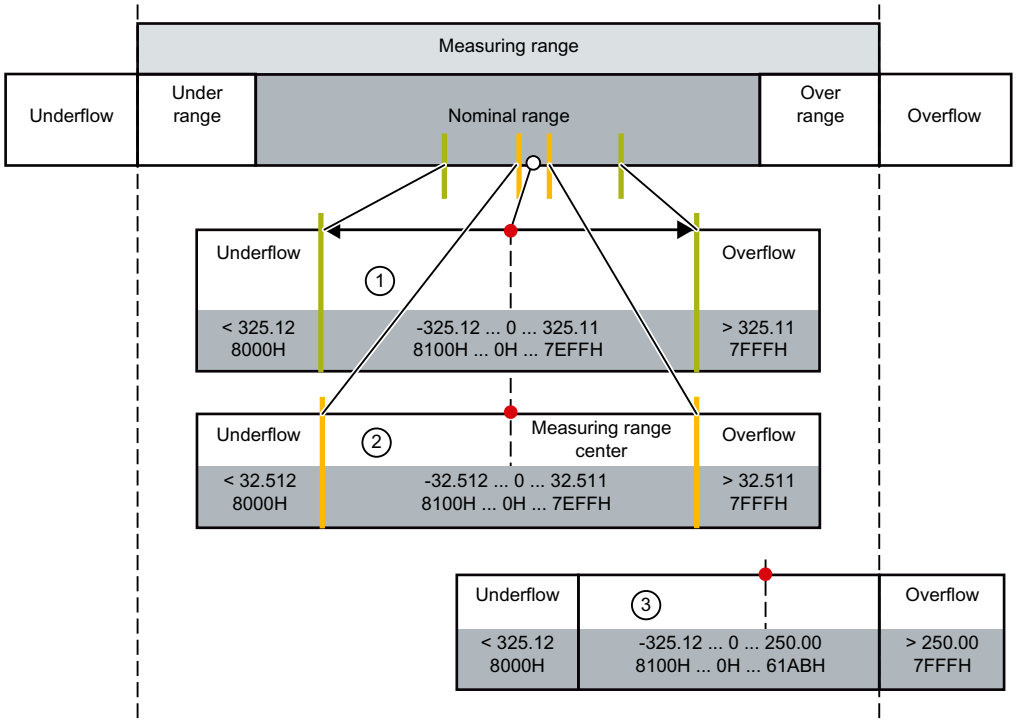
Example:

- You have set the measuring range center 500 °C (see example in the section "Example of a parameter assignment"). For the resolution, you selected "2 decimal places".
- From the module, you receive the hexadecimal value "0100" in S7 format:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

- The hexadecimal value "0100" corresponds to the decimal value 256.
- Since you have selected a resolution of 0.**01** °C, the number 256 corresponds to the temperature value "2.**56** °C".
- You now add 500 °C and 2.56 °C and obtain the measured value 502.56 °C.

Scalable measuring range in the standard measuring range



① Scalable measuring range with 2 decimal places, temperature values in S7 format.

② Scalable measuring range with 3 decimal places, temperature values in S7 format.

③ Scalable measuring range cut off at the overflow of the standard measuring range ("Clipping").

The sum of the measuring range center (for example 750 °C) and the measured value returned by the module must not extend into the overflow of the standard measuring range. For this reason, in the example above, the maximum value that the module can return is limited to 250 °C

Clipping

STEP 7 limits the maximum of the scalable measuring range so that the sum of the measured value center and the maximum measured value that the module can return is not located in the overflow of the standard measuring range. In the same way, STEP 7 restricts the minimum of the scalable measuring range.

See also

Special features of AI 4xRTD/TC 2-/3-/4-wire HF (Page 1300)

ET 200AL

ET 200AL distributed I/O system

SIMATIC ET 200AL

The SIMATIC ET 200AL distributed I/O system is a scalable and highly flexible, distributed I/O system for connecting process signals to a superordinate control with a field bus.

Properties

- Connection to PROFINET, PROFIBUS or integration in ET 200SP
- Up to 32 modules on an ET 200AL
- Integration in ET 200SP: Up to 16 AT modules can be connected to an ET 200AL
- Connection of modules via ET-Connection
- Spatially separated mounting possible
- Module widths of 30 and 45 millimeters
- Degree of protection IP65/IP67
- Suitable for temperatures from -25 to +55 °C and accelerations up to 5 g.
- Installation in all positions
- Color coding of the cables and connections
- CA-compliant labeling of the interfaces
- PROFIenergy integrated
- Configuration control
- Connection of sensors and actuators using M8 and M12 connection system

Area of application

The SIMATIC ET 200AL distributed I/O system is especially well suited for use in tight spaces, moving applications and for assembly and handling technique. Thanks to its scalable construction, you have the option precisely customize its configuration to your on site needs.

The SIMATIC ET 200AL distributed I/O system features protection type IP65/IP67 and is suited for distributed use on a machine or assembly line.

Structure

The SIMATIC ET 200AL distributed I/O system is made up of the following components:

- Interface modules (PROFINET/PROFIBUS)
- Digital and analog I/O modules
- Communications module

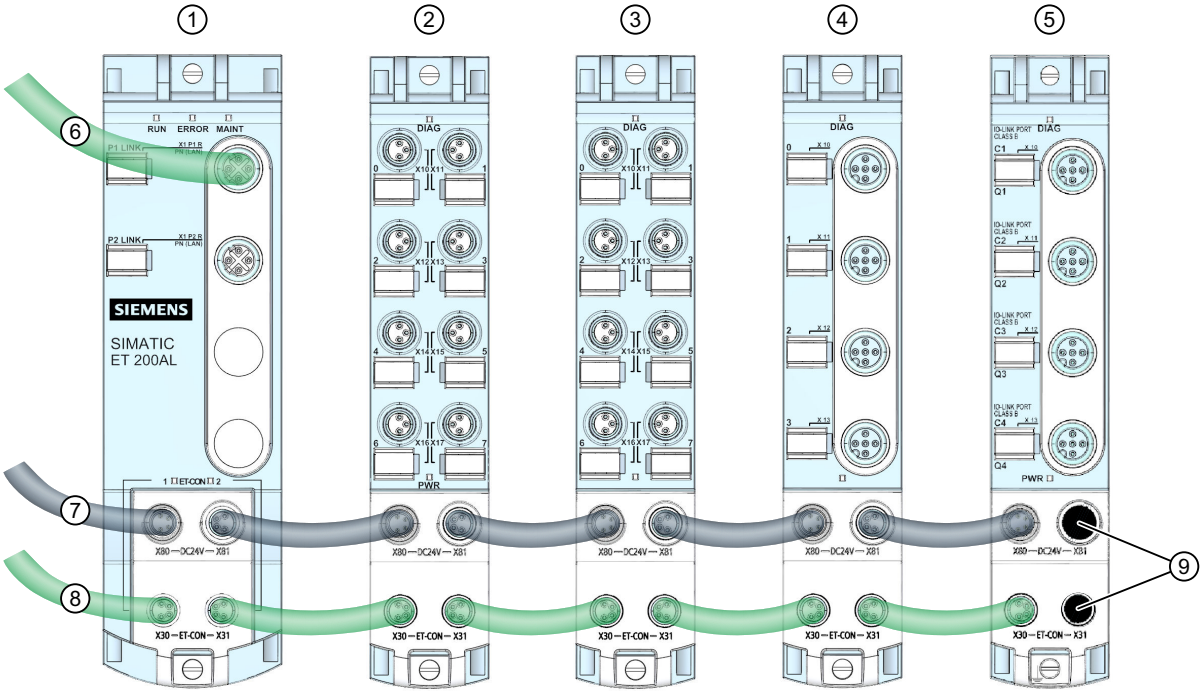
After an interface module you can configure 2 lines (ET-Connection), each with 16 modules.

Alternatively, you can configure a line with 16 I/O modules on the SIMATIC ET 200SP distributed I/O system with BaseUnit BU-Send and the BusAdapter BA-Send 1xFC.

The ET-Connection backplane bus is designed as a cable. This allows you to create spatial distances of up to 10 m between the modules.

Configuration example

The figure below shows a configuration example of the SIMATIC ET 200AL distributed I/O system with a PROFINET interface module.



① Interface module (PROFINET)
② Digital input/Digital output module
③ Digital input module
④ Analog input module
⑤ Communications module
⑥ PROFINET cable
⑦ Power supply cable
⑧ ET-Connection line
⑨ Sealing caps

Figure 10-7 Example configuration of the ET 200AL

Configuring ET 200AL

Introduction

ET 200AL is a distributed I/O system with an IP65/67 degree of protection. It is therefore designed for use on site, for example right beside a machine (no control cabinet required).

The system includes interface and communications modules as well as input and output modules.

There are two use cases for the ET 200AL:

1. As I/O device or DP slave: The system interface module is connected to a field bus (PROFINET or PROFIBUS) and connected to the PN or DP interface of a CPU.
2. As an ET 200SP expansion: The ET 200AL modules are connected to the ET 200SP over the "BA Send 1xFC" module ("mixed mode").

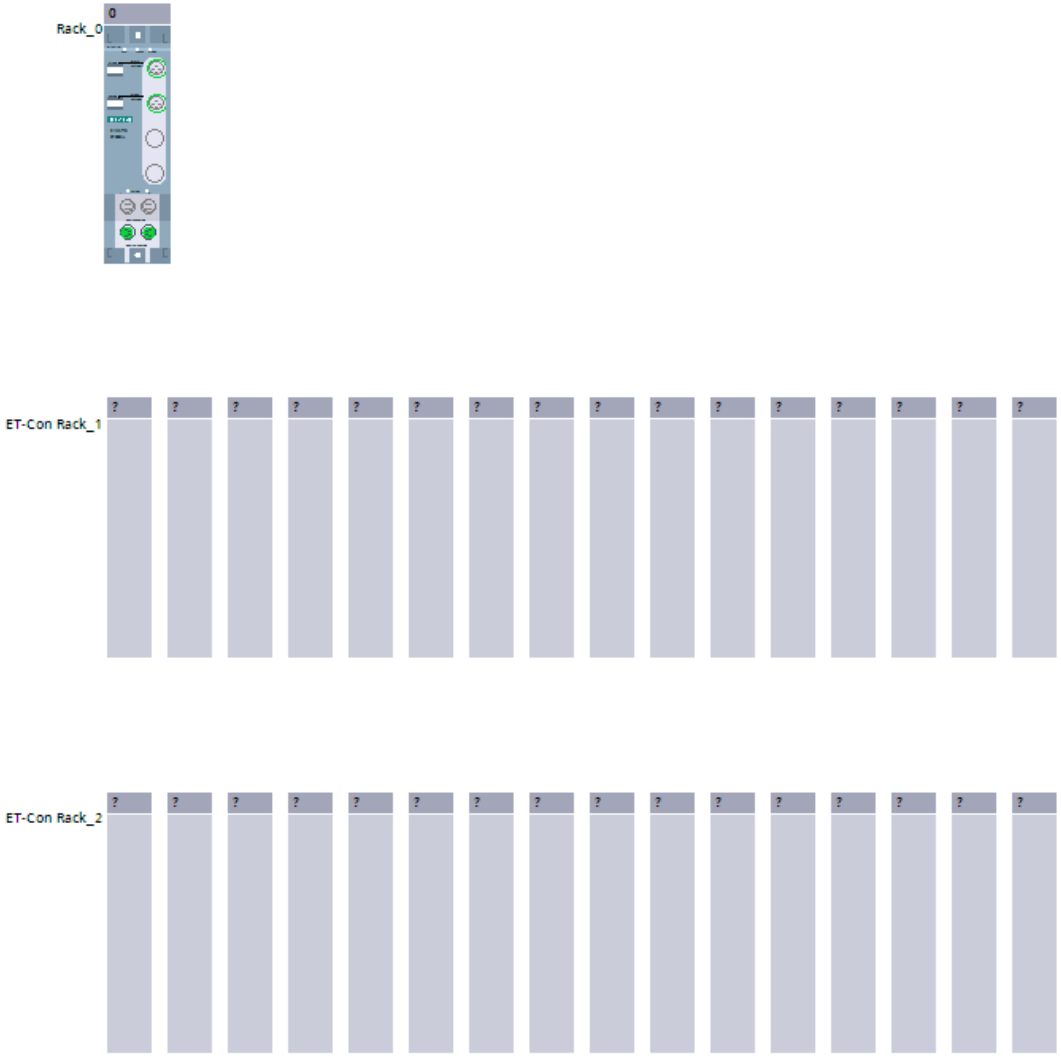
The following sections describe how to configure an ET 200AL as I/O device or DP slave (use case 1).

For use case 2, see the link under "See also".

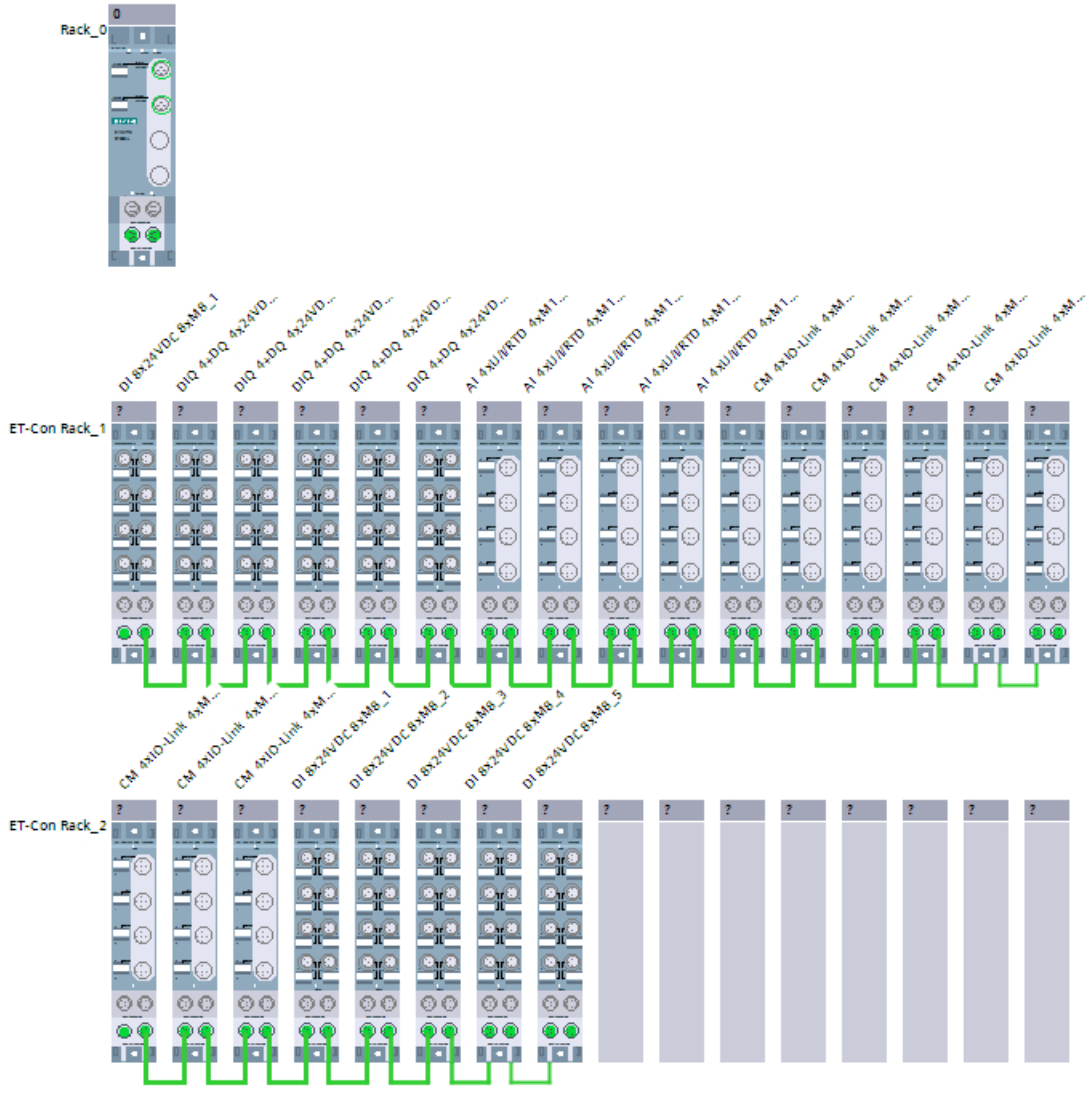
Procedure

Follow the steps below to configure an ET 200AL in STEP 7:

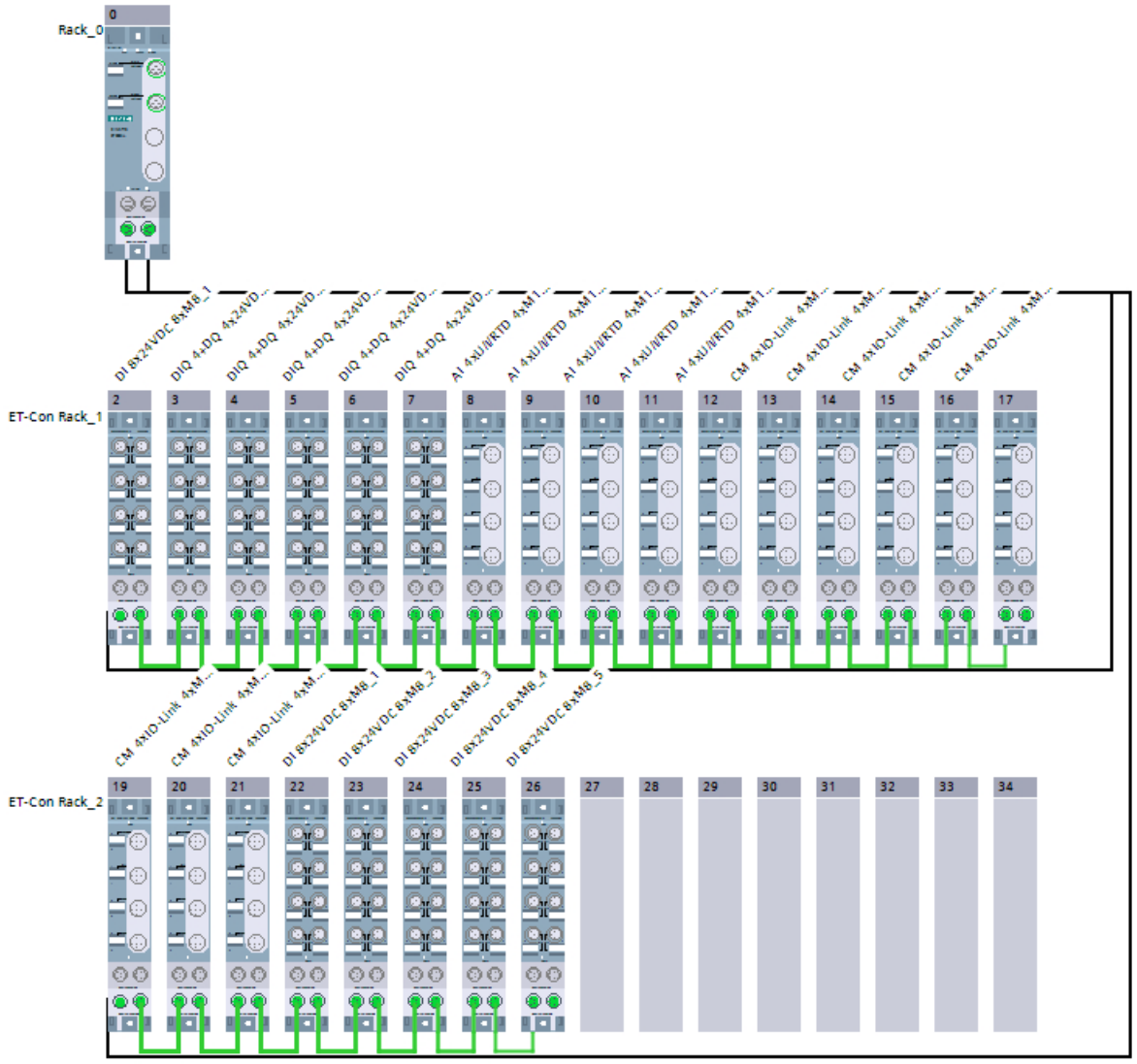
1. Copy an interface module (PROFINET or PROFIBUS) from the ET 200SP series to the network view (drag-and-drop operation from the hardware catalog).
2. Go to the device view. Double-click on the interface module you have just inserted.
   The interface module and two ET-Connection racks are displayed in the device view (figure below).
   There are no slot numbers assigned yet, which is why question marks are displayed above the slots.



3. Now select the modules (input, output and communication modules) from the hardware catalog (ET 200AL folder) and drag them to the free slots (outlined in blue; not shown). You can place up to 16 ET 200AL modules in each ET-Connection rack (figure below). An ET-Connection rack is a virtual rack that sets the order of the connected ET 200AL modules.



4. Now connect the interface module to the two ET-Connection racks.
   First click on an ET-Connection interface of the interface module, then hold and drag a line to the left-hand ET connection of the first module in one of the two ET-Connection racks.
   Repeat this step for the second ET-Connection interface of the interface module and the second ET-Connection rack (if used).



5. Double-click on a module to access the module properties and set the module parameters.

Rules

- The ET 200AL modules must be configured with no gaps.
- The first module in an ET-Connection rack must be connected to the interface module.

See also

ET 200AL distributed I/O system (Page 1309)

Configuration control with ET 200AL

Operating principle

With configuration control, you can change the original configuration of an ET 200AL (created by configuring with STEP 7) with a user program and operate the ET 200AL in this modified configuration. STEP 7 is no longer required for this configuration: You use your user program to signal to the ET 200AL the slot in which a configured module is actually inserted.

You use control data record 196 for this. In this data record, you code which modules are missing or located in different slots in the real configuration compared to the configuration with STEP 7. The configuration control has no effect on the parameter assignment of the modules (for example, the enabling of diagnostic alarms).

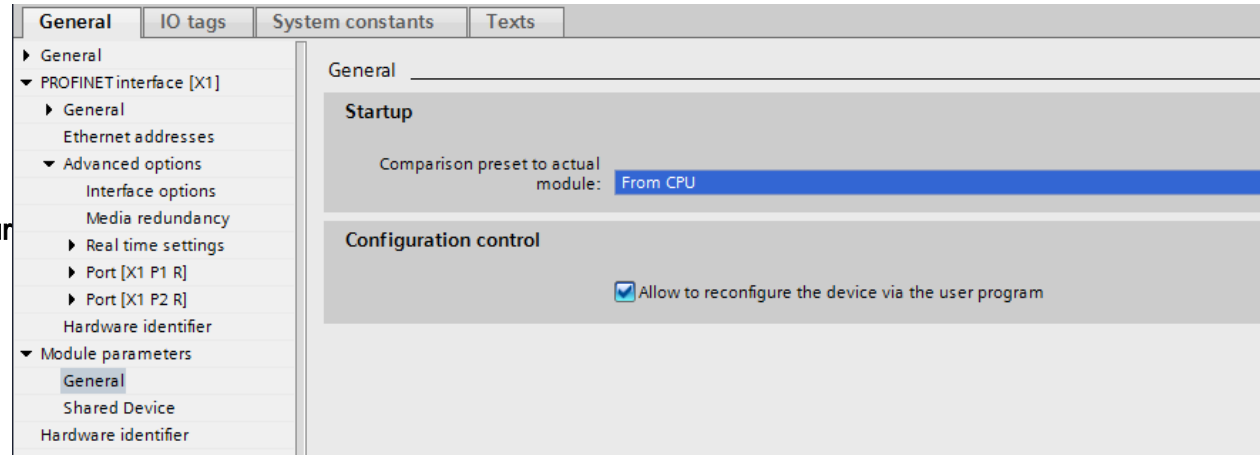
You then call the "WRREC" instruction and use it to write the data record to the interface module of the ET 200AL.

Configuration control gives you the flexibility to vary the configuration of an ET 200AL as long as the real configuration can be derived from a preset maximum configuration (originally created with STEP 7).

The following sections describe how to enable configuration control and how to structure the required data record 196 for the ET 200AL.

Requirement

Enabling configur



Structure of control data record 196

The configuration of the data block corresponds to the original configuration of the ET 200AL with STEP 7.

There are two bytes in the data record for each module. The position of these two bytes in the data record codes a module in the original configuration with STEP 7.

- Bytes 4 and 5 in the data record correspond to the module in slot 1 in the original configuration.
- Bytes 6 and 7 in the data record correspond to the module in slot 2 in the original configuration.
- Bytes 8 and 9 in the data record correspond to the module in slot 3 in the original configuration.
- etc.

The current (real) slot is coded by the number that is assigned to the "Slot_x" byte (by its value): Examples:

- The value "2" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 2 in the current configuration.
- The value "3" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 3 in the current configuration.
- The value "4" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 4 in the current configuration.
- etc.

Creating control data record 196

The figure below shows part of control data record 196 for configuration of an ET 200AL.

This configuration is given as an example:

- ET-Con1 is inserted in "slot_1" (fixed setting). The two ET-Connection submodules "ET-Con1" and "ET-Con2" are submodules of the interface module of the ET 200AL. They are integrated as fixed modules in the IM module. They cannot be inserted individually.
- ET-Con2 is inserted in "slot_18" (fixed setting).
- In this configuration, 16 AL modules are connected to ET-Con1 ("slot_2" to "slot_17" in the following data record). This is the maximum configuration.
- In this configuration, one AL module is connected to ET-Con2 ("slot_19"). However, a total of 16 AL electronic modules could be connected to ET-Con2 (as to ET-Con1).

The ET 200AL originally configured with STEP 7 is now to be reconfigured from the user program.

The new configuration has the following properties:

- ET-Con1 is inserted in "slot_1" (fixed setting).
- Module 2 is also operated in slot 2 in the modified configuration.
- Module 3 is not used.
- Module 4 is now inserted in slot 3.
- Module 5 is now inserted in slot 4.
- No other modules at ET-Con1 are used.

- ET-Con2 is inserted in "slot_18" (fixed setting)
- The module at ET-Con2 is used.

Record_196_ET_200AL

	Name	Data type	Default value
1	block_length	USInt	42
2	block_ID	USInt	196
3	version	USInt	2
4	subversion	USInt	1
5	slot_1_ET_Con1	USInt	1
6	reserve_1_ET_Con1	USInt	0
7	slot_2	USInt	2
8	reserve_2	USInt	0
9	slot_3	USInt	0
10	reserve_3	USInt	0
11	slot_4	USInt	3
12	reserve_4	USInt	0
13	slot_5	USInt	4
14	reserve_5	USInt	0
15	slot_6	USInt	0
16	reserve_6	USInt	0
17	slot_7	USInt	0
18	reserve_7	USInt	0
19	slot_8	USInt	0
20	reserve_8	USInt	0
21	slot_9	USInt	0
22	reserve_9	USInt	0
23	slot_10	USInt	0
24	reserve_10	USInt	0
25	slot_11	USInt	0
26	reserve_11	USInt	0
27	slot_12	USInt	0
28	reserve_12	USInt	0
29	slot_13	USInt	0
30	reserve_13	USInt	0
31	slot_14	USInt	0
32	reserve_14	USInt	0
33	slot_15	USInt	0
34	reserve_15	USInt	0
35	slot_16	USInt	0
36	reserve_16	USInt	0
37	slot_17	USInt	0
38	reserve_17	USInt	0
39	slot_18 ET_Con2	USInt	1
40	reserve_18_ET_Con2	USInt	0
41	slot_19	USInt	1
42	reserve_19	USInt	0

The components of control data record 196 (definition in section "Control data record 196" below):

- block_length: Note the length of the control data record here; in the example: 42 (bytes). The length of the control data block is calculated using the following formula: 2 x "number of modules" + 4.
- block_ID: Enter the figure 196 here. This number identifies the data record as the data record for configuration control.
- version: The ET 200AL uses version 2 of control data record 196.
- subversion: The ET 200AL uses subversion 1 of control data record 196.
- slot_1: The ET-Connection 1 submodule is always inserted in slot 1 of the ET 200AL.
- reserve_1: This byte is not used (value "0").
- slot_2: The configured module 2 is inserted in slot 2 (value "2").
- reserve_2: This byte is not used (value "0").
- slot_3: The configured module 3 is not present in the current configuration (value "0").
- reserve_3: This byte is not used (value "0").
- slot_4: The configured module 4 is inserted in slot 3 in the current configuration (value "3").
- reserve_4: This byte is not used (value "0").
- slot_5: The configured module 5 is inserted in slot 4 in the current configuration (value "4").
- reserve_5: This byte is not used (value "0").
- slot_6: The configured module 6 is not present in the current configuration (value "0").
- reserve_6: This byte is not used (value "0").
- slot_7: The configured module 7 is not present in the current configuration (value "0").
- reserve_7: This byte is not used (value "0").
- etc.
- slot_18: The ET-Connection 2 submodule is always inserted in slot 18 of the ET 200AL (value "18").
- reserve_18: This byte is not used (value "0").
- slot_19: The configured module 19 is inserted in slot 19 in the current configuration (value "19").
- reserve_19: This byte is not used. (value "0")

Control data record 196 for ET 200AL

A control data record 196 containing a slot assignment is defined for configuration control.

Byte	Element	Value	Description	
0	Block length in bytes	e.g. 14 for ET 200AL with 5 modules	The length of the data record is calculated using the formula: 4 + (number of modules x 2) bytes	Header
1	Block ID	196	ID for control data record 196	
2	Version	2	Version 2 of control data record 196	
3	Subversion	1	Subversion 1 of control data record 196	
4	Configured module 1 (ET-Connection 1)	1	ET-Connection 1 is always assigned to slot 1 (fixed). The value "1" must therefore always be entered in byte 4.	Assignment for configured module 1 (ET-Connection 1) to real slot 1
5	Reserve for configured module 1	0	Not used	
6	Configured module 2	Real slot of module 2 Possible values: 2 up to the number of modules (except 18) 0 (if the configured module 2 is not present)	The configured module 2 can be inserted in any slot from 2 to slot 34. Slot 18 is reserved for ET-Con 2. If the configured module is not used, this byte contains the value "0".	Assignment for the configured module 2 to a real slot
7	Reserve for configured module 2	0	Not used	
8	Configured module 3	Real slot of module 3 Possible values: 2 up to the number of modules (except 18) 0 (if the configured module 2 is not present)	The configured module 3 can be inserted in any slot from 2 to slot 34. Slot 18 is reserved for ET-Connection 2. If the configured module is not used, this byte contains the value "0".	Assignment for the configured module 3 to a real slot
9	Reserve for configured module 3	0	Not used	
:	:	:	:	:
39	Configured module 18 (ET-Connection 2)	18	ET-Connection 2 is always assigned to slot 18 if AT modules are connected at this submodule.	Assignment for configured module 18 (ET-Connection 2) to real slot 18
40	Reserve for configured module 18	0	Not used	
:	:	:	:	:

(Bytes 4 to 70, not byte 39)	Configured module x	Real slot of module x Possible values: 2 up to the number of modules (except 18) 0 (if there is no configured module x)	The configured module x can be inserted in any real slot from 2 to slot 34. Slot 18 is reserved for ET-Con 2 (bytes 39 and 40 in the control data record)	Assignment for configured module x to a real slot y
(Bytes 5 to 71, not byte 40)	Reserve for configured module x	0	Not used	

Rules

- The ET-Connection 1 and ET-Connection 2 submodules must be treated like real modules in configuration control. Restriction: ET-Connection 1 is always placed in slot 1 and ET-Connection 2 is always placed in slot 18 (fixed assignment).
- There are no reserve modules for the ET 200AL (unlike for the ET 200S or with BU cover modules for the ET 200SP). For this reason, bit 7 of "slot_x" must not be set (i.e. only the values 0 to 127 may be used).
- The value "0" of "slot_x" indicates that this module is not inserted in the current configuration.
- Gaps must not be left between the AL modules when configuring with STEP 7.
- If no modules are connected to ET-Con2 when STEP 7 is configured, ET-Con2 is not configured: This shortens data record 196.
- If fewer than 16 modules are connected to ET-Con1 during configuration with STEP 7 and there are also modules connected to ET-Con2, control data record 196 must contain all unassigned slots for ET-Con 1. They are assigned zero as the value for the real slot.

Writing a data record

Transfer the control data record to the ET 200AL module.

To do so, call the extended WRREC (Write data record) instruction, and transfer the control data record created.

If you do not transfer a control data record, the interface module uses the original configuration created with STEP 7. In this case, the following applies: Configured module x is inserted in real slot x.

Addressing the interface module using the HW identifier

To transfer data record 196 with the instruction WRREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Error messages

The following error messages are returned if an error occurs when writing control data record 196:

Table 10-99 Error messages

Error code	Meaning
16#80A2	DP protocol error on layer 2. Indicates that a data record has not been acknowledged due to the system.
16#80B1	Invalid length; the length information in data record 196 is not correct.
16#80B5	Configuration control parameters not assigned.
16#80B2	Invalid slot: The configured slot is not assigned.
16#80B8	Parameter error; module signals invalid parameters.
16#80C5	DP slave or module not available. Indicates that a data record has not been acknowledged due to the system.

Readback data record 197 for ET 200AL

Readback data record 197 is used to read the actual configuration of a station (in this case of an ET 200AL).

This data record allows you to check the real configuration of the ET 200AL (actual configuration). The readback data record for each configured module specifies whether or not it is actually available.

- The value "1" means that the correct module is inserted in the correct slot.
- The value "0" codes all other options (wrong module, empty slot, BU cover).

Example:

A module has been configured with STEP 7 for slot 4.

This module has then been moved to slot 3 in the current configuration using data record 196.

If this module is also really in slot 3, this is coded by the value "1" (status_slot_4 = 1).

Configuration details:

The configuration of the data block corresponds to the original configuration of the ET 200AL with STEP 7.

There are two bytes in the data record for each module. The position of these two bytes in the data record corresponds to the position of a module in the original configuration with STEP 7.

Sequence of bytes:

- "status_slot_1_ET_Con1" and "reserve_slot_1_ET-Con1" (bytes 4 and 5 in the data record) correspond to the module in slot 1 in the configuration,
- "status_slot_2" and "reserve_slot_2" (bytes 6 and 7) correspond to the module in slot 2 in the configuration
- "status_slot_3" and "reserve_slot_3" (bytes 8 and 9) correspond to the module in slot 3 in the configuration,
- etc.

Example

The following readback data record 197 is returned by the ET 200AL that was reconfigured with control data record 196 in the example above (section "Creating control data record 196").

Record_197_ET_200AL

	Name	Data type	Default value
1	block_length	USInt	42
2	block_ID	USInt	197
3	version	USInt	2
4	subversion	USInt	1
5	status_slot_1_ET_Con1	USInt	1
6	reserve_1_ET_Con1	USInt	0
7	status_slot_2	USInt	1
8	reserve_2	USInt	0
9	status_slot_3	USInt	0
10	reserve_3	USInt	0
11	status_slot_4	USInt	1
12	reserve_4	USInt	0
13	status_slot_5	USInt	1
14	reserve_5	USInt	0
15	status_slot_6	USInt	0
16	reserve_6	USInt	0
17	status_slot_7	USInt	0
18	reserve_7	USInt	0
19	status_slot_8	USInt	0
20	reserve_8	USInt	0
21	status_slot_9	USInt	0
22	reserve_9	USInt	0
23	status_slot_10	USInt	0
24	reserve_10	USInt	0
25	status_slot_11	USInt	0
26	reserve_11	USInt	0
27	status_slot_12	USInt	0
28	reserve_12	USInt	0
29	status_slot_13	USInt	0
30	reserve_13	USInt	0
31	status_slot_14	USInt	0
32	reserve_14	USInt	0
33	status_slot_15	USInt	0
34	reserve_15	USInt	0
35	status_slot_16	USInt	0
36	reserve_16	USInt	0
37	status_slot_17	USInt	0
38	reserve_17	USInt	0
39	status_slot_18 ET_Con2	USInt	1
40	reserve_18_ET_Con2	USInt	0
41	status_slot_19	USInt	1
42	reserve_19	USInt	0

Modules 2, 4 and 5 are actually connected to ET-Con1.

None of the other modules that were connected to ET-Con1 in the configuration with STEP 7 are present in the current configuration (in line with the control data record 196 settings from the example above).

A module is really connected to ET-Con2 as in the original configuration with STEP 7.

Reading readback data record 197

You can read readback data record 197 from the ET 200AL with the instruction RDREC. RDREC operates asynchronously. If you call RDREC in the startup OB, you must call the instruction multiple times using a loop until the "BUSY" or "DONE" output parameter indicates that the data record has been read.

To read data record 197 with the instruction RDREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Further information and examples

Further information on ET 200AL is available here (http://support.automation.siemens.com/WW/view/en/89254863).

Specific examples of configuration control can be found here in this application description (http://support.automation.siemens.com/WW/view/en/29430270).

See also

ET 200AL distributed I/O system (Page 1309)

Expanding ET 200SP with ET 200AL modules (Page 1268)

Configuration control with ET 200SP (Page 1274)

ET 200MP

ET 200MP distributed I/O system

Definition

The ET 200MP distributed I/O system is a scalable and flexible distributed I/O system for connection of process signals to a central controller via a field bus.

Application area

The ET 200MP is a multi-functional distributed I/O system for various fields of application. The scalable design allows you to configure the system exactly to the specific requirements on location.

The ET 200MP complies with IP 20 degree of protection and is intended for installation in a control cabinet.

Structure

The ET 200MP is installed on a mounting rail and comprises:

- An interface module that communicates with all IO controllers conforming to the PROFINET standard IEC 61158
- Up to 30 modules (power supply modules and I/O modules from the S7-1500 I/O range) can be inserted to the right of the interface module.
- If you insert a power supply module to the left of the interface module, this yields a possible maximum configuration of 32 modules in total.
- The number of insertable I/O modules is limited by their power requirements.

Slot rules

- Slot 0: Power supply module (optional)
- Slot 1: Interface module
- Slot 2 to 31: I/O modules or power supply modules

Interface module parameters

Supply voltage L+ connected

Parameter "Supply voltage L+ connected"

This parameter influences the diagnostics and the checking of the power budget.

- Diagnostics of the ET 200MP:
  If the actual configuration does not match the preset configuration with regard to the supply voltage of the interface module, the interface module generates a diagnostic alarm.
  Example: You have deactivated the "Supply voltage L+ connected" option, but you have connected the supply voltage in the actual configuration.
- Power budget check during configuration:
  The power budget changes in accordance with the parameter setting: Either the interface module feeds power into the backplane bus or it draws power from the backplane bus.

The default ("Supply voltage L+ connected" option is **activated**) means that the front of the interface module is supplied with 24 V DC and the power is stored in the backplane bus.

If the "Supply voltage L+ connected" option is **deactivated**, the interface module may not be supplied with 24 V DC on the front.

In this case, insert a power supply unit (PS) on the left next to the interface module that supplies the interface module and the modules to the right of the interface module.

Note

We recommend that you always supply the interface module on the front side with 24 V DC. If a system power supply unit (PS) is inserted and connected additionally **before** or on the left next to the interface module, both the power from the system power supply unit (PS) as well as the power from the integrated power supply of the interface module are then available to the configuration.

In this case, you do not have to change the default setting of the parameter.

Configuration control with ET 200MP

Operating principle

Configuration control allows you to operate various real configurations (options) with a single configuration of the ET 200MP distributed I/O device.

Configuration control provides you with the option of configuring the ET 200MP distributed I/O device with its maximum configuration and still operating it with modules missing. If missing modules are retrofitted later, no new configuration is required and the hardware configuration does not have to be reloaded either.

Using control data record 196, which is transferred to the interface module in the user program, you define a current configuration. You transfer the control data record with the instruction WRREC.

Readback data record 197 is used to read the actual configuration of an ET 200MP.

Requirements

- STEP 7 Professional version V13 SP1 or higher
- The CPU startup parameter "Compare preset to actual configuration" is set to Startup even if mismatch (default setting). This setting is also selected for the startup parameters of the individual modules of the ET 200MP.

Enabling configuration control

In the properties of the interface module under Module parameters > General > Configuration control, select "Enable reconfiguration of device via user program". This activates configuration control.

Control data record 196 for ET 200MP

The figure below shows the start of control data record 196 for the configuration control of an ET 200MP.

The data block is 36 bytes long (maximum configuration with 32 modules). The value "36" therefore appears in the "block_length" element of the data record.

If you configure an ET 200MP in STEP 7 with fewer modules, the data block will be shorter: If there are only five modules, for example, the data record is reduced to 9 bytes (4 bytes for the header plus one byte for each module).

There is one byte in the data record for each module. The position of this byte in the data record codes a module in the original configuration with STEP 7:

- "slot_0 power supply" (byte 4 in the data record below) corresponds to the power supply module in slot 0 in the configuration with STEP 7.
- "slot_1 interface module" (byte 5 in the data record) corresponds to the interface module in slot 1 in the configuration.
- "slot_2" (byte 6 in the data record) corresponds to the module in slot 2 in the configuration.
- "slot_3" (byte 7 in the data record) corresponds to the module in slot 3 in the configuration.
- "slot_4" (byte 8) corresponds to the module in slot 4 in the configuration.
- etc.

Value in slot_x

The current slot is coded by the figure that is assigned to "slot_x" (by its value). Examples:

- The value "2" in slot_2 means you are assigning the module originally inserted in slot 2 to slot 2 in the current configuration (slot_2 = 2).
- The value "3" in slot_2 means you are assigning the module originally inserted in slot 2 to slot 3 in the current configuration (slot_2 = 3).
- The value "4" in slot_2 means you are assigning the module originally inserted in slot 2 to slot 4 in the current configuration (slot_2 = 4).
- etc.

Example for data record 196

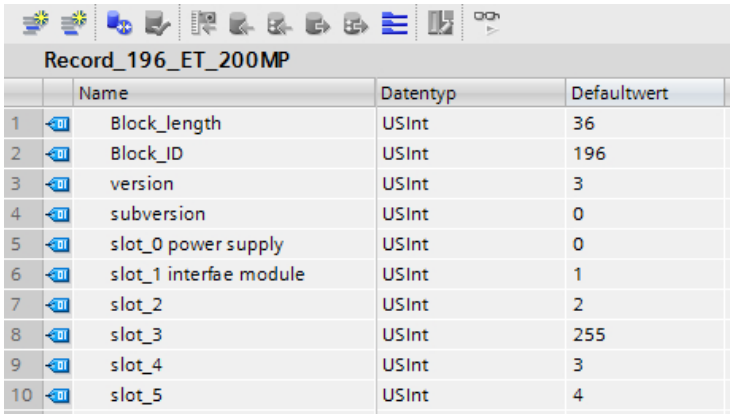
The following data record was created for a configuration that changes the original configuration with STEP 7.

The modified configuration has the following properties:

- The module inserted in slot 0 in the configuration (power supply module) is also inserted in slot 0 in the current configuration (specification).
- The module inserted in slot 1 in the configuration (interface module) is also inserted in slot 1 in the current configuration (specification).
- The module inserted in slot 2 in the configuration (module 2) is also inserted in slot 2 in the current configuration.
- The module inserted in slot 3 in the configuration (module 3) does not exist in the current configuration.
- The module inserted in slot 4 in the configuration (module 4) is inserted in slot 3 in the current configuration.

- The module inserted in slot 5 in the configuration (module 5) is inserted in slot 4 in the current configuration.
- etc.

The bytes "slot_6" to "slot_31" are not shown in the figure below.



Record_196_ET_200MP

	Name	Datentyp	Defaultwert
1	Block_length	USInt	36
2	Block_ID	USInt	196
3	version	USInt	3
4	subversion	USInt	0
5	slot_0 power supply	USInt	0
6	slot_1 interfae module	USInt	1
7	slot_2	USInt	2
8	slot_3	USInt	255
9	slot_4	USInt	3
10	slot_5	USInt	4

Rules

- If a module is not available in the current configuration, this is indicated by the value 255: "slot_x" = 255
- The power supply module is always in slot 0 ("slot_0 power supply" = 0).
- The interface module is always in slot 1 ("slot_1 interface module" = 1).

Addressing the interface module using the HW identifier

To transfer data record 196 with the instruction WRREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Readback data record 197 for ET 200MP

Readback data record 197 is used to read the actual configuration of a station (in this case, of an ET 200MP).

This data record allows you to check the real configuration of the ET 200MP (actual configuration). The readback data record for each configured module specifies whether or not it is actually available.

- The value "1" means that the correct module is inserted in the correct slot.
- The value "0" codes all other options (wrong module, empty slot, reserve module).

Example:

A module has been configured with STEP 7 for slot 4.

This module has then been moved to slot 3 in the current configuration using data record 196.

If this module is also really in slot 3, this is coded by the value "1" (status_slot_4 = 1).

Configuration details:

The configuration of the data block corresponds to the original configuration of the ET 200MP with STEP 7.

There is a byte in the data record for each module. The position of this bytes in the data record corresponds to the position of a module in the original configuration with STEP 7.

Sequence of bytes:

- "status_slot_0 power supply" (byte 4 in the data record below) corresponds to the power supply module in slot 0 in the configuration with STEP 7.
- "status_slot_1 interface module" (byte 5 in the data record) corresponds to the interface module in slot 1 in the configuration.
- "status_slot_2" (byte 6) corresponds to the module in slot 2 in the configuration.
- "status_slot_3" (byte 7) corresponds to the module in slot 3 in the configuration.
- etc.

You can choose any name for the components (for example "status_slot_2").

Meaning of "status_slot_x":

- The value "1" in status_slot_x means that module x is in the correct slot.
- The value "0" in status_slot_x codes all other options (wrong module, module does not exist).

Example:

The figure below shows readback data record 197 for the configuration of an ET 200MP in which there is no module 3 (the module in slot 3 in the configuration).

All other modules are available and correctly plugged.

The bytes "status_slot_6" to "status_slot_31" are not shown in the figure below.

Record_197_ET_200MP

	Name	Datentyp	Defaultwert
1	Block_length	USInt	36
2	Block_ID	USInt	197
3	version	USInt	3
4	subversion	USInt	0
5	status_slot_0 power supply	USInt	1
6	status_slot_1 interfae module	USInt	1
7	status_slot_2	USInt	1
8	status_slot_3	USInt	0
9	status_slot_4	USInt	1
10	status_slot_5	USInt	1

Reading readback data record 197

You can read readback data record 197 from the ET 200MP with the instruction RDREC. RDREC operates asynchronously. If you call RDREC in the startup OB, you must call the instruction multiple times using a loop until the "BUSY" or "DONE" output parameter indicates that the data record has been read.

To read data record 197 with the instruction RDREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Further information and examples

Further information on ET 200MP can be found in the manual for IM 155-5 PN (http://support.automation.siemens.com/WW/view/en/89261636).

Specific examples of configuration control can be found in this application description (http://support.automation.siemens.com/WW/view/en/29430270).

See also

Documentation on configuration control (http://support.automation.siemens.com/WW/view/en/67295970)

Input module parameters

Parameters of the analog input modules

Missing supply voltage L+

Enabling of the diagnostics for missing or insufficient supply voltage L+.

Wire break

Enabling of the diagnostics if the module has no current flow or the current is too weak for the measurement at the corresponding configured input or the applied voltage is too low.

Current limit for wire break diagnostics

Threshold at which a wire break is reported. The value can be set to 1.185 mA or 3.6 mA, depending on the sensor used.

Overflow

Enabling of the diagnostics if the measured value exceeds the overrange.

Underflow

Enabling of the diagnostics if the measured value undershoots the underrange.

Common mode error

Enable diagnostics if the valid common mode voltage is exceeded.

Reference channel error (only for AI 8xU/I/RTD/TC ST)

- Enable diagnostics on error at the temperature compensation channel, e.g. wire break.
- Dynamic reference temperature compensation type is configured and no reference temperature has been transferred to the module yet.

Temperature coefficient

The temperature coefficient depends on the chemical composition of the material. In Europe, only one value is used per sensor type (default value).

The correction factor for the temperature coefficient (α value) specifies how much the resistance of a certain material changes when the temperature is raised by 1 °C.

The further values facilitate a sensor-specific setting of the temperature coefficient and enhance accuracy.

Interference frequency suppression

At analog input modules, this suppresses interference caused by the frequency of AC mains.

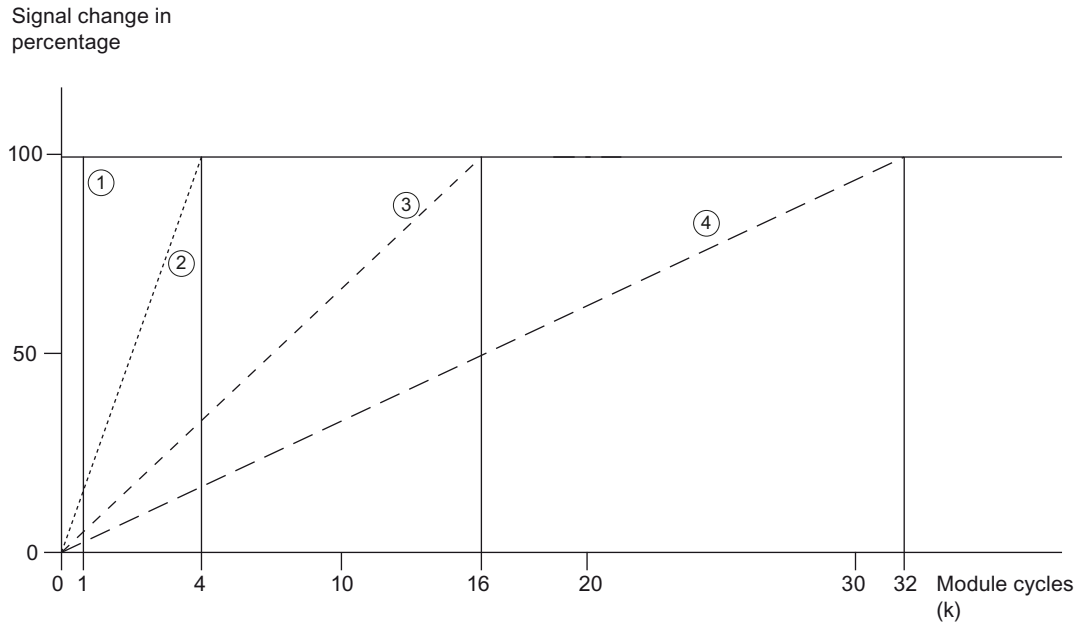
The frequency of the AC network may interfere with measured values, particularly for measurements within low voltage ranges and when thermocouples are being used. With this parameter, you define the mains frequency in your system.

Smoothing

The individual measured values are smoothed using filtering. Smoothing can be set in 4 stages for the analog input modules AI 8xU/I/RTD/TC ST and AI 8xU/I HS.

Smoothing time = number of module cycles (k) x cycle time of the module.

The figure below shows the number of module cycles after which the smoothed analog value is almost 100%, depending on the set smoothing. Is valid for each signal change at the analog input.



① None (k = 1)
② Weak (k = 4)
③ Medium (k = 16)
④ Strong (k = 32)

Reference junction (only for AI 8xU/I/RTD/TC ST)

The following settings can be configured for the reference junction parameter:

Table 10-100 Possible parameter assignments for the reference junction parameter

Setting	Description
Fixed reference temperature	The reference junction temperature is configured and stored in the module as a fixed value.
Dynamic reference temperature	The reference junction temperature is transferred in the user program from the CPU to the module by data records 192 to 199 using the WRREC (SFB 53) instruction.
Internal reference junction	The reference junction temperature is determined using an integrated sensor of the module.
Reference channel of the module	The reference junction temperature is determined using an external resistance thermometer (RTD) at the reference channel (COMP) of the module.

Note

Fixed reference temperature

During parameter assignment of a thermocouple Type B, only the setting "Fixed reference temperature" with a temperature of 0 °C is possible.

Enable hardware interrupt 1 or 2

Enable a hardware interrupt if high limit 1 or 2 is exceeded, or low limit 1 or 2 is violated.

Low limit 1 or 2

Specifies the low limit threshold that triggers hardware interrupt 1 or 2.

High limit 1 or 2

Specifies the high limit threshold that triggers hardware interrupt 1 or 2.

Temperature compensation for thermocouples

Introduction

You have several options of measuring the reference junction temperature in order to obtain an absolute temperature value as a function of the temperature difference between the reference junction and the measuring point.

You can use various compensation options depending on the required location of the reference junction.

Note

During parameter assignment of a thermocouple Type B, only the setting "Fixed reference temperature" with a temperature of 0 °C is possible.

Options of compensating for the reference junction temperature

Compensation options	Explanation	Application case
Internal reference junction	With this compensation, the reference junction temperature is determined using an integrated sensor of the module. **Procedure** Connect the thermocouple to the I/O module directly or with compensating lines.	• For the connection, you use compensating lines matching the thermocouple material. • If the reference junction temperature and the module temperature are identical in your system, you may also use lines made from a different material.
Reference channel of the module	The reference junction temperature is determined using an external resistance thermometer (RTD). **Procedure** Connect the thermocouple to the supply lines at the reference junction, either directly or with compensating lines. You connect the supply lines to the appropriate terminals of the module. Connect the resistance thermometer (RTD) to the reference channel of the module. The resistance thermometer (RTD) must be placed in the area of the reference junction.	• You want to measure the temperature directly at the reference junction. • The measured temperatures of all channels that you have configured for this compensation type is corrected automatically by the temperature value of the reference junction. • You can use inexpensive lines, e.g., copper lines, from the reference junction to the module.
Dynamic reference temperature	The temperature of the reference junction is determined via a module. This temperature value is transferred to other modules via a data record in the user program. **Procedure** Connect the resistance thermometer (RTD) for the reference junction to any channel. The reference junction temperature is communicated from the CPU to the module by data records 192 to 199 using the WRREC instruction.	• You use multiple modules at the reference junction and can therefore compensate all channels using a common temperature value. • You require only one resistance thermometer (RTD) to acquire the temperature value. • You can use inexpensive lines, e.g., copper lines, from the reference junction to the module.
Fixed reference temperature	The reference junction temperature is stored in the module as a fixed value. **Procedure** Connect the thermocouple to the supply lines at the reference junction, either directly or with compensating lines. You connect the supply lines to the appropriate terminals of the module. When configuring the module, specify a fixed temperature value for the reference junction (e.g. 20 °C).	• You keep the reference junction temperature constant and know the temperature value. • You can use inexpensive lines, e.g., copper lines, from the reference junction to the module.

Output module parameters

Parameters of the analog output modules

Missing supply voltage L+

Enabling of the diagnostics, with missing or too little supply voltage L+.

Short-circuit to ground

Enabling of the diagnostics if a short-circuit of the actuator supply to ground occurs.

Wire break

Enabling diagnostics if the line to the encoder is interrupted.

Overflow

Enabling of the diagnostics if the measured value exceeds the overflow range.

Underflow

Enabling of the diagnostics if the measured value falls below the underflow range.

Reaction to CPU STOP

Determines the reaction of the output to the CPU going into STOP state.

Substitute value

The substitute values are values that the outputs (the output) issue in the event of a CPU STOP.

ET 200M

Configuring an ET 200M

Introduction

For the ET 200M series, you can find a wide range of modules in the hardware catalog under "Distributed I/O".

Configuration and parameter assignment

Information on configuration and parameter assignment can be found in the following sections.

ET 200M configuration

Definition

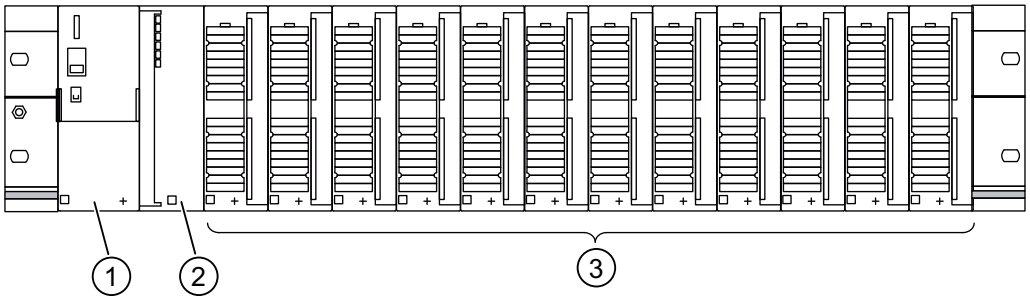
The distributed IO device, ET 200M, is a modular DP slave with an IP 20 degree of protection.

The ET 200M has the configuration technology of the S7-300 automation system and consists of an IM 153-x and I/O modules of the S7-300.

ET 200M supports communication with:

- all DP masters compliant with IEC 61784-1:2002 Ed1 CP 3/1
- all IO controllers compliant with IEC 61158

Configuration of the ET 200M (example)



① Power supply module PS 307
② Interface module IM 153-x
③ up to 12 I/O modules (SM / FM / CP)

Configuration of the 'Module replacement during operation' function

Introduction

The ET 200M supports the "Replace modules during operation" function and the associated pull/plug interrupt.

The "Replace modules during operation" function makes it possible for you to pull modules from or plug modules into the ET 200M rack during operation.

Requirement

You have configured an interface module that supports replacing modules during operation. (as of IM 153-1, article no. 153-1AA02-0XB0).

In addition, the configured CPU must also support the function, e.g. for PROFIBUS an S7-400 with DP interface.

You must use the active backplane bus (bus rail with slots) for the hardware configuration. The conventional profile rail with bus connectors between the modules does not support this function.

Configuring

If the configuration requirements have been met, the "Replace modules during operation" parameter is available for selection in the inspector window's "Module parameters" area. Below this parameter, a table for the configured modules is displayed, which shows the required active bus modules for the hardware configuration.

For a PROFIBUS configuration, the "Startup if preset configuration does not match actual configuration" option is displayed. This option is automatically enabled if "Replace modules during operation" is enabled.

Signal modules for process automation

Fundamentals

Introduction

Signal modules for the process automation are S7-300 models, such as SM 321; DI 16xNAMUR or SM 322; DO 16x24VDC/0.5A.

They are being operated in a DP slave (IM 153-2).

Unlike standard modules, they offer the following additional technical functions, such as pulse extension and chatter monitoring.

See also

Changeover contact (Page 1337)

Technological parameters (Page 1338)

Changeover contact

"Changeover contact" sensor type

If the digital inputs of a channel group are configured as "changeover contacts", the module runs diagnostics for the changeover contact sensor type for this channel group.

Changeover contact

A changeover contact is an auxiliary switch with only one moving switch element with one close setting each for closed and open switching device.

Remember the following rule:

- Always connect a normally open contact to the "even" channel
- Always connect a normally closed contact to the "odd" channel.

The tolerated switchover time between the two channels is fixed at 300 ms.

If the result of the check is negative, then

- the module identifies the value status of the normally open channel as "invalid"
- the module generates a diagnostic entry for the normally open channel
- triggers a diagnostic interrupt (if diagnostic interrupts have been enabled)

The digital input signal and the value status are updated only for the normally open channel. For the normally closed channel, the digital input signal is set permanently to "zero" and the value status to "invalid" since this channel is used only to check the sensor.

Diagnostics depends on the "Selection" parameter (of the sensor). You should also note the special features of diagnostics with the changeover contact sensor type in the "Signal Modules for Process Automation" manual.

See also

Documentation on modules for process automation (http://support.automation.siemens.com/WW/view/de/7215812/0/en)

Technological parameters

Pulse extension and flutter monitoring

Pulse extension is a function for changing a digital input signal. A pulse at a digital input is extended to at least the length set in the parameters. If the input pulse is already longer than the specified length, it is not changed.

If you want the pulse to be extended, click in the box to select the time. If you do not want the pulses to be extended, select the "---" entry.

Flutter monitoring is a process control function for digital input signals. It detects and reports signal changes that are unexpected in process control, for example when an input signal fluctuates too often between "0" and "1".

Flutter monitoring is possible only when group diagnostics has also been enabled for this input.

Monitoring window and number of signal changes

Flutter monitoring works with aid of the two parameters Monitoring window and Number of signal changes.

The first time the signal changes, the time set as the monitoring window is started. If the signal changes more often during this time than allowed by the number of signal changes parameter,

this is signaled as a flutter error. If no flutter error is detected during the monitoring window time, the monitoring window can be restarted at the next signal change.

Note

If you set pulse extension for an input channel, this also affects the flutter monitoring enabled for this channel. The "extended pulse" signal is the input signal for the flutter monitoring. You should therefore make sure that the values set for pulse extension and flutter monitoring are compatible with each other.

See also

Documentation on modules for process automation (http://support.automation.siemens.com/WW/view/de/7215812/0/en)

IQ Sense module

Properties of 8 IQ-SENSE

Properties

The 8 IQ-SENSE module has the following properties:

- Connection of sensors with IQ-SENSE®, photoelectric proximity switches: for example, reflex sensors, diffuse sensors, and laser sensors.
- It can be used centrally in an S7-300 or distributed in an ET 200M.
- You can connect up to 8 sensors to every module. Each sensor requires a two-wire cable.
- Function reserve that can be assigned parameters.
- Time functions, switching hysteresis, synchronous mode that can be assigned parameters
- Sensitivity and distance values can be specified (*IntelliTeach* using the "IQ-SENSE Opto" FB)
- Teach-in
- Sensors can be removed and inserted during operation (automatic reassignment of parameters)

Anti-interference group

Only for optical IQ Sense devices (IQ profile ID 1).

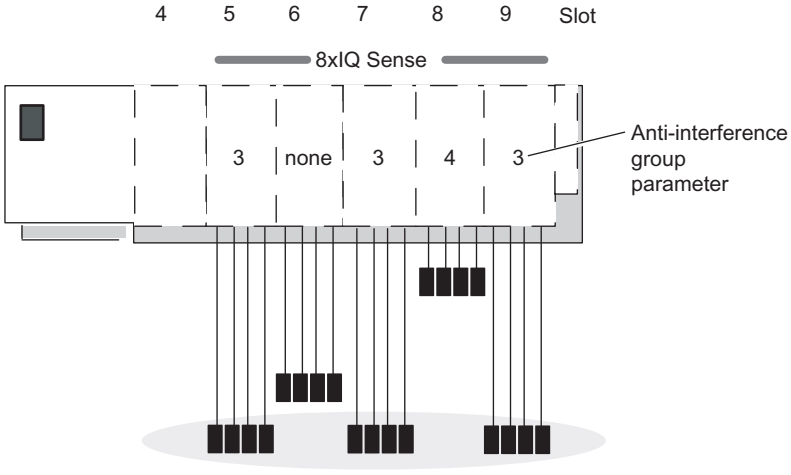
For IQ Sense devices with IQ profile ID 128 (ultrasound), see "Multiplex/synchronous mode" under the channel-specific parameters.

Prevention of interference (e.g., scattered light) by assigning an anti-interference group. This means:

- Anti-interference group: None (= default)
  Optical sensors on one or more modules can mutually influence each other when unfavorably arranged.
- Anti-interference group: 3 or 4
  Optical sensors on the same module with anti-interference group 3 or 4 cannot mutually influence each other. Similarly, optical sensors on different modules with anti-interference group 3 or 4 cannot mutually influence each other. You need not maintain minimum clearance between the IQ Sense devices and can, for example, align two retroreflective sensors on a single reflector.

Operating principle

The diagram below explains the functioning of the anti-interference group parameter:



Mutual interference is only possible between the optical sensors of the modules in slot 5, 6, 7 and 9 because they are in the same anti-interference group 3 or "None" is set.

Note

Sensors in the same anti-interference group must be installed to maintain the minimum clearance (see sensor package insert) and to prevent mutual interference.

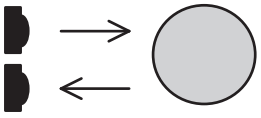
Encoder type

This parameter is used to set the sensor type per channel:

- Reflex sensor or
- Diffuse sensor or
- Disabled

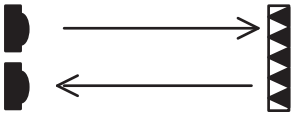
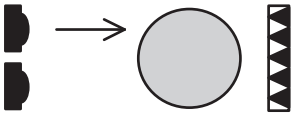
Diffuse sensor

Table 10-101 Diffuse sensor

Diffuse sensor	Object	
Transmitter Receiver		Circuit state 0: No object detected, which means the object is not in the beam. The receiver does not see any light.
Transmitter Receiver		Circuit state 1: Object detected, which means the object is in the beam. The receiver does not see any light.

Reflex sensor

Table 10-102 Reflex sensor

Reflex sensor	Object	
Transmitter Receiver		Circuit state 0: No object detected, which means the object is not in the beam. The receiver sees light.
Transmitter Receiver		Circuit state 1: Object detected, which means the object is in the beam. The receiver does not see any light.

Switching hysteresis

Faults with the diffuse sensor or in the production process can result in signal wobbles. The measured value then changes the switching threshold by 100 % (object detected - object not detected). You can prevent this switching threshold wobble using the switching hysteresis parameter. This will ensure a stable output signal on the sensor.

You can assigned parameters to 5 %/10 %/20 %/50 % for switching hysteresis.

Requirements

You can only set the switching hysteresis parameter for diffuse sensors with background fadeout.

Operating principle

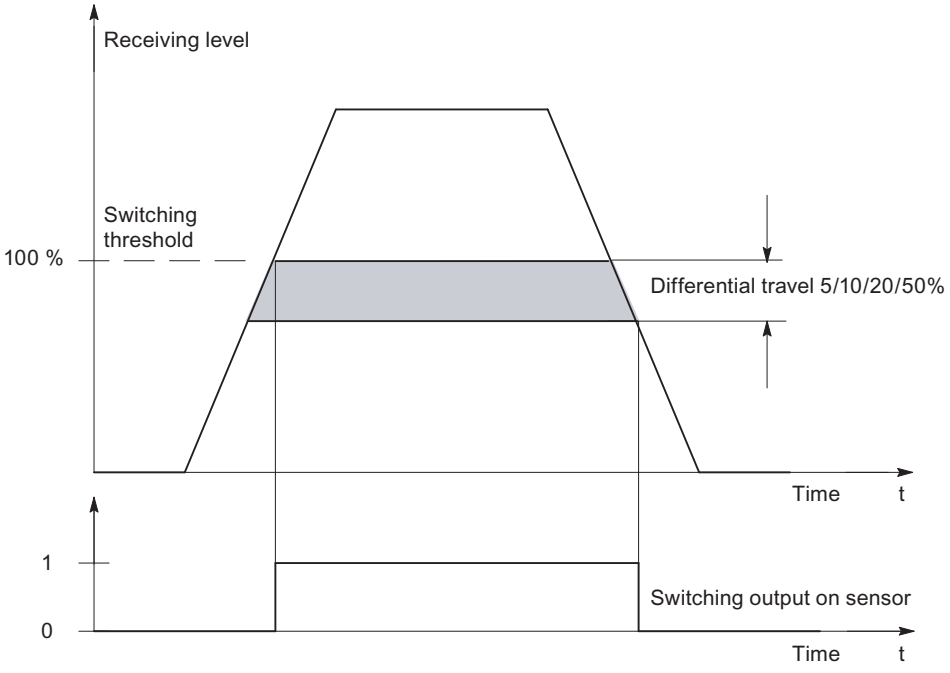


Figure 10-8 Switching hysteresis parameter

Time function,time value

These parameters can be used to set the electronic module for its specific application.

Operating principle

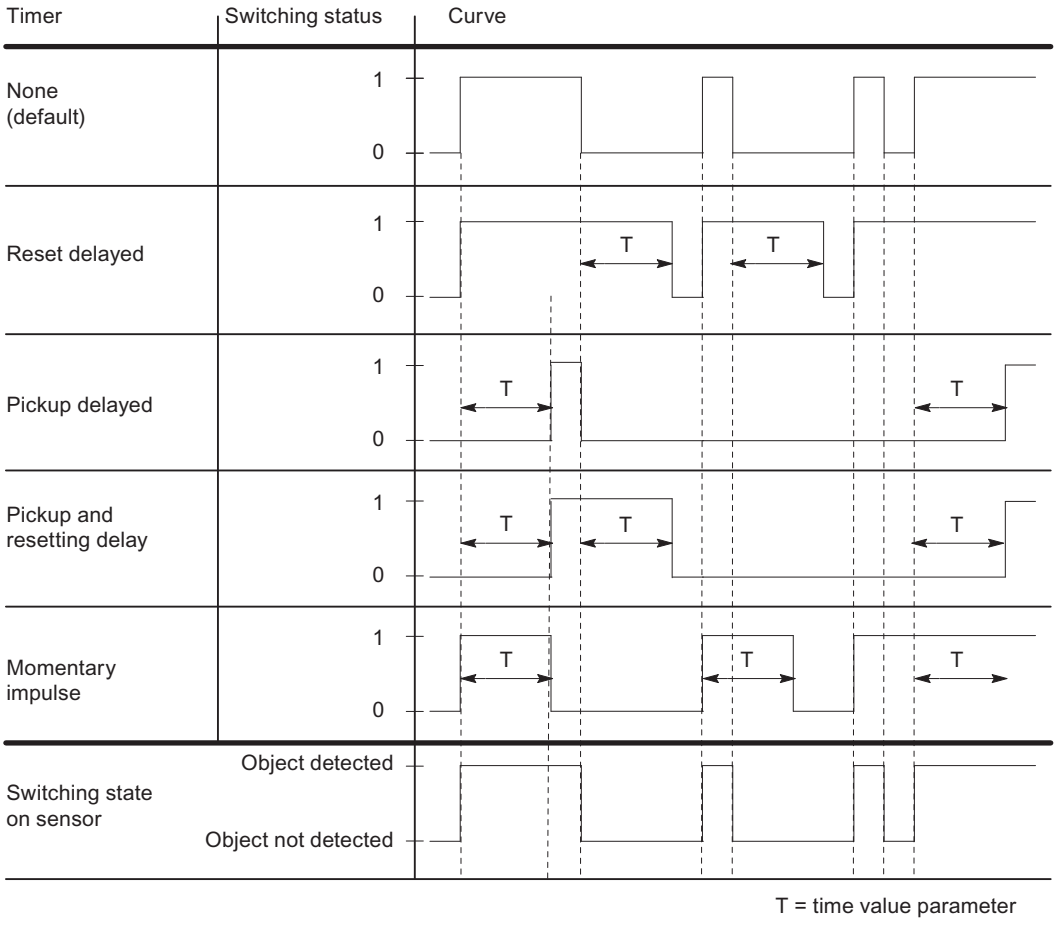


Figure 10-9 Time functions, time values parameters

Multiplex/synchronous mode

For the prevention of mutual influence between IQ Sense ultrasound devices in spatial proximity (devices with IQ profile ID 128), use the "Multiplex/synchronous operation" parameter.

Settings for the multiplex/synchronous mode parameter

Disabled: Mutual influence between IQ Sense ultrasound sensors in spatial proximity is possible (default). The cycle time is determined by the IQ Sense ultrasound sensor.

Multiplex: The IQ Sense ultrasound sensors determine the process value (distance) one after another, preventing them from affecting one another. The cycle time here is the sum of the configured synchronous cycle times of the IQ Sense ultrasound sensors that are to be multiplexed.

Synchronization: The IQ Sense ultrasound sensors determine the process value (distance) at exactly the same time, preventing them from affecting one another. The cycle time here corresponds to the greatest configured synchronous cycle time from among the IQ Sense ultrasound sensors that are to be synchronized.

You can, for example, use synchronous operation for a curtain function in which several IQ Sense ultrasound sensors aligned in parallel share a single extended detection area. The sensors simultaneously emit an ultrasound impulse. When an object enters the detection area, the sensor nearest to the object receives the echo most quickly. The object can therefore not only be detected, it can be located as well.

AFI value

Using the AFI value (application series identifier, as defined in the ISO 15693-3 international standard), transponders can be selected for different applications. Only transponders whose AFI value coincides with the value set on the sensor are processed. If a transponder has the AFI value "0", it can be identified and processed regardless of the AFI value of the sensor.

This parameter is only important if it is supported by the ident system, otherwise any value (normally "0") may be assigned.

Transponder type

Depending on the type of the transponder, you must configure whether it is an ISO transponder or a vendor-specific type.

For transponders in accordance with international standard ISO 15693, the value "1" should be selected; for all other types "0" is set. Based on this setting, one of the two possible air interface drivers is selected in the sensor.

This parameter is only important if it is supported by the ID system, otherwise any value (normally "0") may be assigned.

ET 200S

Configuring an ET 200S

Introduction

For the ET 200S series, you can find a wide range of modules in the hardware catalog under "Distributed I/O".

Assigning parameters

For information on configuration and parameter assignment, refer to "See also".

Frequency converters

Use of the frequency converter

Frequency converters

The frequency converter ICU24 and ICU24F (as fail-safe version) are modular design frequency converters that are completely embedded in the distributed I/O system ET 200S. For parameterization of both modules, please see the following.

Message frame

The message frame number and the operating mode of the module are only displayed and cannot be modified.

Application ID

You indicate the saved parameters in the frequency converter as a whole with the application ID. Enter an application ID from the value range 0 to 65535. During startup (or pull/plug), this ID is compared with the application ID stored on the converter.

Converters that work with identical applications are usually also identically parameterized and should be identified with the same application ID. Converters with the same application ID may be exchanged between each other. Copying of the complete parameterization of a converter to another converter, for example, via an MMC, is only accepted, if both have the same application ID.

Converters that work with different applications and are parameterized differently must be identified by different application IDs. This prevents a converter with unsuitable parameterization from starting on an incorrect slot, i.e. on the wrong application. This also prevents the parameterization that is saved in the converter from being accidently overwritten with any parameteriation that is stored on an MMC.

Enable diagnostic interrupt

You can enable the diagnostic interrupt for the frequency converter. If diagnostic interrupt is enabled, an OB 82 must be available in a CPU to process the diagnostic events.

See also

Documentation for the frequency converter (http://support.automation.siemens.com/WW/view/en/26291825/0/en)

ET 200pro

Configuration control with ET 200pro

Operating principle

Through the configuration control, it is possible to change the original configuration of an ET 200pro (created by configuring with STEP 7) and to operate the ET 200pro in this modified configuration. STEP 7 is no longer required for this configuration: You communicate to the ET 200pro the slot in which a configured module is actually inserted by means of your user program.

You use control data record 196 for this. In this data record, you code which modules are missing or located on different slots in the real configuration as compared to the preset configuration. The configuration control has no effect on the parameter assignment of the modules (for example, the enabling of diagnostic alarms).

You then call the "WRREC" instruction and use it to write the data record to the interface module of the ET 200pro.

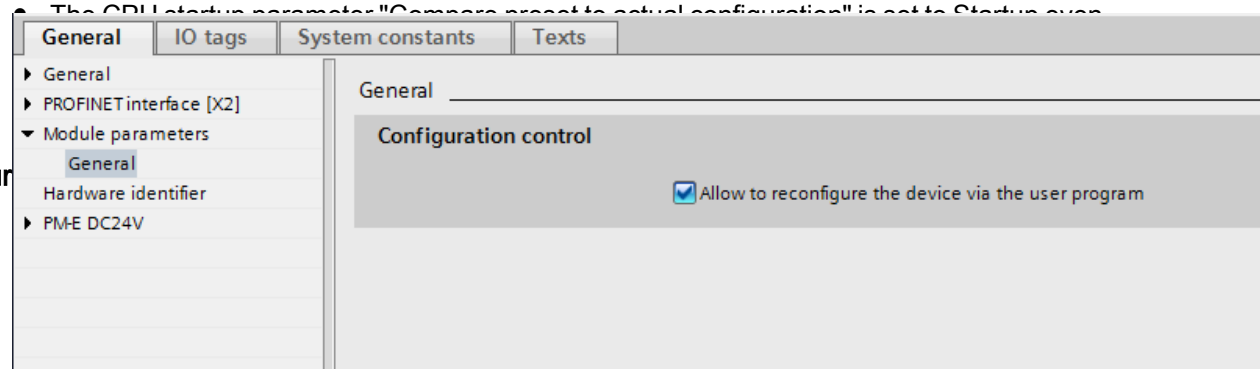
Configuration control gives you the flexibility to vary the configuration of an ET 200pro as long as the real configuration can be derived from a preset maximum configuration (originally created with STEP 7).

The following sections describe how to enable configuration control. They also outline how control data record 196 and readback data record 197 are structured.

Requirements

- STEP 7 Professional version V13 SP1 or higher
- The CPU startup parameter "Compare preset to actual configuration" is set to Startup even

Enabling configur



Configuration of control data record 196 for the ET 200pro

There is a byte in control data record 196 for each module.

The position of this byte in the data record codes one module in the original configuration with STEP 7:

- "slot_IM" (byte 4 in the data record, figure below) corresponds to the module in slot 1 in the configuration
- "slot_2" (byte 5) corresponds to the module in slot 2 in the configuration.
- "slot_3" (byte 6) corresponds to the module in slot 3 in the configuration.
- etc.

"slot_x" byte

The current slot is coded by the figure that is assigned to "slot_x" (by its value). Examples:

- The value "2" in byte 5 means you are assigning the module originally inserted in slot 2 to slot 2 in the current configuration (slot_2 = 2).
- The value "3" in byte 5 means you are assigning the module originally inserted in slot 2 to slot 3 in the current configuration (slot_2 = 3).
- The value "4" in byte 5 means you are assigning the module originally inserted in slot 2 to slot 4 in the current configuration (slot_2 = 4).
- etc.

There are no reserve modules for the ET 200pro (unlike for the ET 200S or with BU cover modules for the ET 200SP). For this reason, bit 7 of "slot_x" must not be set.

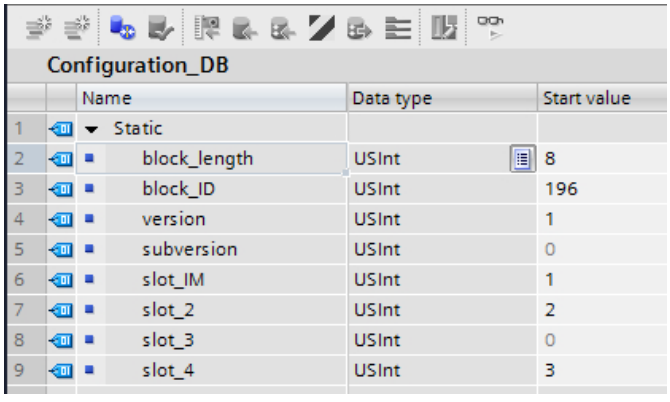
The value "0" in "slot_x" indicates that this module is not inserted in the current configuration.

Example of control data record 196

The figure below shows control data record 196 for a configuration of an ET 200pro with four modules.

This configuration is given as an example:

- The module originally configured with STEP 7 in slot 1 is also inserted in slot 1 in the current configuration.
- The module in slot 2 is in slot 2 in the current configuration.
- The module in slot 3 does not exist in the current configuration.
- The module in slot 4 is actually inserted in slot 3 in the current configuration.



Configuration_DB

		Name	Data type	Start value
1		Static		
2		block_length	USInt	8
3		block_ID	USInt	196
4		version	USInt	1
5		subversion	USInt	0
6		slot_IM	USInt	1
7		slot_2	USInt	2
8		slot_3	USInt	0
9		slot_4	USInt	3

The components of control data record 196 (figure above):

- block_length: Note here the length of the control data record; in the example: 8 (bytes). The length is calculated using the following formula: "Number of assigned slots" + 4.
- block_ID: Enter the figure 196 here.
- version: The ET 200pro uses Version 1 of control data record 196.
- subversion: The ET 200pro uses Subversion 0 of control data record 196.
- slot_IM: The IM module is always inserted in slot 0 of the ET 200pro. Slot 1 always contains the virtual power module that is integrated as a fixed module in the IM module. The "slot_IM" (name can be changed) can contain any values. This byte is not interpreted in the configuration control of the ET 200pro.
- slot_2: The configured module 2 is inserted in slot 2 (value "2").
- slot_3: The configured module 3 is not present in the current configuration (value "0").
- slot_4: The configured module 4 is inserted in slot 3 in the current configuration (value "3").

Definition of control data record 196

A control data record 196 containing a slot assignment is defined for configuration control.

Byte	Element	Value	Description
0	Block length	e.g., 8 for ET 200pro with four modules	The length of the data record is calculated using the formula: 4 + Number of modules in bytes
1	Block ID	196	ID for control data record 196
2	Version	1	Version 1 of control data record 196
3	Subversion	0	Subversion 0 of control data record 196
4	Slot_1	Any values are possible: For example "1"	This byte is not interpreted for the ET 200pro because the power module that is integrated as a fixed module in the IM module of the ET 200pro is always inserted in slot 1.
5	Slot_2	Coding of the actual slot: 2 = Slot 2 3 = Slot 3 4 = Slot 4 etc. Coding for missing module: 0 = No slot, module not present	Byte 5 indicates where the module originally configured with STEP 7 in slot 2 is actually inserted in the current configuration. Example: 2 The module originally configured with STEP 7 in slot 2 is also actually located in slot 2 in the current configuration (value "2").

6	Slot_3	Coding of the actual slot: 2 = Slot 2 3 = Slot 3 4 = Slot 4 etc. Coding for missing module: 0 = No slot, module not present	Byte 6 indicates where the module originally configured with STEP 7 in slot 3 is actually inserted in the current configuration. Example: 0 The module originally configured with STEP 7 in slot 3 is not present in the current configuration (value "0").
7	Slot_4	Coding of the actual slot: 2 = Slot 2 3 = Slot 3 4 = Slot 4 etc. Coding for missing module: 0 = No slot, module not present	Byte 7 indicates where the module originally configured with STEP 7 in slot 4 is actually inserted in the current configuration. Example: 3 The module originally configured with STEP 7 in slot 4 is actually located in slot 3 in the current configuration (value "3").
:	:	:	:

Writing a data record

Transfer the control data record to the ET 200pro module.

To do so, call the extended WRREC (Write data record) instruction, and transfer the control data record created.

If you do not transfer a control data record, the interface module uses the original configuration created with STEP 7. In this case, the following applies: Configured module x is inserted in real slot x.

Addressing the interface module using the HW identifier

To transfer data record 196 with the instruction WRREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Error messages

The following error messages are returned if an error occurs when writing control data record 196:

Table 10-103 Error messages

Error code	Meaning
16#80A2	DP protocol error on layer 2. Indicates that a data record has not been acknowledged due to the system.
16#80B1	Invalid length; the length information in data record 196 is not correct.
16#80B5	Configuration control parameters not assigned.
16#80B2	Invalid slot: The configured slot is not assigned.
16#80B8	Parameter error; module signals invalid parameters.
16#80C5	DP slave or module not available. Indicates that a data record has not been acknowledged due to the system.

Readback data record 197 for ET 200pro

Readback data record 197 is used to read the actual configuration of a station (in this case of an ET 200pro).

This data record allows you to check the real configuration of the ET 200pro (actual configuration). The readback data record for each configured module specifies whether or not it is actually available.

- The value "1" means that the correct module is inserted in the correct slot.
- The value "0" codes all other options (wrong module, empty slot).

Example:

A module has been configured with STEP 7 for slot 4.

This module has then been moved to slot 3 in the current configuration using data record 196.

If this module is also really in slot 3, this is coded by the value "1" (status_slot_4 = 1).

Configuration details:

The configuration of the data block corresponds to the original configuration of the ET 200pro with STEP 7.

There is a byte in the data record for each module. The position of this bytes in the data record corresponds to the position of a module in the original configuration with STEP 7.

Sequence of bytes:

- "status_slot_IM" (byte 4 in the data record) corresponds to the module in slot 1 in the configuration,
- "status_slot_2" (byte 5) corresponds to the module in slot 2 in the configuration
- "status_slot_3" (byte 6) corresponds to the module in slot 3 in the configuration,
- etc.

The following example is for a configuration with 4 modules. The value "8" therefore appears in the "block_length" element of the data record.

If you configure an ET 200pro in STEP 7 with fewer modules, the data block will be shorter.

You can choose any name for the components of the control data record (for example "status_slot_2").

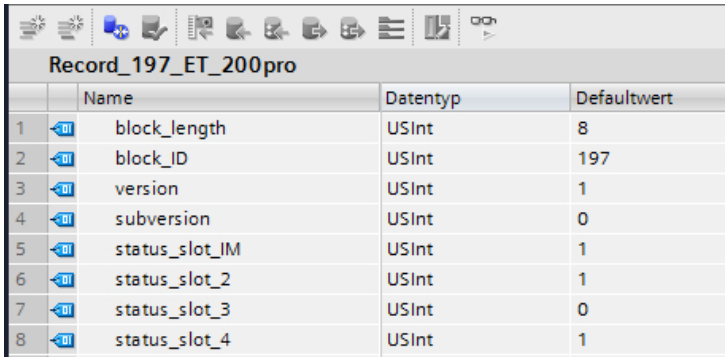
Meaning of "status_slot_x":

- The value "1" in status_slot_x means that module x is inserted in the correct slot
- The value "0" in status_slot_x codes all other options (wrong module, module does not exist).

Example:

The figure below shows readback data record 197 for an ET 200pro with four modules.

- There is no module 3 (this was assigned in control data record 196, see section "Example of control data record 196" above)
- The three other modules are actually inserted in the ET 200pro.



Record_197_ET_200pro

	Name	Datentyp	Defaultwert
1	block_length	USInt	8
2	block_ID	USInt	197
3	version	USInt	1
4	subversion	USInt	0
5	status_slot_IM	USInt	1
6	status_slot_2	USInt	1
7	status_slot_3	USInt	0
8	status_slot_4	USInt	1

Reading readback data record 197

You can read readback data record 197 from the ET 200pro with the instruction RDREC. RDREC operates asynchronously. If you call RDREC in the startup OB, you must call the instruction multiple times using a loop until the "BUSY" or "DONE" output parameter indicates that the data record has been read.

To read data record 197 with the instruction RDREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Further information and examples

Further information on the PN interface module of the ET 200pro is available here (http://support.automation.siemens.com/WW/view/en/98099372).

Specific examples of configuration control can be found here in this application description (http://support.automation.siemens.com/WW/view/en/29430270).

See also

Configuration control with ET 200SP (Page 1274)

Use of the frequency converter

Frequency converters

The frequency converters ET 200pro FC and ET 200pro F-FC (as fail-safe version) are modularly design frequency converters that are completely embedded in the distributed I/O system ET 200pro. The following section describes how to configure the two modules.

Message frame

The message frame number and the operating mode of the module are only displayed and cannot be modified.

Application ID

You indicate the saved parameters in the frequency converter as a whole with the application ID. Enter an application ID from the value range 0 to 65535. During startup (or pull/plug), this ID is compared with the application ID stored on the converter.

Converters that work with identical applications are usually also identically configured and should be identified with the same application ID. Converters with the same application ID may be exchanged between each other. Copying of the complete configuration of a converter to another converter, for example, via an MMC, is only applied, if both have the same application ID.

Converters that work with different applications and are configured differently must be identified by different application IDs. This prevents a converter with unsuitable configuration from starting on an incorrect slot, in other words on the wrong application. This also prevents the configuration that is saved in the converter from being accidently overwritten with any configuration that is stored on an MMC.

Enable diagnostic interrupt

You can enable the diagnostic interrupt for the frequency converter. If diagnostic interrupt is enabled, an OB 82 must be available in a CPU to process the diagnostic events.

10.1.6.4 IPv6 configuration

IPv6 protocol

The Internet Protocol version 6 - called IPv6 below - extends the Internet Protocol version 4 (IPv4) that is used predominantly at the current time.

Address format IPv6: Notation

IPv6 addresses consist of 8 fields each with four-character hexadecimal numbers (128 bits in total). The fields are separated by a colon.

Example:

fd00:ffff:ffff:ffff:ffff:ffff:2f33:8f21

Rules / simplifications:

- Leading zeros within a field can be omitted.
  Example: fd01:0:ffff::2d12:7d23
- If one or more fields have the value 0, a shortened notation is possible.
  The address fd00:0:0:0:0:0:0:8f21 can also be shortened and written as follows:
  fd00::8f21
  To ensure uniqueness, this shortened form can only be used once within the entire address.
- Decimal notation with periods
  The last 2 fields or 4 bytes can be written in the normal decimal notation with periods.
  Example: The IPv6 address fd00::ffff.125.1.0.1 is equivalent to fd00::ffff:7d01:1

Entry and appearance

The entry of IPv6 addresses is possible in the notations described above. IPv6 addresses are always shown in the same notation in which they were entered.

See also

IPv6 with the CP 1543-1 (Page 1353)

IPv6 with the CP 1543-1

Use of IPv6 with the CP 1543-1

The CP supports the Internet Protocol version 4 (IPv4) for all IP services

The additional specification of addresses in IPv6 format can be used with the CP for the following services and applications:

- FETCH/WRITE
  Direct write/read access by PC stations, SIMATIC S5 or third-party devices
- FTP client
  FTP access from the S7-1500 CPU to an FTP server with the program block FTP_CMD
- FTP server
  FTP access from an FTP client to data areas of the S7-1500 CPU
- SNMP
  Data query using MIB objects according to SNMP
- E-mail
  Data transfer from the S7-1500 CPU using the program block T_Mail

10.2 Device and network diagnostics

10.2.1 Hardware diagnostics

10.2.1.1 Overview of hardware diagnostics

Principal methods of hardware diagnostics

Principal methods of hardware diagnostics

Hardware diagnostics can be performed as follows:

- Using the Online and Diagnostics view
- Using the "Online Tools" task card
- Using the "Diagnostics > Device Info" area of the Inspector window
- Using diagnostics icons, for example, in the device view and the project tree

Structure of the Online and Diagnostics view

The Online and Diagnostics view consists of two windows alongside each other:

- The left window shows a tree structure with folders and - when you open the folder - groups.
- The right window contains detailed information on the selected folder or selected group.

The "Online access" group and the "Diagnostics" and "Functions" folders are located here:

- "Online access" group: Displays whether or not there is currently an online connection with the associated target. In addition, you can establish or disconnect the online connection.
- "Diagnostics": Contains several diagnostics groups for the selected module.
- "Functions": Contains several groups, in which you can make settings for the selected module or issue commands to the module.

Function and structure of the "Online Tools" task card

For modules with their own operating mode (such as CPUs), the "Online tools" task card allows you to read current diagnostics information and commands to the module.

If you selected a module without its own operating mode or if you selected several modules before activation of the "Online Tools" task card, the task card relates to the relevant CPU.

The "Online Tools" task card consists of the following panes:

- CPU control panel
- Cycle time
- Memory

Note

A pane is filled with content only if the module controls the associated functions and an online connection exists.

If there is no online connection to the respective module, the display "No online connection" appears in blue in each pane. If an existing online connection was disconnected, then "This target is not available" will be displayed.

Structure or the "Diagnostics" tab of the Inspector window

The "Diagnostics" tab of the Inspector window itself consists of several tabs. Of these tabs, the following is relevant for the hardware diagnostics.

- Device information
  This tab relates to all online devices (e.g. CPUs) to which an online connection has been established and to the devices that are assigned to these online devices (such as PROFINET devices and PROFIBUS slaves). Alarms related to the faulty online devices and the faulty devices are output here.

Note

What is displayed when a module is faulty?

If a module within a device is faulty, only the associated device or its proxy (e.g. head module) are shown, but not the module itself.

The faulty devices are displayed "at the top level" and not in a hierarchical view below their online device (as is the case in the project tree).

See also

Basics on task cards (Page 326)

Inspector window (Page 324)

Determination of which of the devices that are connected online are defective

Overview of the defective devices

In the "Diagnostics > Device Info" area of the Inspector window you will obtain an overview of the defective devices that are or were connected online.

The "Diagnostics> Device Info" area of the Inspector window consists of the following elements:

- Header line with the number of defective devices
- Table with detailed information on each defective device

If you originate the establishment of an online connection to a device which is not reachable or reports one or more faults or is not in RUN mode, it will rank as defective.

Structure of the table with detailed information on the defective devices

The table consists of the following columns:

- Online status: Contains the online status as a diagnostic symbol and in words
- Operating mode: Contains the operating mode as a symbol and in words
- Device / module: Name of the affected device or the affected module
- Message: explains the entry of the previous column
- Details: The link opens the online and diagnostics view for the device, and places it in the foreground. If an online connection does not exist any longer, the link will open the connection establishment dialog.
- Help: The link supplies further information on the defect that has occurred.

See also

Displaying diagnostics status and comparison status using icons (Page 1356)

Displaying diagnostics status and comparison status using icons

Determining diagnostics status online and displaying using icons

When the online connection to a device is established, the diagnostics status of the device and, if applicable, its lower-level components is determined. The operating state of the device is also determined, where applicable.

The following is a description of which icons are displayed in specific views.

- Device view
  - The associated diagnostic icon is displayed for every hardware component (except the signal board on the CPU).
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - For hardware components with their own operating state, the operating state icon is also displayed to the left of or above the diagnostic icon.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: No diagnostic icons are displayed (due to the configuration as GSDML device).
- Device overview
  - The associated diagnostic icon is displayed for every hardware component.
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: For modules assigned to the CPU, the associated diagnostic icon is displayed (modules that are not assigned do not receive a diagnostic icon). The associated diagnostic icon is displayed for plug-in submodules of an assigned module (submodules that are not pluggable are not visible and therefore do not receive a diagnostic icon).
- Network view
  - The associated diagnostic icon is displayed for every device.
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.

- Network overview
  - The associated diagnostic icon is displayed for every hardware component.
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.
- Topology view
  - The associated diagnostic icon is displayed for every device.
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The associated diagnostic icon is displayed for every port. The meaning of the individual colors is described further below.
  - Each cable between two online ports is assigned the color associated with its diagnostics status.
    The color of the cable between two ports depends on the status of the individual ports:

Color of the first port	Color of the second port	Color of the connecting cable
light green	light green	light green
light green	dark green	dark green
green	gray	gray
green	red	red
gray	red	red

  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.
- Topological overview
  - The associated diagnostic icon is displayed for every hardware component.
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.

- Project tree
  - The associated diagnostic icon is displayed behind every hardware component.
  - For a hardware component with lower-level components (e.g., distributed I/O, Slave_1), if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - For hardware components with their own operating state, the operating state icon is also displayed in the top right corner of the diagnostic icon.
  - If forcing is active on a CPU, a red F is displayed at the left margin of the diagnostic icon.
  - The diagnostic icon "Hardware error in lower-level component" is displayed behind the "Local modules" folder when there is a hardware error in at least one of the associated modules.
  - The diagnostic icon "Hardware error in lower-level component" is displayed behind the "Distributed I/O" folder when there is a hardware error in at least one of the associated modules.
  - The diagnostic icon "Hardware error in lower-level component" is displayed behind the project folder when the "Hardware error in lower-level component" diagnostic icon is displayed behind at least one of the "Local modules" or "Distributed I/O" folders.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: The associated diagnostic icon is displayed for modules assigned to the CPU (modules that are not assigned are grayed out and do not receive a diagnostic icon). The associated diagnostic icon is displayed for plug-in submodules of an assigned module (submodules that are not pluggable are not visible and therefore do not receive a diagnostic icon).

Note

If the diagnostic for a hardware component is "not reachable from the CPU", the diagnostic icon "Hardware error in lower-level component" is not additionally shown.

Diagnostic icons for modules and devices

The following table shows the available icons and their respective meaning.

Icon	Meaning
	The connection with a CPU is currently being established.
	The CPU is not reachable at the set address.
	The configured CPU and the CPU actually present are of incompatible types.
	On establishment of the online connection to a protected CPU, the password dialog was terminated without specification of the correct password.
	No fault

Icon	Meaning
	Maintenance required
	Maintenance demanded
	Error
	The module or device is deactivated.
	The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
	No input or output data are available, because the (sub)module has blocked its input or output channels.
	Diagnostics data are not available because the current online configuration data differ from the offline configuration data.
	The configured module or device and the module or device actually present are incompatible (valid for modules or devices under a CPU).
	The configured module does not support display of the diagnostics status (valid for modules under a CPU).
	The connection is established, but the module status has not yet been determined or is unknown.
	The configured module does not support display of the diagnostics status.
	Hardware error in lower-level component: A hardware error is present in at least one lower-level hardware component. (occurs as a separate icon only in the project tree)

Note

Some modules, for example, the FM 450-1, are only indicated as having a problem in the case of an error if you have enabled the diagnostic interrupt when assigning the module property parameters.

Icons for the comparison status

The diagnostic icons can be combined at the bottom right with additional smaller icons that indicate the result of the online/offline comparison. The following table shows the available comparison icons and their meaning.

Icon	Meaning
	Hardware error in lower-level component: The online and offline versions differ (only in the project tree) in at least one lower-level hardware component.
	Software error in lower-level component: The online and offline versions differ (only in the project tree) in at least one lower-level software component.
	Online and offline versions of the object are different
	Object only exists online
	Object only exists offline
	Online and offline versions of the object are the same

Note

If both a comparison icon and the "Error in lower-level component" diagnostic icon are to be displayed at the bottom right in the device view, the following rule applies: The diagnostic icon for the lower-level hardware component has a higher priority than the comparison icon. This means that a comparison icon is only displayed if the lower-level hardware components have no errors.

Display of software errors in the project tree

- The associated comparison icon is shown behind each block.
- Behind each folder, under which exclusively blocks are contained, the diagnostic icon "Software error in lower-level component" is displayed when there is a software error in at least one of the associated blocks.
- For a hardware component with lower-level software components, if there is no hardware error and there is an error in at least one lower-level software component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Software error in lower-level component" is also shown in the lower right corner.

Combined diagnostics and comparison icons

The following table shows examples of icons that are displayed in the diagnostics icon.

Icon	Meaning
	Folder contains objects whose online and offline versions differ (only in the project tree)
	Object only exists online

Operating state icons for CPUs and CPs

The following table shows the available icons and their respective operating states.

Icon	Operating state
	RUN
	STOP
	STARTUP
	HOLD
	DEFECTIVE

Icon	Operating state
	Unknown operating state
	The configured module does not support display of the operating state.

Note

If forcing is active on a CPU, a red F is displayed on a pink background at the bottom right of the operating state icon.

Color marking of ports and Ethernet cables

The following table shows the available colors and their respective meaning.

Color	Meaning
	No fault or maintenance required
	Offline
	Maintenance demanded
	Communication error or topological error
	no diagnostic capability

Start online and diagnostics view

Overview of possible ways of starting the Online and Diagnostics view

You can start the Online and Diagnostics view of a module to be diagnosed at the following locations:

- Overview
- Project tree
- Device view
- Device overview
- Network view
- Network overview
- Topology view
- Topological overview

In the following, examples are used to show how to proceed.

Requirement

The project with the module to be diagnosed is open.

Note

This requirement does not apply if you call the online and diagnostics view from the project tree after you have identified the accessible devices.

Procedure

To start the online and diagnostics view of a module, follow these steps:

1. In the project tree, open the respective device folder.
2. Double click on "Online & Diagnostics".

Or:

1. In the project tree, select the respective device folder.
2. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. In the project tree, open the "Online access" folder.
2. Open the folder for the interface with which you want to establish the online connection.
3. Double click on "Show/Update accessible devices".
4. Select the module to be diagnosed.
5. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. In the project tree, open the "Local modules" folder.
2. Select the respective device or the module that is to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the main menu.

Or:

1. Open the device view in the device configuration.
2. Select the module to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. Open the device view in the device configuration.
2. Establish an online connection to the module to be diagnosed.
3. Double-click on the diagnostics icon above the module.

Or:

1. Open the network view in the device configuration.
2. Select the station with the module to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. Open the topology view in the device configuration.
2. Establish an online connection to the module to be diagnosed.
3. In the topology view, double-click the diagnostic icon associated with the module.

Result

The online and diagnostics view of the module to be diagnosed will be started. If an online connection to the associated CPU had previously been created, the header bar of the Online and Diagnostics view will now have an orange background.

Note

If no online connection exists when the online and diagnostics view is started, no online information is displayed and the display fields remain empty.

Activation of the "Online Tools" task card

Activation of the "Online Tools" task card

You can activate this task card as follows:

1. Start the online and diagnostics view.
2. Click on the "Online Tools" task card.

Or:

1. Start the device view.
2. Click on the "Online Tools" task card.

Or:

1. Start the network view.
2. Click on the "Online Tools" task card.

10.2.1.2 Showing non-editable and current values of configurable module properties

Showing general properties and system-relevant information for a module

Where do I find the information I need?

The general properties and system-relevant information for a module can be found in the "General" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed.

Structure of the "General" group

The "General" group consists of the following areas:

- Module
- Module information
- Vendor information

"Module" area

This area shows the following data of the module:

- Short designation, for example, CPU 1214C DC/DC/DC
- Article no.
- Hardware
- Firmware
- Racks
- Slot

"Module information" area

This area shows the following data of the module that you configured during hardware configuration:

- Module name
- Installation date (not displayed for all modules)
- Additional information (not displayed for all modules)

"Manufacturer information" area

This area shows the following data of the module:

- Manufacturer
- Serial number

- Profile: Profile ID as hexadecimal number

  Note

  You will find the corresponding profile name in the profile ID table for PROFIBUS International (see "www.profibus.com").

- Profile details: Profile-specific type as hexadecimal number

  Note

  You will find the corresponding profile-specific type name in the profile-specific type table for PROFIBUS International (see "www.profibus.com").

Display configured cycle times

Where do I find the information I need?

The required information can be found in the following places:

- In the "Cycle time" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.
- In the "Cycle time" pane of the "Online Tools" task card

Structure of the "Cycle time" group in the "Diagnostics" folder of the Online and Diagnostics view

The "Cycle time" group consists of the following areas:

- Cycle time diagram (graphical display of the assigned and measured cycle times)
- Cycle time configured (display of the assigned cycle times as absolute values)
- Cycle times measured (display of the measured cycle times as absolute values)

Structure of the "Cycle time" pane of the "Online Tools" task card

The "Cycle time" pane displays the cycle time diagram and below it the measured cycle times as absolute values.

Assigned cycle times

The following assigned cycle times are displayed in the cycle time diagram and in the "Cycle time configured" area.

- Minimum cycle time
- Maximum cycle time

In the cycle time diagram, the minimum cycle time and the maximum cycle time correspond to the two markings on the time axis.

In the "Cycle time configured" area, the assigned cycle times are displayed as absolute values.

Show interfaces and interface properties of a module

Where do I find the information I need?

The interfaces and interface properties of a module can be found in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed in the following group:

- PROFINET interface

"PROFINET interface" group

This group is divided into the following areas:

- "Ethernet address" with the "Network connection" and "IP Parameters" subareas
- "Ports"

"Network connection" subarea of the "Ethernet address" area

This subarea shows the following data for the module:

- MAC Address:
  MAC address of the interface.
  The MAC address consists of two parts. The first part ("Basic MAC address") identifies the manufacturer (Siemens, 3COM, ...). The second part of the MAC address differentiates between the various Ethernet devices. Each Ethernet module is assigned a unique MAC address.

"IP Parameters" subarea of the "Ethernet address" area

This subarea shows the following data for the module:

- IP address:
  Internet protocol address of the device on the bus (TCP/IP)
- Subnet mask:
  The subnet mask shows which part of the IP address determines the membership of a particular sub-network.
- Default router:
  If the subnet is connected via a router to other subnets, the IP address of the default router must be known. This is the only way a datagram can be forwarded with a non-matching subnet address.
- IP settings:
  Identifier for the path by which the device has obtained its IP settings (IP address, subnet mask, default router).

Identifier	Meaning
0	IP address is not initialized
1	By configuration (i.e., by the configuration loaded to the device from the device or network view)
2	Via the "Assign IP address" group of the online and diagnostics view

Identifier	Meaning
3	Via the DHCP server (i.e., the IP parameters are obtained by a special service from a DHCP server (Dynamic Host Configuration Protocol) and assigned for a limited time)
4	IP address is set by a user program
5	Source of IP address unknown

- IP setting time:
  Time stamp of the last change to the IP address directly through the Ethernet connection of the module

"Ports" area

This area shows the following data for the module:

- Ethernet ports
  Physical properties of the PROFINET interface

Properties of the PROFINET interface	Meaning
Port no.	Port number The short description of interface (X + interface no.) and port (P + port no.) is specified in parentheses. An "R" in the short description of a port means that it is a ring port.
Status	Displays the status of the port LINK LED. • Status "OK" means another device (such as a switch) is connected to the port and the physical connection is available. • Status "disconnected" means no other device is connected to the port. • Status "deactivated" means that access to the port is blocked.
Settings	• "Automatic" for automatic network settings of the device • Network settings for speed and transmission method for manual network settings of the device
Operating mode	Network settings for speed and transmission method

If you select a line in the port table, additional help information will be provided for the corresponding port.

Displaying IO controllers that access modules of a Shared Device

Where do I find the information I need?

The display of those IO controllers that access the modules of a Shared Device can be found in the Online and Diagnostics view of the interface module of the Shared Device in the "Diagnostics" folder in the following area of the "PROFINET interface" group:

- IO controller

Displaying sync domain properties of a PROFINET device

Where do I find the information I need?

The sync domain properties of a PROFINET device can be found in the following area of the "PROFINET interface" group in the "Diagnostics" folder of the Online and Diagnostics view of the device to be diagnosed:

- Domain

"Domain" area

This area is divided into the following subareas:

- Sync domain
- MRP domain

What is a sync domain?

A sync domain is a group of PROFINET devices that are synchronized to a common clock. Exactly one device has the role of the sync master (clock generator); all other devices assume the role of a sync slave. The sync master is usually an IO controller or a switch.

Non-synchronized PROFINET devices are not part of a sync domain.

"Sync domain" subarea of the "Domain" area

This subarea shows the following properties of the sync domain:

- Name:
  Name of sync domain
- Role:
  Role of the PROFINET device in the sync domain. The following roles are possible:
  - Sync master
  - Sync slave
- Synchronization interval:
  Interval at which the synchronization is performed
- Send clock
  Smallest possible send interval for the data exchange
- Jitter accuracy of the send clock
- Reserved bandwidth for cyclic communication

Displaying MRP domain properties of a PROFINET device

Where do I find the information I need?

The MRP domain properties of a PROFINET device can be found in the following area of the "PROFINET interface" group in the "Diagnostics" folder of the Online and Diagnostics view of the device to be diagnosed:

- Domain

"Domain" area

This area is divided into the following subareas:

- Sync domain
- MRP domain

What is an MRP domain?

The Media Redundancy Protocol (MRP) enables redundant networks to be structured. Redundant transmission paths (ring topology) ensure that, if one transmission path fails, an alternative communication path is available. The PROFINET devices that are part of this redundant network form an MRP domain.

"MRP domain" subarea of the "Domain" area

This subarea shows the following properties of the MRP domain:

- Name:
  Name of MRP domain
- Role:
  Role of the PROFINET device in the MRP domain. The following roles are possible:
  - Manager
  - Manager (Auto)
  - Client
  - Not a device of the ring
- Ring port 1:
  The port of the PROFINET device that has the "Ring port 1" property
- Ring port 2:
  The port of the PROFINET device that has the "Ring port 2" property
- Status of the MRP ring:
  Indicates whether the ring is interrupted ("open" status) or not ("closed" status).

Displaying current firmware of a module

Displaying firmware

You can display the currently installed firmware of a module.

Requirements

- The module supports a firmware update.
- The module is connected online.

Procedure

To display the current firmware, follow these steps:

1. Open the module in the Online and Diagnostics view.
2. Select the "Firmware update" group in the "Functions" folder.
3. You read off the current firmware in the "Online data" area under "Firmware".

10.2.1.3 Showing the current values of dynamic modules properties

Display measured cycle times

Where do I find the information I need?

The measured cycle times can be found at each of the following places:

- In the "Cycle time" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.
- In the "Cycle time" pane of the "Online Tools" task card

Structure of the "Cycle time" group in the "Diagnostics" folder of the Online and Diagnostics view

The "Cycle time" group consists of the following areas:

- Cycle time diagram (graphical display of the assigned and measured cycle times)
- Cycle time configured (display of the assigned cycle times as absolute values)
- Cycle times measured (display of the measured cycle times as absolute values)

Structure of the "Cycle time" pane of the "Online Tools" task card

The "Cycle time" pane displays the cycle time diagram and below it the measured cycle times as absolute values.

Graphical display of the measured cycle times

The following measured cycle times are displayed in the cycle time diagram:

- Shortest cycle time: Duration of the shortest cycle since the last transition from STOP to RUN
  This corresponds to the dashed gray arrow on the left in the diagram.
- Current / last cycle time: Duration of the last cycle
  This corresponds to the green arrow in the diagram. If the current / last cycle time exceeds the maximum cycle time, the arrow will turn red.

  Note

  If the duration of the last cycle comes close to the maximum cycle time, it may be possible that it will be exceeded. Depending on the CPU type, parameter assignment and your user program, the CPU can switch to STOP mode. If for instance you are monitoring the tags in your program, this will increase the cycle time.

  If the cycle lasts longer than double the maximum cycle time, and you do not restart the maximum cycle time in the user program (by calling the extended RE_TRIGR) instruction, the CPU will switch to STOP mode.
- Longest cycle time: Duration of the longest cycle since the last transition from STOP to RUN.
  This corresponds to the dashed blue arrow on the right in the diagram.

A blue band extends between the two dashed lines; this band corresponds to the entire range of the measured cycle times. If a measured cycle time is greater than the maximum cycle time, the portion of the band that lies outside the assigned limits will be colored red.

Display of the measured cycle times as absolute values

The following measured times are displayed in the "Cycle times measured" area and in the "Cycle time" pane.

- Shortest cycle time since the last transition from STOP to RUN.
- Current/last cycle time:
- Longest cycle time since the last transition from STOP to RUN.

Showing the current status of the LEDs of a CPU

Where do I find the information I need?

The current status of the LEDs of a CPU can be found in the display area of the "CPU control panel" pane of the "Online tools" task card.

Display area of the "CPU control panel" pane of the "Online Tools" task card

This area contains the following displays:

- Station name and CPU type (short designation)
- RUN / STOP (corresponds to the "RUN / STOP" LED of the CPU)
- ERROR (corresponds to the "ERROR" LED on the CPU)
- MAINT (corresponds to the "MAINT" LED on the CPU)

Showing fill levels of all types of memory on a CPU

Where do I find the information I need?

The fill levels of all types of memory on a CPU can be found on the following two pages:

- In the display area of the "Memory" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed
- In the display area of the "Memory" pane on the "Online Tools" task card

Display area of the "Memory" group in the "Diagnostics" folder of the online and diagnostics view

This area contains the current memory utilization of the associated module and details of the individual memory areas.

The memory utilization is shown both as a bar diagram and as a numerical value (percentage).

The following memory utilizations are shown:

- Load memory
  If no memory card is inserted, the internal load memory is displayed.
  If a memory card is inserted, the operating system only uses the inserted load memory as the load memory. This is displayed here.
- Work memory
- Retentive memory

Display area of the "Memory" pane of the "Online Tools" task card

This area contains the current memory utilization of the associated module. The available memory is shown both as a bar diagram and as a numerical value (percentage). The numerical value is rounded to an integer value.

Note

If less than 1% of a memory area is utilized, the available portion of this memory area is shown as "99%".

The following memory utilizations are shown:

- Load memory
  If no memory card is inserted, the internal load memory is displayed.
  If a memory card is inserted, the operating system only uses the inserted load memory as the load memory. This is displayed here.
- Work memory
- Retentive memory

See also

Load memory (Page 1171)

Work memory (Page 1172)

Retentive memory areas (Page 1173)

Displaying fill level of all types of memory of an S7-1500 CPU

Where do I find the information I need?

The fill levels of all types of memory of an S7-1500 CPU can be found at the following two places:

- In the display area of the "Memory" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed
- In the display area of the "Memory" pane on the "Online Tools" task card

Display area of the "Memory" group in the "Diagnostics" folder of the online and diagnostics view

This area contains the current memory utilization of the associated module and details of the individual memory areas.

The memory utilization is shown both as a bar diagram and as a numerical value (percentage).

The following memory utilizations are shown:

- Load memory

  Note

  The load memory is located on the SIMATIC memory card.

- Code work memory: work memory for program code
- Data work memory: work memory for data blocks
- Retentive memory

Display area of the "Memory" pane of the "Online Tools" task card

This area contains the current memory utilization of the associated module. The available memory is shown both as a bar diagram and as a numerical value (percentage). The numerical value is rounded to an integer value.

Note

If less than 1% of a memory area is utilized, the available portion of this memory area is shown as "99%".

The following memory utilizations are shown:

- Load memory

  Note

  The load memory is located on the SIMATIC memory card.

- Code work memory: work memory for program code
- Data work memory: work memory for data blocks
- Retentive memory

10.2.1.4 Checking a module for defects

Determining the diagnostic status of a module

Where is the diagnostics status of a module displayed?

The diagnostic status of a module is displayed in the "Diagnostic status" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed.

The "Diagnostics status" group consists of the following areas:

- Status
- Standard diagnostics (for S7-300 and S7-400 only for non-CPU modules)

"Status" area

The following status information is displayed in this area:

- Status of the module as viewed by the CPU, for example:
  - Module available and OK.
  - Module defective.
    If the module experiences a fault and you have enabled the diagnostic error interrupt during configuration, the "Module defective" status is displayed.
  - Module configured, but not available.
    Example: Diagnostics data is not available because the current online configuration differs from the offline configuration.
- Detected differences between the configured and the inserted module. Provided it can be ascertained, the article number will be displayed for the set and actual type.

The scope of the displayed information depends on the selected module.

"Standard diagnostics" area

The following diagnostics information for non-CPU modules is displayed in this area:

- Internal and external faults that relate to the overall module
- Associated diagnostics events

Examples of such diagnostics information are:

- Entire backup failed
- Module defective

Note

Diagnostic interrupts

A diagnostic interrupt can be reported to the CPU only if the module has diagnostic interrupt capability and the diagnostic interrupt has been enabled.

The display of the diagnostic interrupt is a snapshot. Sporadic module defects can be identified in the diagnostics buffer of the respective CPU.

Reading out the diagnostics buffer of a CPU

Where do you read out the diagnostics buffer of a CPU?

You read out the diagnostics buffer of a CPU in the "Diagnostics buffer" group in the "Diagnostics" folder in the Online and Diagnostics view.

Structure of the "Diagnostics buffer" group

The "Diagnostics buffer" group consists of the following areas:

- "Events"
- "Settings"

Diagnostics buffer

The diagnostics buffer is used as a log file for the diagnostics events that occurred on the CPU and the modules assigned to it. These are entered in the order of their occurrence, with the latest event shown at the top.

"Events" area

The "Events" area consists of the following elements:

- Check box "CPU time stamp takes into account local PG/PC time"
- Event table
- "Freeze display" or "Cancel freeze" button
- Details of the event: Event no., event ID, description, time stamp, incoming/outgoing information
- "Help on event", "Open in editor", "Save as ..." buttons

Check box "CPU time stamp takes into account local PG/PC time"

If you have not activated the check box, the diagnostics buffer entries are shown with the module time.

If you have activated the check box, the diagnostics buffer entries are shown with the time given by the following formula:

Displayed time = module time + time zone offset on your programming device / PC

This requires the module time to be identical to UTC.

You should use this setting if you wish to see the times of the diagnostics buffer entries for the module expressed in the local time of your programming device / PC.

Selecting or clearing the check box immediately changes the times displayed for the diagnostics buffer entries.

Note

If you use the "WR_SYS_T" instruction in your program or if you set the real-time clock of the CPU using an HMI device instead of using UTC, we recommend that you clear the "CPU time stamp takes into account local PG/PC time" check box. In this case, the module time is the sole time of concern.

Event table

The following information is displayed in the table for each diagnostics event:

- Sequential number of the entry
  The first entry contains the latest event.
- Date and time of the diagnostics event
  If no date and time are shown, the module has no integral clock.
- Short name of the event and, if applicable, the reaction of the CPU

  Note

  If an individual parameter of a text cannot be determined, the character string "###" is shown in its place.

  If no display text is yet available for new modules or new events, the numbers of the events and the individual parameters are stated as hexadecimal values.

- Icon for information related to incoming/outgoing status
  The following table shows the available icons and their respective meaning.

Icon	Meaning
	Incoming event
	Outgoing event
	Incoming event to which there is no independent outgoing event
	User-defined diagnostics event

- Only for S7-1200 and S7-1500 CPUs: Icon for the severity of the event
  The following table shows the available icons and their respective meaning.

Icon	Meaning
	No maintenance and/or no fault
	Maintenance required
	Maintenance demanded
	Error

You can change the order of the columns, adjust the column widths and remove and add individual columns in the event table. In addition, you can sort as follows: by sequential number, by "Date and time" and by "Event".

"Freeze display" or "Cancel freeze" button

The "Freeze display" or "Cancel freeze" button is only enabled when there is an online connection to the CPU.

The default setting is "Freeze display".

The following happens when you click the "Freeze display" button:

- The current display of the diagnostics buffer entries is frozen.
- The labeling of the button changes to "Cancel freeze".

If an error has occurred in your system, diagnostics events can occur very quickly in succession. This produces a high update rate on the display. Freezing the display allows you to calmly examine the situation in more detail.

If the display is frozen and you click the "Cancel freeze" button, the following happens:

- The display of the diagnostics buffer entries is updated again.
- The labeling of the button changes to "Freeze display".

Note

If you freeze the diagnostics buffer display, the CPU continues to enter events in the diagnostics buffer.

Details of the event

If you select a line in the list of events, you obtain detailed information on the respective event:

- Sequential number of the event in the diagnostics buffer
- Event ID
- Description of the event with event-dependent additional information. Examples of this additional information:
  - Command that caused the event
  - Operating mode switch caused by the diagnostics event
- Time stamp
- Only for S7-1200 and S7-1500 CPUs: Associated I&M data (module and, if required, station and/or device name, rack/slot, plant designation, location designation)
- Priority of the event
- Information on whether the event is an incoming or outgoing event

"Help on event" button

If you click on this button, the selected event is explained in more detail and any remedies given.

Note

For a small number of events, the "Help on event" button is grayed out.

"Open in editor" button

The following table shows if the "Open block" button is active and which function it conceals.

When is the "Open in editor" button enabled?	What happens when you click this button?
If the diagnostics event references the relative address of a block. This is the address of the command that caused the event.	The "Open in editor" function opens the referenced block in the offline view at the programming instruction that causes the error. This allows you to check and, if necessary, change the source code of the block at the specified place and then download it again to the CPU.
If the diagnostics event was triggered by a module.	The "Open in editor" function opens the Device view of the module involved.

"Save as ..." button

If you click this button, the content of the diagnostics buffer is saved in a text file. "Diagnostics", depending on the language, with the extension ".txt" is suggested as the file name. You can however change this name.

"Settings" area

The "Settings" area consists of the following elements:

- "Display events" list
- "Apply settings as default" button
- "Output event information in hexadecimal format" check box

List "Display events:"

There is an check box in this list for every event class (default setting: all check boxes are selected). If you clear a check box, the events of that event class is no longer displayed in the "Events" area. Reselecting the check box displays the associated events once again.

"Apply settings as default" button

If you click this button, the settings are also applied to future occasions when the "Events" tab is opened.

"Output event information in hexadecimal format" check box

If you select the check box, the event IDs in the Events list of the "Events" area is displayed in hexadecimal format. If you clear the check box, the event information is given in text form.

See also

Basic information on the diagnostics buffer (Page 1398)

Saving service data

Purpose

In the event of servicing it may be possible that the SIEMENS Customer Support requires very special information about the state of a module of your system for diagnostic purposes.

If such a case occurs in your system, you will be asked by Customer Support to save the service data of the module and send the resulting file to them.

Where do you carry out the saving of service data of a module?

You carry out the saving of service data of a module in its online and diagnostics view at the following points: In the "Functions" folder in the "Save service data" group"

The "Save service data" group consists of the following areas:

- Online data
- Saving service data

"Online data" area

This area shows the following data of the module:

- Article number
- Firmware version
- Module name (you configured this while configuring the hardware.)
- Rack
- Slot

"Save service data" area

Proceed as follows to create and save a file with special service data:

1. Select the point in the file system at which you want to save the file:
   - You use the path preset in the "Path" field.
   - Click the three-dot (browse) button. In the dialog that opens specify the desired path and enter the file name.
2. Click the "Save data" button.

10.2.1.5 Changing the properties of a module or the programming device/PC

Changing the mode of a CPU

Requirement

There is an online connection to the CPU whose mode you want to change.

Procedure

To change the mode of the CPU, follow these steps:

1. Enable the "Online tools" task card of the CPU.
2. Click the "RUN" button in the "CPU control panel" pane if you want to change the CPU to RUN mode or the "STOP" button if you want to change the CPU to STOP mode.

   Note

   The only button active is the one that can be selected in the current operating mode of the CPU.

3. Acknowledge the confirmation prompt with "OK".

Or:

1. Open the "Online" menu.
2. Choose the "Start CPU" menu command if you want to set the CPU to RUN mode and "Stop CPU" if you want to set the CPU to STOP mode.

   Note

   The only button that is active is the one that can be chosen in the current operating mode of the CPU.

3. Acknowledge the confirmation prompt with "OK".

Or:

1. Click the "Start CPU" button in the toolbar if you want to set the CPU to RUN mode and the "Stop CPU" button if you want to set the CPU to STOP mode.

   Note

   The only button that is active is the one that can be chosen in the current operating mode of the CPU.

2. Acknowledge the confirmation prompt with "OK".

Result

The CPU will be switched to the required operating mode.

Performing a memory reset

Requirement

- There is an online connection to the CPU on which the memory reset is to be performed.
- The CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the memory reset, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To perform a memory reset on a CPU, follow these steps:

1. Enable the "Online Tools" task card of the CPU.
2. Click the "MRES" button in the "CPU control panel" pane.
3. Acknowledge the confirmation prompt with "OK".

Result

The CPU is switched to STOP mode, if necessary, and the memory reset is performed on the CPU.

See also

Basics of a memory reset (Page 1169)

Determining and setting the time of day on a CPU

Where do I find the functions I need?

You determine and change the time of day on a CPU in the "Set time of day" group in the "Functions" folder of the Online and Diagnostics view. This requires an online connection.

Structure of the "Set time of day" group

The "Set time of day" group consists of the following areas:

- Area for reading out and setting the time of day
- Time system (This area does not exist for S7-1200 and will not be examined here.)

Structure of the area for reading out and setting the time of day

This area consists of the following parts:

- Programming device / PC time
  Here the time zone setting, the current date and the current time setting of your programming device / PC are displayed.
- Module time
  Here the date and time values currently read from the module (for example the CPU), are converted to local time and date and displayed.
  If the "Take from PG/PC" check box is selected, when you click the "Apply" button, the date and the PG/PC time converted to UTC are transferred to the module.
  If the "Take from PG/PC" check box is not selected, you can assign the date and time for the integrated clock of the module. After clicking the "Apply" button, the date and the time recalculated to UTC time are transferred to the module.

Updating the firmware of a module

Performing a firmware update

Using firmware files, you can update the firmware of a module.

Requirements

- The module is connected online.
- The module supports a firmware update.
- For those modules that require a supply voltage to perform the firmware update correctly: The supply voltage of the module is secured. For details, see the documentation of the module.

Procedure

To perform a firmware update, follow these steps:

1. Open the module in the Online and Diagnostics view.
2. Select the "Firmware update" group in the "Functions" folder.

   Note

   For S7-1500-CPUs, this group is subdivided into "PLC" and "Display".

3. Click the "Browse" button in the "Firmware update" area in order to select the path to the firmware update files.
4. Select one of these files. The table then lists all modules for which an update is possible with the selected firmware file.

5. Optional: Select the "Run firmware after update" check box to reset the module after the load operation and to start the new firmware.
6. Click the "Start update" button. If the selected file can be interpreted by the module, it is downloaded to the module. If the mode of the CPU needs to be changed, you will be prompted to do this in dialogs.

Warning

Invalid plant states possible

An S7-1500 CPU immediately goes to STOP mode when you start the firmware update, which can have an effect on the operation of an online process or a machine.
Unexpected operation of a process or a machine can result in severe or fatal injuries and/or damage to property.

Note

After you have run a firmware update, you will need to replace the module involved with the same module with the current firmware version in the hardware configuration of your project. The engineering configuration then matches the actual physical configuration again.

"Run firmware after update" check box

If you have not selected the "Run firmware after update" check box, the previous firmware remains active until the module is reset (for example by cycling power). The new firmware only becomes active after the module has been reset.

If you have selected the check box, the module is automatically reset after the firmware has been downloaded and it then continues with the new firmware.

Activating the firmware following the update has the following consequences:

- A station executes a restart. This means that all modules in the station become unavailable.
- If the corresponding CPU is in RUN, activating the firmware can lead to access errors or other problems in the user program and might even mean that the CPU remains permanently in STOP.

  Note

  For some CPUs, the "Run firmware after update" check box is grayed out and deactivated. In this case, you must restart the CPU manually.

  For S7-1500 CPUs, the "Run firmware after update" check box is grayed out and selected. In this case, the new firmware is activated immediately after the download operation.

See also

Replacing a hardware component (Page 574)

Resetting an S7-1200 CPU to factory settings

Requirement

- There is no memory card inserted in the CPU.
- There is an online connection to the CPU that you want to reset to the factory settings.
- The CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the reset operation, you can place it in STOP mode by answering the security prompt with yes.

Procedure

To reset an S7-1200 CPU to factory settings, follow these steps:

1. Open the Online and Diagnostics view of the CPU.
2. Select the "Reset to factory settings" group in the "Functions" folder.
3. Select the "Keep IP address" check box if you want to keep the IP address or the "Delete IP address" check box if you want to delete the IP address.

   Note

   The two check boxes mentioned are only available if the module to be reset is able to choose whether to retain or delete the IP address.

4. Click the "Reset" button.
5. Acknowledge the confirmation prompt with "OK".

Result

The module is switched to STOP mode if necessary and the settings are then reset to factory settings. This means:

- The work memory and the internal load memory and all operand areas are cleared.
- All parameters are reset to their defaults.
- The diagnostic buffer is cleared.
- The time is reset.
- The IP address is kept or deleted depending on which setting you made.

Resetting an S7-1500 CPU to factory settings

Requirement

- If you start a reset to factory settings from the project context, an online connection to the relevant CPU must exist.
- The relevant CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the reset operation, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To reset an S7-1500 CPU to factory settings, follow these steps:

1. Open the Online and Diagnostics view of the CPU (either from the project context or via "Accessible devices").
2. Select the "Reset to factory settings" group in the "Functions" folder.
3. Select the "Keep IP address" check box if you want to keep the IP address or the "Delete IP address" check box if you want to delete the IP address.

   Note

   With "Delete IP address", all IP addresses are deleted. This applies regardless of how you created the online connection.

   If a memory card is inserted, selecting the "Delete IP address" option causes the following: The IP addresses are deleted and the CPU is reset to factory settings. Then, the configuration (including IP addresses) that is stored on the memory card is transferred into the CPU (see below). If the memory card was formatted before resetting to factory settings or if it is empty, no IP address is transferred into the CPU.

4. Click the "Reset" button.
5. Acknowledge the confirmation prompt with "OK".

Result

The module is switched to STOP mode if necessary and the settings are then reset to factory settings. This means:

- The work memory and the internal retentive system memory and all operand areas are cleared.
- All parameters are reset to their defaults.
- The diagnostic buffer is cleared.
- The time of day is reset.
- The I&M data are deleted except for I&M0 data.
- The runtime meters are reset.

- The IP address is kept or deleted depending on which setting you made.
- If a memory card was inserted prior to the reset to factory settings, the configuration contained on the memory card (hardware and software) is downloaded to the CPU.

Formatting an S7-1500 memory card

Requirement

- If you start the formatting of the memory card from the project context, an online connection to the relevant CPU must exist.
- The relevant CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start a formatting operation, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To format an S7-1500 memory card, follow these steps:

1. Open the Online and Diagnostics view of the CPU (either from the project context or via "Accessible devices").
2. Select the "Format memory card" group in the "Functions" folder.
3. Click the "Format" button.
4. Answer the safety prompt with "Yes".

Result

- The memory card is formatted.
- The CPU is temporarily unavailable.
- The project data on the CPU are deleted with the exception of the IP address.
- If you start the formatting of the memory card from the project context, the Online and Diagnostics view remains open. If formatting is started via "Accessible devices", the Online and Diagnostics view will close.

Assigning an IP address to a PROFINET IO device

Basic information on assigning an IP address to a PROFINET IO device

Overview

All PROFINET IO devices work with the TCP/IP protocol and therefore require an IP address for operation on Industrial Ethernet. Once an IP address has been assigned to an IO device, it can be accessed via this address. You can then download configuration data or perform diagnostics, for example.

Requirement

- The Ethernet LAN connection must already be established.
- The Ethernet interface of your programming device or PC must be accessible.
- The IO device that is to be assigned an IP address must be in the same IP band as the programming device or PC.

Starting the address assignment via "Accessible devices"

Requirement

- The devices accessible via the associated interface of the PG/PC are displayed in the project tree (to display these, either double-click "Update accessible devices" in the project tree or select the "Accessible devices..." command in the "Online" menu.).
- You have double-clicked "Online access" -> <Selected interface> -> <PROFINET IO device> -> "Online & Diagnostics" in the project tree to open the Online and Diagnostics view.

Procedure

1. Open the "Functions" folder and the "Assign IP address" group inside this folder. The "MAC address" field displays the MAC address of the PROFINET IO device. The "Accessible devices" button is grayed out.
2. Enter the desired IP address.
3. Enter the subnet mask.
4. If a router is to be used, select the "Use router" check box and enter its IP address.
5. Click the "Assign IP address" button.

Result

The IP address is permanently assigned to the IO device or to the relevant PROFINET interface of the IO device. It is retained even through a startup or a power failure.

Note

For an S7-1500 CPU, you can also use the above-described method to change the IP address of a PROFINET interface, if a project has already been downloaded to the CPU via this interface. This overwrites the IP address downloaded via the project.

See also

Retentivity of IP address parameters and device names (Page 1124)

Starting the address assignment from the project context

Requirement

- An online connection to the PROFINET IO device exists.
- You have opened the Online and Diagnostics view of the PROFINET IO device from the project context.
- The PROFINET IO device is not assigned to any IO controller.

Procedure

1. Open the "Functions" folder and the "Assign IP address" group inside this folder.
2. Click the "Accessible devices" button in order to identify the devices that can be accessed. Note: For an S7-1500 CPU, there are two entries here because it has two PROFINET interfaces.
3. Select the IO device. The "IP address" field, "Subnet mask" field, "Use router" check box and "Router address" field are grayed out and contain the node properties you used to establish the current online access.
4. Click the "Assign IP address" button.

Result

The IP address is permanently assigned to the IO device or to the relevant PROFINET interface of the IO device. It is retained even through a startup or a power failure.

See also

Retentivity of IP address parameters and device names (Page 1124)

Assigning a PROFINET device name

Basic information on assigning a name to a PROFINET IO device

Device name

Before an IO device can be addressed by an IO controller, it must have a device name. This procedure was chosen for PROFINET because names are easier to handle than complex IP addresses.

Assigning a device name to a PROFINET IO device is comparable to setting the PROFIBUS address for a DP slave.

An IO device has no device name in its delivery state. For an IO controller to address an IO device, it must first be assigned a device name using the programming device or PC. It is now ready to transfer the configuration information including the IP address during startup or exchange user data in cyclic operation.

Rules for the device name

The device name is subject to the following limitations:

- Restricted to a total of 240 characters (lower case letters, numbers, dash, or dot)
- A name component within the device name, which is a character string between two dots, must not exceed 63 characters.
- No special characters such as umlauts, brackets, underscore, slash, blank space, etc. The only special character permitted is the dash.
- The device name must not begin or end with the "-" character.
- The device name must not begin with a number.
- The device name form n.n.n.n (n = 0, ... 999) is not permitted.
- The device name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).

Where do I find the function I am seeking?

You can assign the name of a PROFINET IO device at the following places:

- In the online and diagnostics view of the device in the "Assign name" group of the "Functions" folder. The user interface for this group differs depending on how you open the online and diagnostics view:
  - Open via "Accessible devices"
  - Open from the project context
- In the "Assign PROFINET device name" dialog

See also

Assigning a name in the online and diagnostics view opened via "Accessible devices" (Page 1392)

Assigning a name in the online and diagnostics view opened from the project context (Page 1392)

Assigning a name in the "Assign PROFINET device name" dialog (Page 1393)

Assigning a name in the online and diagnostics view opened via "Accessible devices"

Requirement

- You have opened the online and diagnostics view of the PROFINET IO device using "Update accessible devices" (in the project tree) or "Accessible devices..." ("Online" menu).

Procedure

1. Open the "Functions" folder and the "Assign name" group inside this folder. The "Type" field displays the module type of the PROFINET IO device.
2. Enter the required device name in the "PROFINET device name" input box.
3. Optional: Select the "Flash LED" check box in order to run an LED flash test on the PROFINET IO device. In this way, you verify that you are naming the desired IO device.

   Note

   The LED flash test is not supported by all PROFINET IO devices.

   The LED flash test runs until you cancel it. This is done, for example, by clearing the "Flash LED" check box or by closing the online and diagnostics view.
4. Click "Assign name".

Result

The entered name is assigned to the PROFINET IO device.

Assigning a name in the online and diagnostics view opened from the project context

Requirement

- You have opened the online and diagnostics view of the PROFINET IO device from the project context.
- The PROFINET IO device can be accessed using at least one PG/PC interface.

Procedure

1. Open the "Functions" folder and the "Assign name" group inside this folder. The "PROFINET device name" drop-down list displays the current name in the offline project, and the "Type" box shows the module type of the PROFINET IO device.

   Note

   For CPUs with several PROFINET interfaces, the names of all existing PROFINET interfaces are displayed in the offline project.

2. Choose a different name from the drop-down list, if necessary.

   Note

   In steps 3 to 5, you determine the IO devices that are present in the PROFINET subnet.

3. In the "PG/PC interface for assignment" drop-down list, select the PG/PC interface you want to use to establish the online connection.
4. Optional: Use the three check boxes to make a selection from all IO devices available online.
5. Click the icon for determining the IO devices present in the PROFINET subnet. The table is then updated.
6. Select the desired IO device in the table.
7. Optional: Select the "Flash LED" check box in order to run an LED flash test on the PROFINET IO device. In this way, you verify that you are naming the desired IO device.

   Note

   The LED flash test is not supported by all PROFINET IO devices.

   The LED flash test runs until you cancel it. This is done, for example, by clearing the "Flash LED" check box, by selecting another IO device in the table, or by closing the online and diagnostics view.
8. Click "Assign name".

Result

The selected name is assigned to the PROFINET IO device or one of its PROFINET interfaces.

Assigning a name in the "Assign PROFINET device name" dialog

Requirement

- You have opened the "Assign PROFINET device name" dialog from the network view (from the shortcut menu of a PN/IE connection).
- The PROFINET IO device can be accessed using at least one PG/PC interface.

Procedure

1. The following is displayed in the "PROFINET device name" drop-down list:
   - the name of the interface that was used to open the project in the current offline project
   - the names of those IO devices that are connected by means of this interface

   The "Type" field displays the module type of the PROFINET IO device.
   Choose a different name from the drop-down list, if necessary.

   Note

   In steps 2 to 4, you determine the IO devices that are present in the PROFINET subnet.

2. In the "PG/PC interface for assignment" drop-down list, select the PG/PC interface you want to use to establish the online connection.
3. Optional: Use the three check boxes to make a selection from all IO devices available online.
4. Click the icon for determining the IO devices present in the PROFINET subnet. The table is then updated.
5. Select the desired IO device in the table.
6. Optional: Select the "Flash LED" check box in order to run an LED flash test on the PROFINET IO device. In this way, you verify that you are naming the desired IO device.

   Note

   The LED flash test is not supported by all PROFINET IO devices.

   The LED flash test runs until you cancel it. This is done, for example, by clearing the "Flash LED" check box or by selecting another IO device in the table.
7. Click "Assign name".

Result

The selected name is assigned to the PROFINET IO device or the interface that was used to open the dialog.

Calibrating an S7-1500 analog module

Calibrating an S7-1500 analog module - Overview

Where do you calibrate an S7-1500 analog module?

You calibrate an S7-1500 analog module in its Online and Diagnostics view in the "Calibrate" group of the "Functions" folder.

Overview of the function scope of the calibrating function

You can perform the following functions for an S7-1500 analog module in the "Calibrate" group:

- Specifying the current calibration of all channels
- Calibrating a channel
- Canceling a running calibration
- Resetting the calibration of a channel to the factory settings

Requirement for the calibrating function described below

The following is required for the calibrating function described below:

- You have opened the Online and Diagnostics view from the project context (thus not from the project tree or via the "Online" menu).
- There is an online connection to the analog module that is to be calibrated.
- Offline and online configuration are identical.

Calibrating an S7-1500 analog module

Overview of the calibration of a channel of an S7-1500 analog module

The calibration of a channel of an S7-1500 analog module consists of the following steps:

1. Start calibration
2. Perform the second step up to the next to last step of the calibration
3. Complete calibration

These steps are described in more detail in the following section.

Requirement

- You have opened the Online and Diagnostic view of the S7-1500 analog module from the project context and are in the "Calibrate" group of the "Functions" folder.
- The associated CPU is online.
- No calibration is currently running on the analog module (if you want to start the calibration) or the last step initiated has been performed successfully (if you want to resume or complete the calibration).

Procedure for starting the calibration

To start the calibration, follow these steps:

1. In the overview table, select the line that belongs to the channel to be calibrated.
2. Click the "Start manual calibration" button.

The user interface then changes as follows:

- The overview table and the "Start manual calibration" button and "Set to factory settings" buttons become inactive.
- The step display is activated and displays the numbers of the current and last steps.
- The "Command" field becomes active and indicates what the user must do in the next calibration step.
- The "Status" field becomes active and shows the current status of the calibration, e.g., "Calibration successfully started".
- The "Measured value" field becomes active. For an input module a value is displayed here; you must enter a value here for an output module.
- The "Cancel" button becomes active.
- The "Next" button becomes active. This button can be used to advance to the next step of the calibration.

Procedure for the second to the next to last step of the calibration

Follow these steps:

1. Click the "Next" button.

The fields of the user interface described above are then updated.

Procedure for the last step of the calibration

Follow these steps:

1. Click the "Next" button.

The user interface then changes as follows:

- The overview table becomes active.
- The calibration display of the calibrated channel is updated.
- The "Start manual calibration" button and "Set to factory settings" buttons become active.
- The step display is deactivated and the numbers of the current step and last steps are empty.
- The "Command" field becomes inactive and is empty.
- The "Status" field becomes inactive and shows the last status of the calibration, e.g., "Calibration successfully finished".
- The "Measured value" field becomes inactive and is empty.
- The "Cancel" button becomes inactive.
- The "Next" button becomes inactive.

Error occurrence

If an error occurs during the calibration, the module cancels the calibration. Afterwards, the channel that was to be calibrated has the same settings as before the start of the calibration.

Except for the "Status" field, the user interface appears the same after the occurrence of an error as before the calibration. The "Status" field displays the error that the module detected during the calibration.

Canceling a running calibration of an S7-1500 analog module

Requirement

- You have opened the Online and Diagnostic view of the S7-1500 analog module from the project context and are in the "Calibrate" group of the "Functions" folder.
- The associated CPU is online.
- A calibration is currently running on the analog module.

Procedure

To cancel a running calibration, follow these steps:

1. Click the "Cancel" button.

Result

The running calibration is canceled, and afterwards the channel to be calibrated has the same settings as before the calibration.

All operator controls in the user interface are deactivated until the cancelation is complete. Except for the "Status" field, the user interface appears the same afterwards as before the calibration. The "Status" field displays the result of the cancelation.

Resetting an S7-1500 analog module to factory settings

Requirement

- You have opened the Online and Diagnostic view of the S7-1500 analog module from the project context and are in the "Calibrate" group of the "Functions" folder.
- The associated CPU is online.

Procedure

To reset a channel of an S7-1500 analog module to factory settings, follow these steps:

1. Select the line associated with the channel to be reset in the overview table.
2. Click the "Set to factory settings" button.

Result

All operator controls in the user interface are deactivated until the reset operation is complete. Except for the "Status" field, the user interface appears the same afterwards as before the reset operation. The "Status" field displays the result of the reset operation.

10.2.1.6 Diagnostics in STOP mode

Basic information on the diagnostics buffer

Function

The operating system of the CPU enters the errors detected by the CPU and the diagnostics-capable modules into the diagnostics buffer in the order in which they occurred. This includes the following events:

- Every mode change of the CPU (POWER UP, change to STOP mode, change to RUN mode)
- Every hardware and diagnostic error interrupt

The top entry contains the most recent event. The entries in the diagnostics buffer are stored permanently. They are retained even if the power supply fails and can only be deleted by resetting the CPU to factory settings.

A diagnostics buffer entry contains the following elements:

- Time stamp
- Error ID
- Additional information specific to the error ID

Advantages of the diagnostics buffer

The diagnostics buffer offers the following advantages:

- After the CPU has changed to STOP mode, you can evaluate the last events prior to the STOP so that you can locate and identify the cause of the STOP.
- You can detect and eliminate the causes of errors more quickly and thus increase the availability of the system.
- You can evaluate and optimize the dynamic system response.

Organization of the diagnostics buffer

The diagnostics buffer is a ring buffer. The maximum number of entries for the S7-1200 CPUs is 50. When the diagnostics buffer is full and a further entry needs to be made, all existing entries are shifted by one position (which means that the oldest entry is deleted) and the new entry is made at the top position that is now free (FIFO principle: first in, first out).

Evaluation of the diagnostics buffer

The contents of the diagnostics buffer can be accessed as follows:

- Using the Online and Diagnostics view

The evaluation of events occurring prior to the error event (e.g., transition to STOP mode) allows you to obtain a picture of the possible causes or to zero in more closely or specify in more detail the possible causes (depending on the error type).

Read the detailed information about the events carefully and use the "Help on event" button to obtain additional information and possible causes of individual entries.

Note

To make the best use of the time stamp information on the diagnostics buffer entries in time-critical systems, it is advisable to check and correct the date and time of day on the CPU occasionally.

Alternatively, it is possible to perform a time-of-day synchronization using an NTP time server.

See also

Resetting an S7-1200 CPU to factory settings (Page 1386)

Determining the cause of a STOP of a CPU (Page 1399)

Determining and setting the time of day on a CPU (Page 1383)

Assigning the clock parameters (Page 1193)

Determining the cause of a STOP of a CPU

Requirement

The CPU you want to analyze is in STOP mode.

Procedure

To find out the reason why a CPU changed to STOP, follow these steps:

1. Open the online and diagnostics view of the CPU.
2. Select the "Diagnostics buffer" group from the "Diagnostics" folder.
3. Evaluate the events occurring prior to the transition to STOP mode. Use this to obtain a picture of the possible causes or to zero in on or specify in more detail the possible causes (depending on the error type).
   Read the detailed information about the events carefully and use the "Help on event" button to obtain additional information and possible causes of individual entries.

Result

You were able to zero in on or determine in more detail the cause of the CPU STOP.

Note

If the analysis does not enable you to overcome the problem, contact Customer Support. In this case, use the "Save as" button to back up the content of the diagnostics data to a text file and submit it to Customer Support.

See also

Reading out the diagnostics buffer of a CPU (Page 1376)

10.2.1.7 Online accesses in the Online and Diagnostics view

Displaying status of the online connection

Requirement

- The associated device can be accessed using at least one PG/PC interface.

Procedure

1. Open the online and diagnostics view for the device whose online connection status you want to display.
2. Select the "Online access" group.

   Note

   The "Online access" group exists only for CPUs and some CPs. However, if you have opened the online and diagnostics view using the "Show/update accessible devices" function, it is not displayed.

Result

The status of the online connection is displayed in the "Status" area both graphically and in text form.

Specifying a PG/PC interface, going online

Requirement

- The associated device can be accessed using at least one PG/PC interface.
- There is currently no online connection to the relevant device.

Procedure

1. Open the Online and Diagnostics view of the device to which you want to establish an online connection.
2. Choose the "Online access" group and the "Online access" area within this group.

   Note

   The "Online access" group exists only for CPUs and some CPs. If you have opened the Online and Diagnostics view using the "Show/update accessible devices" function, it is not displayed.

3. If an online connection was established previously for the device, the associated data for this online connection is preset in the drop-down lists. In this case, you can immediately continue with the last step of this operating instruction, provided you have not changed the IP address in the meantime using the Online and Diagnostics view.
4. Choose the interface type in the "Type of PG/PC interface" drop-down list.
   The "PG/PC interface for online access" drop-down list then shows only the interfaces of the programming device or PC that match the selected interface type.
5. In the "PG/PC interface for online access" drop-down list, select the programming device or PC interface via which you want to establish the online connection.
6. Optional: Click the "Properties" button to change the properties of the associated CP.
7. In the "Connection to subnet" drop-down list, select the subnet via which the device is connected to the PG/PC interface.

   Note

   The PG/PC interface is connected to an interface of a device.

   If you only want to access this device, select the setting "Directly at slot <interface name>" in the drop-down list.

   If, however, you want to access another device by means of routing, create a subnet at this interface in the device configuration and then select this subnet in the drop-down list.

8. If the device is accessible via a gateway, select the gateway that connects the two subnets involved in the "1st gateway" drop-down list.
9. In the "Device address" entry field, enter the IP address of the device to which you want to establish an online connection, if necessary.

   Note

   For CPUs with multiple IP addresses, select the IP address of the PROFINET interface you want to use to establish an online connection from the "Device address" drop-down list.

10. Alternatively: Click the "Show accessible devices" button and choose the device from the list of accessible devices to which you want to establish an online connection.
11. Click the "Go online" button.

Result

The online connection to the desired device is established.

Going offline

Requirement

- There is currently an online connection to the relevant device.

Procedure

1. Open the online and diagnostics view of the device for which you want to disconnect the online connection.
2. Choose the "Online access" group and the "Online access" area within this group.

   Note

   The "Online access" group exists only for CPUs and some CPs. However, if you have opened the online and diagnostics view using the "Show/update accessible devices" function, it is not displayed.

3. Click the "Go offline" button.

Result

The online connection to the desired device will be disconnected.

Performing the flash test for a device with an online connection

Requirement

- There is currently an online connection to the relevant device.
- The FORCE function is not active.

Procedure

1. Open the online and diagnostics view of the device for which you want to perform a flash test.
2. Choose the "Online access" group and the "Status" area within this group.

   Note

   The "Online access" group exists only for CPUs and some CPs. However, if you have opened the online and diagnostics view using the "Show/update accessible devices" function, it is not displayed.

3. Select the "LED flash test" check box.

Result

- On an S7-1200 CPU, the RUN/STOP, ERROR and MAINT LEDs flash.
- On an S7-1500 CPU, the RUN/STOP, ERROR, and MAINT LEDs flash.
- On an S7-300 or S7-400 CPU, the FRCE LED flashes.

The LEDs flash until you cancel the flash test. This is done, for example, by clearing the "LED flash test" check box, by changing to another group of the online and diagnostics view, or by changing settings in the "Online access" area.

10.2.1.8 Checking PROFIBUS DP subnets for faults

Basic information on the diagnostic repeater

What is the diagnostic repeater?

The diagnostic repeater is a repeater that can monitor a segment of an RS 485 PROFIBUS subnet (copper cable) during operation and signal line errors to the DP master via a diagnostics message frame.

The diagnostic repeater detects, localizes and visualizes line errors during operation at an early stage. As a result, problems in the system are identified early and production downtimes will be minimized.

Function of the diagnostic repeater

The diagnostic repeater can perform line diagnostics on the DP2 and DP3 segments because it has a measuring circuit for these segments.

The line diagnostics run in two steps:

- Step 1: Topology determination
  You start the topology determination by calling the "DP_TOPOL" instruction in your program. The diagnostic repeater then determines the PROFIBUS addresses and the distance of the devices and creates a topology table.
- Step 2: Error localization
  The diagnostic repeater checks the lines during operation. It determines the distance to the point of the error and the reason for the error; it then issues a diagnostics alarm with relative information on the error location.

Display of detailed information on the determined error location

You receive detailed information on the determined error location in the Online and Diagnostics view of the diagnostic repeater.

- By means of icons
- By means of a display with graphics and text

See also

Displaying the status of the segment diagnostics using icons (Page 1404)

Displaying the status of the segment diagnostics using graphics and text (Page 1404)

Displaying the status of the segment diagnostics using icons

Where do I find the information I need?

The icons for the status of the segment diagnostics are available:

- In the expanded "Segment diagnostics" folder in the navigation pane of the Online and Diagnostics view of the relevant diagnostic repeater.

The diagnostics icon associated with the segment will be displayed behind the segment designation. It must be noted here that line errors will be displayed for the DP2 and DP3 segments only. The DP1 and programming device segments do not display errors in the form of a diagnostics icon; rather, they signal only a few bus errors.

Diagnostics icons

The following table shows the available icons and their meaning.

Icon	Meaning
	Segment is error-free
	Segment contains errors
	Segment is deactivated

Displaying the status of the segment diagnostics using graphics and text

Where is the status of the segment diagnostics displayed with graphics and text?

The status of the segment diagnostics will be displayed using graphics and text in the "DP1", "DP2", "DP3", and "PG" groups of the "Segment diagnostics" folder in the Online and Diagnostics view of the relevant diagnostic repeater.

Structure of the "DP1", "DP2" "DP3", and "PG" groups

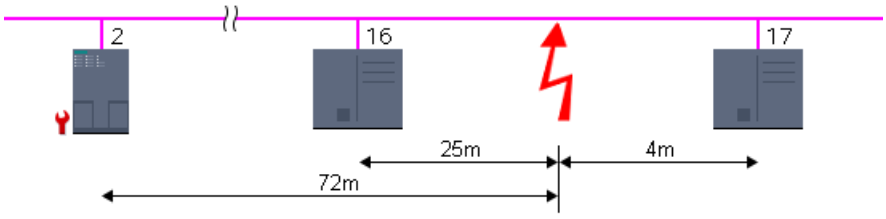
The "DP1", "DP2", "DP3", and "PG" groups consist of the following elements:

- "Error location" field
- "Error" field
- "Resolution" field
- "Help on event" button
- "Freeze display" or "Cancel freeze" button

"Error location" field

This field displays the error location graphically, provided the diagnostic repeater can determine the location.

The following picture shows an example for a line error occurring in the DP2 segment.



In this example, the diagnostic repeater has the PROFIBUS address 2, and a line error has occurred between the devices with PROFIBUS addresses 16 and 17. This line error is located 25 m from device 16, 4 m from device 17, and 72 m from the diagnostic repeater.

"Error" field

The error is explained in plain text in this field.

"Resolution" field

Here, you will find actions for resolving the error.

"Help on event" button

Click this button to obtain a more detailed explanation of the error and additional details on resolving the error, if applicable.

"Freeze display" or "Cancel freeze" button

The "Freeze display" or "Cancel freeze" button is only enabled when there is an online connection to the diagnostic repeater.

The default setting is "Freeze display".

The following happens when you click the "Freeze display" button:

- The current display of the segment diagnostics is frozen.
- The labeling of the button changes to "Cancel freeze".

If the display is frozen and you click the "Cancel freeze" button, the following happens:

- The display of the segment diagnostics is updated again.
- The labeling of the button changes to "Freeze display".

10.2.2 Connection diagnostics

10.2.2.1 Overview of connection diagnostics

Basics

Connection diagnostics, as described below, refers to the diagnostics of communication connections.

The connection diagnostics is started each time an online connection is established to a module (CPU or CP) that participates in one or more communication services. The connection status is updated automatically in the background.

In the case of one-way connections, an online connection must exist to the communication partner that has established the communication connection.

On connections configured at both ends, a distinction between the following two situations must be made:

- If there is an online connection to only one connection endpoint, only the part of the connection belonging to this connection endpoint can be diagnosed.
- If there is an online connection to both connection endpoints, both parts of the connection (and therefore the entire connection) can be diagnosed.

Basic connections diagnostics options

Connection diagnostics can be performed as follows:

- Using icons on the connection status display
  This display is generated in the connection table.
- Through detailed connection diagnostics
  This step is available in the "Diagnostics > Connection information" area of the Inspector window.

Requirement for the connection diagnostics described below

You can display the details of either all the communication connections created in the project (default) or selected communication connections in the connection table.

The connection diagnostics described in the following assume that you display the details of selected communication connections. To do this, clear the "Show all connections" option in the shortcut menu.

10.2.2.2 Displaying the connection status using icons

Content of connection table without an online connection

- For a CPU or CP, the connection table lists the communication connections (including properties) configured offline, if an online connection is not established.

Content of connection table with an online connection

After the online connection has been established, the properties of the communication connections listed offline will be expanded to include diagnostics icons for the connection status ("Online status" column).

In addition, entries for all communication connections that exist online only (e.g., connections for the instructions for Open User Communication, programming device and OP connections, connections for web server access) will now be added to the connection table.

For connections that exist online or offline only, the diagnostics icon at the bottom right is combined with a smaller additional comparison status icon.

Diagnostics icons for communication connections

The following table shows the diagnostics icons for communication connections.

Icon	Meaning
	Connection setup
	Connection not setup / is being setup
	Connection not available

Diagnostics icons for the comparison status

The diagnostic icons for communication connections can be combined at the bottom right with additional smaller icons that indicate the result of the online/offline comparison. The following table shows the available comparison icons and their meaning.

Icon	Meaning
	Connection exists online only
	Connection exists offline only

10.2.2.3 Detailed connection diagnostics

Detailed connection diagnostics - overview

Where do I perform detailed connection diagnostics?

To perform detailed connection diagnostics, go to the "Diagnostics > Connection" information of the Inspector window.

How do I open the "Diagnostics > Connection information" area of the Inspector window?

The following options are available for opening the "Connection information" tab of the Inspector window.

- Select the line of the relevant connection in the connection table. Click the "Diagnostics" and "Connection information" tabs one after the other in the Inspector window.
- Double-click the diagnostics icon of the relevant connection in the connection table.
- This step takes you to the programming editor for a S7 communication instruction or open user communication instruction. Double-click the diagnostic icon of the instruction (stethoscope).

Structure or the "Diagnostics > Connection information" area of the Inspector window

Requirements: the content of the "Connection information" tab has been filled, and an online connection to at least one end point of the relevant connection has been established.

If a module has been selected (network view), the tab will contain the following group:

- Connection resources (for S7-1200 and S7-1500)

If a connection has been selected (connection table), it will contain the following groups:

- Connection details
- Address details of the connection (for S7-1200 and S7-1500)

Determining online connection resources for S7-1200

Where do you determine the online connection resources?

The online connection resources are obtained in the "Connection resources" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window. It is displayed only if you have selected a module in the network view to which an online connection exists.

Number of connection resources

- Maximum number: Specifies the maximum number of available connection resources of the module.
- Not assigned: Indicates how many connection resources are not yet assigned. If connection resources are already reserved for certain types of communication, then the unreserved connection resources cannot always be used for the various connection types.

Reserved and currently assigned connection resources

For the communication types indicated below, the connection resources that are reserved and currently assigned by the module will be displayed.

Communication type	Meaning
PG communication	Resources for connections between the module and programming devices (for example, for the establishment of a connection from the project tree, for online diagnostics, etc.)
HMI communication	Resources for connections between the module and HMI devices
Open User Communication	Resources for connection of open user communication instructions
S7 communication	Resources for configured S7 connections, through which data can be exchanged by calling instructions in the user program.
Other communication	Specifies other assigned connection resources for which connection resources are not reserved.

Determining online connection resources for S7-1500

Where do you determine the online connection resources?

The online connection resources are obtained in the "Connection resources" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window. It is displayed only if you have selected a module in the network view to which an online connection exists.

Description of the detailed display of the connection resources

The detailed display of the connection resources includes:

- Number of available connection resources
- Number of configured connection resources
- Number of connection resources still available

For a description of these, go to here .

Determining connection details

Where do I determine the connection details?

The connection details are obtained in the "Connection details" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window.

When is the "Connection details" group filled in?

The following requirements must be met to fill in the "Connection details" group on the "Connection information" tab:

- An online connection to the end point of the relevant connection must exist.
- You have selected a line in the connection table.

Structure of the "Connection details" group

The "Connection details" group consists of the following elements:

- Local ID (hex)
- Connection type (for S7-1200 and S7-1500)
- Protocol
- Connection status: icon and description
- Details
- Last status change (for S7-300 and S7-400 only)

Determining the address details of a connection

Where do I determine the address details of a connection?

The address details of a connection are obtained in the "Connection address details" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window.

For which CPUs is the "Connection address details" group available?

The "Connection address details" group of the "Connection information" tab is available for S7-1200 and S7-1500 CPUs.

When is the "Connection address details" group filled in?

The following requirements must be met to fill in the "Connection address details" group on the "Connection information" tab:

- An online connection to the end points of the relevant connection must exist.
- You have selected a line in the connection table.

Structure of the "Connection address details" group

The address details relevant to the connection type are specified for the two communication partners.

Programming the PLC 11

11.1 Creating the user program

11.1.1 Programming basics

11.1.1.1 Operating system and user program

Operating system

Function

The operating system is contained in every CPU and organizes all CPU functions and sequences that are not associated with a specific control task.

The tasks of the operating system, for example, include the following:

- Processing a warm restart
- Updating the process image of the inputs and outputs
- Calling the user program
- Detecting interrupts and calling interrupt OBs
- Detecting and handling errors
- Managing memory areas

The operating system is a component of the CPU and is already installed there upon delivery.

See also

User program (Page 1411)

User program

Function

The user program contains all functions that are necessary for processing your specific automation task.

The tasks of the user program include:

- Checking the requirements for a (warm) restart using startup OBs, for example, limit switch in correct position or safety relay active.
- Processing process data, e.g. linking binary signals, reading in and evaluating analog values, defining binary signals for output, and outputting analog values
- Reaction to interrupts, for example, diagnostic error interrupt if the limit value of an analog expansion module is overshot.
- Error handling in normal program execution

You write the user program and load it into the CPU.

See also

Operating system (Page 1411)

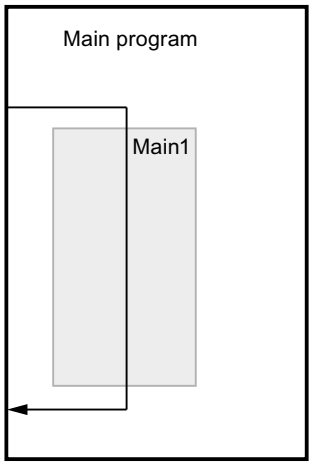
11.1.1.2 Blocks in the user program

Linear and structured programming

Linear programming

Solutions for small automation tasks can be programmed linearly in a program cycle OB. This is only recommended for simple programs.

The following figure shows a linear program schematically: The "Main1" program cycle OB contains the complete user program.



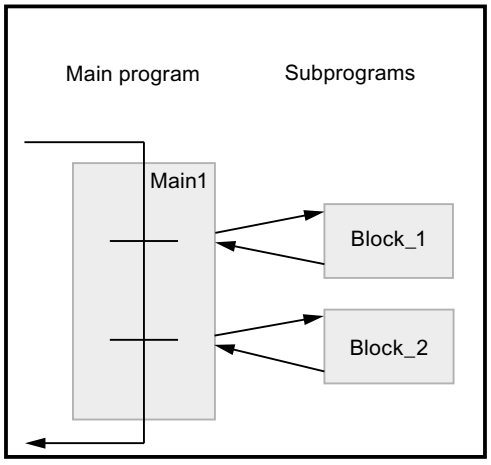
Structured programming

Complex automation tasks can be more easily handled and managed by dividing them into smaller sub-tasks that correspond to the technological functions of the process or that can be reused. These sub-tasks are represented in the user program by blocks. Each block is then an independent section of the user program.

Structuring the program offers the following advantages:

- Extensive programs are easier to program through the structure.
- Individual program sections can be standardized and used repeatedly with changing parameters.
- Program organization is simplified.
- Changes to the program can be made more easily.
- Debugging is simplified since separate sections can be tested.
- Commissioning is simplified.

The following figure shows a structured program schematically: The "Main1" program cycle OB calls subprograms one after the other that execute defined subtasks.



Overview of the block types

Block types

Different BLOCK types are available to perform tasks within an automation system. The following table shows the available block types:

Block type	Brief description
Organization blocks (Page 1414) (OB)	Organization blocks define the structure of the user program.
Functions (Page 1414) (FC)	Functions contain program routines for recurring tasks. They have no "memory".
Function blocks (Page 1415) (FB)	Function blocks are code blocks that store their values permanently in instance data blocks, so that they remain available even after the block has been executed.

Block type	Brief description
Instance data blocks (Page 1417)	Instance data blocks are assigned to a function block when it is is called for the purpose of storing program data.
Global data blocks (Page 1416)	Global data blocks are data areas for storing data that can be used by any blocks.

Organization blocks (OB)

Definition

Organization blocks (OBs) form the interface between the operating system and the user program. They are called by the operating system and control, for example, the following operations:

- Startup characteristics of the automation system
- Cyclic program processing
- Interrupt-driven program execution
- Error handling

You can program the organization blocks and at the same time determine the behavior of the CPU. Various organization blocks are available to you depending on the CPU used.

For more information on organization blocks, refer to the descriptions of the modes of operation of CPUs in the "Additional information on configurations" chapter in "Configuring Hardware and Networks".

Start information of organization blocks

When certain organization blocks are started, the operating system provides information that can be evaluated in the user program. Refer to the descriptions of the organization blocks to find out which information is provided, if any.

See also

Creating organization blocks (Page 1506)

Functions (FCs)

Definition

Functions (FCs) are code blocks without memory. You have no data memory in which values of block parameters can be stored. Therefore, when a function is called, all formal parameters must be assigned actual parameters.
Functions can use global data blocks to store data permanently.

Application

A function contains a program that is executed when the function is called by another code block. Functions can be used, for example, for the following purposes:

- To return function values to the calling block, e.g. for mathematical functions
- To execute technological functions, e.g. individual controls using bit logic operations

A function can also be called several times at different points in a program. As a result, they simplify programming of frequently recurring functions.

Note

Parameter transfer when calling functions

To avoid errors when working with functions, observe the information in chapter "Auto-Hotspot".

See also

Creating functions and function blocks (Page 1507)

Function blocks (FB)

Definition

Function blocks are code blocks that store their input, output and in-out parameters permanently in instance data blocks, so that they remain available even after the block has been executed. Therefore they are also referred to as blocks "with memory".

Function blocks can also operate with temporary tags. Temporary tags are will not be stored in the instance DB, but are available for one cycle only.

Application

Function blocks contain subroutines that are always executed when a function block is called by another code block. A function block can also be called several times at different points in a program. As a result, they simplify programming of frequently recurring functions.

Instances of function blocks

A call of a function block is referred to as an instance. An instance data block is required for each instance of a function block; it contains instance-specific values for the formal parameters declared in the function block.

The function block can store its instance-specific data in its own instance data block or in the instance data block of the calling block.

Access modes

S7-1200 and S7-1500 offer two different access options for the instance data blocks, which can be assigned to a function block when this is called:

- Data blocks with optimized access
  Data blocks with optimized access have no firmly defined memory structure. The data elements contain only a symbolic name in the declaration, no fixed address within the block.
- Data blocks with standard access (compatible with S7-300/400)
  Data blocks with standard access have a fixed memory structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block.

Note

To avoid errors when working with function blocks, refer to the section "Auto-Hotspot".

See also

Creating functions and function blocks (Page 1507)

Multi-instances (Page 1427)

Instance data blocks (Page 1417)

Basics of block access (Page 1419)

Global data blocks (DB)

Definition

Data blocks are used to store program data. Data blocks thus contain variable data that is used by the user program. Global data blocks store data that can be used by all other blocks.

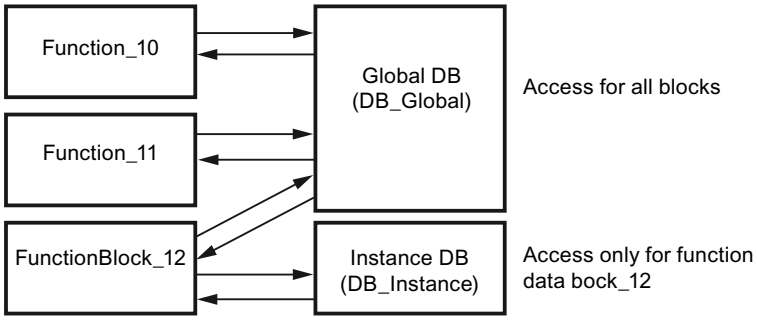
The maximum size of data blocks varies depending on the CPU. You can define the structure of global data blocks anyway you please.

You also have the option of using PLC data types (UDT) as a template for creating global data blocks.

Global data blocks in the user program

Every function block, function, or organization block can read the data from a global data block or can itself write data to a global data block. This data remains in the data block even after the data block is exited. A global data block and an instance data block can be open at the same time.

The following figure shows the different accesses to data blocks:



Access modes

S7-1200 and S7-1500 offer two different access options for global data blocks:

- Data blocks with optimized access
  Data blocks with optimized access have no fixed defined structure. In the declaration, the data elements are assigned only a symbolic name and no fixed address within the block.
- Data blocks with standard access (compatible with S7-300/400)
  Data blocks with standard access have a fixed structure. In the declaration, the data elements are assigned both a symbolic name and a fixed address within the block.

ARRAY data blocks (S7-1500)

ARRAY data blocks are a particular type of global data block. These consist of an ARRAY of any data type. For example, an ARRAY of a PLC data type (UDT) is possible. The DB contains no other elements besides the ARRAY. Because of their flat structure, ARRAY data blocks facilitate access to the ARRAY elements and their transfer to called blocks.

The "Optimized block access" attribute is always enabled for ARRAY data blocks. ARRAY data blocks with standard access are not possible.

The "Move operations" section of the "Instructions" task card offers options for addressing of ARRAY DBs.

See also

Creating data blocks (Page 1508)

Basics of block access (Page 1419)

Instance data blocks

Definition

The call of a function block is referred to as an instance. The data with which the instance works is stored in an instance data block.

The maximum size of instance data blocks varies depending on the CPU. The tags declared in the function block determine the structure of the instance data block.

Access modes

S7-1200 and S7-1500 offer two different access options for the instance data blocks, which can be assigned to a function block when this is called:

- Data blocks with optimized access
  Data blocks with optimized access have no firmly defined structure. The declaration elements contain only one symbolic name in the declaration, and no fixed address within the block.
- Data blocks with standard access (compatible with S7-300/400)
  Data blocks with standard access have a fixed structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block.

See also: Auto-Hotspot

See also

Creating data blocks (Page 1508)

Basics of block access (Page 1419)

CPU data blocks

Definition

CPU data blocks are generated by the CPU at runtime. To this purpose, insert the "CREATE_DB" instruction into your user program. You can use the data block that is generated at runtime to save your data.

CPU data blocks are indicated by means of a small CPU icon in the "Program blocks" folder of an available node. You can monitor the values of the variables of a CPU data block in online mode, similar to those of a different data block type.

You cannot create CPU data blocks in your offline project.

Loading CPU data blocks

The CPU data block that the user program has generated by means of the "CREATE_DB" instruction is initially only available on the device in online mode. All CPU data blocks will be included with the other blocks the next time you perform a complete download from the device to the project. The CPU data blocks are marked with a small CPU icon in the process. However, you cannot upload these CPU data blocks to your device again.

Restrictions on CPU data blocks in the project

Once the CPU data blocks have been loaded into your offline project, you can open and view their content. However, note that the CPU data blocks in the project are write-protected. The CPU data blocks in the project are therefore subject to the following restrictions:

- You cannot edit CPU data blocks, or convert these into a different data block type.
- CPU data blocks cannot be assigned a know-how protection.
- You cannot change the programming language of a CPU data block.
- CPU data blocks cannot be compiled or downloaded to a device.

Comparing CPU data blocks

Once the CPU data blocks have been loaded into your offline project, you can run an online/offline comparison for the CPU DBs loaded. The comparison editor provides you with a corresponding overview of the differences. It is possible to synchronize the online and off-line version of CPU data blocks if differences are found, but not by downloading the offline version to the device.

Deleting CPU data blocks

You can delete CPU data blocks both from the project and from the CPU.

See also

Deleting CPU data blocks (Page 1528)

Blocks with optimized access

Basics of block access

Introduction

STEP 7 offers data blocks with different access options:

- Data blocks with optimized access (S7-1200/S7-1500)
- Data blocks with standard access (S7-300 / S7-400 / S7-1200 / S7-1500)

Within one program you can combine the two types of blocks.

Data blocks with optimized access

Data blocks with optimized access have no fixed defined structure. In the declaration, the data elements are assigned only a symbolic name and no fixed address within the block. The elements are saved automatically in the available memory area of the block so that there are no gaps in the memory. This makes for optimal use of the memory capacity.

Tags are identified by their symbolic names in these data blocks. To address the tag, enter its symbolic name. For example, you access the "Fill Level" tag in the "Data" DB as follows:
`"Data".Fill Level`

Blocks with optimized access offers the following advantages:

- You can create data blocks with any structure without paying attention to the physical arrangement of the individual data elements.
- Quick access to the optimized data is always available because the data storage is optimized and managed by the system.
- Access errors, as with indirect addressing or from the HMI, for example, are not possible.
- You can define specific individual tags as retentive.
- Optimized blocks are equipped with a memory reserve by default which lets you expand the interfaces of function blocks or data blocks during operation. You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

Note

The "Optimized block access" attribute is always enabled for the following blocks and cannot be deselected.

- GRAPH blocks
- ARRAY data blocks

Data blocks with standard access

Data blocks with standard access have a fixed structure. In the declaration, the data elements are assigned both a symbolic name and a fixed address within the block. The address is shown in the "Offset" column.

Tags in these data blocks can be addressed in both symbolic and absolute form.
`"Data".Fill Level`
`DB1.DBW2`

Setting Retentivity for Optimized Access or Standard Access

If you define data as retentive, its values are retained even after a power failure or a network off. A retentive tag is not initialized after the hot restart but retains the value it had prior to the power failure. If a DB tag is defined as retentive, it is stored in the retentive memory area of the data block.

The options for setting the retentivity depend on the access type of the block.

- In data blocks with standard access, you cannot set the retentive behavior of individual tags. The retentivity setting is valid for all tags of the data block.
- In data blocks with optimized access you can define the retentive behavior of individual tags.
  For structured data type tags, the retentivity setting always applies to the entire structure. You cannot make any individual retentivity setting for separate elements within the data type.

Setting Addressing Options for Optimized Access or Standard Access

Blocks with optimized access permit only "type-safe" access. Type-safe access addresses tags by their symbolic name only. This means even changes to the block or the block interface will not result in inconsistencies in the program or access errors.

The following table shows the permitted addressing options for optimized data:

Addressing	Block with standard access	Block with optimized access
Symbolic addressing	x	x
Indexed addressing of ARRAYs		x
Slice access	x	x
Overlapping with AT	x	-
Absolute addressing	x	-
Indirect addressing via ANY	x	-
Indirect addressing via POINTER and VARIANT	x	with symbolic notation only

See also

Setting up block access (Page 1421)

Setting up block access

Introduction

Block access is set up automatically when you create a block:

- Blocks created on CPUs of the S7-1200/1500 product range provide optimized access by means of a default setting.
- New blocks created on CPUs of the S7-300/S7-400 product range provide standard access by means of a default setting.

Access to a block that you copy or migrate to a CPU of a different product range is not converted automatically. However, in certain situations it may be useful to change block access in manual mode, e.g., in order to utilize the full functional scope of the CPU.

In most cases, you will have to recompile and load the program after block access has been converted.

Notice

Optimized block access for GRAPH blocks

The "Optimized block access" attribute is always enabled for GRAPH blocks in S7-1500 and cannot be deselected.

Procedure

To set the block access, proceed as follows:

1. Open the "Program blocks" folder in the project tree.
2. Right-click on the block whose block access you want to change.
3. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the block opens.
4. Click "Attributes" in the area navigation.
5. Enable or disable the "Optimized block access" option.
6. Confirm your entries with "OK".

Restrictions and special features

As a matter of principle, you can only convert block access on CPUs of the S7-1200/1500 product range, as only these support the "optimized" access mode.

The following restrictions or special features apply in this context:

- Instance data blocks
  The block access of instance data blocks is always determined by the assigned function block and cannot be changed in manual mode. If you change the access mode on a function block, you also need to update the assigned instance data blocks. This update procedure adapts the access mode of the instance data block.
- System blocks and know-how protected blocks
  You cannot manually edit the block access for system blocks and know-how protected blocks.
- Organization blocks
  The start information of an OB with standard access is always stored in the first 20 bytes of the "Temp" section in the block interface. By contrast, the start information of an OB with optimized access is always written to the "Input" section. For this reason, the block interface of OBs will also change whenever you convert block access. Additional information is provided in the following sections.

Converting block access from "standard" to "optimized".

A block copied from the CPU of the S7-300/400 product range to a CPU of the S7-1200/1500 product range will initially retain the "standard" access mode. However, you can significantly increase the performance of program execution by using blocks with optimized access, which is why it may be useful to modify the access mode manually.

The blocks are adapted as follows in the course of conversion:

- Function blocks
  All interface parameters are assigned the "Non-retain" retentivity setting.
- Global data blocks
  The retentivity setting that was assigned centrally to the entire data block is transferred to the individual interface parameters. It is now possible to manipulate the retentivity setting of the various parameters.
  However, the following rule will still apply: For structured data type tags, the retentivity setting always applies to the entire structure. You cannot assign separate retentivity settings to the various elements within a structured data type. It therefore follows that you cannot assign individual retentivity settings to the tags of data blocks that are based on PLC data types.
- Organization blocks
  All interface parameters that are stored in the first 20 bytes of the "Temp" section will be deleted. New CPU-specific start information is entered in the "Input" section. Naming conflicts with user-defined interface parameters occurring in the process are resolved by renaming the user-defined interface parameters.

Caution

The conversion of the block access has the following consequences:

- Absolute addressing of the interface parameters of the block is no longer possible after conversion of block access to the "optimized mode.
  Example: `#L0.1` is no longer valid.
- Since conversion to the "optimized" block access mode of organization blocks also modifies the OB interface,

you may possibly have to adapt, recompile and load the program again due to these changes.

See also: Auto-Hotspot

Converting block access from "optimized" to "standard".

If you want to copy or move a block from the CPU of the S7-300/400 product range to a CPU of the S7-1200/1500 product range, you first need to set the "standard" access mode.

The blocks are adapted as follows in the course of conversion:

- Function blocks and global data blocks.
  You can no longer set a retentivity in the function block. The corresponding setting is made in the instance data block.
  All interface parameters in the instance DB or global DB are assigned the same retentivity setting. The conversion is subject to the following rule:
  - If all interface parameters in the original block were retentive, the entire block will be retentive after conversion.
  - If all interface parameters in the original block were non-retentive, the entire block will be non-retentive after conversion.
  - If the interface parameters in the original block had different retentivity settings, the entire block will be non-retentive after conversion.
- Organization blocks
  All interface parameters stored in the "Input" section will be deleted. New CPU-specific start information is entered in the "Temp" section. This data is written to the first 20 bytes. Naming conflicts with user-defined interface parameters occurring in the process are resolved by renaming the user-defined interface parameters.

Caution

The conversion of the block access has the following consequences:

Since a conversion to "standard" block access mode might change the retentivity settings of the interface parameters, you may possibly have to adapt, recompile and load the program again due to these changes.

See also: Auto-Hotspot

See also

Basics of block access (Page 1419)

11.1.1.3 Block calls

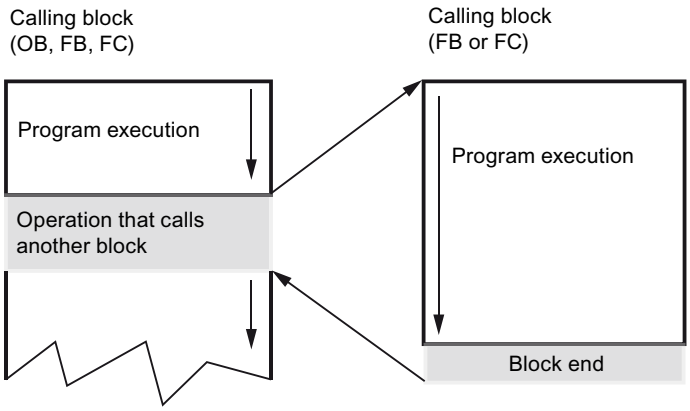
Principles of block calls

Function of block calls

For your blocks to be executed in the user program, they need to be called from another block.

When one block calls another block, the instructions of the called block are executed. Only when execution of the called block has been completed does execution of the calling block resume. The execution is continued with the instruction that follows on the block call.

The following figure shows the sequence of a block call within a user program:



Parameter transfer

When a block is called, you must assign values to the parameters in the block interface. By providing input parameters you specify the data with which the block is executed. By providing the output parameters you specify where the execution results are saved.

See also

Call hierarchy (Page 1425)

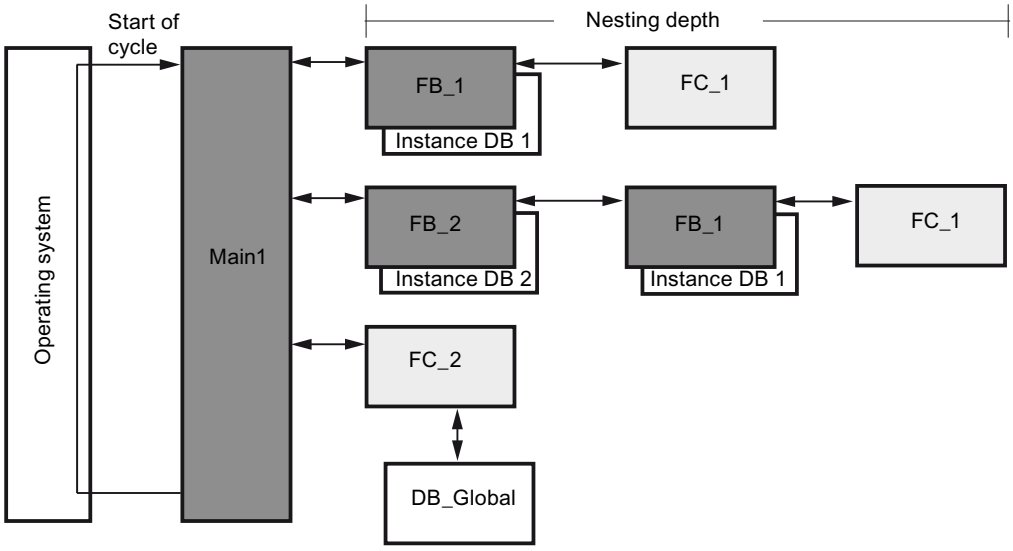
Principles for single instances and multi-instances (Page 1426)

Call hierarchy

Definition

The order and nesting of block calls is referred to as the call hierarchy. The permissible nesting depth depends on the CPU.

The following figure shows an example of the order and nesting of block calls within an execution cycle:



See also

Principles for single instances and multi-instances (Page 1426)

Principles of block calls (Page 1424)

Call function blocks as single or multi-instances

Principles for single instances and multi-instances

Use of single instances and multiple instances

Function blocks (FBs) store their data in instance data blocks. Instance data blocks store the values of the block parameters and the static local data of the function blocks.

You can assign instance data blocks as follows:

- Single instance:
  One instance data block for each instance of a function block
- Multiple instance:
  An instance data block for the instance of a function block and all instances of function blocks called in it.

See also

Principles of block calls (Page 1424)

Multi-instances (Page 1427)

Single instances (Page 1427)

Call hierarchy (Page 1425)

Single instances

Definition

The call of a function block, which is assigned its own instance data block, is called a single instance data block.

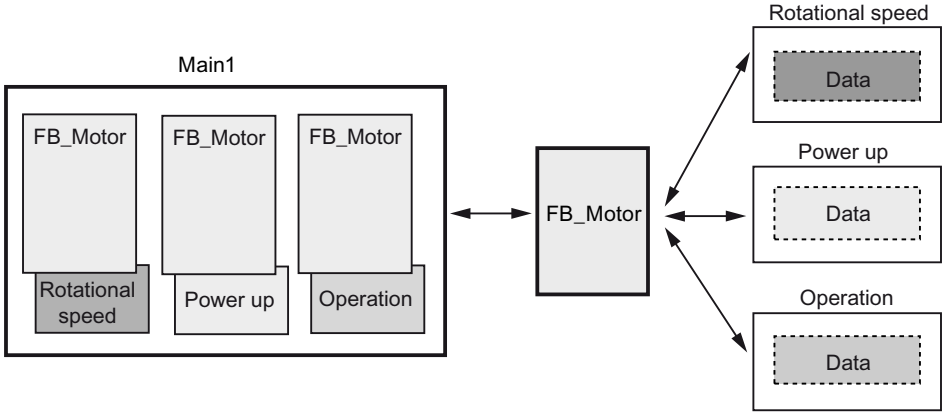
By assignment of the instance data block, you specify where the instance data of the FB is to be stored. By assigning a different instance data block to each call, you can use the same FBs several times with different instance data in each case.

Example of a single instance

You can control several motors using one function block. For this purpose, you assign a different instance data block for each function block call for motor control.

The different data for the various motors, such as speed, ramp-up time, total operating time, are saved in the different instance data blocks. A different motor will be controlled, depending on the instance data block assigned.

The following figure shows the control of three motors using one function block and three different data blocks:



See also

Principles for single instances and multi-instances (Page 1426)

Multi-instances (Page 1427)

Multi-instances

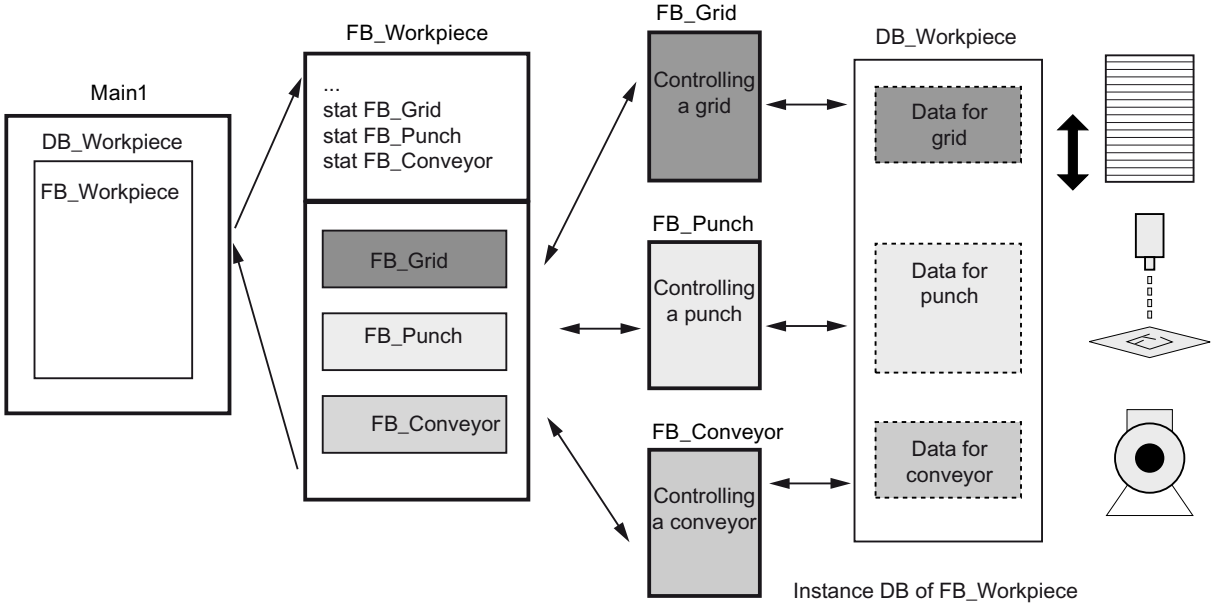
Definition

Multi-instances enable a called function block to store its data in the instance data block of the calling function block.

This allows you to concentrate the instance data in one instance data block and thus make better use of the number of instance data blocks available.

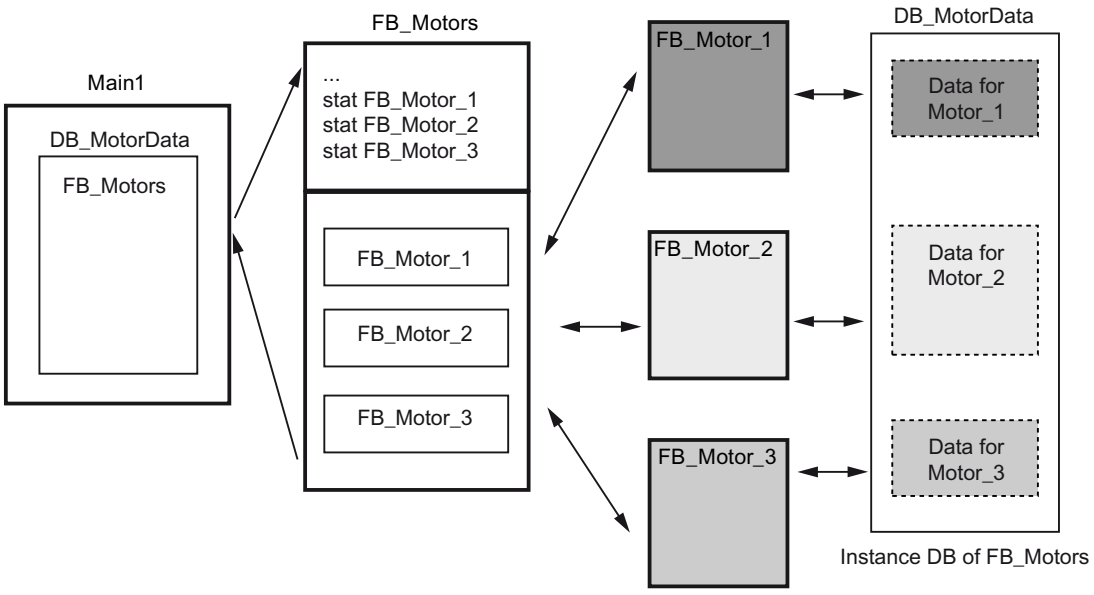
One instance data block for the instances of different function blocks

The following figure shows how multiple different function blocks store their data in a calling block. The FB_Workpiece calls the following on after the other: FB_Grid, FB_Punch and FB_Conveyor. The called blocks store their data in the DB_Workpiece, the instance data block of the calling block.



One instance data block for multi-instances of a function block

The following figure shows how a function block that is called in multi-instances stores the data for all the instances in one instance data block.



The function block FB_Motors calls three instances of the FB_Motor. The instances are "Motor_1", "Motor_2" and "Motor_3". Each call uses different instance data. However, all instance data are located in a single instance data block, DB_MotorData.

See also

Principles for single instances and multi-instances (Page 1426)

Single instances (Page 1427)

Parameter transfer at block call

Basics of block parameters

Introduction

The calling block gives the called block the values with which it is to work. These values are referred to as block parameters. The input parameters provide the called block with the values that it has to process. The block returns the results via the output parameters.

Block parameters are therefore the interface between the calling and the call block.

You use input parameters when you want to only query or read values, and output parameters when you want to set or write these values. If block parameters are read and written you have to create these as in-out parameters.

Formal and actual parameters

The block parameters are defined in the interface of the called block. These parameters are referred to as formal parameters. They are placeholders for the parameters that are transferred to the block when it is called. The parameters transferred to the block when it is called are referred to as actual parameters.

Rules for using the block parameters

The following rules apply to the use of block parameters within the block:

- Input parameters may only be read.
- Output parameters may only be written.
- In/out parameters may be read and written.

See also

Parameter assignment to function blocks (Page 1433)

Parameter assignment to functions (Page 1431)

General rules for assigning parameters (Page 1430)

Using tags within the program (Page 1447)

Keywords (Page 1442)

Supplying block parameters during call

General rules for assigning parameters

Introduction

When you call a block with block parameters, assign actual parameters to its formal parameters. The rules described below apply here.

Compatible data types

The data types of actual and formal parameters must be identical or convertible according to the rules of data type conversion.

Transferring ARRAYs

You can transfer ARRAYs as parameters. If a block has an input parameter of ARRAY type, you must transfer as actual parameter an ARRAY with identical structure. This means the data type, the number of dimensions and the number of field components must be identical. You can also transfer individual elements of an ARRAY as actual parameter if the element corresponds to the data type of the formal parameter.

Transferring PLC data types

You can also transfer tags that are declared as PLC data type as actual parameters. If the formal parameter is declared as PLC data type in the tag declaration, you must transfer a tag that has the same PLC data type as actual parameter.

An element of a tag declared by means of PLC data type can also be transferred as actual parameter at block call, provided that the data type of the element of the tag matches the data type of the formal parameter.

Transferring structures (STRUCT)

You can transfer structures as parameters. If a block has an input parameter of the STRUCT type, you must transfer as actual parameter a STRUCT with identical structure. This means that the names and data types of all structure components have to be identical.

You can also transfer individual elements of an STRUCT as actual parameter if the element corresponds to the data type of the formal parameter.

Note

We recommend programming structures as PLC data types. PLC data types make programming easier, since they can be used multiple times and modified centrally.

See also

Parameter assignment to function blocks (Page 1433)

Parameter assignment to functions (Page 1431)

Basics of block parameters (Page 1429)

PLC data types (Page 1954)

Parameter assignment to functions

Parameters of functions (FC)

Functions have no data memory in which values of block parameters can be stored. Therefore, when a function is called, all formal parameters must be assigned actual parameters.

Input parameters (Input)

Input parameters are read only once per cycle, namely before the block call. Therefore, the rule is that writing an input parameter within the block does not affect the actual parameter. Only the formal parameter is written.

Output parameters (Output)

Output parameters are read only once per cycle, namely after the block call. Therefore, the rule is that output parameters should not be read within the block. If you nevertheless read an output parameter, please note that only the value of the formal parameter is read. The value of the actual parameter cannot be read within the block.

If an output parameter of a function is not written in this function, the value that is predefined for the specified data type is used. For example, the value "false" is predefined for BOOL. However, structured output parameters are not pre-assigned with a value.

To prevent unintentional further processing of the predefined value or an undefined value, note the following when programming the block:

- Make sure that the output parameters are written with values for all possible program paths within the block. In doing so, note that jump commands may skip instruction sequences in which outputs are set, for example.
- Note that the set and reset commands are dependent on the result of the logic operation. If the value of an output parameter is determined with these commands and RLO = 0, a value will not be generated.
- If possible, assign a default value for the output parameters of functions.

In/out parameters (InOut)

In/out parameters are read before the block call and written after the block call. If you read or write the parameter within the block, you only access its formal parameter.

An exception is in/out parameters with a structured data type. Structured data types consist of several data elements, for example ARRAY or STRUCT. These are passed to the called block through a POINTER. You therefore always access the actual parameter when you read or write a structured in/out parameter within a block.

When an in/out parameter of a function is not written to this function, the old output value or the input value is used as a value. Nevertheless, you should observe the information provided above for output parameters so that old values are not inadvertently processed further.

Temporary local data (Temp)

Temporary local data is only available within a cycle. It is treated differently depending on the block type:

- Standard access
  The following rule applies to code blocks with standard access as well as to all tags with retentivity setting "Set in IDB":
  If you are using temporary local data, you must ensure that the values are initialized prior to use. Otherwise, the values will be random. Temporary data of the STRING of WSTRING data type is an exception: They are automatically pre-assigned the maximum length of 254 characters and the actual length 0.
- Optimized access
  The following rule applies to code blocks with optimized access:
  If a temporary tag is not written within a function, the value that is predefined for the specified data type is used. For example, the value "false" is predefined for BOOL. Elements of the PLC data types are pre-assigned with the default value that is specified in the declaration of the PLC data type (UDT). ARRAY elements are pre-assigned with the value "0", even if they are used within a PLC data type. STRINGs and WSTRINGs are automatically pre-assigned the maximum length of 254 characters and the actual length 0.

Function value (Return)

Functions normally calculate a function value. This function value can be returned to the calling block via the RET_VAL output parameter. For this, the RET_VAL output parameter must be declared in the interface of the function. RET_VAL is always the first output parameter of a function. All data types are permitted for the RET_VAL parameter except ARRAY and STRUCT, as well as parameter types TIMER and COUNTER.

In the SCL programming language functions can be call directly in an expression. The result of the expression is then formed with the calculated function value. Therefore, the data type ANY is not permitted in SCL for the function value.

See also

Parameter assignment to function blocks (Page 1433)

Basics of block parameters (Page 1429)

General rules for assigning parameters (Page 1430)

Calling functions (Page 1685)

Examples for calling functions in SCL (Page 1688)

Parameter assignment to function blocks

Supplying parameters of function blocks (FB)

In the case of function blocks the parameter values will be stored in the instance data.

If the input, output, or in-out parameters of a function block were not assigned with values, the stored values are used.

In some cases, it is mandatory to specify an actual parameter.

The following table shows which parameters of a function block must be assigned actual parameters:

Parameter	Elementary data type	Structured data type	Parameter type
Input (Input)	optional	optional	required
Output (Output)	optional	optional	required
In-out (InOut)	optional	required	Permitted with S7-1200 only, parameter assignment required
Temporary (Temp)	required S7-1500: Optional with optimized block access	required S7-1500: Optional with optimized block access	required

See also

Basics of block parameters (Page 1429)

General rules for assigning parameters (Page 1430)

Parameter assignment to functions (Page 1431)

Parameter types (Page 1953)

Transfer parameter as copy or as pointer

Introduction

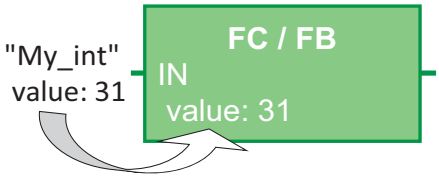
When a block is called, you transfer data to the parameters in the block interface. At the input parameters, you transfer the data with which the block is to work. At the output parameters, you specify where the results of the processing are saved. In/out parameters are used to transfer data to the called block as well as to return results.

Internally, STEP 7 recognizes two different methods of parameter transfer: The data is transferred either as pointer or as copy, depending on the transfer range and data type of the parameter.

Transfer as copy (Call by value)

During a block call, the value of the operand is copied to the input parameter of the called block. With function blocks, the copy is stored in the instance DB; with functions, it is stored in the block stack. Additional storage space is required for the copy.

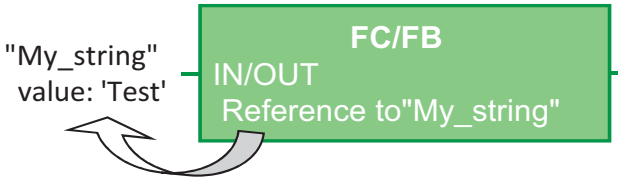
This means that the called block always works with the value that the specified operand had at the time of the block call. It cannot access the operand directly. Write access modifies only the copy, but not the actual value of the specified operand. Read access reads only the copy that was created at the time of the block call.



Transfer as pointer (Call by reference)

The parameters are referenced via a pointer during the block call.

This means that the called block can directly access the memory address of the operand that is specified as parameter: Write access directly results in the changing of the specified operand. Read access reads the value of the operand directly at the time of access. As no copy is created, no additional memory is required.



Note

Declare structured data types in the "InOut" area

If possible, use the "InOut" area in the block interface for structured tags (e.g., of data type ARRAY, STRUCT, STRING, …). As structured in/out parameters are always transferred as pointer, the required data memory is not increased more than necessary.

Parameter transfer with S7-1200/1500

The following table shows how block parameters with elementary or structured data type are transferred in S7-1200/1500. Elementary data types are, for example, BOOL, INT or BYTE. Structured data types are, for example, ARRAY, STRUCT or STRING.

		Elementary data types	Structured data types
FC	Input	Copy	Pointer
	Output	Copy	Pointer
	InOut	Copy	Pointer
FB	Input	Copy	Copy
	Output	Copy	Copy
	InOut	Copy	Pointer

Note

Parameter transfer between blocks with optimized access and blocks with standard access

When optimized data is transferred to a block with the property "Standard access", it is always transferred as copy. If the block contains numerous structured parameters, this can quickly lead to the temporary memory area (local data stack) overflowing.

You can avoid this by setting the same access type for both blocks.

See also: Auto-Hotspot

Parameter transfer with S7-300/400

The following table shows how block parameters with elementary or structured data type are transferred in S7-300/400.

		Elementary data types	Structured data types
FC	Input	Copy*	Pointer
	Output	Copy*	Pointer
	InOut	Copy*	Pointer
FB	Input	Copy	Copy
	Output	Copy	Copy
	InOut	Copy	Pointer

* Exception: Operands from the memory areas I, Q, M, P, L and partly qualified DB addresses (for example, "DW 2") are transferred as pointer.

Note

Special aspects of transfer as pointer in S7-300/400

In cases in which the parameters are transferred via a pointer, it is not possible to forward output parameters or in/out parameters from the calling block to the input parameters of the called block.

Forwarding of block parameters

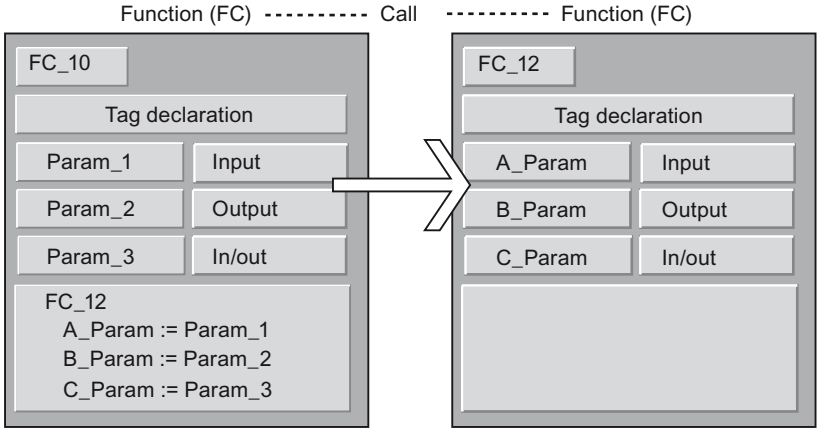
Basic information on forwarding block parameters

Introduction

Definition

The "Forwarding" of block parameters is a special type of parameter use. In this case the block parameters of the calling block are forwarded to the parameters of the called block. The called block uses the values that are currently present at the block parameters of the calling block as the actual parameters.

The following figure shows how the parameters of the function FC_10 are forwarded to the function FC_12:



Rules for LAD/FBD

The following general rules apply in LAD and FBD:

- Input parameters can only be forwarded to input parameters.
- Output parameters can only be forwarded to output parameters.
- In/out parameters can be forwarded to all parameter types.
- In S7-300/400, the two block parameters must have the same data type.
- In S7-1200/1500, the parameters can also be converted according to the rules of implicit conversion.

Rules for STL

The following general rules apply in STL:

- Input parameters can only be forwarded to input parameters.
- Output parameters can only be forwarded to output parameters.

- In/out parameters can be forwarded to all parameter types.
- Both block parameters must have the same data type. In STL, this rule applies to all CPU families.

Rules for SCL

The rules for SCL are less stringent. So that programs from previous SCL versions can be taken over more easily, additional parameter transfer options are permissible, but subject to warning. You can, for example, forward an in/out parameter to an input parameter, but a warning is output as the transferred in/out parameter cannot be written by the program.

Additional rules are described in detail in the following chapters.

See also

Calling a function by another function (Page 1438)

Call of a function by a function block (Page 1439)

Call of a function block by a function (Page 1440)

Call of a function block by another function block (Page 1440)

Calling a function by another function

Permissible data types for the call of a function by another function

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FC calls FC		Data types					
Actual parameter (calling block)	Formal parameters (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any
Input	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 as of V2 S7-1500
Output	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

InOut	**Output**	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	**InOut**	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

See also

Basic information on forwarding block parameters (Page 1437)

Call of a function by a function block

Permissible data types for the call of a function by a function block

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FB calls FC		**Data types**					
Actual parameter (calling block)	**Formal parameters (called block)**	**Standard data types**	**ARRAY, STRUCT, STRING, WSTRING, DT**	**ANY, POINTER**	**VARIANT**	**Parameter types (TIMER, COUNTER, BLOCK_XX)**	**DB_Any**
Input	**Input**	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 as of V2 S7-1500
Output	**Output**	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	**Input**	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-
InOut	**Output**	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	**InOut**	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

See also

Basic information on forwarding block parameters (Page 1437)

Call of a function block by a function

Permissible data types for the call of a function block by a function

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FC calls FB		Data types					
Actual parameter (calling block)	Formal parameters (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any
Input	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-300/400 S7-1500	S7-1200 as of V2 S7-1500
Output	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-
InOut	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	InOut	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

See also

Basic information on forwarding block parameters (Page 1437)

Call of a function block by another function block

Permissible data types for the call of a function block by another function block

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FB calls FB		Data types					
Actual parameter (calling block)	Formal parameters (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any

Input	**Input**	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-300/400 S7-1500	S7-1200 as of V2 S7-1500
Output	**Output**	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	**Input**	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-
InOut	**Output**	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	**InOut**	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

See also

Basic information on forwarding block parameters (Page 1437)

11.1.1.4 Using and addressing operands

Basic information about operands

Introduction

When you program instructions you must specify which data values the instruction should process. These values are referred to as operands. You can, for example, use the following elements as operands:

- PLC tags
- Constants
- Tags in instance data blocks
- Tags in global data blocks

Absolute address and symbolic name

Operands are identified by means of an absolute address and a symbolic name. You define the names and addresses in the PLC tag table or in the tag declaration of the blocks.

Data blocks with optimized access (S7-1200, S7-1500)

Data elements in data blocks with optimized access only receive a symbolic name and no absolute address in the declaration. For more information on this, refer to "See also".

See also

Displaying symbolic and absolute addresses (Page 1539)

Basics of block access (Page 1419)

Keywords

SIMATIC recognizes a range of key words whose definitions are fixed and which have a certain meaning in the program. You should not use these keywords as names for tags or constants.

Table of keywords

The following table shows all keywords.

Keywords German mnemonics	Keywords English mnemonics	Description
&	&	And logical operation of logical expressions
A	Q	Output, bit
A1	CC1	Condition code bit
A0	CC0	Condition code bit
AB	QB	Output, byte
AD	QD	Output, double word
AND	AND	And logical operation of logical expressions
ANY	ANY	Data type, pointer
AR1	AR1	Address Register 1
AR2	AR2	Address Register 2
ARRAY	ARRAY	Introduces the specification of an array and is followed by the index list between "[" and "]"
AT	AT	Overlaying tag declaration
AUTHOR	AUTHOR	Name of the author, company name, department name, or other name (max. 8 characters, no spaces)
AW	QW	Output, word
B	B	Byte
BEGIN	BEGIN	Introduces the instruction part for code blocks or initialization part for a data block
BIE	BR	Binary result

Keywords German mnemonics	Keywords English mnemonics	Description
BLOCK_FB	BLOCK_FB	Parameter type for specification of an FB
BLOCK_FC	BLOCK_FC	Parameter type for specification of an FC
BLOCK_SDB	BLOCK_SDB	Parameter type for specification of an SDB
BOOL	BOOL	Data type
BY	BY	Increment of the FOR loop
BYTE	BYTE	Data type
CALL	CALL	Call
CASE	CASE	Introduction to the CASE statement
CHAR	CHAR	Elementary data type
CODE_VERSION1	CODE_VERSION1	Label, whether an FB is multiple instance capable or not. If you want to declare multiple instances, the FB must not have this characteristic.
CONST	CONST	Start of the constant declaration
CONTINUE	CONTINUE	Instruction to exit a loop in SCL
COUNTER	COUNTER	Parameter type for specification of a counter
DATA_BLOCK	DATA_BLOCK	Introduces the data block
DATE	DATE	Data type
DATE_AND_TIME	DATE_AND_TIME	Data type
DB	DB	Data block
DB_ANY	DB_ANY	Data type
DBB	DBB	Data block, data byte
DBD	DBD	Data block, data double word
DBLG	DBLG	Data block length
DBNO	DBNO	Data block number
DBW	DBW	Data block, data word
DBX	DBX	Data block, data bit
DI	DI	Instance data block
DIB	DIB	Instance data block, data byte
DID	DID	Instance data block, data double word
DILG	DILG	Instance data block length
DINO	DINO	Instance data block number
DINT	DINT	Data type
DIW	DIW	Instance data block, data word
DIX	DIX	Instance data block, data bit
DO	DO	Introduction of the instruction part in FOR and WHILE instruction

Keywords German mnemonics	Keywords English mnemonics	Description
DT	DT	Data type
DTL	DTL	Data type
DWORD	DWORD	Data type
E	I	Input (via process image), bit
EB	IB	Input (via process image), byte
ED	ID	Input (via process image), double word
ELSE	ELSE	Alternative branch in IF and CASE statement
ELSIF	ELSIF	Alternative condition of the IF instruction
EN	EN	System operand of the EN/ENO mechanism
ENO	ENO	System operand of the EN/ENO mechanism
END_CASE	END_CASE	End of the CASE statement
END_DATA_BLOCK	END_DATA_BLOCK	Ends the data block
END_FOR	END_FOR	End of the FOR statement
END_FUNCTION	END_FUNCTION	Ends the function
END_FUNCTION_BLOCK	END_FUNCTION_BLOCK	Ends the function block
END_IF	END_IF	End of the IF instruction
END_ORGANIZATION_BLOCK	END_ORGANIZATION_BLOCK	Ends the organization block
END_REPEAT	END_REPEAT	End of the REPEAT statement
END_STRUCT	END_STRUCT	Ends the specification of a structure
END_SYSTEM_FUNCTION	END_SYSTEM_FUNCTION	Ends the system function
END_SYSTEM_FUNCTION_BLOCK	END_SYSTEM_FUNCTION_BLOCK	Ends the system function block
END_TYPE	END_TYPE	Ends the PLC data type
END_VAR	END_VAR	Ends a declaration block
END_WHILE	END_WHILE	End of the WHILE instruction
EW	IW	Input (via process image), word
EXIT	EXIT	Instruction to exit a loop in SCL
FALSE	FALSE	Predefined Boolean constant: Logical condition false, value equal to 0
FAMILY	FAMILY	Block family name: e.g. controller
FB	FB	Function block
FC	FC	Function
FOR	FOR	Introduction of the FOR statement
FUNCTION	FUNCTION	Introduces the function
FUNCTION_BLOCK	FUNCTION_BLOCK	Introduces the function block

Keywords German mnemonics	Keywords English mnemonics	Description
GOTO	GOTO	Introduction of the GOTO statement
IF	IF	Introduction of the IF instruction
INSTANCE	INSTANCE	Data type
INT	INT	Data type
KNOW_HOW_PROTECT	KNOW_HOW_PROTECT	Block protection
L	L	Local data bit
LB	LB	Local data byte
LD	LD	Local data double word
LDT	LDT	Data type
LINT	LINT	Data type
LTIME	LTIME	Data type
LTOD	LTOD	Data type
LW	LW	Local data word
LWORD	LWORD	Data type
M	M	Memory bit
MB	MB	Memory byte
MD	MD	Memory double word
MOD	MOD	Modulo operator
MW	MW	Memory word
NAME	NAME	Block name
NETWORK	NETWORK	Network
NOT	NOT	Logic inversion
NULL	NULL	Zero pointer
OB	OB	Organization block
OF	OF	Introduction of the data type specification / Introduction of the instruction part of the CASE statement
OR	OR	Or logical operation of logical expressions
ORGANIZATION_BLOCK	ORGANIZATION_BLOCK	Introduces the organization block
OS	OS	Save overflow
OV	OV	Overflow
PA	PQ	Output (direct peripherals), bit
PAB	PQB	Output (direct peripherals), byte
PAD	PQD	Output (direct peripherals), double word
PAW	PQW	Output (direct peripherals), word
PE	PI	Input (direct peripherals), bit
PEB	PIB	Input (direct peripherals), byte
PED	PID	Input (direct peripherals), double word

Keywords German mnemonics	Keywords English mnemonics	Description
PEW	PIW	Input (direct peripherals), word
POINTER	POINTER	Data type
READ_ONLY	READ_ONLY	Write protection for data blocks
REAL	REAL	Data type
REPEAT	REPEAT	Introduction of the REPEAT statement
RET_VAL	RET_VAL	Return value
RETURN	RETURN	RETURN statement in SCL
S5T	S5T	Syntax for data type S5TIME
S5TIME	S5TIME	Data type
S7_	S7_	Keywords for system attributes
SDB	SDB	System data block
SFB	SFB	System function block
SFC	SFC	System function
SINT	SINT	Data type
STRING	STRING	Data type
STRUCT	STRUCT	Introduces the specification of a structure and is followed by a list of components
STW	STW	Status word
SYSTEM_FUNCTION	SYSTEM_FUNCTION	System function
SYSTEM_FUNCTION_BLOCK	SYSTEM_FUNCTION_BLOCK	System function block
T	T	Time element (timer)
THEN	THEN	Introduction of the instruction part of an IF instruction
THIS	THIS	Syntax for access to an ARRAY data block
TIME	TIME	Elementary data type for time information
TIME_OF_DAY	TIME_OF_DAY	Data type
TIMER	TIMER	Parameter type for specification of a timer
TITLE	TITLE	Optional block title or network title
TO	TO	Definition of the full-scale value of a FOR statement
TOD	TOD	Data type
TRUE	TRUE	Predefined Boolean constant: Logical condition true, value not equal to 0
TYPE	TYPE	Introduction of the PLC data type
UDT	UDT	Global or PLC data type
UDINT	UDINT	Data type
UINT	UINT	Data type

Keywords German mnemonics	Keywords English mnemonics	Description
ULINT	ULINT	Data type
UNLINKED	UNLINKED	Marking 'non runtime-related'
UNTIL	UNTIL	End of the instruction part of a REPEAT statement
USINT	USINT	Data type
UO	UO	Query after (Q1=1) AND (Q0=1)
VAR	VAR	Introduces a declaration block
VAR_IN_OUT	VAR_IN_OUT	Introduces a declaration block
VAR_INPUT	VAR_INPUT	Introduces a declaration block
VAR_OUTPUT	VAR_OUTPUT	Introduces a declaration block
VAR_TEMP	VAR_TEMP	Introduces a declaration block
VARIANT	VARIANT	Data type
VERSION	VERSION	Version number of the block
VOID	VOID	Function has no return value
WCHAR	WCHAR	Data type
WSTRING	WSTRING	Data type
WHILE	WHILE	Introduction of a WHILE instruction
WORD	WORD	Data type
XOR	XOR	Logic operation
Z	C	Counter

Using tags within the program

Definition

A variable is a placeholder for a data value that can be changed in the program. The format of the data value is defined. The use of variables makes your program more flexible. For example, you can assign different values to variables that you have declared in the block interface for each block call. As a result, you can reuse a block you have already programmed for various purposes.

A variable consists of the following elements:

- Name
- Data type
- Absolute address
  - PLC tags and DB tags in blocks with standard access have an absolute address.
  - DB variables in blocks with optimized access have no absolute address.
- Value (optional)

Declaring Variables

You can define variables with different scopes for your program:

- PLC tags that apply in all areas of the CPU
- DB variables in global data block that can be used by all blocks throughout the CPU.
- DB tags in instance data blocks that are predominantly used within the block in which they are declared.

The following table shows the difference between the variable types:

	PLC tags	Variables in instance DBs	Variables in global DBs
Range of application	• Are valid throughout the entire CPU. • Can be used by all blocks on the CPU. • The name is unique within the CPU.	• Are predominantly used in the block in which they are defined. • The name is unique within the instance DB.	• Can be used by all blocks on the CPU. • The name is unique within the global DB.
Permissible characters	• Letters, numbers, special characters • Quotation marks are not permitted. • Reserved keywords are not permitted.	• Letters, numbers, special characters • Reserved keywords are not permitted.	• Letters, numbers, special characters • Reserved keywords are not permitted.
Use	• I/O signals (I, IB, IW, ID, Q, QB, QW, QD) • Bit memory (M, MB, MW, MD)	• Block parameters (input, output and in-out parameters), • Static data of a block	• Static data
Location of definition	PLC tag table	Block interface	Declaration table of the global DB

See also

Keywords (Page 1442)

Basic information about operands (Page 1441)

Displaying symbolic and absolute addresses (Page 1539)

Valid names of PLC tags (Page 1481)

Permissible addresses and data types of PLC tags (Page 1482)

Constants

Basics of constants

Definition

Constants are data with a fixed value that you cannot change during program runtime. Constants can be read by various program elements during the execution of the program but cannot be overwritten. There are defined notations for the value of a constant, depending on the data type and data format. A distinction is made between the typed and non-typed notation.

Non-typed constants

In the non-typed notation, you only enter the value of the constant without a data type. Non-typed constants do not receive their data type until the first arithmetic or logical operation in which they are used.

The example below shows the non-typed notation:

```
SCL
#My_Int1 := #My_Int2 + 12345        (*The data type of the constant "12345"
                                    results from the addition with My_Int 2. "12345"
                                    receives the data type INT.*)
#My_Real1 := #My_Real2 + 12345      (*The data type of the constant "12345" results
                                    from the addition with My_Real2. "12345" re-
                                    ceives the data type REAL.*)
```

Typed constants

In the typed notation, you specify a data type in addition to the value of the constant.

The example below shows the typed notation:

```
SCL
#My_Int1 :=  INT#12345                    (*The data type of the constant is always
                                          INT.*)
```

Note

Constants of BOOL type in LAD/FBD

Constants of the BOOL type may not be used as inputs of instructions in S7-300/400.

Constants of the BOOL type may be used as inputs of instructions which are a system-internal function block (FB) in S7-1200/1500. These instructions are identified by the fact that the "Call options" dialog opens when you insert the instruction into a network. Boolean constants may not be used as inputs for all other instructions.

Additional information

Additional information on the data types of constants, their input formats and value ranges:

Data types (Page 1908)

Calculating with constants in SCL (Page 295)

See also

Layout of the block interface (Page 1551)

Declaration of symbolic names for constants

Symbolic constants

You have the option of declaring symbolic names for constants and thus making constant values available under a name in the program. This makes a program more readable and easier to maintain when changing constant values.

A symbolic constant consists of the following elements:

- Name
- Data type
  Symbolic constants always have a data type; non-typed notation is not possible for symbolic constants.
- Constant value
  You can select any value from the value range of the specified data type as constant value. For information on the value ranges, refer to the "Data types" chapter.

Declaration of constants

You can define constants with different scopes of validity:

- Global constants that apply to all areas of the CPU
- Local constants that only apply within a block

The table below shows the difference between the constant types:

	Global constants	Local constants
Scope of validity	• Are valid throughout the entire CPU • The name is unique within the CPU.	• Are valid only in the block in which they were declared. • The name is unique within the block.
Permitted characters	• The permitted characters in constant names are letters, digits and special characters.	• The permitted characters in constant names are letters, digits and special characters.

	Global constants	Local constants
Location of definition	"Constants" tab from the PLC tag table	Block interface
Representation	In quotation marks, for example.: `"Glob_Const"`	Prefixed with a number sign, for example: `#Loc_Const`

Note

Downloading constant declarations (S7-300/400)

Local and global constant declarations are not downloaded into the CPU. If you download a program from a device, the constant declarations may no longer be available.

Additional information

Additional information on the procedure for declaring constants:

Overview of the valid data types (Page 1908)

Auto-Hotspot

Auto-Hotspot

Calculating with constants in SCL (Page 295)

Data types of constants

Permitted data types

For constants, all basic data types as well as all derived data types are permitted:

- Binary numbers
- Bit strings
- Integers
- Floating-point numbers
- Timers
- Date and time
- Character strings

All general rules for explicit and implicit type conversion apply.

Data types of non-typed constants

Non-typed constants do not contain an explicit type specification. They do not receive their data type until the first arithmetic or logical operation in which they are used.

The example below shows how non-typed constants are used:

```
SCL
#My_Int1 := #My_Int2 + 12345        (*The data type of the constant "12345"
                                    results from the addition with My_Int 2. "12345"
                                    receives the data type INT.*)
#My_Real1 := #My_Real2 + 12345      (*The data type of the constant "12345" results
                                    from the addition with My_Real2. "12345" re-
                                    ceives the data type REAL.*)
```

Note

STEP 7 always uses the data type with the highest possible precision

Unless the data type of a constant can be clearly defined in an expression, the data type with the highest precision available on the current CPU is used.

Example:

```
#My_Real := #My_Int / 3.5
```

In this expression an integer tag is combined with a non-typed floating point constant. In S7-300/400 the right part of the assignment is calculated in the REAL format. In S7-1200/1500, calculation is performed using the highest possible precision, which in this case means LREAL. As a result, the assignment to a REAL tag is invalid or generates a warning.

To precisely define the data type of a constant, use the typed notation.

Example:

```
#My_Real := #My_Int / REAL#3.5
```

Additional information

Additional information on the data types of constants, their input formats and value ranges:

Data types (Page 1908)

Calculating with constants in SCL (Page 295)

Examples of using constants

Use in instructions, assignments and expressions

Constants can be used in place of tags in instructions or assignments. You can also use constants in expressions in SCL. But because constants cannot be written, they may only be used as inputs.

The example below shows possible uses of constants:

```
SCL
#My_Int :=   3;
```

```
SCL
#My_Real1 := #My_Real2 * 3;
#My_Real1 := #My_Real2 * #My_local_const;
#My_Real1 := #My_Real2 * "My_global_const";
```

Use as a default value

You can use constants as the default value of a tag. To do so, enter either the value or the symbolic name of the constant in the "Default value" column of the block interface. The data type of the constant must match the data type of the tag or be convertible with it according to the implicit conversion with IEC check.

Name conflicts can occur if you have declared a local and a global constant with the same symbolic name and have used this doubly specified name as the default value of a tag. In this case, the local constant is automatically used.

Use as an ARRAY limit

You can use local or global constants of data type INT or DINT as ARRAY limits.

The example below shows the use of constants as ARRAY limits:

```
SCL
Array[#My_local_const1..#My_local_const2] of REAL
Array["My_global_const1".."My_global_const1"] of REAL
```

Note

Constants as ARRAY limits

- Constants which are used as ARRAY limits cannot be changed if the memory reserve of the block is activated. This applies to both local and global constants. To change these constants, you must first disable the memory reserve.
- Changes to global constants result in inconsistencies in the blocks which use them as ARRAY limits. The inconsistencies are marked in red in the block used. To remedy these inconsistencies, the data blocks have to be updated.
  See also: Updating data blocks (Page 1708)

Additional information

Additional information on the data types of constants, their input formats and value ranges:

Data types

Calculating with constants in SCL (Page 295)

See also

Overview of the valid data types (Page 1908)

Addressing operands

Addressing global variables

Addressing global variables

To address a global PLC variable, you can use the absolute address or the symbolic name.

Note

The LWORD, LINT, ULINT, LREAL, LTIME, LTOD and LDT data types can only be addressed by means of their symbolic name.

Addressing global variables in symbolic form

When you use addressing in symbolic form, you enter the variable name from the PLC tag table. The symbolic name of global variables are automatically enclosed in quotation marks.

You address structured tags that are based on a PLC data type with the symbolic name of the PLC tag. You can also indicate the names of the individual components separated by a dot.

Addressing global variables in absolute form

When you use addressing in absolute form, you enter the address of the variables from the PLC tag table. The absolute address uses numerical addresses starting with zero for each operand range. The address identifier % is set automatically as prefix for the absolute address of global tags.

Examples

The following examples show applications of symbolic and absolute addressing:

Addressing	Description
%Q1.0	Absolute address: Output 1.0
%I16.4	Absolute address: Input 16.4
%IW4	Absolute address: Input word 4
"Motor"	Symbolic address "Motor"
"Value"	Symbolic address "Value"
"Structured_Tag"	Symbolic address of a tag that is based on a PLC data type
"Structured_Tag".Component	Symbolic address of the component of a structured tag.

See also: Permissible addresses and data types of PLC tags (Page 1482)

See also

Displaying symbolic and absolute addresses (Page 1539)

Accessing I/O devices (Page 1455)

Accessing I/O devices

Description

The process image of the CPU is updated once in a cycle. In time-critical applications, however, it can be that the current state of a digital input or output has to be read or transferred more often than once per cycle. For this purpose you can use a suffix for I/O access identifiers on the operand to directly access the I/O.

If you want to read the input directly from the peripherals, use the peripheral inputs memory area (PI) instead of the process input image (I). The peripherals memory area can be read as a bit, byte, word, or double word.

If you want to write directly to the output, use the peripheral output (PQ) memory area instead of the process output image (Q). The peripheral output memory area can be written as a bit, byte, word, or double word.

To read or write a signal directly from a peripheral input, you can add the suffix for I/O access ":P", to the operand.

Components of structured PLC tags can also be addressed with ":P". However, access to the higher-level tag with ":P" is not possible.

Warning
Direct writing of the I/O

Immediate writing to the I/O can lead to hazardous states, for example when writing multiple times to an output in one program cycle.

Syntax

```
<Operand>:P
```

Example

The following example shows applications of I/O access identifiers:

Addressing	Description
"Motor"	Addresses the "Motor" tag in the process image.
"Motor":P	Addresses the "Motor" tag in the I/O memory area (PI or PQ).
"Structured_Tag".Component	Addresses the component of a structured PLC tag in the process image.

Addressing	Description
"Structured_Tag".Component:P	Addresses the component of a structured PLC tag in the I/O memory area (PI or PQ).

See also

Addressing global variables (Page 1454)

Addressing variables in data blocks

Addressing variables in global data blocks

Description

Tags in global data blocks can be addressed in symbolic or absolute form. For symbolic addressing, you use the name of the data block and the name of the tag, separated by a dot. The name of the data block is enclosed in quotation marks.

For absolute addressing, you use the number of the data block and the absolute address of the tags in the data block, separated by a dot. The address identifier % is set automatically as prefix for the absolute address.

The S7-1200/1500 provides you with an option of accessing a data block that is not yet known during programming. For this purpose, create a block parameter of data type DB_ANY in the block interface of the accessing block. The data block name or data block number is transferred to this parameter during runtime. In order to access the internal tags of the data block, use the name of the block parameter of data type DB_ANY and the absolute address of the tag, separated by a dot.

Note

Transfer DB with memory reserve to the parameter DB_ANY.

It is not possible to transfer a DB with memory reserve to a block parameter of data type "DB_ANY".

Note

Addressing DB tags in absolute form

Absolute addressing is not possible for the following tags:

- Tags in blocks with optimized access.
- Tags of data type LWORD, LINT, ULINT, LREAL, LTIME, LTOD and LDT.

Best practice is to use the more convenient symbolic addressing for these tags.

ARRAY data blocks

ARRAY data blocks are a particular type of global data block. These consist of an ARRAY of any data type. For example, an ARRAY of a PLC data type (UDT) is possible.

You address elements in ARRAY data blocks with the help of the keyword "THIS". The index is then specified in square brackets. The index can be a constant as well as a tag. Integers with a width of up to 32 bits are permitted as tags for the index.
Extended options for addressing ARRAY DBs are available in the "Move" section of the "Instructions" task card. These instructions give you the option, for example, to also address the DB name indirectly.

Syntax

```
"<DBname>".TagName
%<DBnumber>.absoluteAddress
#<DBAny_name>.%absoluteAddress
"<ArrayDBname".THIS[#i].<Component>.<ComponentElement>
```

SCL:
```
"<ArrayDBname"."THIS"[#i].<Component>.<ComponentElement>
```

The following table show the possible absolute addresses of tags in data blocks:

Data type	Absolute address	Example	Description
BOOL	%DBn.DBXx.y	%DB1.DBX1.0	Data bit 1.0 in DB1
BYTE, CHAR, SINT, USINT	%DBn.DBBy	%DB1.DBB1	Data bit 1 in DB1
WORD, INT, UINT	%DBn. DBWy	%DB1.DBW1	Data word 1 in DB1
DWORD, DINT, UDINT, REAL, TIME	%DBn.DBDy	%DB1.DBD1	Data double word 1 in DB1

Example

The following examples show the addressing of tags in global data blocks:

Addressing	Description
"Motor".Value	Symbolic addressing of the "Value" tag in the "Motor" global data block.
%DB1.DBX1.0	Absolute addressing of the "DBX1.0" tags in the "DB1" global data block.
#MyDBAny.%DBX30.0	Absolute addressing of the "DBX30.0" tag in the global data block that is transferred at runtime at the "MyDBany" parameter.
"MyARRAY_DB".THIS[#MyIndex].MyComponent.MyComponentElement	Addressing an ARRAY data block. The ARRAY index is specified with the "MyIndex" tag. The ARRAY element has two additional substructures: "MyComponent" and "MyComponentElement".

See also

Using the DB_ANY data type (Page 275)

Addressing structured variables (Page 1459)

Addressing areas of a tag with slice access (Page 1461)

Basics of indirect addressing (Page 1465)

Addressing instance data (Page 1458)

Addressing instance data

Description

You can address data elements from the interface of the current block. These tags are stored in the instance data block.

Note

Tags in blocks with optimized access can only be addressed in symbolic form.

To address a tag from the interface of the current block, enter the character # followed by the symbolic tag name.

You can also access the tags of a multiple instance block. Within the multiple instance block, also use the character # followed by the tag name to address the data. You access the data of the multiple instance block from the calling block using #<Multiple instanceName.TagName>.

Syntax

Use the following syntax for addressing tags in instance data blocks:

```
#<TagName>
#<Multiple instanceName.TagName>
```

Examples

The following examples show the addressing of tags in instance data blocks:

Addressing	Description
#Value	Addressing the "Value" tag in the instance data block.
#On	Addressing the "On" tag within the multiple instance block
#Multi.On	Addressing the "On" tag of the multiple instance block from the calling block

See also

Addressing variables in global data blocks (Page 1456)

Addressing structured variables (Page 1459)

Addressing areas of a tag with slice access (Page 1461)

Basics of indirect addressing (Page 1465)

Addressing structured variables

Addressing data elements of an ARRAY

You access an element in an ARRAY using the syntax `ArrayName[i,j,k...]`.

The index of the element is specified in square brackets. The index includes an integer value (-2147483648 ... 2147483647) for each ARRAY dimension.

Access errors result when you access an element during runtime which is located outside the declared ARRAY limits. The various CPU families react differently to violations of the ARRAY limits:

- S7-300/400
  - The CPU changes to "STOP" mode.
  - You can program the program execution error OB (OB 85) to prevent this.
  - In SCL, you also have the option of enabling the attribute "Check ARRAY limits" in the block properties. This causes the enable output ENO to be set to FALSE in the case of ARRAY access errors.
- S7-1200
  - The CPU generates a diagnostic buffer entry and remains in "RUN" mode.
- S7-1500
  - The CPU changes to "STOP" mode.
  - You can program the programming error OB (OB 121) to prevent this.
  - You also have the option of programming the local error handling with the instructions "GET_ERROR: Get error locally" or "GET_ERROR_ID: Get error ID locally".

Note

Monitoring ARRAY access errors with ENO

The enable output ENO is not set to the signal state FALSE if the ARRAY limits are violated during execution of an instruction. The only exception is SCL blocks on CPUs of the S7-300/400 series for which the block property "Check ARRAY limits" is set.

See also:

Array (Page 1941)

Indirect indexing of ARRAY components (Page 1467)

Addressing ARRAY data blocks

ARRAY data blocks are a particular form of the ARRAY. ARRAY data blocks are global data blocks that consist of exactly one ARRAY. You address elements in ARRAY data blocks using the following syntax:

```
"<GlobArrayDBname".THIS[#i].<componentname>."<elementname>"
```

SCL:

```
"<GlobArrayDBname"."THIS"[#i].<componentname>."<elementname>".
```

The "Move" section of the "Instructions" task card offers extended options for addressing ARRAY DBs. These instructions give you the option, for example, to also address the DB name indirectly.

Addressing data elements in structures

You access individual elements in a structure using `StructureName.ElementName`.

See also:

Structures (Page 1945)

Addressing data elements of an PLC data type

The syntax `PLCDataTypeName.ElementName` is used to access elements of a PLC data type.

See also:

Auto-Hotspot

Addressing individual characters of a STRING or WSTRING (S7-1200/1500)

Use the syntax `StringName[i]` to access an individual character of a STRING or WSTRING tag. The counting index "i" begins with "1". Thus, you access the first character of the string with StringName[1].

You cannot access individual characters of a STRING or WSTRING constant.

Errors result when you access a character during runtime which is located outside the STRING length. On read access to the STRING, you receive the character '$00' or '$0000'; write access to the STRING is not executed. If the instruction has the enable output ENO, ENO is set to the signal state FALSE. The CPU does not change to STOP.

See also:

Character strings (Page 1936)

Examples:

The following examples show the addressing of structured data type tags:

Addressing	Description
`Motor.Value_1x3[2]`	Addressing of a one-dimensional array
`Motor.Value_2x4[2,4]`	Addressing of a two-dimensional array

Addressing	Description
Motor.Value_4x7[2,4,1,3]	Addressing of a four-dimensional array
Batch_1.Temperature	Addressing of the element "Temperature" in the structure "Batch_1"
Values.Temperature	Addressing of the "Temperature" element in the "Values" tag, which is based on a PLC data type.
STRING[3]	Addresses the third character of the STRING.
WSTRING[3]	Addresses the third character of the WSTRING.

See also

Basics of indirect addressing (Page 1465)

Addressing areas of a tag with slice access

Description

You have the option to specifically address areas within declared tags. You can access areas of the 1-bit, 8-bit, 16-bit, or 32-bit width. The type of access is referred to as "slice access".

Structures, constants and tags overlaying AT cannot be addressed with slice access.

Syntax

The following syntax is used for addressing:

```
<Tag>.X<Bit number>
<Tag>.B<BYTE number>
<Tag>.W<WORD number>
<Tag>.D<DWORD number>
```

The syntax has the following components:

Part	Description
<Tag>	Tag that you access. The tag must be of the "Bit string" data type. In the case of deactivated IEC check, access to tags of the "Integer" data type is also possible.
X	ID for the access width "Bit (1Bit)"
B	ID for the access width "Byte (8 Bit)"
W	ID for the access width "Word (16 Bit)"
D	ID for access width "DWord (32-bit)"
<BIT number>	Bit number within <tag> that is accessed. Number 0 accesses the least significant BIT.
<BYTE number>	Byte number within <tag> that is accessed. The number 0 accesses the least significant BYTE.

Part	Description
<WORD number>	Word number within <tag> that is accessed. The number 0 accesses the least significant WORD.
<DWORD number>	DWord number within <tag> that is accessed. The number 0 accesses the least significant DWORD.

Examples

The following examples show the addressing of individual bits:

Addressing	Description
"Engine".Motor.X0 "Engine".Motor.X7	"Motor" is a tag of the BYTE, WORD, DWORD or LWORD data type in the global data block "Engine". X0 addresses the bit address 0, X7 the bit address 7 within "Motor".
"Engine".Speed.B0 "Engine".Speed.B1	"Speed" is a tag of the WORD, DWORD or LWORD data type in the global data block "Engine". B0 addresses the byte address 0, B1 the byte address 1 within "Speed".
"Engine".Fuel.W0 "Engine".Fuel.W1	"FUEL" is a tag of the DWORD or LWORD data type in the global data block "Engine". W0 addresses the word address 0, W1 the word address 1 within "Fuel".
"Engine".Data.D0 "Engine".Data.D1	"Data" is a tag of the LWORD data type in the global data block "Engine". D0 addresses the double word address 0, D1 the double word address 1 within "Data".

See also

Addressing with Slice access (Page 250)

Overlaying tags with AT

Description

To access data areas within a declared tag, you can overlay the declared tags with an additional declaration. This provides you with the option of addressing an already declared tag with a different data type. You can, for example, address the individual bits of a tag of WORD data type with an ARRAY of BOOL.

Rules

The following general rules are valid for tag overlaying:

- Overlaying is possible in S7-1200 and S7-1500 in STL, LAD, FBD and GRAPH.
- SCL supports overlaying in all CPU families.

- Overlaying of tags is possible in the following blocks:
  - In code blocks with standard access
  - In code blocks with optimized access for tags with the retain setting "Set in IDB"
- The data width of the overlaying tag must be equal to or less than that of the overlaid tag.
- It is not possible to overlay tags of the VARIANT and INSTANCE data types.
- Blocks from libraries which are declared as parameters in the interface cannot be overlaid.
- Structured PLC tags that are declared as parameters in the interface cannot be overlaid.
- You cannot address overlaying tags with slice access.

Note

S7-1200/1500: Using AT in FCs

The data widths of the overlaying tag and the overlaid tag must be identical for FCs in S7-1200/1500. If this is not possible in your program, check to see if you can use slice access instead of the AT construct.

See also: Addressing areas of a tag with slice access (Page 1461)

The following combination rules are also valid:

		Overlaying tag	Overlaid tag			
Elementary	Structured *	Any/Pointer	DB_ANY			
FB	Input	Elementary	x	x		x
		Structured *	x	x	x	x
		Any/Pointer		x		
	Temp	Elementary	x	x		
		Structured	x	x	x	
		Any/Pointer		x		
	Static, Output	Elementary	x	x		x
		Structured	x	x		x
		Any/Pointer				
	InOut	Elementary	x			x
		Structured		x		
		Any/Pointer				
FC	Temp	Elementary	x	x		
		Structured	x	x	x	
		Any/Pointer		x		
	Input, Output, InOut	Elementary (both tags must have the same bit width)	x			x
		Structured		x	x	
		Any/Pointer				

		Overlaying tag	Overlaid tag			
OB	Temp	Elementary	x	x		
		Structured	x	x	x	
		Any/Pointer		x		

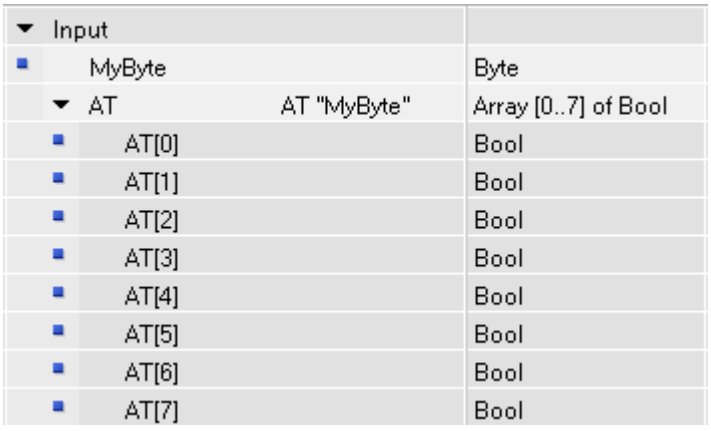
* Structured data types consist of several data elements, e.g. ARRAY or STRUCT.

Declaration

To overlay a tag, declare an additional tag directly after the tag that is to be overlaid and identify it with the keyword "AT".

Example

The following figure shows the declaration of an overlaid tag in the interface of a FB:



Input		
MyByte		Byte
AT	AT "MyByte"	Array [0..7] of Bool
AT[0]		Bool
AT[1]		Bool
AT[2]		Bool
AT[3]		Bool
AT[4]		Bool
AT[5]		Bool
AT[6]		Bool
AT[7]		Bool

When a block is called with the shown tag declaration, the "MyByte" tag is assigned. Within the block there are now two options for interpreting the data:

- as a byte
- As one-dimensional ARRAY of BOOL

Addressing operands indirectly

Basics of indirect addressing

Introduction

Indirect addressing offers the option of addressing operands whose address is not calculated until during runtime. With indirect addressing, program sections can be executed several times and a different operand can be used during each run.

Warning

Risk of access errors

Since operands are only calculated during runtime with indirect addressing, there is a risk that access errors may occur and that the program will operate with incorrect values. In addition, memory areas may inadvertently be overwritten with incorrect values. The automation system can then react in unexpected manner.

Therefore, use indirect addressing only with caution.

Indirect addressing

Basics of indirect addressing

General indirect addressing options in S7-1200 and S7-1500

The following indirect addressing options are available in all programming languages:

- Indirect addressing via pointer
- Indirect indexing of ARRAY components
- Indirect addressing of a data block via DB_ANY data type.

Language-specific options of indirect addressing

The following specific addressing options are also available in the various programming languages:

- In STL, you can address operands indirectly via the address register.
- In SCL, you can read or write a variable memory area with the following instructions:
  - POKE - Write memory address
  - POKE_BOOL - Write memory bit
  - PEEK - Read memory address
  - PEEK_BOOL - Read memory bit
  - POKE_BLK - Write memory area

For a detailed description of these addressing options, refer to "See also".

See also

Addressing variables in global data blocks (Page 1456)

POKE: Write memory address (Page 2852)

POKE_BOOL: Write memory bit (Page 2853)

PEEK: Read memory address (Page 2848)
PEEK_BOOL: Read memory bit (Page 2850)
POKE_BLK: Write memory area (Page 2855)
Indirect addressing via pointer (Page 1466)
Indirect indexing of ARRAY components (Page 1467)
Indirect addressing in STL (Page 1471)

Indirect addressing via pointer

Description

For indirect addressing, a special data format is required that contains the address and possibly also the range and the data type of an operand. This data format is referred to as pointer. The following types of pointers are available to you:

- POINTER (S7-1500)
- ANY (S7-1500, only for blocks with standard access)
- VARIANT (S7-1200/1500)

For more information on the pointer data types, refer to "See also".

Note

In SCL the use of the POINTER is restricted. The only option available is to forward it to the called block.

Example

The following example shows an indirect addressing with an area-internal pointer:

Addressing in STL	Explanation
L P#10.0	// Load pointer (P#10.0) in accumulator 1
T MD20	// Transfer pointer to the operand MD20
L MW [MD20]	// Load MW10 in accumulator 1
....	// Any program
L MD [MD20]	// Load MD10 in accumulator 1
....	// Any program
= M [MD20]	// If RLO=1, set the memory bit M10.0

The pointer P#10.0 is transferred to the operand MD20. If the operand MD20 in square brackets is programmed, this will be replaced in runtime by the address that is contained in the pointer.

See also

Basics of indirect addressing (Page 1465)

Pointer (Page 1946)

Indirect indexing of ARRAY components

Description

For addressing the components of an ARRAY, you can specify tags of the integer data type as well as constants as the index. Integers with a length of up to 32 bits are allowed here. When tags are used, the index is calculated during runtime. You can, for example, use a different index for each cycle in program loops.

Note

When you call a block and transfer an indirectly indexed ARRAY component ("<Data block>".<ARRAY>["i"]) to it as in/out parameter (InOut), you cannot change the value of the index tag [i] while the block is being executed. The value is therefore always written back to the same ARRAY component from which it was read.

Syntax

The following syntax is used for the indirect indexing of a ARRAY:

```
"<Data block>".<ARRAY>["i"] // one-dimensional ARRAY
"<Data block>".<ARRAY>["i"] // one-dimensional ARRAY of STRUCT
"<Data block>".<ARRAY>["i"] // multidimensional ARRAY
"<Data block>".<ARRAY>["i"] // multidimensional ARRAY of STRUCT
```

The syntax has the following components:

Part	Description
Data block	Name of the data block in which the ARRAY is located
ARRAY	Tag of the ARRAY data type
i, j	PLC tags of the integer data type that are used as pointers
a	Additional partial tag of the structure

Examples

The following example shows indirect array indexing of an ARRAY component in STL:

Several axes traverse at different angles. The values for axis number and angle are stored in the two-dimensional ARRAY "control_axis".

You can use the "SEL" instruction to select the components of the "control_axis" ARRAY to be written at the "#out" output parameter.

The axis number is defined by the constants "Constant_Axis_NoX" and "Constant_Axis_NoY"; the angle is defined by the "#Angle" tag.

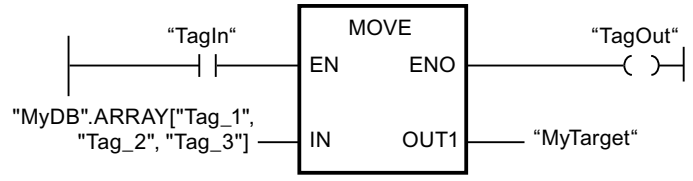
Addressing in STL

```
CALL  SEL
    value_type:=Int
    G   :="Select"
    IN0 :=#control_axis["Constant_Axis_NoX", #Angle]
    IN1 :=#control_axis["Constant_Axis_NoY", #Angle]
    OUT :=#out
```

The following examples are based on SCL and demonstrate indirect indexing of an ARRAY component. "MOTOR" is a one-dimensional ARRAY_of_INT with three rows. "VALUES" is a PLC tag of data type "Integer".

Addressing in SCL	Explanation
`MOTOR[2] := VALUES;`	`(*Direct addressing: Assignment of VALUES to the second row of the ARRAY MOTOR*)`
`MOTOR["Tag_1"] := VALUES;`	`(*Indirect addressing: Assignment of VALUES to the rows of ARRAY MOTOR*) specified by "Tag_1"`
`#MOTOR["Tag_2"+"Tag_3"] := #Values;`	`(*Indirect addressing: Assignment of VALUES to the row of the MOTOR*) ARRAY specified by the expression "Tag_2"+"Tag_3"`

The following example shows the indirect indexing of an ARRAY component as an example of LAD. "ARRAY" is a three-dimensional ARRAY. "Tag_1", " "Tag_2" and ""Tag_3" are PLC tags of the "Integer" data type. Depending on their values, one of the "ARRAY" components will be copied to the "MyTarget" tag.



Indexing ARRAY components using the "FieldRead" and "FieldWrite" instructions

You may also use the following instructions for indirect indexing of ARRAY components in LAD and FBD:

- FieldWrite - Write field
- FieldRead - Read field
  For more information on these instructions, refer to the "References" chapter.

Additional information

For more information on the ARRAY data type, refer to "See also".

See also

Basics of indirect addressing (Page 1465)

Array (Page 1941)

Addressing structured variables (Page 1459)

Indirect addressing of individual characters of a STRING

Description

For addressing the individual characters of a STRING or WSTRING, you can specify constants and also tags as the index. The tags must be of the Integer data type. When tags are used, the index is calculated during runtime. You can, for example, use a different index for each cycle in program loops.

If you transfer a STRING or WSTRING with variable index to an in/out parameter during a block call, please note that: The index tag [i] is read once at the start of the block call and cannot be changed by the called block while it is being executed.

Note

Monitoring STRING access in runtime

When a STRING or WSTRING that exceeds the defined length is written in runtime, unwanted reactions may occur in the program. Violation of the STRING or WSTRING length is monitored in S7-1200/1500. On read access to the STRING, you receive the character '$00' or '$0000'; write access to the STRING is not executed. If the instruction has the enable output ENO, ENO is set to the signal state FALSE. The CPU does not change to STOP.

Syntax

The following syntax is used for the indirect indexing of a STRING or WSTRING:

```
"<Data block>".<STRING>["i"]
"<Data block>".<WSTRING>["i"]
```

Example

The example below shows indirect indexing of a STRING using SCL as an example. "STRING", "WSTRING", "CHAR" and "WCHAR" are tags. "Tag_1" is a PLC tag of the "Integer" data type.

Addressing in SCL	Description
`STRING["Tag_1"] := CHAR;`	`(*Indirect addressing: Assignment of "CHAR" to the character of the STRING*) specified by "Tag_1"`

Addressing in SCL	Description
`WSTRING["Tag_1"] := WCHAR;`	`(*Indirect addressing: Assignment of "WCHAR" to the character of the WSTRING*) specified by "Tag_1"`
`WCHAR := WSTRING["Tag_1"];`	`(*Indirect addressing: Assignment of the WSTRING character specified by "Tag_1" to WCHAR*)`

Additional information

You can find additional information on the STRING and WSTRING data types under "See also".

See also

STRING (Page 1937)

WSTRING (Page 1939)

Indirect addressing in STL

Basic information about address registers

Introduction

Two address registers are available for the indirect addressing of operands: address register 1 (AR1), and address register 2 (AR2). The address registers are equal and are 32 bits in length. You can store area-internal and cross-area pointers in the address registers. To define the address of an operand, you can call the stored data in the program.

Data is exchanged between the registers and the other available memory areas with the assistance of load and transfer instructions.

Note

In S7-1500, special rules apply to data exchange via address register and data block register:

- The values in the registers do not remain in existence beyond the block limits.
- The registers are reset when the language is changed within a block.
- You can only reference data in blocks with optimized access if these have the retain setting "Set in IDB".
- It is not possible to reference local data in blocks with optimized access with the help of the address registers (across areas).

Additional information

For more information on the statements that address registers use and on indirect addressing, refer to "See also".

See also

Indirect addressing in STL (Page 1471)

Addressing areas of a tag with slice access (Page 1461)

Indirect addressing in STL

In STL, the following options are available for indirect addressing:

- Memory-indirect addressing
- Register-indirect area-internal addressing
- Register-indirect cross-area addressing

Memory-indirect addressing

In the case of memory-indirect addressing, you store the address in a tag. The tag can be of WORD or DWORD data type. The tag can be located in the memory areas "Data" (DB or DI), "Bit memory" (M) or "Temporary local data" (L). In S7-1500, FB parameters can also be used to store the address. If the tag is located in a data block, it must be a data black with standard access.

The following example shows applications of memory-indirect addressing:

Addressing in STL	Explanation
`U E [MD 2]`	`// Execute an AND logic operation with a variable input bit. The address of the input bit is located in the memory double word MD2.`
`= DIX [DBD 2]`	`// Assign the RLO to a variable data bit. The address of the data bit is located in the data double word DBD2.`
`L EB [DID 4]`	`// Load a variable input byte to ACCU 1. The address of the input byte is located in the instance double word DID4.`
`AUF DB [LW 2]`	`// Open a variable data block. The number of the data block is located in the local data word LW2.`

Register-indirect area-internal addressing

Register-indirect addressing uses one of the address registers (AR1 or AR2) to pick up the address of the operand.

In the case of register-indirect, area-internal addressing, you index only the bit address and the byte address via the address register (e.g. `P#10.0`). You do not enter the memory area for which the address in the address register is to apply until during programming of the instruction. The address in the address register then moves to the memory area specified in the instruction.

Possible memory areas are "Inputs" (I), "Outputs" (Q), "I/O" (PI or PQ), "Bit memory" (M), "Temporary local data" (L) and "Data" (DB or DI). If the operand is located in a data block, it must be a data black with standard access.

When you enter register-indirect, area-internal addressing, specify an offset after the specification of the address register. This offset is added to the contents of the address register without changing the address register. This offset also has the format of a pointer. The specification of a pointer is mandatory and must be entered as constant (e.g. `P#0.0` or `P#2.0`).

The following example shows an application of register-indirect area-internal addressing:

STL	**Explanation**
`LAR1 P#10.0`	`// Load pointer (P#10.0) to address register 1`
`L IW [AR1, P#2.0]`	`// Increase contents of address register 1 (P#10.0) by offset P#2.0.`
	`// Load contents of input word IW12 into accumulator 1`
`L IW [AR1, P#0.0]`	`// Increase contents of address register 1 (P#10.0) by offset P#0.0.`
	`// Load contents of input word IW10 into accumulator 1`

Register-indirect cross-area addressing

In the case of register-indirect, cross-area addressing, use the address register to index the entire address of the operand, in other words, the bit address and byte address, as well as the memory area. Possible memory areas are "Inputs" (I), "Outputs" (Q), "I/O" (P), "Bit memory" (M), "Temporary local data" (L) and "Data" (DB or DI). If the operand is located in a data block, it must be a data black with standard access or the operand must have the retain setting "Set in IDB".

In the instruction, program only the operand width. Possible operand widths are bit, byte, word, and double word.

The following example shows an application of register-indirect cross-area addressing:

`LAR1 P#M10.0`	`// Load cross-area pointer (P#M10.0) to address register 1`
`L W [AR1, P#2.0]`	`// Increase contents of address register 1 (P#M10.0) by offset P#2.0.`
	`// Load contents of memory word "MW12" into accumulator 1`
`LAR1 P#A10.0`	`// Load cross-area pointer (P#A10.0) to address register 1`
`L W [AR1, P#2.0]`	`// Add contents of address register 1 (P#A10.0) by offset P#2.0`
	`// Load contents of output word QW12.0 into accumulator 1`

Note

Special features in S7-1500

In S7-1500, special rules apply to data exchange via address register and data block register:

- The values in the registers do not remain in existence beyond the block limits. The registers are also reset when the language is changed within a block.
- If you access an operand of the BYTE, WORD or DWORD type using register-indirect addressing, the address must begin at a byte limit.
  Examples:
  LAR1 P#0.0
  L MW [AR1, P#0.0] // P#0.0 + P#0.0 = P#0.0 - The addressing is allowed, because P#0.0 points to a byte limit.
  L MW [AR1, P#2.1] // P#0.0 + P#2.1 = P#2.1 - The addressing is not allowed, because P#2.1 does not point to a byte limit.

See also

Basics of indirect addressing (Page 1465)

Addressing structured variables (Page 1459)

Basic information about address registers (Page 1470)

11.1.1.5 Program flow control

EN/ENO mechanism

Basics of the EN/ENO mechanism

Introduction

Runtime errors that require a program abort can occur during the processing of instructions. You can use the EN/ENO mechanism to avoid such program aborts. This mechanism can be used at two levels:

- EN/ENO mechanism for individual instructions
- EN/ENO mechanism for block calls

EN/ENO mechanism for instructions in LAD/FBD

In LAD and FBD, certain instructions have an enable input EN and an enable output ENO.

You can use the enable input EN to make the execution of the instruction dependent on conditions. The instructions are only executed if the signal state is "1" at the enable input EN.

You can use the enable output ENO to query runtime errors in instructions and react to these.

The enable output ENO returns the signal state "1" if one of the following conditions applies:

- No error occurred during processing.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- The EN input has signal state "0".
- An error occurred during processing.

The EN/ENO mechanism is used for the following basic instructions:

- Mathematical functions
- Move operations
- Conversion operations
- Word logic operations
- Shift + rotate

In LAD and FBD, you can switch the evaluation of the enable output ENO on and off by means of the shortcut menu specifically for certain instructions.

EN/ENO mechanism for block calls in LAD/FBD

All blocks that you call in LAD or FBD are provided with an enable input EN and an enable output ENO when called. This applies to all called blocks, regardless of the programming language in which they were created.

You can use the enable input EN to call the block depending on conditions. The block is only executed if the signal state is "1" at the enable input EN.

You can query the error status of the block with the enable output ENO. It has signal "1" as soon as the execution of the block starts. If you do not explicitly set the output ENO to "0" in the program code, it retains signal "1".
However, you can explicitly set it to "0" to return an error statement to the called block. In LAD or FBD, the output ENO is set with the instruction "RET: Return".

See also:

Example of the use of the EN/ENO mechanism in LAD (Page 1476)

Example of the use of the EN/ENO mechanism in FBD (Page 1476)

EN/ENO mechanism for STL

In STL, the EN/ENO mechanism is not required for individual instructions. This function is mapped by language-specific instruction sequences.

Blocks that you call from an STL block are not provided with the EN and ENO parameters. Regardless of the programming language in which they were created, you can transfer an error statement to the calling STL block using the BR bit of the status word.

In STL, you can evaluate the error status of the called block by linking the BR bit of the status word with the RLO. It has signal "1" as soon as the execution of the block starts. If you do not explicitly set it to "0" in the program code, it retains signal "1".
However, you can explicitly set it to "0" to return an error statement to the calling block. In STL, the error statement is set with the instructions "SAVE" or "JNB".

See also: Example of the simulation of the EN/ENO mechanism in STL (Page 1478)

EN/ENO mechanism in SCL

With SCL, the use of the EN/ENO mechanism for instructions is optional. You can activate it with the block property "Set ENO automatically". If the property is active, all blocks implicitly receive an error handling.

You can implement a conditional block call with the enable input EN. Use the enable input EN in the parameter list as a normal input parameter. If EN has signal "1" or when EN is not used, the block is called. If EN has signal "0", the block is not called.

Note

When you call functions in SCL, you cannot use the release mechanism via EN. Use an IF statement instead to call functions conditionally.

You can query the error status of the block with the enable output ENO. If the ENO has signal "1", the block was processed without errors. If the ENO has signal "0", an error occurred during processing. To query the state of the enable output, insert an additional output parameter with the name ENO in the parameter list during a block call.

See also: Example of the use of the EN/ENO mechanism in SCL (Page 1477)

ENO in GRAPH (S7-1500)

In GRAPH, you can use the ENO operand to evaluate whether an action has been completed successfully. You can select the "Set ENO automatically" option for this.

- Select the option in the program properties in order for it to be used as the default setting for new GRAPH blocks.
- Select the option in the block properties in order for it to be used for specific blocks.

If this option is selected, the ENO operand is displayed when testing with program status. This operand has the value "TRUE" if the action was successful and "FALSE" if the action failed.

The ENO operand can be used in the following cases:

- When blocks are called that have the enable output ENO.
- For instructions in which errors can occur, e.g., conversions or mathematical functions.

EN/ENO mechanism for memory and I/O access errors

You cannot evaluate memory and I/O access errors with the EN/ENO mechanism. You do this either with the global troubleshooting via OBs (S7-300/400 and S7-1200/1500) or local troubleshooting using the "GetError" instruction (S7-1200/1500 only). If a memory access error occurred for an instruction, you can evaluate the associated ENO.

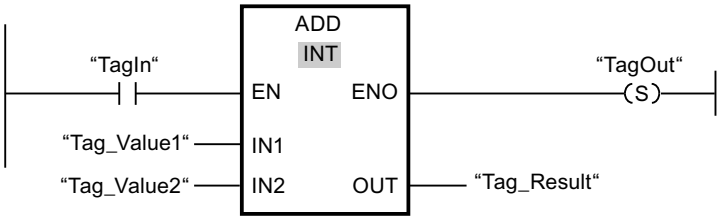
See also

Enabling and disabling the EN/ENO mechanism (Page 1608)

Example of the use of the EN/ENO mechanism in LAD

Description

The following figure shows an ADD instruction with EN and ENO protective circuit:



After the normally open contact, the RLO contains the previous result of logic operation:

- If "TagIn" signal is "0", the addition is not executed. EN and ENO both lead to the signal state "0".
- If "TagIn" signal is "1", EN is also "1" and the addition is executed. If no errors occur during the processing of the instruction, the output ENO also has the signal state "1" and the output ""TagOut"" is set.

See also

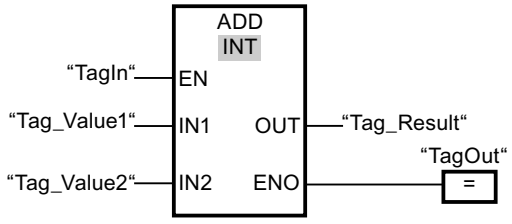
Basics of the EN/ENO mechanism (Page 1473)

ADD: Add (Page 2303)

Example of the use of the EN/ENO mechanism in FBD

Description

The following figure shows an ADD instruction with EN and ENO protective circuit:



- If "TagIn" signal is "1", EN is also "1" and the addition is executed. If no errors occur during the processing of the instruction, the output ENO also has the signal state "1" and the output ""TagOut"" is set.
- If "TagIn" signal is "0", the addition is not executed. EN and ENO both lead to the signal state "0".

See also

Basics of the EN/ENO mechanism (Page 1473)

Example of the use of the EN/ENO mechanism in SCL

Example of the EN/ENO mechanism for basic instructions

To use the EN/ENO mechanism for instructions in SCL, you have to activate the block property "Set ENO automatically". The following example shows the use of the enable output ENO for the "a/b" instruction.

SCL

```
"MyoutputREAL" :=a/b;
IF ENO
    THEN "MyOutputBool":=1;
    ELSE "MyOutputBool":=0;
END_IF;
```

If the "a/b" instruction is executed error-free, MyOutputBool has signal "1".

Example of the use of the EN/ENO mechanism in block calls

The following example shows the use of the enable output ENO for a block call.

SCL

```
"MyDB"."MyFB"(EN:="MyTag1">"MyTag2",
              in1:="MyInputBool1",
              in2:="MyInputBool1",
              EN0=>"MyOutputBool");
```

If MyTag1 is not greater than MyTag2 the block call is not processed. EN and ENO both lead to the signal state "0".

If MyTag1 is greater than MyTag2, EN has signal "1" and the block call is executed.

If all instructions within MyFB are executed error-free, MyOutputBool has signal "1".

See also

Basics of the EN/ENO mechanism (Page 1473)

Example of the simulation of the EN/ENO mechanism in STL

Description

The following example shows an program section for adding values with EN and ENO connected:

STL	Description
A"Tag_Input_1"	// Query whether the signal state of the operand is "1" and AND with current RLO
JNBMyLABEL	// Evaluation of the EN input
	// If RLO="0" jump to jump label "MyLABEL" and save the current RLO in the BR
	// Execute next instruction if RLO="1"
L"Tag_Input_2"	// Load first value of addition
L"Tag_Input_3"	// Load second value of addition
+I	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result"
AN OV	// Query if errors occurred
SAVE	// Transfer signal state of the RLO to the BR bit
CLR	// Reset RLO to "0"
MyLABEL: U BR	// Jump label "MyLABEL"
	// Query BR and AND with RLO
= "Tag_Output"	// Assign signal state of the RLO to the operand "Tag_Output"

The query of the operand "U" Tag_Input_1"" provides the result of the preceding logic operation (RLO). The instruction "Jump at RLO = 0 and save RLO (SPBNB)" saves the RLO to the BR. The instruction "Jump if RLO = 0 and save RL0" also evaluates the RLO and executes one of the following actions depending on the evaluation:

- If the RLO is "0", the processing of the program is continued at the jump label "MyLABEL" with the query of the BR. The addition is not executed. Assign the current RLO to the operand "Tag_Output".
- If the RLO is "1", the addition is executed. A query of the overflow bit (OV) shows if an error occurred during the addition. The query result is saved in the BR. The operation "CLR" resets the RLO to "0". The BR is then queried for "1" and AND'd with the current RLO. The result is assigned to the operand "Tag_Output". The signal state of the BR and of the operand "Tag_Output" shows if the addition was carried out with any error

See also

Basics of the EN/ENO mechanism (Page 1473)

11.1.2 Declaring PLC tags

11.1.2.1 Overview of PLC tag tables

Introduction

PLC tag tables contain the definitions of the PLC tags and symbolic constants that are valid throughout the CPU. A PLC tag table is created automatically for each CPU used in the project. You can create additional tag tables and use these to sort and group tags and constants.

In the project tree there is a "PLC tags" folder for each CPU of the project. The following tables are included:

- "All tags" table
- Standard tag table
- Optional: Other user-defined tag tables

All tags

The "All tags" table gives an overview of all PLC tags, user constants and system constants of the CPU. This table cannot be deleted or moved.

Standard tag table

There is one standard tag table for each CPU of the project. It cannot be deleted, renamed or moved. The default tag table contains PLC tags, user constants and system constants. You can declare all PLC tags in the default tag table, or create additional user-defined tag tables as you want.

User-defined tag tables

You can create multiple user-defined tag tables for each CPU to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. User-defined tag tables can contain PLC tags and user constants.

See also

Structure of the PLC tag tables (Page 1480)

Using tags within the program (Page 1447)

Basics of constants (Page 1449)

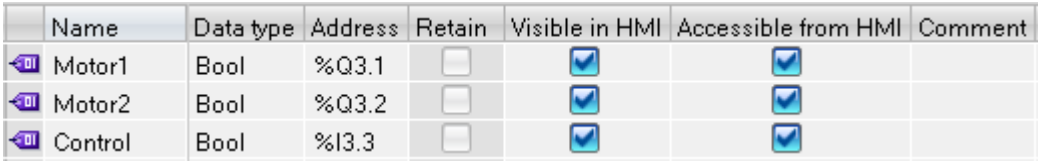
11.1.2.2 Structure of the PLC tag tables

Introduction

Each PLC tag table contains a tab for tags and a tab for user constants. The default tag table and the "All tags" table also have a "System constants" tab.

Structure of the "PLC tags" tab

In the "Tags" tab you declare the global PLC tags that you require in the program. The following figure shows the tab structure. The number of columns shown may vary.



	Name	Data type	Address	Retain	Visible in HMI	Accessible from HMI	Comment
	Motor1	Bool	%Q3.1	☐	☑	☑	
	Motor2	Bool	%Q3.2	☐	☑	☑	
	Control	Bool	%I3.3	☐	☑	☑	

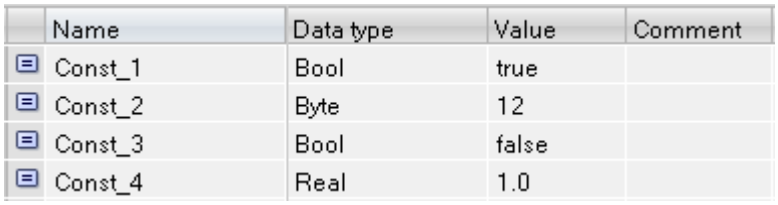
The following table shows the meaning of the individual columns. The number of columns shown may vary. You can show or hide the columns as required.

Column	Description
	Symbol you can click on to drag-and-drop a tag to a program for use as an operand.
Name	Unique name for the constants throughout the CPU.
Data type	Data type of the tags.
Address	Tag address.
Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off.
Accessible from HMI	Shows whether HMI can access this tag during runtime.
Visible in HMI	Shows whether the tag is visible by default in the operand selection of HMI.
Monitor value	Current data value in the CPU. This column only appears if an online connection is established and you select the "Monitor all" button.
Tag table	Shows which tag table includes the tag declaration. This column is only available in the "All tags" table.
Comment	Comment to document the tags.

Structure of the "User constants" and "System constants" tabs

In the "User constants" you define symbolic constants that are valid throughout the CPU. The constants required by the system are shown in the "Systems constants" tab. System constants can be hardware IDs, for example, which can be used for identification of modules.

The following figure shows the structure of both tabs. The number of columns shown may vary.



	Name	Data type	Value	Comment
	Const_1	Bool	true	
	Const_2	Byte	12	
	Const_3	Bool	false	
	Const_4	Real	1.0	

The following table shows the meaning of the individual columns. You can show or hide the columns as required.

Column	Description
	Symbol you can click to move a tag into a network via a drag-and-drop operation for use as an operand.
Name	Unique name for the constants throughout the CPU.
Data type	Data type of the constants
Value	Value of the constants
Tag table	Shows which tag table includes the constant declaration. This column is only available in the "All tags" table.
Comment	Comments to document the tags.

See also

Hardware data types (Page 1957)

Using tags within the program (Page 1447)

Basics of constants (Page 1449)

Overview of PLC tag tables (Page 1479)

Show and hide table columns (Page 1502)

Editing tables (Page 347)

11.1.2.3 Rules for PLC tags

Valid names of PLC tags

Permissible characters

The following rules apply to the use of names for PLC tags:

- Letters, numbers, special characters are permitted.
- Quotation marks are not permitted.

Unique tag names

The names of the PLC tags must be unique throughout the CPU, even if the tags are located in different tag tables of a CPU. A name that is already used for a block, another PLC tag or a constant within the CPU, cannot be used for a new PLC tag. The uniqueness check does not differentiate between use of small and capital letters.

If you enter an already assigned name another time, a sequential number is automatically appended to the second name entered. For example, if you enter the name "Motor" a second time, the second entry is changed to "Motor(1)".

Unique table names

The names of the PLC tag tables must also be unique throughout the CPU. A unique name is automatically suggested when user-defined PLC tag tables are being created.

See also

Using tags within the program (Page 1447)

Permissible addresses and data types of PLC tags (Page 1482)

Keywords (Page 1442)

Permissible addresses and data types of PLC tags

The addresses of PLC tags are made up of the particulars of the operand area and the address within this area.

The addresses must be unique throughout the CPU. If you enter an address that is already assigned to another tag, the address will be highlighted at both places in yellow and an error message will be issued.

Operand areas

The following table shows the possible operand areas. The available data types depend on the CPU you use:

Operand area		Description	Data type	Format	Address area:		
International mnemonics	German mnemonics				S7-1200	S7-300/400	S7-1500
I	E	Input bit	BOOL	I x.y E x.y	0.0..1023.7	0.0..65535.7	0.0..32767.7
I	E	Input (64-bit)	LWORD, LINT, ULINT, LTIME, LTOD, LDT, LREAL, PLC data type	I x.0 E x.0	-	-	0.0..32760.0

Operand area		Description	Data type	Format	Address area:		
Interna-tional mnemon-ics	German mnemon-ics				S7-1200	S7-300/400	S7-1500
IB	EB	Input byte	BYTE, CHAR, SINT, USINT, PLC data type	IB x EB y	0..1023	0..65535	0..32767
IW	EW	Input word	WORD, INT, UINT, DATE, S5TIME, PLC data type	IW x EW y	0..1022	0..65534	0..32766
ID	ED	Input double word	DWORD, DINT, UDINT, RE-AL, TIME, TOD, PLC da-ta type	ID x ED y	0..1020	0..65532	0..32764
Q	A	Output bit	BOOL	Q x.y A x.y	0.0..1023.7	0.0..65535.7	0.0..32767.7
Q	A	Output (64-bit)	LWORD, LINT, ULINT, LTIME, LTOD, LDT, LREAL, PLC data type	Q x.0 A x.0	-	-	0.0..32760.0
QB	AB	Output byte	BYTE, CHAR, SINT, USINT, PLC data type	QB x AB y	0..1023	0..65535	0..32767
QW	AW	Output word	WORD, INT, UINT, DATE, S5TIME, PLC data type	QW x AW y	0..1022	0..65534	0..32766
QD	AD	Output dou-ble word	DWORD, DINT, UDINT, RE-AL, TIME, TOD, PLC da-ta type	QD x AD y	0..1020	0..65532	0..32764
M	M	Memory bit	BOOL	M x.y	0.0..8191.7	0.0..65535.7	0.0..16383.7
M	M	Bit memory (64-bit)	LREAL	M x.0	0.0..8184.0	-	0.0..16376.0
M	M	Bit memory (64-bit)	LWORD, LINT, ULINT, LTIME, LTOD, LDT	M x.0	-	-	0.0..16376.0
MB	MB	Memory byte	BYTE, CHAR, SINT, USINT	MB x	0..8191	0..65535	0..16383

Operand area		Description	Data type	Format	Address area:		
Interna-tional mnemon-ics	German mnemon-ics				S7-1200	S7-300/400	S7-1500
MW	MW	Memory word	WORD, INT, UINT, DATE, S5TIME	MW x	0..8190	0..65534	0..16382
MD	MD	Memory dou-ble word	DWORD, DINT, UDINT, RE-AL, TIME, TOD	MD x	0..8188	0..65532	0..16380
T	T	Time function (for S7-300/400 only)	Timer	T n	-	0..65535	0..2047
C	Z	Counter func-tion (for S7-300/400 only)	Counter	Z n C n	-	0..65535	0..2047

Addresses

The following table shows the possible addresses of tags:

Data type	Address	Example
BOOL	Tags with BOOL data type are addressed with a byte number and a bit number. The numbering of the bytes begins for each operand area at 0. The numbering of the bits goes from 0 to 7.	A 1.0
BYTE, CHAR, SINT, USINT	Tags with BYTE, CHAR, SINT, and USINT data type are addressed with a byte number.	MB 1
WORD, INT, UINT, DATE, S5TIME	Tags with WORD, INT, UINT, DATE, S5TIME data type consist of two bytes. They are addressed with the number of the lowest byte.	IW 1
DWORD, DINT, UDINT, RE-AL, TIME, TOD	Tags with DWORD, DINT, UDINT, REAL, TIME, TOD data type con-sist of four bytes. They are addressed with the number of the lowest byte.	AD 1
LWORD, LINT, ULINT, LTIME, LTOD, LDT, LREAL	Tags of data type LWORD, LINT, ULINT, LTIME, LTOD, LDT, and LREAL consist of eight bytes. They are addressed with it number 0 and the number of the lower byte.	I 1.0

Mnemonics used

The addresses that you enter in the PLC tag table are automatically adapted to the set mnemonics.

See also

Setting the mnemonics (Page 1539)

Using tags within the program (Page 1447)

Valid names of PLC tags (Page 1481)

Overview of the valid data types (Page 1908)

11.1.2.4 Creating and managing PLC tag tables

Creating a PLC tag table

You can created multiple user-defined PLC tag tables in a CPU. Each tag table must have have a unique name throughout the CPU.

Requirement

The project view is open.

Procedure

To created a new PLC tag table, follow these steps:

1. Open the "PLC tags" folder under the CPU in the project tree.
2. Double-click the "Add new tag table" entry.
   A new PLC tag table with the default name "TagTable_x" is created.
3. Select the PLC tag table in the project tree.
4. Select the "Rename" command in the shortcut menu.
5. Type in a name that is unique throughout the CPU.

Result

A new PLC tag table is created. You can now declare tag and constants in this table.

See also

Overview of PLC tag tables (Page 1479)

Structure of the PLC tag tables (Page 1480)

Grouping PLC tag tables

You can gather the user-defined tag tables of the CPU into groups. You cannot, however, move the standard tag table and the "All tags" table into a group.

Requirement

Multiple user-defined tag tables are contained in the "PLC tags" folder of the CPU.

Procedure

To gather multiple PLC tag tables into a group, follow these steps:

1. Select the "PLC tags" folder under the CPU in the project tree.
2. Select the "Insert > Group" menu command.
   A new group with the standard name "Group_x" is inserted.
3. Select the newly inserted group in the project tree.
4. Select the "Rename" command in the shortcut menu.
5. Assign the new group a unique name throughout the CPU.
6. Drag to the new group the tables you want to group together.

Result

The tag tables are gathered in the new group.

See also

Overview of PLC tag tables (Page 1479)

Structure of the PLC tag tables (Page 1480)

Opening the PLC tag table

Procedure

To open the PLC tag table in a CPU, proceed as follows:

1. Open the "PLC tags" folder under the CPU in the project tree.
2. Double-click the PLC tag table in the folder.
3. Select the desired tab in the upper corner.

Result

The PLC tag table associated with the CPU opens. You can declare the required tags and constants.

See also

Overview of PLC tag tables (Page 1479)

Structure of the PLC tag tables (Page 1480)

11.1.2.5 Declaring PLC tags

Entering a PLC tag declaration

Declaring tags in the PLC tag table

Requirements

The "Tags" tab of the PLC tag table is open.

Procedure

To define PLC tags, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
   An address corresponding to the data type is automatically appended.
3. Optional: Click on the arrow key in the "Address" column and enter an operand identifier, an operand type, an address and a bit number in the dialog which then opens.
4. Optional: Enter a comment in the "Comments" column.
5. Repeat steps 1 to 4 for all the tags you require.

See also: Permissible addresses and data types of PLC tags (Page 1482)

Syntax check

A syntax check is performed automatically after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. As long as the tag declaration contains syntax errors and the tag is used in the program, you will not be able to compile the program.

See also

Valid names of PLC tags (Page 1481)

Declaring PLC tags in the program editor (Page 1487)

Structure of the PLC tag tables (Page 1480)

Editing tables (Page 347)

Declaring PLC tags in the program editor

Requirement

- The program editor is open.

Procedure

To declare operands as global PLC tags, follow these steps:

1. Insert an instruction in your program.
   The "<???>", "<??.?>" or "..." strings represent operand placeholders.
2. Replace an operand placeholder with the name of the PLC tag to be created.
3. Select the tag name.
   If you want to declare multiple PLC tags, select the names of all the tags to be declared.
4. Select the "Define tag" command in the shortcut menu.
   The "Define tag" dialog box opens. This dialog displays a declaration table in which the name of the tag is already entered.
5. Click the arrow key in the "Section" column and select one of the following entries:
   - Global Memory
   - Global Input
   - Global Output
6. In the other columns, enter the address, data type, and comments.
   See also: Permissible addresses and data types of PLC tags (Page 1482)
7. If the CPU contains multiple PLC tag tables, you can use an entry in the "PLC tag table" column to indicate in which table the tag is to be inserted. If you make no entry in the column, the new tag will be inserted in the default tag table.
8. Click the "Define" button to complete your entry.

Result

The tag declaration is written to the PLC tag table and is valid for all blocks in the CPU.

See also

Valid names of PLC tags (Page 1481)

Editing tables (Page 347)

Declaring tags in the PLC tag table (Page 1487)

Setting the retentivity of PLC tags

Retentive behavior of PLC tags

Retentive PLC tags

Each CPU has a memory area whose content remains available even after the supply voltage has been switched off. This area is referred to as retentive memory area.

To avoid data loss during power failure, you can save specific PLC tags to this memory area. You specify the retain setting of PLC tags in the PLC tag table.

Depending on the CPU family, the retentive memory area can accommodate various type of PLC tags. The following table provides an overview of the options of the various CPUs:

CPU type	Retentive bit memories	Retentive SIMATIC timers	Retentive SIMATIC counters
S7-300/400 series	✓	-	-
S7-1200 series	✓	-	-
S7-1500 series	✓	✓	✓

See also

Setting the retentive behavior of PLC tags (Page 1489)

Setting the retentive behavior of PLC tags

Introduction

In the PLC tag table you can specify the width of the retentive memory area for PLC tags. All tags with addresses in this memory area are then designated as retentive. You can recognize the retentivity setting of a tag by the check mark set in the "Retain" column of the PLC tag table.

Requirement

The "PLC tags" tab of the PLC tag table is open.

Procedure

To define the width of the retentive memory area for PLC tags, follow these steps:

1. On the toolbar, click the "Retain" button.
   The "Retain memory" dialog will open.
2. Specify the width of the retentive memory area by entering the number of retentive bytes, timers or counters in the input field.
3. Click the "OK" button.

Result

The width if the retentive memory area is defined. In the "Retain" column of the tag table a check mark is automatically set for all tags that are located within the retentive memory area.

See also

Retentive behavior of PLC tags (Page 1488)

Editing tables (Page 347)

11.1.2.6 Grouping PLC tags for inputs and outputs in structures

Other useful information regarding structured PLC tags

Use of structured PLC tags (S7-1200 V4 and higher/S7-1500)

To make your program easier to view, you can group several input or output addresses in a higher-level PLC tag. The higher-level PLC tag represents a structure that contains several logically related inputs or outputs. When the block is called, you transfer the higher-level tag and thus need only an input or output parameter for all associated inputs or outputs.

Principle of operation

To create a structured PLC tag, you initially define a PLC data type (UDT). In it you declare the necessary data elements and specify their names and data types.

Then, you switch to the PLC tag table and specify the higher-level PLC tag there. As a data type for the tag, you select your PLC data types. The system now reserves a certain number of input or output addresses starting from the start address of the higher-level tag. The number of reserved addresses depends on the length of your PLC data type.

If you call a block that requires the reserved inputs or outputs for program execution, you transfer the higher-level tag as a block parameter.

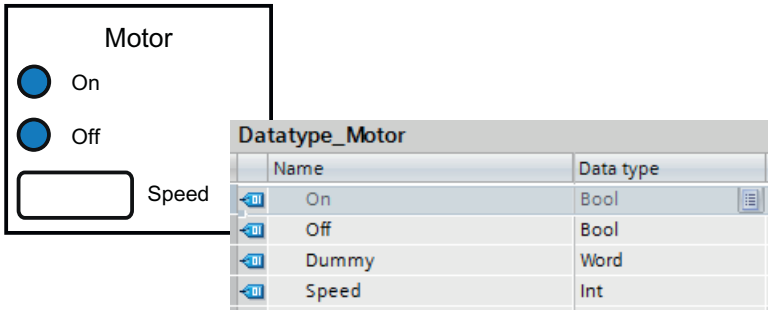
You can address the individual PLC tags like structure elements in the program code.

A detailed description of the individual handling steps can be found in the following chapters.

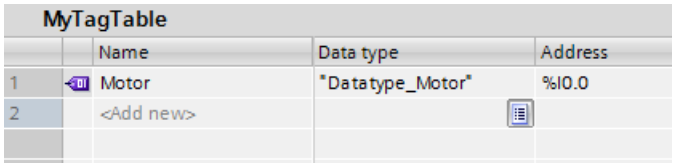
Application example

You can use structured PLC tags in order to group the inputs or outputs of a function module. The following figure shows the schematic representation of a motor: A component was created in the "Datatype_Motor" PLC data type for each of the three inputs.

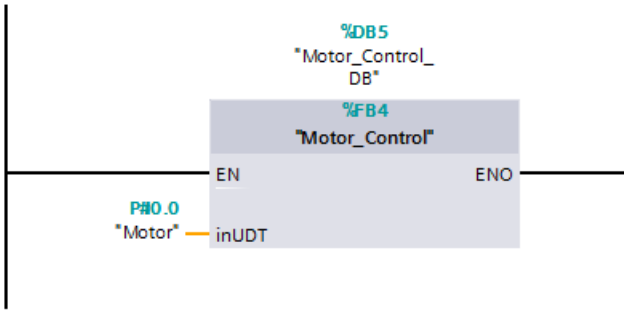
The memory areas of the declared tags must not overlap. In the example, you see that the "Speed" component has the data type "Integer" and therefore must start at a word address. For this reason, the first input word has been filled with the "Dummy" fill tag. This means that "Speed" is located on the second input word.



The following figure shows the higher-level "Motor" PLC tag that is based on the "Data type_Motor" data type. With the declaration of "Motor", the addresses IW0 and IW1 are reserved on the input module.



The following figure shows the transfer of the "Motor" PLC tag as an input parameter of the "Motor_Control" block.



You can address the individual components of the tag in the "Motor_Control" block.

Addressing	Description
"Motor"	Addressing the higher-level PLC tag.
"Motor".On	Addressing a component of a structured PLC tag.
"Motor".On:P	Addressing an I/O input or output (PI or PQ).

Rules for use of structured PLC tags

Note the following rules when creating and using structured PLC tags.

- Structured PLC tags can be used in the "Inputs" and "Outputs" operand areas.
- Structured tags are not permitted in the bit memory address area.
- Structured PLC tags cannot be addressed from HMI.

Observe the following rules when creating the PLC data type that is to serve as the basis for a PLC tag:

- The memory areas of the individual elements must not overlap.
  See also: Permissible addresses and data types of PLC tags (Page 1482)
- Do not group inputs and outputs in one PLC data type but create different PLC data types for inputs and outputs.

- Do not group inputs or outputs from different modules in a PLC data type because it is not guaranteed that the process images of the modules are updated synchronously.
- In the lower-level PLC data types, all data types are permitted except for "STRING" and "WSTRING".

See also

Creating structured PLC tags (Page 1492)

Creating structured PLC tags

Rules

Note the following rules when creating structured PLC tags:

- Use separate PLC data types for the "Inputs" and "Outputs" operand areas.
- Structured tags are not permitted in the bit memory address area.
- Do not group inputs or outputs from different modules in a PLC data type because it is not guaranteed that the process images of the modules are updated synchronously.

Procedure

To create a structured PLC tag, follow these steps:

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare all the necessary components in the PLC type. All data types except for "STRING" and "WSTRING" are permitted.
3. Select the PLC data type in the project tree and select the "Compile > Software (only changes)" command from the shortcut menu.
   The PLC data type is compiled and can then be used in the PLC tag table.
   Even when you make changes to existing PLC data types, you must recompile the program. This updates all locations of use of the PLC data type.
4. Open a PLC tag table within the same CPU.
5. Declare a new tag or select an existing tag.
6. In the "Data type" column, select the PLC data type and assign it to the PLC tag.
   The PLC tag receives the structure of the PLC data type. A suitable address is assigned automatically. Structured PLC tags always start at word addresses.
   The highest structure element is displayed without its subelements in the table.

Note

Assignment rules and default values

- For the declaration of the PLC data type, note that the memory areas of the individual tags must not overlap. For example, tags of data type "Integer" must start at a word limit. If necessary, insert "fill tags" to prevent overlaps.
  See also: Permissible addresses and data types of PLC tags (Page 1482)
- It is not possible to assign default values for the individual components. Values that you enter in the "Default value" column are not evaluated. Tags of data types "DT" and "DTL" may therefore contain invalid values.

See also

Other useful information regarding structured PLC tags (Page 1490)

11.1.2.7 Declaring global constants

Rules for global user constants

Permitted characters

Names of global constants may consist of the following characters:

- Letters, numbers, special characters are permitted.
- Quotation marks are not permitted.

Unique constant names

The names of global constants must be unique throughout the CPU, even if the constants are located in different tag tables of a CPU. A name that is already used for a block, a PLC tag or another constant within the CPU, cannot be used for new constant. The uniqueness check does not differentiate between use of small and capital letters.

If you enter an already assigned name another time, a sequential number is automatically appended to the second name entered. For example, if you enter the name "Motor" a second time, the second entry is changed to "Motor(1)".

Permitted data types

For constants, all data types supported by the CPU are permitted, with the exception of structured data types.

Permitted values

You can select any value from the value range of the specified data type as constant value. For information on the value ranges, refer to the "Data types" chapter.

See also: Auto-Hotspot

See also

Basics of constants (Page 1449)

Declaring global constants (Page 1495)

Rules for global system constants

Definition

System constants are global constants, unique throughout the CPU, that are required and automatically created by the system. System constants can, for example, be used to address and identify hardware objects.

Rules

System constants are assigned automatically when components are inserted in the device or network view, and entered in the default tag table ("System constants" tab). A system constant is created for each module, but also for each submodule. An integrated counter, for example, therefore receives a system constant as well. System constants consist of a symbolic name and a numeric HW identifier and cannot be changed.

Names of system constants

The names of the system constants are structured hierarchically. They consist of a maximum of four hierarchical levels, each of which is separated by a tilde "~". Based on the name, you can therefore recognize the "path" to the relevant hardware module.

Example

A system constant with the name "`Local~PROFINET_interface_1~Port_1`" designates Port 1 of the PROFINET interface 1 of the local CPU.

See also:

Auto-Hotspot

Auto-Hotspot

Instructions for address conversion (Page 3358)

Declaring global constants

Introduction

You declare constants in the "User constants" tab of a PLC tag table. During declaration you have to to enter a symbolic name, a data type and a fixed value for each constant. You can select any value from the value range of the specified data type as constant value. For information on the value ranges, refer to the "Data types" chapter.

See also: Auto-Hotspot

Procedure

To declare constants, follow these steps:

1. Open a PLC tag table.
2. Open the "User constants" tab.
   The constants table opens.
3. Enter a constant name in the "Name" column.
4. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
5. Enter a constant value in the "Value" column; this constant value must be valid for the selected data type.
6. If you want, enter comments on the constants in the "Comments" column. The entry of a comment is optional.
7. If you want to declare additional constants, place the cursor in the next row and repeat steps 3 to 6.

Syntax check

A syntax check is performed automatically after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. As long as the tag declaration contains syntax errors and the constant is used in the program, you will not be able to compile the program.

See also

Opening the PLC tag table (Page 1486)

Inserting a table row in the PLC tag table (Page 1499)

Structure of the PLC tag tables (Page 1480)

Rules for global user constants (Page 1493)

Editing tables (Page 347)

11.1.2.8 Editing properties

Editing the properties of PLC tags

Properties of PLC tags

Overview

The following table provides an overview of the properties of PLC tags. The display of properties may vary depending on the CPU type.

Group	Property	Description
General	Name	A unique name within the CPU.
	Data type	Data type of the tags.
	Address	Tag address.
	Retain	Shows whether the tag is in the retentive memory area.
	Comment	Comment on the tag.
History	Date created	Time when the tag was created (cannot be changed).
	Last modified	Time when the tag was last changed (cannot be changed).
Usage	Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
	Accessible from HMI	Shows whether HMI can access this tag during runtime.

See also

Editing the properties of PLC tags (Page 1496)

Editing the properties of PLC tags

Editing properties in a PLC tag table

To edit the properties of one or more tags, follow these steps:

1. In the project tree, double-click the PLC tag table that contains the tags.
   The PLC tag table opens.
2. Change the entries in the columns.

Editing addresses in the program editor

To edit the address of a tag in the program editor, follow these steps:

1. Select the tag name.
2. Select the "Rewire tag" command in the shortcut menu.
   The "Rewire tag" dialog will open. The dialog shows a declaration table.

3. Enter the new address in the "Address" column.
4. Click the "Change" button to confirm the input.

Editing names in the program editor

To edit the name of a tag in the program editor, follow these steps:

1. Select the tag name.
2. Select the "Rename tag" command in the shortcut menu.
   The "Rename tag" dialog opens. The dialog shows a declaration table.
3. Enter the new name in the "Name" column.
4. Click the "Change" button to confirm the input.

Effect in the program

In the case of a change of the tag's name, data type or address, each location of use of the tag is automatically updated in the program.

See also

Properties of PLC tags (Page 1496)

Editing properties of global constants

Properties of global constants

Overview

The following table gives an overview of the properties of constants:

Group	Property	Description
General	Name	A unique name within the table
	Data type	Data type of the constants
	Value	Value that you defined for the constants. This value must be compatible with the declared data type. See also: Auto-Hotspot
	Comment	Comment on the constants
History	Date created	Time when the constant was created (cannot be changed)
	Last modified	Time when the constant was last changed (cannot be changed)

Editing properties of global constants

Editing properties in a PLC tag table

To edit the properties of one or more constants, follow these steps:

1. In the project tree, double-click the PLC tag table that contains the constants.
   The PLC tag table opens.
2. Open the "User constants" tab.
3. Change the entries in the "Name", "Data type", "Value", or "Comments" column.

Effect in the program

In the case of a change of a constant's name, data type or value, each location of use of the constant is automatically updated in the program.

See also

Editing tables (Page 347)

11.1.2.9 Monitoring of PLC tags

Monitoring of PLC tags

You can monitor the current data values of the tags on the CPU directly in the PLC tag table.

Requirement

An online connection to the CPU exists or is possible.

Procedure

To monitor the data values, proceed as follows:

1. Open a PLC tag table.
2. Start monitoring by clicking the "Monitor all" button.
   - If no online connection to the CPU exists, this is established.
   - Monitoring is started with the trigger setting "Permanent".
   - The additional "Monitor value" column is displayed in the table. This shows the current data values.
   - The symbol for the forcing of tags is displayed if a tag is currently being forced.
3. End the monitoring by clicking the "Monitor all" button again.

Note

Editing PLC tags during the monitoring of tags

If the editing of tags is started and the PLC tag table is edited, for example by adding new tags, the monitoring is re-started after editing is complete.

Note

You also have the option of copying PLC tags to a monitor or force table so that you can monitor, control or force them in the table.

See also

Structure of the PLC tag tables (Page 1480)

Introduction to testing with the watch table (Page 1854)

Introduction for testing with the force table (Page 1881)

Copying entries in the PLC tag table (Page 1499)

11.1.2.10 Editing PLC tag tables

Inserting a table row in the PLC tag table

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

See also

Editing tables (Page 347)

Copying entries in the PLC tag table

You can copy PLC tags within a table or to other tables.

Procedure

To copy a tag, follow these steps:

1. Select the tags you want to copy.
   You can also select several tags by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last tag.
2. Select "Copy" in the shortcut menu.
3. Position the insertion pointer at the location where you want to insert the tags.
4. Select "Paste" in the shortcut menu.

Or

1. Select the tag.
2. Hold down the left mouse button.
3. At the same time, press <Ctrl>.
4. Drag the tag to the destination.

Result

- The tag is copied to the destination.
- If there is a name conflict, a number is automatically appended to the tag name. For example, "Tag" becomes "Tag(1)".
- All other properties of the tag remain unchanged.

See also

Editing tables (Page 347)

Deleting entries in the PLC tag table

Procedure

Follow the steps below to delete elements:

1. Select the row with the element to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

See also

Editing tables (Page 347)

Sorting rows in the PLC tag tables

You can sort the rows in the tables alphanumerically by name, data type, or address.

Procedure

To sort the table rows, follow these steps:

1. Select the column by which you want to sort.
2. Click the column header.
   The column will be sorted in order of increasing values.
   An up arrow shows the sort sequence.
3. In order to change the sort sequence, click the arrow.
   The column will be sorted in order of decreasing values. A down arrow shows the sort sequence.
4. To restore the original sequence, click a third time on the column header.

See also

Editing tables (Page 347)

Automatically filling in cells in the PLC tag table

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

If you fill the cells in the column "Address" automatically, the addresses will be increased depending on the indicated data type.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically. If entries are already present in the cells that are to be automatically updated, a dialog appears in which you can indicate whether you want to overwrite the existing entries or whether you want to insert new rows for the new tags.

See also

Editing tables (Page 347)

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.
5. To hide or show several columns, click "More" and activate or deactivate the check box of the corresponding columns in the "Show/Hide" dialog.

See also

Editing tables (Page 347)

Editing PLC tags with external editors

Basics for importing and exporting

Introduction

You can export PLC tag tables to a standardized XLSX format for editing with external table editors. Similarly, you can import PLC tag tables created with external table editors to the TIA Portal.

Overwriting existing PLC tags and constants during import

Existing entries of the same name will be overwritten during import if they have the same name as the entries that will be imported.

Link to existing objects

References to PLC tags or constants that already exist in the project are updated automatically during import. The update is executed based on the name of the PLC tags and constants.

See also

Format of the export file (Page 1503)

Exporting PLC tags (Page 1503)

Importing PLC tags (Page 1504)

Format of the export file

Introduction

During the export of PLC tag tables, a standardized XSLX format will be generated that you can edit with external table editors.

This format is also expected during the import of tables.

Format of the export file

The sheet is always named "PLCTags". This sheet can contain the displayed columns. The sorting order of columns can vary. The sheet does not necessarily have to include all columns. During import, the following values will be identified by a <no value> entry.

The names of the column headers are also clearly defined and are always expected in English.

The following table specifies the contents expected for the individual columns:

Element	Description
Name	Name of the tags
Path	Group and name of the PLC tag table
Data Type	The notation of the data type corresponds to the notation used in the PLC tag table.
Logical Address	The address can be specified with German or international mnemonics.
Comment	Free-form comments
Hmi Visible	The value TRUE or FALSE is expected.
Hmi Accessible	The value TRUE or FALSE is expected.

See also

Basics for importing and exporting (Page 1502)

Exporting PLC tags (Page 1503)

Importing PLC tags (Page 1504)

Exporting PLC tags

Requirement

A PLC tag table is open.

Procedure

To export PLC tags and constants, follow these steps:

1. In the PLC tag table, click the "Export" button.
   The "Export to Excel" dialog opens.
2. Select the path to which you want to save the export file.
3. Select whether to export tags and/or constants.
4. Click the "OK" button.

Result

The export file will be generated. Errors and warnings generated during export are indicated in the "Info" tab of the Inspector window.

See also

Basics for importing and exporting (Page 1502)

Format of the export file (Page 1503)

Importing PLC tags (Page 1504)

Importing PLC tags

Requirement

A table exists and it conforms to format specifications.

Procedure

To import a PLC tag table, follow these steps:

1. Open the "All tags" table.
2. Click the "Import" button.
   The "Import from Excel" dialog opens.
3. Select whether to import PLC tags and/or constants.
4. Select the table you want to import.
5. Click the "OK" button.

Result

The PLC tag table will be imported.

Errors and warnings generated during export are indicated in the "Info" tab of the Inspector window.

See also

Basics for importing and exporting (Page 1502)

Format of the export file (Page 1503)

Exporting PLC tags (Page 1503)

Editing individual PLC tags with external editors

To edit individual PLC tags in external editors outside the TIA portal, you can export or import these tags using copy and paste. However, you cannot copy structured tags to an editor.

Requirement

A PLC tag table and an external editor are opened.

Procedure

To export and import individual PLC tags, follow these steps:

1. Select one or more PLC tags.
2. Select "Copy" in the shortcut menu.
3. Switch to the external editor and paste the copied tags.
4. Edit the tags as required.
5. Copy the tags in the external editor.
6. Switch back to the PLC tag table.
7. Select "Paste" in the shortcut menu.

Note

You also have the option of export or importing PLC tags as mass data.

See also: Auto-Hotspot

11.1.3 Creating and managing blocks

11.1.3.1 Creating blocks

Block folder

Function

You can find a "Program blocks" folder in the project tree, in which you can create and manage the following blocks:

- Organization blocks (OB)
- Function blocks (FB)
- Functions (FCs)
- Data blocks (DB)

A "System blocks" subfolder containing another subfolder, "Program resources", is also created in the "Program blocks" folder the first time you drag an instruction to your program which is an internal system function block. The instance data block of the internal system function block is also pasted to the "Program resources" folder. You can move or copy such instance data blocks from the "Program resources" folder to any other folder and rename or delete them. You can also move your blocks into the "Program resources" folder. Blocks in the "Program resources" folder that are not required to run the user program are removed during the next compilation. If the "Program resources" folder contains no more blocks then it is also deleted with the "System blocks" folder.

A program cycle OB is automatically generated for each device and inserted in the "Program blocks" folder.

See also

Creating functions and function blocks (Page 1507)

Creating data blocks (Page 1508)

Creating organization blocks (Page 1506)

Using blocks from libraries (Page 1510)

Creating organization blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create an organization block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Organization block (OB)" button.
3. Select the type of new organization block.
4. Enter a name for the new organization block.
5. Enter the properties of the new organization block.
6. To enter additional properties for the new organization block, click "Additional information".
   An area with further input fields is displayed.
7. Enter all the properties you require.
8. Activate the "Add new and open" check box if the organization block does not open as soon as it is created.
9. Confirm your entries with "OK".

Result

The new organization block is created. You can find the organization block in the project tree in the "Program blocks" folder. You can assign additional parameters to some organization blocks in the inspector window or device view after they have been created. The organization block description will state whether the newly created organization block has additional parameters.

See also

Organization blocks (OB) (Page 1414)

Block folder (Page 1506)

Creating functions and function blocks (Page 1507)

Creating data blocks (Page 1508)

Using blocks from libraries (Page 1510)

Entering a block title (Page 1515)

Entering a block comment (Page 1516)

Creating functions and function blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a function (FC) or a function block (FB), follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Function block (FB)" or "Function (FC)" button.
3. Enter a name for the new block.
4. Enter the properties of the new block.
5. To enter additional properties for the new block, click "Additional information".
   An area with further input fields is displayed.
6. Enter all the properties you require.
7. Activate the "Add new and open" check box if the block does not open as soon as it is created.
8. Confirm your entries with "OK".

Result

The new block is created. You can find the block in the project tree in the "Program blocks" folder.

See also

Function blocks (FB) (Page 1415)

Functions (FCs) (Page 1414)

Basics of block access (Page 1419)

Block folder (Page 1506)

Creating organization blocks (Page 1506)

Creating data blocks (Page 1508)

Using blocks from libraries (Page 1510)

Entering a block title (Page 1515)

Entering a block comment (Page 1516)

Creating data blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a data block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Select the type of the data block. You have the following options available to you:
   - To create a global data block, select the list entry "Global DB".
   - To create an ARRAY data block, select the "ARRAY DB" entry in the list.
   - To create an instance data block, select the function block to which you want to assign the instance data block from the list. The list contains only the function blocks that were previously created for the CPU.
   - To create a data block based on a PLC data type, select the PLC data type from the list. The list contains only the PLC data types that were previously created for the CPU.
   - To create a data block based on a system data type, select the system data type from the list. The list contains only those system data types that have already been inserted to program blocks in the CPU.
4. Enter a name for the data block.
5. Enter the properties of the new data block.
6. If you have selected an ARRAY DB as the data block type, enter the ARRAY data type and the high limit for the ARRAY.
   You can change the high limit for the ARRAY at any time in the property window of the created block. The ARRAY data type cannot be changed subsequently.
7. To enter additional properties of the new data block, click "Additional information".
   An area with further input fields is displayed.
8. Enter all the properties you require.
9. Activate the "Add new and open" check box if the block does not open as soon as it is created.
10. Confirm your entry with "OK".

Result

The new data block is created. You can find the data block in the project tree in the "Program blocks" folder.

See also

Global data blocks (DB) (Page 1416)

Instance data blocks (Page 1417)

Block folder (Page 1506)

Creating organization blocks (Page 1506)

Creating functions and function blocks (Page 1507)

Using blocks from libraries (Page 1510)

Basics of block access (Page 1419)

System data types (Page 1955)

Using blocks from libraries

You can save blocks in the project library or in a global library, so that you can use them more than once within a user program. You can insert the blocks as master copies or as types.

See also: Library basics (Page 470)

Requirement

- The "Libraries" task card is displayed.
- No write protection is set for global libraries.

Adding blocks as master copies to the project library or to a global library

To add new blocks as master copies to the project library or to a global library, follow these steps:

1. Maximize the project library or the global library.
2. Use drag-and-drop to move the block you wish to add to the library to the "Master copies" folder or any one of the "Master copies" subfolders in the project library or a global library. Do not release the left mouse button until a small plus sign appears underneath the mouse pointer.

Or:

1. Copy the element you want to add as master copy.
2. Maximize the project library or the global library.
3. Right-click the "Master copies" folder or any of its subfolders.
4. Select "Paste" in the shortcut menu.

Adding blocks as types to the project library or to a global library

To add new blocks as types to the project library or to a global library, follow these steps:

1. Maximize the project library or the global library.
2. Drag-and-drop the element you want to add as a type into the "Types" folder or any of its subfolders in the project library or a global library. Do not release the left mouse button until a small plus sign appears underneath the mouse pointer.

Or:

1. Copy the element you want to add as a type.
2. Maximize the project library or the global library.

3. Right-click the "Types" folder or any of its subfolders.
4. Select "Paste" in the shortcut menu.

Using blocks of the project library or a global library

To use a block from the project library or a global library in your project, follow these steps:

1. Maximize the project library or the global library so that you can see the block you wish to use.
2. Use a drag-and-drop operation to move the block to the CPU block folder. If the selected insertion points is not allowed, the mouse pointer will appear as a circle with a slash.

Note

If you derive an instance from a type in a global library, the type is also inserted into the project library. The instance is then only linked to the type in the project library.

See also

Using libraries (Page 470)

Copying and pasting blocks

Basics of copying and pasting blocks

Function

You can also create new blocks by copying existing blocks and pasting the copy. Note the following principles when copying to CPUs of the same device family:

- You can copy organization blocks (OBs), functions (FCs), function blocks (FBs), and global data blocks (DBs) without restriction.
- You can copy instance data blocks only for the same function block, since the assignment to the function block cannot be changed afterwards. However, the assignment is canceled if you copy the instance data block to a different CPU. If a function block with the same name exists there, the instance data block will be assigned to this function block. If you copy the instance data block together with the function block into the other CPU, the instance data block is assigned to the copy of the function block.

The various device families sometimes support different blocks, especially in the case of organization blocks. However, function blocks and functions can also be programmed on the

various devices with different access types. Therefore, not all blocks are supported on all devices. Note the following principles when copying to a different device family:

- Copying to an S7-1200 CPU:
  - Organization blocks with "Optimized" access type can be copied to an S7-1200. If the copied OB type is supported by the S7-1200 CPU, the copied OB retains the properties of its event. You must, however, compile it again.
  - Although organization blocks with the "Standard" access type can be copied to an S7-1200, they are not supported by the CPU.
  - Function blocks (FBs), functions (FCs) and global data blocks (DBs) with "Optimized" access type can be copied to an S7-1200. However, they must be recompiled after this.
  - Although function blocks (FBs), functions (FCs) and global data blocks (DBs) with "Standard" access type can be copied to an S7-1200, they are not supported by the CPU.
  - Instance data blocks: If there is a function block in the target CPU with the name that was assigned to the instance data block in the source CPU, the instance data block is assigned to the function block in the target CPU. If you copy the instance data block together with the function block to which it was assigned in the source CPU into the target CPU, the instance data block is assigned to the copy of the function block.
- Copying to an S7-1500 CPU:
  - Organization blocks with "Optimized" access type can be copied to an S7-1500. If the copied OB type is supported by the S7-1500 CPU, the copied OB retains the properties of its event. You must, however, compile it again. OB types that are not supported receive a no parking symbol.
  - Organization blocks with "Standard" access type can be copied to an S7-1500. If the OB derives from an S7-300/400 CPU, it receives the standard event of the corresponding OB type. If the OB derives from an S7-1200/1500 CPU, it receives the properties of its event. However, it must be compiled again.
  - Function blocks (FBs), functions (FCs) and global data blocks (DBs) with "Optimized" access type can be copied to an S7-1500. However, they must be recompiled after this.
  - Although function blocks (FBs), functions (FCs) and global data blocks (DBs) with "Standard" access type can be copied to an S7-1500, they are not supported by the CPU.
  - Instance data blocks: If there is a function block in the target CPU with the name that was assigned to the instance data block in the source CPU, the instance data block is assigned to the function block in the target CPU. If you copy the instance data block together with the function block to which it was assigned in the source CPU into the target CPU, the instance data block is assigned to the copy of the function block.
- Copying to S7-300/400 CPUs:
  - Organization blocks can be copied as required between S7-300 and S7-400.
  - Although organization blocks from S7-1200/1500 CPUs can be copied to S7-300/400 CPUs, they are not supported by the target CPU.
  - Function blocks (FBs), functions (FCs) and global data blocks (DBs) can be copied as required between S7-300 and S7-400.

- Although function blocks (FBs), functions (FCs) and global data blocks (DBs) can be copied from S7-1200/1500 CPUs to S7-300/400 CPUs, they are not supported by the target CPU.
- Instance data blocks: If there is a function block in the target CPU with the name that was assigned to the instance data block in the source CPU, the instance data block is assigned to the function block in the target CPU. If you copy the instance data block together with the function block to which it was assigned in the source CPU into the target CPU, the instance data block is assigned to the copy of the function block.

In the project tree, blocks that are not supported are indicated by the no parking symbol. Blocks with a no parking symbol cannot be edited, but only used again as copy source.

Note

When blocks are copied between different device families, it is possible that the copied block needs to be recompiled. This applies also to the coping of blocks between CPUs and software controllers. If the block has know-how protection, re-compilation is only possible with the correct password.

Copying data

With paste, all the block data is copied and forwarded to the copy. This data includes:

- Block interface tags
- All networks
- Comments in all existing compilations
- Messages defined in the block
- The entire program code of the copied block including the call instructions contained in the block.
  However, called blocks and their associated instance data blocks are not copied.

Avoiding name conflicts during pasting

When pasting copied blocks with identical names to already existing blocks, the following mechanisms are used to avoid name conflicts:

- Pasting the copied block into the same CPU:
  The copy of the block gets a name that is extended by a number. For example, if block "A" is copied, a possible name for the copy is "A_1". Consecutive numbering is not used, but rather the smallest free number. The copy of block "A" can also get the name "A_25", if no lower number is available.
- Pasting the copied block into another CPU:
  A dialog box opens in which you can select whether the block with the same name will be replaced or the copied block will be pasted with a duplicate designation (Name_Number).

Note

Number conflicts

Number conflicts may occur, if the pasted block has the same block number as an existing block. The block number is not automatically changed during pasting. This double number may have an effect on block calls. When you copy blocks you should therefore check the block number carefully and correct duplicate block numbers manually or using the block properties. Duplicate block numbers lead to a compilation error.

See also

Copying blocks (Page 1514)

Pasting blocks (Page 1514)

Copying blocks

Requirement

The "Program blocks" folder is opened in the project tree.

Procedure

To copy a block, follow these steps:

1. Right-click the block that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

A copy of the block is now on the clipboard and can be pasted either into the same CPU or into another one.

See also

Basics of copying and pasting blocks (Page 1511)

Pasting blocks (Page 1514)

Pasting blocks

Requirement

You have copied a block.

Procedure

To paste a copied block and its data into a CPU, follow these steps:

1. In the project tree, open the folder structure for the CPU into which you want to paste the copied block.

   Note

   Please note that you can only paste the copied block into a CPU which supports the block programming language and type.

2. Right-click on the "Program blocks" folder.
3. Select "Paste" in the shortcut menu.
   - If you are pasting the block into the same CPU as the original block, "_<consecutive number>" will be appended to the name of the copy.
   - If you are pasting the block into a different CPU where a block of the same name already exists, the "Paste" dialog box opens. Select the required option and confirm with "OK".

See also

Basics of copying and pasting blocks (Page 1511)

Copying blocks (Page 1514)

Entering a block title

The block title is the title of the block. It is not the same thing as the block name, which was assigned when the block was created. The length of the block title is restricted to one row. You can enter the block title for open and closed blocks.

Requirement

A code block is available.

Enter block title for open block

To insert the block title in an open block, follow these steps:

1. Click on the title bar of the block in the program editor.
2. Enter the block title.

Enter block title for closed blocks

To insert the block title in a closed block, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.

3. Select the entry "Information" in the area navigation.
4. Enter the block title in the "Title" input field.
5. Confirm your entry with "OK".

See also

Creating organization blocks (Page 1506)

Creating functions and function blocks (Page 1507)

Entering a block comment (Page 1516)

Entering a block comment

You can use the block comment to document the entire code block. For example, you can indicate the purpose of the block or draw attention to special characteristics. You can enter the block comment for open and closed blocks.

Requirement

A code block is available.

Enter block comment for open blocks

To insert a block comment in an open block, proceed as follows:

1. Click the small arrow in front of the block title.
   The right arrow becomes a down arrow, and the comment area is displayed.
2. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
3. Enter the block comment.

Enter block comments for closed blocks

To insert the block comment in a closed block, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.
3. Select the entry "Information" in the area navigation.
4. Enter the block comment in the "Comment" input field.
5. Confirm your entry with "OK".

See also

Creating organization blocks (Page 1506)

Creating functions and function blocks (Page 1507)

Entering a block title (Page 1515)

11.1.3.2 Specifying block properties

Basics of block properties

Block properties

Each block has certain properties, which you can display and edit. These properties are used amongst other things to:

- Identify the block
- Display the memory requirements and the compilation status of the block
- Display the time stamp
- Display the reference information
- Specify the access protection

See also

Overview of block properties (Page 1518)

Block time stamps (Page 1520)

Displaying and editing block properties (Page 1523)

Setting the mnemonics (Page 1539)

Overview of block properties

Overview

The properties of the blocks are block and CPU-specific. Not all properties are therefore available for all blocks or in all CPU families. The following table gives an overview of block properties:

Group	Property	Description
General	Name	Unique block name within the station
	Constant name	Name of the constant pasted for the OB in the PLC tag table
	Type	Block type (cannot be changed)
	Number	Block number
	Event class	Event class of an OB (cannot be changed)
	Language	Programming language of the block
	Language in networks	Language used to program conditions in GRAPH blocks.
	Version	Version of the GRAPH block
	Process image partition number	Display of the process image partitions that are assigned to the organization block (cannot be changed)
	ARRAY data type	Data type of an ARRAY data block (cannot be changed)
	ARRAY limit	High limit of an ARRAY data block The "Move operations" section of the "Instructions" task card offers options for addressing of ARRAY data blocks.
Information	Title	Block title
	Comment	Block comment
	Version	Version number of the block
	Family	Block family name
	Author	Name of the author, company name, department name, or other names
	User-defined ID	ID created by the user
Time stamp	Block	Times of creation and time of change of the block (cannot be changed)
	Interface	Time of creation of the block interface (cannot be changed)
	Code	Code modification time (non-editable)
	Data	Data modification time (non-editable)
	Load-relevant	Time of the last load-relevant modification (cannot be changed)
Compilation	Status	Details of the last compilation run (cannot be changed)
	Lengths	Details of the block lengths (cannot be changed)
Protection	Protection	Setting up know-how protection and copy protection for the block See also: Protecting blocks

Group	Property	Description
Attributes	Optimized block access	The tag declaration for blocks with optimized access contains only the symbolic names of the data elements. The system automatically optimizes and manages the addresses. The CPU performance increases and access errors, for example, from SIMATIC HMI, are prevented. See also: Auto-Hotspot
	IEC check	The compatibility of the operands in comparison operations and arithmetic operations are tested according to IEC 61131. You have to explicitly convert non-compatible operands. See also: Overview of data type conversion (Page 2091)
	Local error handling within block	Troubleshooting inside the block with the GET_ERROR or GET_ERR_ID instruction (cannot be changed). See also: Auto-Hotspot
	Create extended status information	Allows you to monitor all tags in an SCL block. This option does, however, increase the program memory space required and the execution times.
	Check ARRAY limits	Checks whether array indexes are within the range declared for an ARRAY during the runtime of an SCL block. The block enable output ENO is set to "0" if an array index is outside of the permitted range.
	Set ENO automatically	Checks whether errors occur in the processing of certain instructions during the runtime of an SCL block. The block enable output ENO is set to "0" if a runtime error occurs.
	Create minimized DB	Creates instance data blocks for GRAPH blocks of the S7-300 and S7-400 in minimized format. This option reduces the GRAPH FB memory space required, however you will only receive limited program status information.
	Skip steps	If the transitions before and after a step become valid simultaneously in a GRAPH block, the step will not be activated and will be skipped.
	Acknowledge supervision alarms	Any supervision error which occurs during the operation of a GRAPH block must be acknowledged before the program can continue.
	Permanent processing of all interlocks in manual operation	Permanently monitors all interlock conditions in a GRAPH block in manual mode.
	Lock operating mode selection	Prevents a GRAPH block operating mode being selected.
	Data block write-protected in the device	Indicates whether the data block is read-only in the target system, and cannot be overwritten while the program is running (for data blocks only)
	Only store in load memory	On activation the data block is stored only in the load memory, occupies no space in the work memory, and is not linked into the program. The "Move operations" section of the "Instructions" task card offers options for transferring data blocks to the work memory (for data blocks only).
	Start information	Here you specify the structure of the start information of the OB for S7-1500 CPUs: either like for S7-300 and S7-400 CPUs or optimized start information.
	Priority	Shows the priority set for organization blocks. The CPU family used and the type of the organization block determine whether you can change the priority.

Group	Property	Description
	Parameter assignment by means of registers	Enables parameter input by means of registers in an STL block of S7-1500. This feature allows you to use the "Conditional call of blocks" (CC) and "Unconditional call of blocks" (UC) instructions in the block.
	Block can be used as know-how protected library item	Shows whether or not the block in the library can be used with know-how protection.
	Enable readback	Enables you to mark individual parameters of the block for readback. The "Tag readback" function is relevant if the block is used in a CFC chart.
	Block representation	Specifies how the block is represented in a CFC chart.
Time-of-day interrupt	Time-of-day interrupt	Parameters of the time-of-day interrupt OB: active (yes or no), execution, start date and time, local time or system time
Cyclic interrupt	Cyclic interrupt	Cycle time and phase shift of the cyclic interrupt OB
Triggers	Triggers	Display of start events of the hardware interrupt OB
Isochronous mode	Isochronous mode	Parameters of the isochronous mode interrupt OB: application cycles, automatic setting (yes or no), delay time. Also indicated is the PROFINET IO system or DP master system whose IO devices or DP slaves are assigned to the isochronous mode interrupt OB.
Download without reinitialization	Reserve in volatile memory	Defines the reserve in volatile memory that can be used for interface extensions. The number of currently available bytes is displayed in parentheses. This information is updated with each compilation.
	Activate download without reinitialization for retentive tags	Enables definition of a reserve in retentive memory.
	Reserve in retentive memory	Defines the reserve in retentive memory that can be used for interface extensions. The number of currently available bytes is displayed in parentheses. This information is updated with each compilation.

See also

Basics of block properties (Page 1517)

Block time stamps (Page 1520)

Displaying and editing block properties (Page 1523)

Basics of block access (Page 1419)

Block time stamps

Introduction

Blocks receive a number of different time stamps which show you when the block was created and when it was last changed. These time stamps are also used for automatic consistency checks before compilation.

Code block time stamps

The following time stamps are generated for code blocks (OBs, FBs, FCs):

- Block: Created on, Modified on
- Interface: Modified on
- Code/data: Modified on
- Load-relevant: Modified on

A time stamp conflict is displayed during compilation if the time stamp for the block calling is older than that of the interface for the block called.

Time stamps for code blocks are updated as follows:

- Block: The time stamp for the last block modification is always the same as the time stamp either of the interface or of the code depending on which area was modified last.
- Interface: The interface time stamp is updated each time the interface is modified. Even if you manually undo a change to the interface, for example change the name back, that is also a change which updates the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.
- Code/data: The code time stamp is updated each time the block code is changed. Even if you manually undo a change to the code, for example remove an instruction, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.
- Load-relevant: The time stamp for "Load-relevant" is updated each time a code block is changed. These changes can affect the code, the data or the interface.

Global data block time stamps

The following time stamps are generated for global data blocks:

- Block: Created on, Modified on
- Interface: Modified on
- Data: Modified on
- Load-relevant: Modified on

Time stamp conflict is reported during compilation of a global data block based on a PLC data type if the time stamp of the global data block is older than the time stamp of the PLC data type used.

The time stamps for global data blocks are updated as follows:

- Block: The time stamp for the last change to a global data block is always the same as the time stamp for the interface and the data.
- Interface and data: The interface time stamps and the data are updated each time the global data block is changed. Even if you manually undo a change, for example remove a tag, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamps will be reset to the value they had before the undone change.
- Load-relevant: The time stamp for "Load-relevant" is updated each time a global data block is changed. These changes can affect the data or the interface.

Instance data block time stamps

The following time stamps are generated for instance data blocks:

- Block: Created on, Modified on
- Interface: Modified on
- Data: Modified on
- Load-relevant: Modified on

A time stamp conflict will be reported during compilation of an instance data block if the interface time stamps of the instance data block are not identical to those of the function block.

The time stamps for instance data blocks are updated as follows:

- Block: The time stamp for the last change to an instance data block is always the same as the time stamp for the interface and the data.
- Interface and data: The interface time stamps and the data are updated each time the instance data block is changed. Even if you manually undo a change, for example cancel the tag retain setting, that is also a change which will update the time stamps. However, if you undo the change using the "Undo" function, the time stamps will be reset to the value they had before the undone change.
- Load-relevant: The time stamp for "Load-relevant" is updated each time an instance data block is changed. These changes can affect the data or the interface.

PLC data type time stamps

The following time stamps are generated for PLC data types:

- Block: Created on, Modified on
- Interface: Modified on
- Load-relevant: Modified on

The time stamps for PLC data types are updated as follows:

- Block: The time stamp for the last change to a PLC data type is always the same as the interface time stamp.
- Interface: The interface time stamp is updated each time the PLC data type is changed. Even if you manually undo a change, for example delete the content of a PLC data type, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.
- Load-relevant: The time stamp for "Load-relevant" is updated each time a PLC data type is changed.

See also

Basics of block properties (Page 1517)

Overview of block properties (Page 1518)

Displaying and editing block properties (Page 1523)

Basic information on compiling blocks (Page 1782)

Displaying and editing block properties

The properties of the blocks are block and CPU-specific. Not all properties are therefore available for all blocks or in all CPU families. Properties that can only be displayed are write-protected.

Displaying and editing properties of a closed block

To display and edit the properties of a closed block, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Right-click the block whose properties you want to display or edit.
3. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the block opens.
4. In the area navigation, click a group whose properties you want to display or edit.
5. Change the relevant property.
6. Confirm your entries with "OK".

Displaying and editing properties of an open block

To display or edit the properties of an open block, follow these steps:

1. Select the "Inspector window" check box in the "View" menu.
   The Inspector window opens.
2. Click the "Properties" tab.
   The properties of the block are shown in the "Properties" tab of the Inspector window.
3. In the area navigation, click a group whose properties you want to display or edit.
4. Change the relevant property.

Result

The properties of the block will be changed. The changes are not saved until the project is saved.

See also

Basics of block properties (Page 1517)

Overview of block properties (Page 1518)

Block time stamps (Page 1520)

11.1.3.3 Managing blocks

Opening blocks

You can open both the blocks in the project (offline blocks) and also the blocks in the device (online blocks).

To open an offline block, you have the following options:

- Open offline block directly
  You can open a block directly if the corresponding block folder is open in the project tree or in the overview window.
- Find and open offline block
  You can look for blocks within a project, a device or the "Program blocks" folder and then open them.

Note the following special features when opening online blocks:

- You cannot edit online blocks.
- It is possible that another user can carry out a loading process on the selected CPU through joint parallel working on a CPU. Consequently, it can happen that the online block that you have open is deleted by the loading process if the block only exists in the device. In this case, the online block is closed and a message is displayed in the Inspector window.

Open offline block directly

To open an offline block directly, you have the following options:

1. Open the folder with the block you wish to open in the project tree or in the overview window.
2. Double-click on the block you wish to open.
   The block will open in the program editor.

Find and open offline block

To find an offline block and then open it, follow these steps:

1. Right-click on the project, a device, the "Program blocks" folder or a lower-level folder of "Program blocks" in the project tree.
2. In the shortcut menu select the "Open block/PLC data type" command or press the <F7> key.
   The "Open block/PLC data type" dialog box opens.
3. Enter the name, the address or the type of block you are looking for.
   The block list is filtered further with each letter entered. The block list is closed if there is no block that matches the input. You can show the complete block list at any time by clicking the button to the right of the text box. Keep in mind that there is no filtering in this case. If you want to filter for your inputs again, click the button once again.
4. In the block list, click the block you wish to open.
   The block is opened in the program editor and displayed as selection in the project tree.

Open online block

To open an online block directly, you have the following options:

1. Open the "Online access" folder in the project tree.
2. Click on the arrow symbol to the left of the interface to show all the objects that are arranged below the interface.
3. Double-click on the "Update accessible devices" command below the interface.
   All devices that are accessible via this interface are displayed.
4. Open the folder of the device that contains the block you want to open.
5. Open the "Program blocks" folder.
   All the device's blocks are displayed.
6. Double-click on the block you wish to open.
   The block will open in the program editor.

See also

Saving blocks (Page 1525)

Closing blocks (Page 1526)

Renaming blocks (Page 1526)

Deleting blocks offline (Page 1527)

Deleting blocks online (Page 1527)

Opening know-how protected blocks (Page 1806)

Saving blocks

Blocks are always saved together with the project. Faulty blocks can also be saved. This allows the error to be resolved at a convenient time.

Procedure

To save a block, follow these steps:

1. Select the "Save" or "Save as" command in the "Project" menu.
   See also: Saving projects (Page 383)

See also

Opening blocks (Page 1524)

Closing blocks (Page 1526)

Renaming blocks (Page 1526)

Deleting blocks offline (Page 1527)

Deleting blocks online (Page 1527)

Closing blocks

Procedure

To close a block, follow these steps:

1. Click the "Close" button in the title bar of the program editor.

Note

Note that the block will not be saved on closing.

See also

Opening blocks (Page 1524)
Saving blocks (Page 1525)
Renaming blocks (Page 1526)
Deleting blocks offline (Page 1527)
Deleting blocks online (Page 1527)

Renaming blocks

Requirement

The "Program blocks" folder is opened in the project tree.

Procedure

To change the name of a block, follow these steps:

1. Right-click the block that you want to rename.
2. Select the "Rename" command in the shortcut menu.
   The block name in the project tree changes to an input field.
3. Input the new name for the block.
4. Confirm your entry with the Enter key.

Result

The name of the block is now changed at all points of use in the program.

See also

Opening blocks (Page 1524)

Saving blocks (Page 1525)

Closing blocks (Page 1526)

Deleting blocks offline (Page 1527)

Deleting blocks online (Page 1527)

Deleting blocks offline

Requirement

The "Program blocks" folder opens in the project tree.

Procedure

To delete a block that exists offline, proceed as follows:

1. In the project tree in the "Program blocks" folder, right-click on the block that you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. Confirm the safety prompt with "Yes".
   The block is deleted offline from the project.

Note

If you are deleting organization blocks, note that events may be assigned to these blocks. If you delete such organization block the program cannot respond to parameterized events.

See also

Opening blocks (Page 1524)

Saving blocks (Page 1525)

Closing blocks (Page 1526)

Renaming blocks (Page 1526)

Deleting blocks online (Page 1527)

Deleting blocks online

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To delete a block that exists online, follow these steps:

1. In the "Program blocks" folder in the project tree, right-click on the block that you wish to delete from the device.
2. Select the "Delete" command from the shortcut menu.
   The "Delete" dialog opens.
3. Select the "Delete from device" check box.
4. Click "Yes".
   The block will be deleted from the device online.

See also

Opening blocks (Page 1524)

Saving blocks (Page 1525)

Closing blocks (Page 1526)

Renaming blocks (Page 1526)

Deleting blocks offline (Page 1527)

Deleting CPU data blocks

You may delete CPU data blocks both in offline and online mode.

Deleting CPU data blocks in offline mode

Proceed as follows to delete a CPU data block from the offline project:

1. Right-click the CPU data blocks that you want to delete in the project navigation, "Program blocks" folder.
2. Select the "Delete" command from the shortcut menu.
3. Confirm the safety prompt with "Yes".
   The CPU data block is deleted from the offline project.

Deleting CPU data blocks in online mode

Proceed as follows to delete a CPU data block from the online project:

1. Establish an online connection to the device containing the CPU data block that you want to delete.
2. Right-click the CPU data block that you want to delete from the device in the project navigation, "Program blocks" folder.
3. Select the "Delete" command from the shortcut menu.
   The "Delete" dialog opens.

4. Select the "Delete from device" check box.
5. Click "Yes".
   The CPU data block is deleted from the online device.

See also

CPU data blocks (Page 1418)

11.1.4 Programming blocks

11.1.4.1 Program editor

Overview of the program editor

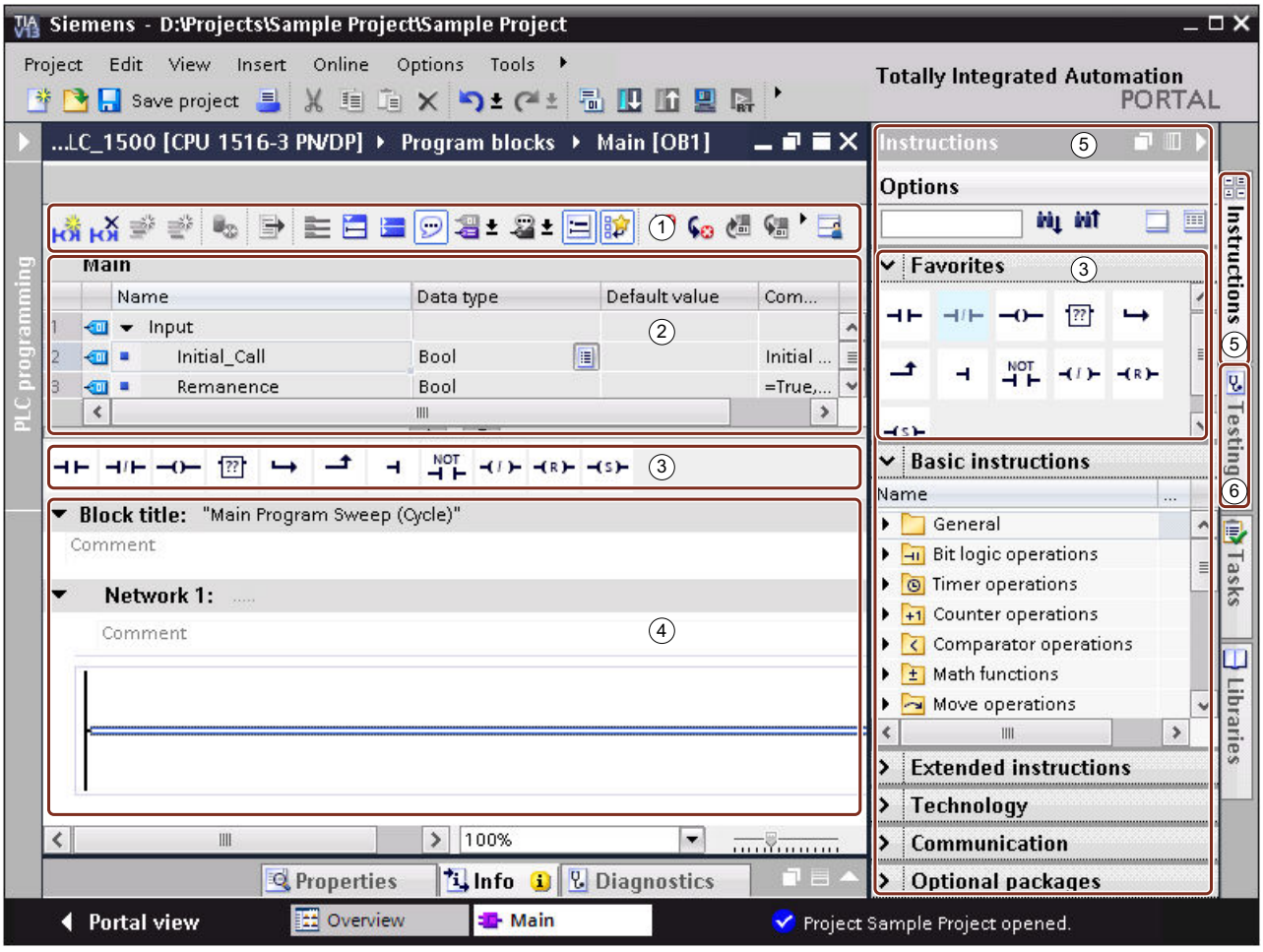
Function of the program editor

The program editor is the integrated development environment for programming functions, function blocks, and organization blocks. If offers comprehensive support for programming and troubleshooting.

The appearance and functionality of the program editor can vary depending on the CPU, programming language and block type used.

Structure of the program editor

Using LAD as an example, the following figure shows the components of the program editor:



① Toolbar
② Block interface
③ "Favorites" pane in the "Instructions" task card and favorites in the program editor
④ Programming window
⑤ "Instructions" task card
⑥ "Testing" task card

Toolbar

The toolbar allows you to access the principal functions of the program editor, such as:

- Show and hide absolute operands
- Showing and hiding favorites
- Skip to syntax errors

- Update block calls
- Show and hide program status

The functions available in the toolbar can vary depending on the programming language used.

Block interface

The block interface contains the declarations for local tags used solely within the block. The sections available depend on the block type.

Favorites

You can save frequently used instructions as favorites. These favorites are then displayed in the "Instructions" task card and the "Favorites" pane. You can also display favorites in the program editor using the program editor toolbar. This allows you to access your favorites even when the "Instructions" task card is not visible.

Programming window

The programming window is the work area of the program editor. You can enter the program code in this window. The appearance and functionality of the program window can vary depending on the programming language used.

"Instructions" task card

The "Instructions" task card offers you easy access to all instructions available for creating your program. The instructions are broken down by area into a number of different palettes. You can search the palettes for specific instructions.
See also: Find instructions (Page 1540)

You can show additional information on the instructions via the "Show column headers and additional columns" button in the task card toolbar. You can modify the arrangement of columns by clicking a column header and moving the column to the new position by means of drag-and-drop.

If an instruction profile is active, the offered instructions may vary.
See also: Using instruction profiles

"Testing" task card

You can set settings in the "Testing" task card for troubleshooting using the program status. The functions of the "Testing" task card are only available in online mode. It contains the following panes, which are displayed depending on the selected CPU and the configured programming language of the block:

- CPU operator panel
  The CPU operator panel allows you to switch the CPU operating mode.
- Breakpoints
  You can test blocks you created in the textual programming languages STL and SCL in single step mode. To do this, set breakpoints in the program code.
  In the "Breakpoints" pane, you can find all breakpoints that you have set and you can activate, delete, navigate to, or set the call environment for the individual breakpoints.
- PLC register
  This pane allows you to read off the values for the PLC registers and the accumulators.
- Sequence control
  In this pane you can set the operating mode for testing sequencers for GRAPH blocks.
- Test settings
  This pane allows you to specify the test settings for the GRAPH block.
- Call environment
  This pane allows you to specify the call environment for the block.
- Call hierarchy
  In this pane, you can trace the call hierarchy of the blocks. You only see the call hierarchy during block monitoring.

See also

Layout of the block interface (Page 1551)

Enlarging the programming window (Page 1537)

Using the keyboard in the program editor

Navigating in the editor

Function	Shortcut key
Open "Instructions" task card	<Ctrl+Shift+C>
Open "Testing" task card	<Ctrl+Shift+O>
Add new block	<Ctrl+N>
Expand all networks	<Alt+F11>
Collapse all networks	<Alt+F12>
Navigate to the next point of use of the selected block or operand	<Ctrl+Shift+F>
Navigate to the previous point of use of the selected block or operand	<Ctrl+Shift+G>

Function	Shortcut key
Navigate to the next read/write access	<Alt+F8>
Navigate to the previous read/write access	<Alt+F9>

Navigating in the program code (LAD/FBD)

Function	Selected object	Shortcut key
Navigating between objects in the network	Object in the network	Arrow keys
Navigate to the first element of the network	Object in the network	<Home>
Navigate to the last element of the network	Object in the network	<End>
Navigate to the next element of the network	Object in the network	<Tab>
Navigate to the previous element of the network	Object in the network	<Shift+Tab>
Insert network	Any	<Ctrl+R>

Navigating in the program code (STL/SCL)

Function	Position of the cursor	Shortcut key
Navigating in the program code	Line	Arrow keys
One word to the right/left	Line	<Ctrl+arrow keys>
Cursor to start of line	Line	<Home>
Cursor to end of line	Line	<End>
Cursor to start of code section	Line	<Ctrl+Home>
Cursor to end of code section	Line	<Ctrl+End>
To the next network (STL only)	Network title	<Down arrow>
To the next network (STL only)	Line	<Tab> Repeat the shortcut keys until the insertion point is in the next network.
To the previous network (STL only)	Network title	<Up arrow>
To the previous network (STL only)	Line	<Shift+Tab> Repeat the shortcut keys until the insertion point is in the previous network.
Insert network	Any	<Ctrl+R>

Inserting instructions (LAD)

Function	Selected object	Shortcut key
Insert normally open contact	Rung	<Shift+F2>
Insert normally closed contact	Rung	<Shift+F3>
Insert empty box	Rung	<Shift+F5>
Insert assignment	Rung	<Shift+F7>
"Insert "Open branch"	Rung	<Shift+F8>
"Insert "Close branch"	Rung	<Shift+F9>

Inserting instructions (FBD)

Function	Selected object	Shortcut key
Insert assignment	Network, input or output	<Shift+F7>
Insert empty box	Network	<Shift+F5>
"Insert "Open branch"	Connection line between two boxes	<Shift+F8>
Invert RLO	Network, input or output	<Ctrl+Shift+4>
Insert input	Network, input or output	<Ctrl+Shift+3>

Entering operands (LAD/FBD/GRAPH)

Function	Selected object	Shortcut key
Activate the input field for the first operand of the instruction	Instruction	<Return> Or <Any letters/numbers> An empty input field opens on <Return>, any letters or numbers will be entered in the input field.
Activate input field for the operand	Operand	<F2>
Delete operand	Operand	<Del>
Define tag	Operand	<Ctrl+Shift+I>
Rewire tag	Operand	<Ctrl+Shift+P>
Rename tag	Operand	<Ctrl+Shift+T>
Entering operands	Input field for operands	<Any letters/numbers>
Confirm entry of the operand	Input field for operands	<Return>
Open autocompletion	Input field for operands	<Ctrl+I>
Discard current change	Input field for operands	<Esc> The input field is deactivated and the previous contents restored.

Process instructions (STL/SCL)

Function	Selected object	Shortcut key
Indent line (SCL only)	Line	<Tab> or <Ctrl+R>
Outdent line (SCL only)	Line	<Shift+Tab> or <Ctrl+Shift+R>

Function	Selected object	Shortcut key
Automatically formatting selected text (SCL only)		<Ctrl+Shift+W>
Open "Call options" dialog	Cursor after block call	<Return>
Define tag	Operand	<Ctrl+Shift+I>
Rewire tag	Operand	<CtrlShift+P>
Rename tag	Operand	<Ctrl+Shift+T>
Expand/collapse parameter list (SCL only)	Operand	<Ctrl+Shift+Space>
Expand/collapse code section	Insertion mark within the code section	<Ctrl+Shift+Num+> <Ctrl+Shift+Num->
Expand/collapse all code sections	Any	<Ctrl+Shift+Num*> <Ctrl+Shift+Num/>
Open autocompletion	Any	<Ctrl+I> or <Ctrl+Spacebar>
Set/delete bookmarks		<Ctrl+Shift+M>
To the next bookmark		<Ctrl+Shift+6>
To the previous bookmark		<Ctrl+Shift+5>
Comment out selection	Line	<Ctrl+Shift+Y>
Remove comment	Line	<Ctrl+Shift+U>

Programming window of GRAPH

Function	Area	Shortcut key
Page Up/Down	Navigation view, single step view, sequence view, permanent instructions	<Page Up>/ <Page Down>
Navigate in the navigation view	Navigation view	<Up arrow> <Down arrow>
Expand/collapse object	Navigation view	<+> or <Right arrow> <-> or <Left arrow>
Switch between single step view and sequence view when a step or a transition is selected	Navigation view	<Return>
Switch between navigation view and work area	Navigation view, single step view, sequence view, permanent instructions	<ALT+F6>
Go to first element in a network	Single step view	<Home>
Go to last element in a network	Single step view	<End>
Switch to interlock	Single step view	<Ctrl+Home>
Switch to transition	Single step view	<Ctrl+End>
Navigate in the structure	Sequence view	Arrow keys
Go to first step	Sequence view	<Home> or <Ctrl+Home>
Go to last step	Sequence view	<End> or <Ctrl+End>
Open branch	Sequence view	<Shift+F8>

Function	Area	Shortcut key
Close branch	Sequence view	<Shift+F9>
Insert sequence end	Sequence view	<Shift+F7>
Insert jump	Sequence view	<Shift+F12>
Insert step and transition	Sequence view	<Shift+F5>
Delete element	Sequence view	<Del>
Go to first editable element	Permanent instructions	<Home>
Go to next editable element	Permanent instructions	<Tab>
Go to last editable element	Permanent instructions	<End>
Go to previous editable element	Permanent instructions	<Shift+Tab>
Jump to the start of the "Action" cell	Actions	<Home>
Jump to the end of the "Action" cell	Actions	<End>
Insert new action	Actions	<Return>

Monitor program

Function	Shortcut key
Set/remove breakpoint (STL, SCL)	<Ctrl+Shift+F9>
Step over breakpoint (STL, SCL)	<Ctrl+Shift+F10>
Jump to lower-level block (STL, SCL)	<Ctrl+Shift+F11>
Return to calling block (STL, SCL)	<Ctrl+Shift+F12>
Run program up to marker (cursor position) (STL, SCL)	<Ctrl+F3>
Display program status (SCL, STL)	<Ctrl+T>
Enable all breakpoints (STL, SCL)	<Ctrl+Shift+F2>
Disable all breakpoints (STL, SCL)	<Ctrl+Shift+F3>
Modify to 0 (LAD, FBD)	<Ctrl+Shift+9>
Modify to 1 (LAD, FBD)	<Ctrl+Shift+1>
Modify operand (LAD, FBD)	<Ctrl+Shift+2>

See also

Keyboard operation in the TIA Portal (Page 339)

Using project-related functions (Page 341)

Arranging windows (Page 341)

Editing tables (Page 347)

Text editing (Page 346)

Enlarging the programming window

Introduction

The programming window is relatively small when all components of the application are shown. If the program code is large, you may find you have to rearrange the work area constantly. To avoid this problem, you can hide or minimize the display of the following components of the application and of the program editor:

- Project tree
- Task cards
- Block interface
- Favorites
- Comments
- Networks

Note

You can also use the "Reduce automatically" option for the task cards, project tree, and Inspector window. These windows will then be minimized automatically when you do not need them.

See also: Maximizing and minimizing the work area

Hiding and showing the project tree

The project tree allows you to access all areas of the project. You can hide the project tree while you are creating a program so you have more space for the programming window.

To show and hide the project tree, follow these steps:

1. To hide the project tree, deselect the "Project tree" check box in the "View" menu, or click on "Collapse" on the project tree title bar.
2. To show the project tree, select the "Project tree" check box in the "View" menu or click on "Extend" on the project tree title bar.

Opening and closing task cards

The task cards are located at the right-hand edge of the programming window.

To open or close the task cards, follow these steps:

1. To close the task cards, deselect the "Task card" check box in the "View" menu or click "Collapse" on the task cards title bar.
2. To open the task cards, select the "Task card" check box in the "View" menu or click "Expand" on the task cards title bar.

Hiding and showing the block interface

The block interface is shown in the upper section of the program editor. During programming you can show and hide it as required.

To show and hide the block interface, follow these steps:

1. In the lower part of the interface within the window splitter, click on the Up arrow or Down arrow.

Showing and hiding favorites

To hide or show the favorites in the program editor, follow these steps:

1. Click the "Display favorites in the editor" button in the program editor toolbar.

Showing and hiding comments

Within a block you can enter a comment for the block or for each network. These two types of comments are shown and hidden differently.

To show or hide a block comment, follow these steps:

1. Click the the triangle at the start of the line with the block title.

To show or hide network comments, follow these steps:

1. Click "Network comments on/off" on the program editor toolbar.

Note

The comments available can vary depending on the programming language used.

Opening and closing networks

Some programming languages use networks. You can open or close these networks as required.

To open or close networks, follow these steps:

1. If you want to open a network, click the right arrow in front of the network title. If you want to close a network, click the down arrow in front of the network title.

To open and close all networks, follow these steps:

1. Click "Open all networks" or "Close all networks" in the program editor toolbar.

Note

Networks are not used in every programming language.

See also

Overview of the program editor (Page 1529)

Maximizing and minimizing the work area (Page 318)

Setting the mnemonics

You can program blocks using German or international mnemonics. When you open the TIA Portal for the first time, international mnemonics is set as default. You can change the mnemonics at any time.

Procedure

To set the mnemonics, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation.
3. In the "General settings" group, select the mnemonics that you want to use.
   The mnemonics is changed in all blocks.

Displaying symbolic and absolute addresses

You have the following options for displaying operands in the program editor:

- Symbolic representation
  The symbolic operands are displayed in the program. The corresponding absolute addresses are shown in tooltips if you hold the mouse pointer over the operand.
- Absolute representation
  The absolute addresses are displayed in the program. The corresponding symbolic operands are displayed in tooltips.
- Symbolic and absolute representation
  Symbolic operands and absolute addresses are displayed in program.

Requirement

The program editor is open.

Procedure

To change the representation of the operands, follow these steps:

1. Click the "Absolute/symbolic operands" button in the program editor toolbar.
   Each time you click the button, the representation and the symbol on the button change.

Or:

1. Click the small arrow next to the "Absolute/symbolic operands" button in the program editor toolbar.
   A drop-down list is displayed.
2. Select the required representation from the drop-down list.
   The symbol on the button changes.

See also

Basic information about operands (Page 1441)

Find instructions

You have the option of finding specific instructions in the "Instructions" task card and then inserting them in your program. Note the following rules when entering search terms:

- No distinction is made between upper and lower case text.
- The search function considers parts of a search term.
- You cannot use placeholders, such as "*" and "?".
- If an instruction name includes underscores, the instruction is found even if you do not enter the underscore.
- The text in the columns "Name" and "Description" are taken into consideration in the search.

Requirement

- A block is open.
- Die "Instructions" task card is open.

Procedure

To search for a specific instruction in the "Instructions" task card, follow these steps:

1. Select a starting point for the search if you want to start the search at a specific point. If you do not select a starting point, the search starts either at the top or bottom in the task card, depending on the type of search you have selected.
2. Enter a search term in the text box of the task card toolbar.
3. Click "Search down" if you want to search the task card from top to bottom.
4. Click "Search up" if you want to search the task card from bottom to top.
   The first match with the search term found is displayed as the result. If you want to continue the search, click the "Search down" or "Search up" button again. If no matches are found, you will receive a corresponding message.

See also

Overview of the program editor (Page 1529)

Use instruction versions

Basic information on instruction versions

The instructions available to you for programming the user program are managed in system libraries. If a new version of a system library is installed by an update, the newer versions of the instructions of this system library may also be installed.

If there are several versions for an instruction, these are listed in the "Instructions" task card after the respective instruction. If the instruction versions are not shown, you can show them via the "Show column headers and additional columns" button in the toolbar of the "Instructions" task card. You can then select the versions of the instructions to be used in the program from the dropdown-list box of the "Version" column. If you do not select any versions, the most recent versions are used.

Note

Please note the following:

- You can only ever use the same version of an instruction within a device.
- If you change the version of an instruction that other instructions depend on, the versions of the dependent instructions are also changed.
- If you select a version for an instruction that can not be run on the CPU used, the instruction is shaded out. This means that you cannot use this instruction in this version with your CPU.
- The system block in the project tree displays the block version number in its properties; it does not have to match the instruction version number of the associated instruction in the task card.
- If you change the version of an instruction in the task card, you must compile the associated system block in the project tree before you update its block version number.

Changes in the versions

New versions can be main versions or secondary versions. New versions, such as 2.0 or 3.0, have more substantial changes to them. New main versions may therefore result in changes to the block interface. New secondary versions, such as 1.3 or 1.4, contain lesser changes or remedies to errors.

Using instruction versions

You can decide within a device which version of an instruction you want to use. If you select another version for an instruction, the new version is specified for all locations of use of this instruction within your program. These instructions are identified in the program by a red frame. You must then download your program to the device to use the new instruction version.

Using instruction profiles

Basics of instruction profiles

Introduction

The TIA Portal provides you with numerous instructions that you can use to program the user program. However, you may filter out specific instructions that you do not want to use. To this end you can create instruction profiles in which you can explicitly specify the instructions to be listed in the "Instructions" task card. However, although you may create several instruction profiles in a project, only one of these profiles may be active at any given time. You can exchange instruction profiles with other users by means of shared libraries.

Note

Please note the following:

- The use of instructions that are not allowed in the active profile in a block will trigger the output of a block compilation error. Such a situation may be triggered if you drag-and-drop a block from the library to your program.
- Instructions of a profile that are not supported by the currently installed products are deleted from the profile the next time it is edited. If you transfer this profile to an engineering system in which these instructions are supported by the installed products, the instructions are again present in the profile but they are disabled. You can enable these instructions as required at any time.
- If you want to make changes to the active profile, you must recompile the blocks in the project. This is also necessary when you disable or delete the active profile or when you enable a profile.

See also

Creating new instruction profiles (Page 1542)

Opening and editing instruction profiles (Page 1544)

Activating and deactivating instruction profiles (Page 1545)

Deleting instruction profiles (Page 1546)

Creating new instruction profiles

Requirement

The "Common data > Instruction profiles" folder is open in the project navigation.

Procedure

Proceed as follows to create a new instruction profile:

1. Double-click the "Add new profile" command.
   The Instruction Profile Editor opens and displays the new instruction profile. All instructions are activated for the new instruction profile.
2. Edit the new instruction profile to suit your requirements.

If necessary, you can rename the new instruction profile. To do this, follow these steps:

1. Right-click on the new instruction profile.
2. Select the "Rename" command in the shortcut menu.
3. Enter a name for the new instruction profile.

Note

The first instruction profile that you create will be used as active profile. In this case, compile all blocks in the project. If other instruction profiles are already available you must explicitly activate the new one in order to use it as active profile. You can identify the active profile by its icon in the project navigation.

See also

Basics of instruction profiles (Page 1542)

Opening and editing instruction profiles (Page 1544)

Activating and deactivating instruction profiles (Page 1545)

Deleting instruction profiles (Page 1546)

Opening and editing instruction profiles

Once you have opened an instruction profile, you can edit it as follows:

- Activating and deactivating instructions
  You can explicitly specify the instructions to be allowed in the instruction profile.

  Note

  Note that dependencies exist between some instructions. As a result, it is possible that several instructions may be activated or deactivated by an action. The check box icon indicates the folders in which instructions are deactivated.

- Activating and deactivating instruction versions
  Certain instructions are available with different versions. You can explicitly specify the instruction versions to be allowed in the instruction profile.
- Renumbering blocks
  An instruction representing an internal function block (FB) or function (FC) in the system is assigned a specific block number by the system. You can replace this block number with your own block number. Within a version, there are several implementations for certain instructions. The block numbers in such instructions can only be changed for the specific implementation.

  Note

  If an instruction from the instruction profile is used in the program and the specified block number is already in use by a different block, the specified block number of the instruction will be replaced by a free block number.

Requirement

The "Common data > Instruction profiles" folder is open in the project navigation.

Opening instruction profiles

Proceed as follows to open an instruction profile:

1. Double-click the instruction profile that you want to edit.
   The instruction profile opens in the Instruction Profile Editor.

Editing instruction profiles

Proceed as follows to edit an instruction profile in the Instruction Profile Editor:

1. Select the device that you want to edit from the "Device family" drop-down list box.
2. Select the programming language for which you want to edit the instruction profile from the "Language" drop-down list box.
3. Deactivate the instructions or instruction versions that you want to exclude from the instruction profile. You can deactivate a folder to deactivate all subordinate instructions.

4. Activate the instructions or instruction versions that you want to allow in the instruction profile.
5. You may assign your own block numbers.

Note

You can assign the number up to 65535 for CPUs of the S7-1200/1500 series. For CPUs of the S7300/400 series you find the restrictions of the number ranges in the respective CPU manual.

Note

A new compilation process is required for all blocks in the project when you change the active profile.

See also

Basics of instruction profiles (Page 1542)

Creating new instruction profiles (Page 1542)

Activating and deactivating instruction profiles (Page 1545)

Deleting instruction profiles (Page 1546)

Use instruction versions (Page 1541)

Activating and deactivating instruction profiles

You first need to activate an instruction profile in order to include filtering of its instructions. You can always deactivate the instruction profile to reset the instructions task card to the default scope of instructions.

Note

A new compilation process is required for all blocks in the project.

Requirement

The "Common data > Instruction profiles" folder is open in the project navigation.

Activating instruction profiles

Proceed as follows to activate an instruction profile:

1. Right-click on the instruction profile that you want to activate.
2. Select the "Activate profile" command from the shortcut menu.
   The selected instruction profile is now active. Instructions can now only be used in accordance with the settings of this profile.

Deactivating instruction profiles

Proceed as follows to deactivate the instruction profile:

1. Right-click on the instruction profile that you want to deactivate.
2. Select the "Deactivate profile" command from the shortcut menu.
   No instruction profile is active and the "Instructions" task card once again shows all instructions that are available for use.

See also

Basics of instruction profiles (Page 1542)

Creating new instruction profiles (Page 1542)

Opening and editing instruction profiles (Page 1544)

Deleting instruction profiles (Page 1546)

Deleting instruction profiles

Requirement

The "Common data > Instruction profiles" folder is open in the project tree.

Procedure

Proceed as follows to delete an instruction profile:

1. Right-click on the instruction profile that you want to delete.
2. Select the "Delete" command in the shortcut menu.

Note

A new compilation process is required for all blocks in the project when you delete the active profile.

Result

The selected instruction profile is deleted. If you deleted the active instruction profile, no more active profiles are available and the "Instructions" task card once again shows all instructions that are available for use.

See also

Basics of instruction profiles (Page 1542)

Creating new instruction profiles (Page 1542)

Opening and editing instruction profiles (Page 1544)

Activating and deactivating instruction profiles (Page 1545)

Using autocompletion

Basics of autocompletion

Function

You can use autocompletion in the program window of the program editor as an easy way to access available tags or instructions during programming. Autocompletion means a context-specific list appears in a dialog from which you can select the tags or instructions you need.

See also

Using autocompletion in graphic programming languages (Page 1547)

Using autocompletion in textual programming languages (Page 1548)

Using autocompletion in graphic programming languages

Inserting tags using autocompletion

To insert tags in graphic programming languages using autocompletion, follow these steps:

1. Select an operand of the instruction to which you wish to assign a tag.
   The input field for the operand opens. The autocompletion button will appear beside the input field.
2. Either click the autocompletion button or use the shortcut <Ctrl+l>.
   Autocompletion opens. It contains only the local and global tags, data blocks and multiple instances which are admissible for the operand in the given context. You can exit autocompletion at any time by pressing <Esc>.

3. Select the required tag from the list. If necessary, you can also filter the list:
   - For example, enter the first few letters of the name of the tag or instruction you wish to insert. Autocompletion will be filtered further with each letter entered. If there is no tag or instruction starting with the letters entered, autocompletion will remain at the last match.
   - Enter # to access the local tags from the block interface.
   - Enter " to access the global tags.
   - Enter % to access absolute addresses.

   If the tag is a structured tag, a data block or a multiple instance, then an arrow is displayed at the end of the row. Click on the arrow to display the lower-level elements. You can navigate to the very last level in this way. If a structure is allowed as a data type for the operand, you can choose "None" from the list. This assigns the entire structure to the operand as a tag. Use the <Backspace> key to return to the previous level.
4. Press the <Return> key to apply the tag.

See also

Basics of autocompletion (Page 1547)

Using autocompletion in textual programming languages (Page 1548)

Using autocompletion in textual programming languages

Inserting tags and instructions using autocompletion

To insert tags and instructions in textual programming languages using autocompletion, follow these steps:

1. Enter the first few letters of the name of the tag or instruction you wish to insert. If necessary, you can directly filter the kind of tags:
   - Enter # to access the local tags from the block interface.
   - Enter " to access the global tags.
   - Enter % to access absolute addresses.

   Autocompletion opens. It contains only the local and global tags, data blocks, multiple instances and instructions which are admissible at the current position. You can exit autocompletion at any time by pressing <Esc>.
2. Enter more letters of the name of the tag or instruction you wish to insert. You can use <Enter> or <Tab> to apply the tag or instruction and close autocompletion. Autocompletion will be filtered further with each letter entered. If there is no tag or instruction starting with the letters entered, autocompletion only contains the previous matches.

3. Select the tag or instruction required from the list.
   If a tag is a structured tag, a data block or a multiple instance, first select the tag, the data block or multiple instance from the autocompletion and apply the selection with <Enter>.
   To select the additional components of the structure, data block, or multiple instance, enter a period. Autocompletion then reopens and you can select the next component.
4. Press the <Return> key to apply the tag.

See also

Basics of autocompletion (Page 1547)

Using autocompletion in graphic programming languages (Page 1547)

General settings for the PLC programming

Overview of the general settings

Overview

The following table shows the general settings that you can make:

Group	Setting	Description
View	With comments	Network comments are shown.
	Tag information	Additional information for the tags used is displayed in the program editor. When you select the option "Tag information with hierarchy", the comments of the higher structure levels are also displayed for structured tags.
Compilation	Delete actual parameters on interface update	Actual parameters are deleted if the associated formal parameters were deleted in the called block, and you run the "Update block call" function or compile the block.
Default settings for new blocks	IEC check	The compatibility of operands in comparison operations and arithmetic operations are tested according to IEC rules. You have to explicitly convert non-compatible operands.
Additional settings	Set network title automatically	Sets the title of a network based on the comment of the output parameter of the first writing instruction in the network. See also: Inserting network title (Page 1583)
	Show autocompletion list	The autocompletion list is displayed.
	Mnemonics	German or international designation of operations and operands

Group	Setting	Description
Download without reinitialization	Memory reserve	Defines the size of reserve in volatile memory that can be used for interface extensions.
Block interface / data blocks	Set "HMI accessible" for new elements and ARRAY data blocks	Enables the option "HMI accessible" for new tags in the block interface and in data blocks. The use of this option makes sense especially when working with large amounts of data in ARRAY data blocks. See also: Basic principles for programming of data blocks (Page 1703)

See also

Layout of the block interface (Page 1551)

Changing the settings (Page 1550)

Permissible addresses and data types of PLC tags (Page 1482)

Overview of the print settings (Page 305)

Basics of block access (Page 1419)

Setting and canceling the IEC check (Page 2093)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the general settings (Page 1549)

11.1.4.2 Programming code blocks

Declaring the block interface

Layout of the block interface

Introduction

The interface contains the declarations of local tags and constants that are used within the block. The tags are subdivided into two groups:

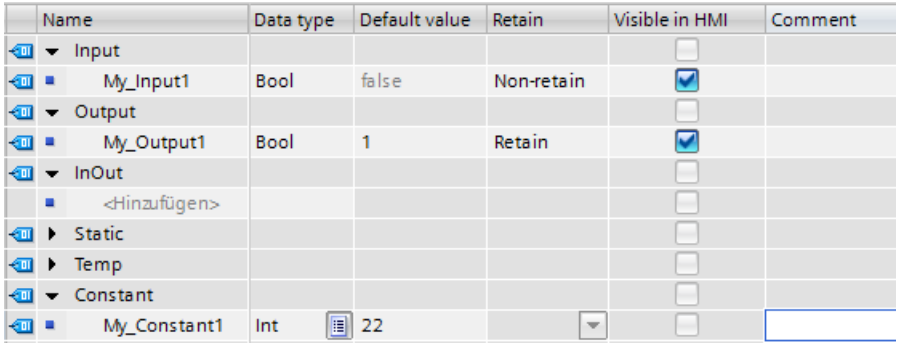
- Block parameters that form the block interface when it is called in the program.
- Local data that are used for storage of intermediate results.

You use the tag declaration to define the call interface of a block in the program and the names and data types of tags and constants that you want to use in the block.

The interface of function blocks also defines the structure of the instances that are assigned to the function block.

Layout of the block interface

The following figure shows the structure of the block interface. The number of columns and sections varies depending on the type of block.



Block parameters

The following table shows the types of block parameters:

Type	Section	Function	Available in
Input parameters	Input	Parameters whose values are read by the block.	Functions, function blocks and some types of organization blocks
Output parameters	Output	Parameters whose values are written by the block.	Functions and function blocks

Type	Section	Function	Available in
In/out parameters	InOut	Parameters whose values are read by the block when it is called, and whose values are written again by the block after execution.	Functions and function blocks
Return value	Return	Value that is returned to the calling block.	Functions

Depending on the type of block opened, additional sections may be displayed.

Local data

The following table shows the types of local data:

Type	Section	Function	Available in
Temporary local data	Temp	Tags that are used to store temporary intermediate results. Temporary local data are retained for only one cycle. If you use temporary local data, you have to make sure that the values are written within the cycle in which you want to read them. Otherwise the values will be random.	Functions, function blocks and organization blocks **Note**: Temporary local data is not shown in instance data blocks.
Static local data	Static	Tags that are used for storage of static intermediate results in the instance data block. Static data is retained until overwritten, which may be after several cycles. The names of the blocks, which are called in this code block as multiple instance, will also be stored in the static local data.	Function blocks
Constant	Constant	Constants with declared symbolic names that are used within the block.	Functions, function blocks and organization blocks **Note**: Local constants are not shown in instance data blocks.

Meaning of the columns

The following table shows the meaning of the individual columns. You can show or hide the columns as required. The number of columns displayed varies depending on the CPU series and the type of the open object.

Column	Description
	Symbol you can click on to drag-and-drop an element to a program for use as an operand.
Name	Name of the element.
Data type	Data type of the element.
Offset	Relative address of a tag. The column is only visible in blocks with standard access.

Column	Description
Default value	Value with which you can pre-assign specific tags in the interface of the code block, or the value of a local constant. Specification of the default value is optional for tags. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL. The default value of a tag is applied as the start value in the corresponding instance data block. You can replace these values with instance-specific start values in the instance data block. Constants always have the default value declared in the block interface. They are not shown in instance data blocks and cannot be assigned instance-specific values there.
Retentivity	Marks a tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This column is only visible in the interface of the function block with optimized access.
Visible in HMI	Shows whether a tag is visible by default in the HMI selection list.
Accessible from HMI	Shows whether HMI can access this tag during runtime.
Setting value	Marks a tag as a setpoint. Setting values are the values that will probably have to be fine tuned during commissioning. The column is only available in the interface of function blocks.
Comment	Comments on documentation of the element.

See also

Using tags within the program (Page 1447)

Keywords (Page 1442)

Valid data types in the block interface (Page 1555)

Setting the retentivity of local tags (Page 1568)

Basics of constants (Page 1449)

Rules for declaring the block interface

General rules for declaring the block interface

Using POINTER

The following rules apply to the use of block parameters within the block:

- Input parameters may only be read.
- Output parameters may only be written.
- In/out parameters may be read and written.

Assigning default values to block parameters

You can assign default values to specific parameters in the function block interface. The possibility of the assignment depends on the declaration subsection and data type of the parameter.

The following table shows which parameters can be assigned a default value:

Parameter type	Section	Assignment of a default value is possible		
		Elementary data types	Structured data types	Parameter types
Input parameters	Input	X	X	-
Output parameters	Output	X	X	-
In/out parameters	InOut	X	- (1)	-
Static local data	Static	X	X	-
Temporary local data	Temp	-	-	-
Constants	Constant	X	-	-

(1) Exception: In blocks with optimized access, you have the option to use PLC data types as default values under certain conditions.

See also: Declaring PLC data types (UDT) as actual parameters for in-out parameters. (Page 1571)

See also

Using tags within the program (Page 1447)

Keywords (Page 1442)

Valid data types in the block interface

Valid data types in the block interface in S7-300/400

The following table shows which data types you can assign to the parameters in the individual sections of the interface.

Section	Standard Data types	ARRAY STRUCT STRING DT	Parameter types	VOID	POINTER	ANY
Organization block						
Temp	X	X	-	-	-	X
Constant	X	X(3)	-	-	-	-
Function block						
Input	X	X	X	-	X	X
Output	X	X	-	-	-	-
InOut	X	X (1)	-	-	X	X

Section	Standard Data types	ARRAY STRUCT STRING DT	Parameter types	VOID	POINTER	ANY
Static	X	X	-	-	-	-
Temp	X	X	-	-	-	X
Constant	X	X(3)	-	-	-	-
Function						
Input	X	X(1)	X	-	X	X
Output	X	X(1)	-	-	X	X
InOut	X	X(1)	-	-	X	X
Temp	X	X	-	-	-	X
Return	X	X	-	X	X	X(2)
Constant	X	X(3)	-	-	-	-

(1) STRING can only be defined in the standard length of 254 characters.

(2) In SCL, ANY is not permitted as a function value.

(3) Constants with the ARRAY or STRUCT data type are not permitted.

Valid data types in the block interface

Valid data types in the block interface in S7-1200

The following table shows which data types you can assign in the parameters in the individual sections of the interface.

Section	Standard Data types	ARRAY STRUCT STRING / WSTRING DT	VOID	VARIANT
Organization block				
Temp	X	X	-	X
Constant	X	X(2)	-	-
Function block				
Input	X	X	-	X
Output	X	X	-	-
InOut	X	X(1)	-	X
Static	X	X	-	-
Temp	X	X	-	X
Constant	X	X(2)	-	-
Function				
Input	X	X(1)	-	X
Output	X	X(1)	-	X
InOut	X	X(1)	-	X

Section	Standard Data types	ARRAY STRUCT STRING / WSTRING DT	VOID	VARIANT
Temp	X	X	-	X
Return	X	X	X	-
Constant	X	X(2)	-	-

(1) You cannot make length declarations for STRING and WSTRING in these sections. Here, STRINGs always have the standard length 254, WSTRINGs the standard length 16832. A declaration in the format MyString[3] would not be permitted. WSTRING is only permitted in blocks with optimized access in these sections.

(2) Constants with the ARRAY or STRUCT data type are not permitted.

Valid data types in the block interface in S7-1500

The following table shows which data types you can assign in the parameters in the individual sections of the interface.

Section	Standard Data types	ARRAY STRUCT STRING / WSTRING DT	Parame-ter types	VOID	DB_ANY	POINTER	ANY	VARIANT
Organization block								
Temp	X	X	-(4)	-	X	-	X(3)	X
Constant	X	X(5)	-	-	-	-	-	-
Function block								
Input	X	X	X	-	X	X	X	X
Output	X	X	-	-	X	-	-	-
InOut	X	X(1)	-(4)	-	X	X	X	X
Static	X	X	-	-	X	-	-	-
Temp	X	X	-(4)	-	-	-	X(3)	X
Constant	X	X(5)	-	-	-	-	-	-
Function								
Input	X	X(1)	X	-	X	X	X	X
Output	X	X(1)	-	-	X	X	X	X
InOut	X	X(1)	-(4)	-	X	X	X	X
Temp	X	X	-(4)	-	X	-	X(3)	X
Return	X	X	-	X	X	X	x(2)	-

Section	Standard Data types	ARRAY STRUCT STRING / WSTRING DT	Parame-ter types	VOID	DB_ANY	POINTER	ANY	VARIANT
Constant	X	X(5)	-	-	-	-	-	-

(1) You cannot make length declarations for STRING and WSTRING in these sections. Here, STRINGs always have the standard length 254, WSTRINGs the standard length 16832. A declaration in the format MyString[3] would not be permitted. WSTRING is only permitted in blocks with optimized access in these sections.

(2) In SCL, ANY is not permitted as a function value.

(3) ANY can only be used in blocks with standard access in the "Temp" section.

(4) The "INSTANCE" parameter type is the only exception permissible in the "TEMP" and "InOut" sections.

(5) Constants with the ARRAY or STRUCT data type are not permitted.

Declaring local tags and constants

Declaring local tags and constants in the block interface

Requirement

The block interface is open.

Procedure

To declare a tag or constant of the elementary data type, follow these steps:

1. Select the appropriate declaration section in the interface.
2. Enter a name for the element in the "Name" column.
3. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
4. For constants, enter a value in the "Default value" column.
5. Optional: Change the properties that are displayed in the other columns of the block interface.

Result

The element is created.

Syntax check

A syntax check is performed after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. However, you will not be able to compile the program if the tag declaration contains syntax errors.

Note

If you change the interface of a block, the calls of the block in the program will possibly become inconsistent. The call locations are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent blocks have to be updated manually.

See also:

Updating block calls in LAD (Page 1596)

Updating block calls in FBD (Page 1638)

See also

Editing tables (Page 347)

Basic information on start values (Page 1713)

Using tags within the program (Page 1447)

Keywords (Page 1442)

Properties of local tags and constants (Page 1567)

Setting the retentivity of local tags (Page 1568)

Declaring a local tag in the program editor

Requirement

The program editor is open.

Procedure

To declare a local tag, follow these steps:

1. Insert an instruction in your program.
   The "<???>", "<??.?>" or "..." strings represent operand placeholders.
2. Replace an operand placeholder with the name of the tag to be created.
3. Select the name of the element.
   If you want to declare multiple elements, select the names of all the elements to be declared.
4. Select the "Define tag" command in the shortcut menu.
   The "Define tag" dialog box opens. It displays a declaration table in which the name of the element is already entered.

5. To declare a local tag, select one of the following sections:
   - Local In
   - Local Out
   - Local InOut
   - Local Static
   - Local Temp
6. In the other columns, enter data type and comments.
7. Click the "Define" button to complete your entry.

Result

The declaration is written directly into the block interface and is valid within the entire block.

Note

If you change the interface of a block, the calls of the block in the program will possibly become inconsistent. The call locations are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent blocks have to be updated manually.

See also:

Updating block calls in LAD (Page 1596)

Updating block calls in FBD (Page 1638)

See also

Editing tables (Page 347)

Using tags within the program (Page 1447)

Keywords (Page 1442)

Basic information on start values (Page 1713)

Properties of local tags and constants (Page 1567)

Setting the retentivity of local tags (Page 1568)

Declaring tags of the ARRAY data type

Requirement

The block interface is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Select the appropriate declaration section in the interface.
2. Enter a tag name in the "Name" column.
3. In the "Data type" column, click the button for the data type selection.
   A list of the permissible data types is opened.
4. Select the "Array" data type.
   The "Array" dialog opens.
5. In the "Data type" text box, specify the data type of the array elements.
6. In the "ARRAY limits" text box, specify the high and low limit for each dimension.
   Example of a one-dimensional ARRAY:
   `[0..3]`
   Example of a three-dimensional ARRAY:
   `[0..3, 0..15, 0..33]`
7. Confirm your entry.
8. Optional: Change the properties of the tags that are displayed in the other columns of the block interface.

Result

The tag of ARRAY data type is created.

Note

You cannot define specific default values for ARRAY elements. However, you can assign them start values in the instance.

See also

Array (Page 1941)

Using tags within the program (Page 1447)

Keywords (Page 1442)

Properties of local tags and constants (Page 1567)

Setting the retentivity of local tags (Page 1568)

Editing tables (Page 347)

Declaring tags of STRUCT data type

Requirement

The block interface is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Select the appropriate declaration section in the interface.
2. Enter a tag name in the "Name" column.
3. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.
   An empty, indented row is inserted after the new tag.
4. Insert the first structural element in the first empty row.
   An additional empty row is inserted after the element.
5. Select a data type for the structure element.
6. Optional: Change the properties of the structural element that is displayed in the other columns of the block interface.
7. Repeat the step 4 to 7 for all additional structure elements.
   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
8. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

Note

S7-1500: A maximum of 252 structures in one data block

A maximum of 252 structures is permitted in one data block. If you need additional structures, you must restructure your program. You can, for example, create the structures in several global data blocks.

See also

Using tags within the program (Page 1447)

Keywords (Page 1442)

Properties of local tags and constants (Page 1567)

Setting the retentivity of local tags (Page 1568)

Editing tables (Page 347)

Declaring tags based on a PLC data type

Requirement

A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a PLC data type, follow these steps:

1. Select the appropriate declaration section in the interface:
2. Enter the PLC data type in the "Data type" column. You will be supported by Autocomplete during input.

Result

The tag is created.

Note

You define the default values of tags within a PLC data type when the PLC data type is created. You cannot change these values at the point of use of the PLC data type.

If you change or delete PLC data types that are used in the block interface, the interface becomes inconsistent. To remedy this inconsistency, the interface has to be updated.

See also: Updating the block interface (Page 1564)

See also

Editing tables (Page 347)

Basics of PLC data types (Page 1734)

Declaring higher-level tags

Introduction

To access data areas within a declared tag, you can overlay the declared tags with an additional declaration. This provides you with the option of addressing an already declared tag with a different data type. You can, for example, address the individual bits of a tag of WORD data type with an ARRAY of BOOL.

Overlaying tags

To overlay a tag with a new data type, follow these steps:

1. Open the block interface.
2. In the interface, select the tag that you want to overlay with a new data type.
3. Click "Add row" in the toolbar.
   A row is inserted after the tag to be overlaid. The overlaying tag must be declared in the row directly after the tag that is to be overlaid.
4. Enter a tag name in the "Name" column.
5. Enter the "AT" entry in the "Data type" column. You will be supported by Autocomplete in this step.
   The following is added to the entry in the "Name" column.
   `"AT<Name of the higher-level tag>"`
6. Click the data type selection button again and select the data type for the new tag.
   The tag is created. It points to the same data as the higher-level tag, however interprets this data with the new data type.

Removing overlay

To remove the overlay of a tag, follow these steps:

1. Select the overlaid tag that you want to remove.
2. Select the "Delete" command in the shortcut menu.
3. The overlay is removed.

See also

Editing tables (Page 347)

Overlaying tags with AT (Page 1462)

Declaring multi-instances

Requirement

- The function block to be called exists in project tree and is multi-instance capable.
- The block interface of the calling function block is open.

Procedure

To declare a function block to be called as a multi-instance, follow these steps:

1. In the "Name" column of the "Static" section, enter a designation for the block call.
2. In the "Data type" column, enter the symbolic name for the function block to be called.

Note

The program editor will declare the multi-instance automatically if you program a block call in a network and then specify in the "Call options" dialog that you want to call the block as a multiple instance.

See also

Updating the block interface (Page 1564)

Updating the block interface

Introduction

If you change or delete PLC data types or multiple instances that are used in the block interface, the interface will become inconsistent. To remedy this inconsistency, the interface has to be updated.

You have two options for updating the block interface:

- Explicit updating of the block interface.
  The used PLC data types and multiple instances will be updated. The instance data blocks that belong to the block are not implicitly updated during this process.
- Implicit updating during compilation.
  All used PLC data types and multiple instances as well as the related instance data blocks will be updated.

Explicit updating of the block interface

To explicitly update the block interface, follow these steps:

1. Open the block interface.
2. Select the "Update" command in the shortcut menu.

Implicit Updating during Compilation

Proceed as follows to implicitly update all uses of PLC data types and multiple instances as well as the instance data blocks during compilation:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Basics of PLC data types (Page 1734)

Declaring tags based on a PLC data type (Page 1562)

Editing tables (Page 347)

Basic information on start values (Page 1713)

Using tags within the program (Page 1447)

Keywords (Page 1442)

Properties of local tags and constants (Page 1567)

Setting the retentivity of local tags (Page 1568)

Updating block calls in LAD (Page 1596)

Declaring multi-instances (Page 1563)

Extending the block interface

Description

In order to enable the editing of PLC programs that have already been commissioned and that are running without error on a system, CPUs of the S7-1500 series and most CPUs of the S7-1200 V4 and higher series support the option of extending the interfaces of function blocks during runtime.

You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Principle of operation

Each function block is always assigned a default memory reserve. The memory reserve is not used initially. Activate the memory reserve if you decide on loading interface changes after having compiled and downloaded the block. All tags that you subsequently declare will be saved to the memory reserve. The subsequent download does not influence any tags that are already loaded or have a negative impact on runtime.

If you decide to review your program at a later time while the plant is not in operation, you are also provided an option of reworking the memory layout of individual or several blocks in a single pass. With this action, you move all tags from the reserve area to the regular area. The memory reserve is now cleared and made available for further interface extensions.

Requirements

This "Download without reinitialization" function is available if the following requirements are met:

- The project is in the "TIA Portal V12" format or a higher version.
- You are working with a CPU that supports "Download without reinitialization".
- The blocks were created in LAD, FBD, STL, or SCL.
- The blocks were created by the user, i.e. they are not included with the blocks delivered in your package.
- These blocks are assigned the optimized access attribute.

Basic steps

Perform the following steps if you want to extend the interface of a function block and then load the block without re-initialization.

1. All blocks have a default memory reserve of 100 bytes. You can adapt this memory reserve to suit your requirements.
2. Activate the memory reserve.
3. Extend the block interface.
4. Compile the block.
5. Download the block to the CPU as usual.

For more information on the various steps, refer to chapter "Loading blocks (S7-1200/1500) ".

Note

The full scope of the "Download without reinitialization" function is only available on CPUs of the S7-1500 and S7-1200 V4 series.

However, all CPU families support the option of extending the interface of function blocks and downloading newly declared tags without repercussion:

- You may add new tags in the "Temp" section and download these without influencing the process.
- You may create new tags of a structured data type in the "InOut" section and download these without influencing the process.

Editing the properties of local tags and constants

Properties of local tags and constants

Properties

The table below provides an overview of the properties of local tags and constants:

Group	Property	Description
General	Name	Name of the element.
	Data type	Data type of the element.
	Default value	Value with which you can pre-assign specific tags in the interface of the code block, or the value of a local constant. Specification of the default value is optional for tags. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL. The default value of a tag is applied as the start value in the corresponding instance. You can then replace these adopted values with instance-specific start values.
	Comment	Comment for the element.

Group	Property	Description
Attributes	Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This attribute is only available in the interface of the function block with optimized access.
	Accessible from HMI	Indicates whether the tag can be used in HMI. When the attribute is set, you have read or write access to the tag from the HMI. When the attribute is not set, you cannot access the tag from the HMI. Please note, however, that you cannot implement general access protection for the tag with the "Accessible from HMI" attribute. Read or write access from other applications is possible even if the attribute is not enabled.
	Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
	Hidden parameter	Indicates whether the tag should be hidden for the block call. This is only possible if you have specified a valid predefined actual parameter beforehand.
	Predefined actual parameter	Defines a parameter that is to be used as actual parameter during the block call.
	Configurable	Indicates whether a parameter is configurable in CFC.
	For test	Indicates whether a parameter is registered for the CFC test mode.
	Visible	Indicates whether a parameter is visible in CFC.
	Interconnectable	Indicates whether a parameter is interconnectable in CFC.
	Enable tag readback	Indicates whether a parameter is relevant for the "Read back chart" function in CFC.
	Enumeration texts	Assigns a parameter to an enumeration in CFC.
	Engineering unit	Assigns a parameter to a unit in CFC.
	Low limit	Defines the low limit for the parameter in CFC.
	High limit	Defines the high limit for the parameter in CFC.

See also

Setting the retentivity of local tags (Page 1568)

Changing properties of local tags and constants (Page 1569)

Keywords (Page 1442)

Setting the retentivity of local tags

Introduction

Function blocks store their data in an instance. To prevent data loss in the event of power failure, you can mark the data as retentive. This data is stored in a retentive memory area. The option of setting the retentivity depends on the set access type of the function block.

Retentive behavior in blocks with standard access

In blocks with standard access you cannot set the retentive behavior of individual tags. You can only define them as retentive in the assigned instance. All tags contained in the block are then considered as retentive.

Retentivity for optimized block access

In data blocks with optimized access you can define the retentive behavior of individual tags.

For structured data type tags, the retentivity setting always applies to the entire structure. You can make no individual retentivity setting for individual elements within the structure.

You cannot create retentive tags of the structured data type in the "InOut" section. In/out parameters with structured data type, for example ARRAY, STRUCT, or STRING, are always non-retentive.

The following settings are available:

- Retentive
  The values of the tags or the structure are available even after a power failure.
- Non-retentive
  The values of the tags or the structure are lost in the event of a power failure.
- Set in IDB
  The retentivity can be set in the instance data block. The setting that is made in the instance data block than applies, however, centrally to all tags that are selected with "Set in IDB".

See also

Properties of local tags and constants (Page 1567)

Basics of block access (Page 1419)

Changing properties of local tags and constants

Editing properties of an element in the block interface

To edit the properties of an element in the block interface, follow these steps:

1. Open the block interface.
2. Select the required element in the table.
3. Change the entries in the columns.

Editing properties of several elements in the block interface

You can also simultaneously set or reset the "Retain", "Visible in HMI", "Accessible in HMI" and "Setpoint" columns for one or more selected elements.

To change one of these properties for several elements, follow these steps:

1. Open the block interface.
2. Hold down the CTRL key.
3. In the required column, select each of the table cells whose value you want to change.
4. Select the "Set <property>" or "Reset <property>" command in the shortcut menu.

Editing properties in the properties window

To edit the properties of an individual tag or constant, follow these steps:

1. Select an element in the table.
   The properties of the element are shown in the Inspector window.
2. Change the entries in the inspector window.

Renaming tags directly in the program editor

To rename one or more elements, follow these steps:

1. Select one or more elements in the program.
2. Select the "Rename tag" command in the shortcut menu.
   The "Rename tag" dialog opens. It displays a declaration table with the selected elements.
3. Change the entries in the "Name" column.
4. Confirm the input by clicking the "Change" button.

Editing the data type or comment in the program editor

Proceed as follows to edit the data type or tag comment in the program editor:

1. Select the name of the tag.
2. Select the "Rewire tag" command in the shortcut menu.
   The "Rewire tag" dialog will open. The dialog shows a declaration table.
3. Change the entry in the "Data type" or "Comment" columns.
4. Click the "Change" button to confirm the input.

Effect in the program

In case of a change of the name, data type or address of a tag or constant, each location of use of the tag is automatically updated in the program.

Note

If you change the interface of a block, the program may become inconsistent. The inconsistencies are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent calls are marked in red. You than have to manually updated the inconsistencies.

See also:

Updating block calls in LAD (Page 1596)

Updating block calls in FBD (Page 1638)

See also

Layout of the block interface (Page 1551)

Editing tables (Page 347)

Properties of local tags and constants (Page 1567)

Setting the retentivity of local tags (Page 1568)

Basic information on start values (Page 1713)

Using tags within the program (Page 1447)

Keywords (Page 1442)

Updating the block interface (Page 1564)

Declaring PLC data types (UDT) as actual parameters for in-out parameters.

Use of PLC data types as predefined actual parameters

In blocks with optimized access, you have the option to use PLC data types as actual parameters for in/out parameters (InOut) under certain conditions. This can be useful if you are using program blocks as library elements and want to store information about the actual parameters to be used along with the library element.

Additionally, you have the option of hiding in/out parameters that have a vaild predefined actual parameter when the block is called. Hidden parameters are not visible initially when the block is called but can be displayed via a small arrow at the bottom edge of the box.

Requirements

- The in/out parameter (InOut) is based on a PLC data type or a system data type.
- The in/out parameter has the retentivity setting "Retain" or "Non-retain". For in/out parameters with setting "Set in IDB", it is not possible to predefine actual parameters.
- The program block is a block with optimized access.

Procedure

To predefine the actual parameter of an in/out parameter, follow these steps:

1. Open the block interface.
2. Select an in/out parameter (InOut) in the block interface.
3. Open the "Properties" tab in the Inspector window.
4. Select the "Attributes" group in the area navigation.
5. Enter the required actual parameter in the "Predefined actual parameter" input box.
6. Select the "Hidden parameter" check box (optional).

Result

- An actual parameter is predefined. If you save the program block as a library element, it thus also contains information about the actual parameter to be used.
- If the library element is used in the program, a check is made to determine whether the actual parameter you have predefined can be addressed. If so, it is automatically used as the actual parameter.
- If the actual parameter is not found in the program, a syntax error is signaled. The parameter is not hidden and the parameter must be initialized manually.

Editing the block interface

Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

See also

Editing tables (Page 347)

Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

See also

Editing tables (Page 347)

Deleting tags

Procedure

Follow the steps below to delete elements:

1. Select the row with the element to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

See also

Editing tables (Page 347)

Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

See also

Editing tables (Page 347)

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.
5. To hide or show several columns, click "More" and activate or deactivate the check box of the corresponding columns in the "Show/Hide" dialog.

Editing tags with external editors

To edit individual tags in external table editors, such as Excel, you can export or import these tags using copy and paste. However, you cannot copy structured tags to an editor.

Requirements

The block interface and an external editor are opened.

Procedure

To export individual tags to an external editor and import them again, follow these steps:

1. Select one or more tags.
2. Select "Copy" in the shortcut menu.
3. Switch to the external editor and paste the copied tags.
4. Edit the tags as required.
5. Copy the tags in the external editor.
6. Select the tags in the external editor.
7. Switch back to the block interface.
8. Select "Paste" in the shortcut menu.

Creating program code

Creating LAD programs

Basic information on LAD

LAD programming language

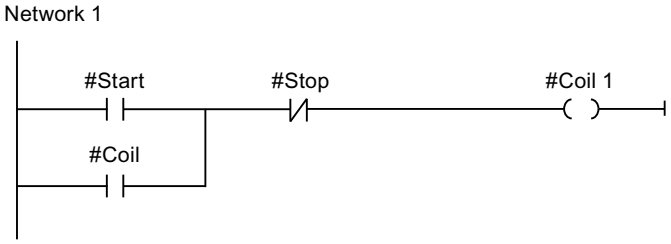
Overview of the Ladder Logic (LAD) programming language

LAD is a graphical programming language. The representation is based on circuit diagrams.

The program is mapped in one or more networks. A network contains a power rail on the left where the rungs originate. The binary signal scans are arranged in the form of contacts on the rungs. The serial arrangement of the elements on a rung creates a series connection; arrangement on simultaneous branches creates a parallel connection. Complex functions are represented by boxes.

Example of networks in LAD

The following figure shows a LAD network with two normally open contacts, one normally closed contact and one coil:



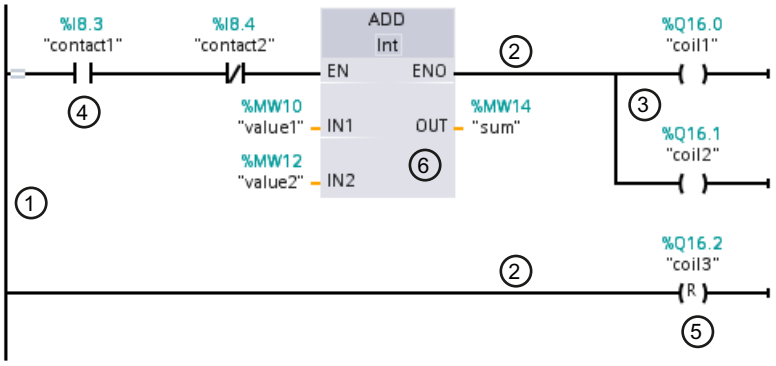
Overview of the LAD elements

LAD elements

A LAD program consists of separate elements that you can arrange in series or parallel on the power rail of a network. Most program elements must be supplied with tags.

There is at least one rung from the power rail. Network programming starts at the left edge of the rung. You can expand the power rail by several rungs and branches.

For example, the following figure shows elements of a LAD network:



1) Power rail

2) Rung

3) Branch

4) Contact

5) Coil

6) Box

Power rail

Each LAD network consists of a power rail that contains at least one rung. A network can be extended by adding additional rungs. You can use branches to program parallel connections in the specific rungs.

Contacts

You can use contacts to create or interrupt a current-carrying connection between two elements. The current is relayed from left to right. You can use contacts to query the signal state or the value of an operand and control it depending on the result of the current flow.

The following types of contact are available to you in a LAD program:

- Normally open contact:
  Normally open contacts forward the current if the signal state of a specified binary operand is "1".
- Normally closed contacts:
  Normally closed contacts forward the current if the signal state of a specified binary operand is "0".
- Contact with additional function:
  Contacts with additional function forward the current if a specific condition is met. With these contacts you can also execute an additional function, such as an RLO edge detection and a comparison.

Coils

You can use coils to control binary operands. Coils can set or reset a binary operand depending on the signal state of the result of logic operation.

The following types of coils are available to you in a LAD program:

- Standard coils:
  Standard coils set a binary operand if current flows in the coil. The "Assignment" instruction is an example of a standard coil.
- Coils with additional function:
  These coils have additional functions in addition to the evaluation of the logic operation result. Coils for RLO edge detection and program control are examples of coils with additional function.

Boxes

Boxes are LAD elements with complex functions. The empty box is an exception. You can use the empty box as a placeholder in which you can select the required instruction.

The following types of boxes are available to you in a LAD program:

- Boxes without EN/ENO mechanism:
  A box is executed depending on the signal state at the box inputs. The error status of the processing cannot be queried.
- Boxes with EN/ENO mechanism:
  A box is only executed if the enable input "EN" carries the signal state "1". If the box is processed correctly, the "ENO" enable output has signal state "1". If an error occurs during the processing, the "ENO" enable output is reset.

Calls of code block are also shown in the network as boxes with EN/ENO mechanism.

See also

Rules for the use of LAD elements (Page 1586)

Settings for LAD

Overview of the settings for LAD

Overview

The following table shows the settings that you can make:

Group	Setting	Description
Font	Font size	Font size in program editor
View	Layout	Compact or wide Changes the vertical spacing between operands and other objects (such as operand and contact). The change becomes visible once the block is reopened.
	With absolute information	Additional display of the absolute addresses
Operand field	Maximum width	Maximum number of characters that can be entered horizontally in the operand field. This setting recalculates the layout of the networks.
	Maximum height	Maximum number of characters that can be entered vertically in the operand field. This setting recalculates the layout of the networks.

See also

Changing the settings (Page 1579)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for LAD (Page 1578)

Working with networks

Using networks

Function

The user program is created in the block within networks. For a code block to be programmed, it must contain at least one network. To achieve a better overview of the user program, you can also subdivide your program into several networks.

See also

Inserting network title (Page 1583)

Entering a network comment (Page 1584)

Navigating networks (Page 1585)

Inserting networks

Requirement

A block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Result

A new empty network is inserted into the block.

See also

Selecting networks (Page 1580)
Copying and pasting networks (Page 1581)
Deleting networks (Page 1582)
Expanding and collapsing networks (Page 1582)
Inserting network title (Page 1583)
Entering a network comment (Page 1584)
Navigating networks (Page 1585)

Selecting networks

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.
   The first and last networks and all those in between are selected.

See also

Inserting networks (Page 1579)
Copying and pasting networks (Page 1581)
Deleting networks (Page 1582)
Expanding and collapsing networks (Page 1582)
Inserting network title (Page 1583)
Entering a network comment (Page 1584)
Navigating networks (Page 1585)

Copying and pasting networks

Copied networks can be pasted within the block or in another block. Networks that were created in LAD or FBD can also be inserted in blocks of the respective other programming language.

Requirement

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Inserting networks (Page 1579)
Selecting networks (Page 1580)
Deleting networks (Page 1582)
Expanding and collapsing networks (Page 1582)
Inserting network title (Page 1583)
Entering a network comment (Page 1584)
Navigating networks (Page 1585)

Deleting networks

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting networks (Page 1579)
Selecting networks (Page 1580)
Copying and pasting networks (Page 1581)
Expanding and collapsing networks (Page 1582)
Inserting network title (Page 1583)
Entering a network comment (Page 1584)
Navigating networks (Page 1585)

Expanding and collapsing networks

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Inserting networks (Page 1579)

Selecting networks (Page 1580)

Copying and pasting networks (Page 1581)

Deleting networks (Page 1582)

Inserting network title (Page 1583)

Entering a network comment (Page 1584)

Navigating networks (Page 1585)

Inserting network title

The network title is the header of a network. The length of the network title is limited to one line. You can enter the title manually or set it automatically. When you set it automatically, you can do this for individual networks or use the settings to specify that the network title is always set automatically.

For automatic insertion of the network title, the comment of the operand in one of the following instructions in the network is evaluated:

- Assignment
- Set output
- Reset output

The instruction that is listed first in the network is used.

The network title is only inserted automatically when the following conditions are fulfilled:

- The network does not have a title yet.
- The operand of the instruction used for the comment has a comment.

Note

Note the following restrictions for automatic insertion of the network title:

- The network title is not adapted if you change the comment of the operand at a later time.
- The network title is not adapted if you change the operand of the instruction.
- The network title is only set by the writing instructions listed above.
- If the operand is of the data type array, the comment of the array is used and not the comments of the array elements.
- Comments of invalid operands are not taken into consideration.

Entering the network title manually

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

Setting the network title automatically

To specify that the network titles are always set automatically, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Select the "Set network title automatically" check box in the "Additional settings" group.
   The network titles are set automatically as of this time if the conditions listed above are fulfilled.

To set an individual network title automatically, follow these steps:

1. Right-click "Network <Number of the network>" in the title bar of a network.
2. Select the "Set network title automatically" command in the shortcut menu.
   The title of the selected network is set based on the comment of the operand if the conditions listed above are fulfilled.

See also

Using networks (Page 1579)

Inserting networks (Page 1579)

Selecting networks (Page 1580)

Copying and pasting networks (Page 1581)

Deleting networks (Page 1582)

Expanding and collapsing networks (Page 1582)

Entering a network comment (Page 1584)

Navigating networks (Page 1585)

Entering a network comment

You can use network comments to provide comments on the program contents of individual networks. For example, you can indicate the function of the network or draw attention to special characteristics.

Requirement

A network is available.

Procedure

To enter a network comment, follow these steps:

1. Click on the right arrow before the network title.
2. If the comment area is not visible, click "Network comments on/off" in the toolbar.
   The comment area is displayed.

3. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
4. Enter the network comment.

See also

Using networks (Page 1579)

Inserting networks (Page 1579)

Selecting networks (Page 1580)

Copying and pasting networks (Page 1581)

Deleting networks (Page 1582)

Expanding and collapsing networks (Page 1582)

Inserting network title (Page 1583)

Navigating networks (Page 1585)

Navigating networks

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click in the code area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.
   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 1579)

Inserting networks (Page 1579)

Selecting networks (Page 1580)

Copying and pasting networks (Page 1581)

Deleting networks (Page 1582)
Expanding and collapsing networks (Page 1582)
Inserting network title (Page 1583)
Entering a network comment (Page 1584)

Inserting LAD elements

Rules for the use of LAD elements

Rules

Note the following rules when inserting LAD elements:

- Every LAD network must terminate with a coil or a box. However, the following LAD elements must not be used to terminate a network:
  - Comparator boxes
  - Instructions for positive and negative RLO edge detection
- The starting point of the branch for a box connection must always be the power rail. Logic operations or other boxes can be present in the branch before the box.
- Only contacts can be inserted into simultaneous branches with preceding logic operations. The contact for negating the result of logic operation (-|NOT|-) is an exception here. The contact for negating the result of logic operation, as well as coils and boxes, can be used in simultaneous branches if they originate directly from the power rail.
- Constants (e.g. TRUE or FALSE) cannot be assigned to normally closed or normally open contacts. Instead, use operands of the BOOL data type.
- Only one jump instruction can be inserted in each network.
- Only one jump label can be inserted in each network.
- Instructions with positive or negative edge detection may not be arranged directly at the left margin of the rung as they requires a prior logic operation.

Placement rules for S7-1200/1500 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
SET_BF	Set bit field	No
RESET_BF	Reset bit field	No
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
JMP_LIST	Define jump list	No

Instruction		Preceding logic operation required
Mnemonics	Name	
SWITCH	Jump distributor	No
RET	Return	No

Placement rules for S7-300/400 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
S	Set output	Yes
R	Reset output	Yes
SP	Start pulse timer	Yes
SE	Start extended pulse timer	Yes
SD	Start on-delay timer	Yes
SS	Start retentive on-delay timer	Yes
SF	Start off-delay timer	Yes
SC	Set counter value	Yes
CU	Count up	Yes
CD	Count down	Yes
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
RET	Return	No
OPN	Open global data block	No
OPNI	Open instance data block	No
CALL	Call block	No
SAVE	Save RLO in BR bit	No
MCRA	Enable MCR range	No
MCRD	Disable MCR range	No
MCR<	Open MCR ranges	No
MCR>	Close MCR ranges	No

See also

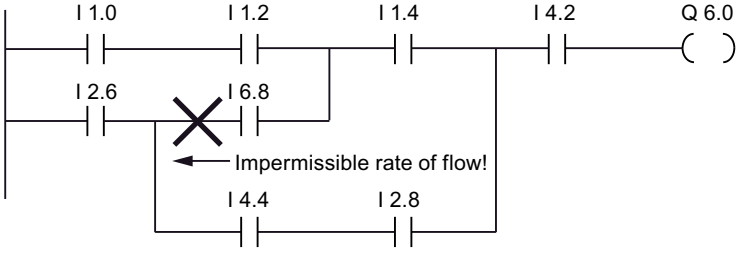
Prohibited interconnections in LAD (Page 1588)

Overview of the LAD elements (Page 1576)

Prohibited interconnections in LAD

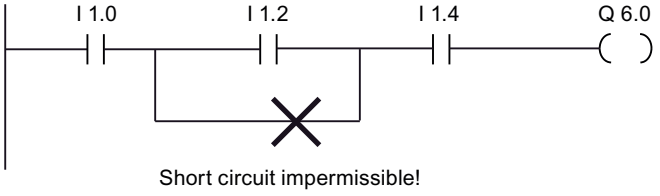
Power flow from right to left

No branches can be programmed that could result in a power flow in the reverse direction.



Short-circuit

No branches may be programmed that would cause a short-circuit.



Logic operations

The following rules apply to logic operations:

- Only Boolean inputs can be combined with preceding logic operations.
- Only the first Boolean output can be combined with a further logic operation.
- Only one complete logical path can exist per network. Paths that are not connected can be linked.

See also

Rules for the use of LAD elements (Page 1586)

Inserting LAD elements using the "Instructions" task card

Requirement

A network is available.

Procedure

To insert a LAD element into a network using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to the LAD element that you want to insert.
3. Use drag-and-drop to move the element to the desired place in the network.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multiple instance", these are located in the block interface in the "Static" section.

Or:

1. Select the point in the network at which you want to insert the element.
2. Open the "Instructions" task card.
3. Double-click on the element you want to insert.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multiple instance", these are located in the block interface in the "Static" section.

Result

The selected LAD element is inserted with placeholders for the parameters.

Inserting LAD elements using an empty box

Requirement

A network is available.

Procedure

To insert an LAD element into a network using an empty box, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Empty box" in the "Basic instructions" palette.
3. Use a drag-and-drop operation to move the "Empty box" element to the desired place in the network.

4. Position the cursor over the triangle in the top right-hand corner of the empty box.
   A drop-down list is displayed.
5. Select the required LAD element from the drop-down list.
   If the element is an internal system function block (FB), the "Call option" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multi-instance", these are located in the block interface in the "Static" section.

Result

The empty box is changed to the respective LAD element. Placeholders are inserted for the parameters.

Selecting the data type of a LAD element

Selecting a data type

Introduction

Some instructions can be executed with several different data types. If you use one of these instructions in the program, you have to specify a valid data type for the instruction at the specific point in the program. For some instructions, you have to select the data types for the inputs and outputs separately.

Note

The valid data type (BOOL) for the tags on the enable input EN and the enable output ENO is predefined by the system and cannot be changed.

The valid data types for an instruction are listed in the instruction drop-down list. You specify the data type of the instruction by selecting an entry from the drop-down list. If the data type of an operand differs from the data type of the instruction and cannot be converted implicitly, the operand is displayed in red and a rollout with the corresponding error message appears.

Data type selection of mathematical instructions

Some mathematical instructions provide you with the option of having the data type automatically set corresponding to the data types of the operand. In the drop-down list for data type selection, these instructions have the entry "Auto" in addition to the actual data types. If you select this entry and then allocate the first operand, the data type of the operand is selected as data type for the instruction. The entry in the drop-down list changes to "Auto (<Data type>)", e.g. "Auto (Real)". If you allocate additional operands, the automatically set data type of the instruction is adjusted according to the following criteria:

- You supply all other operands with tags of the same data type:
  The data type of the instruction is not changed.
- You supply all other operands with tags whose data type is smaller than the data type of the instruction:
  The data type of the instruction is not changed. For the operand with the smaller data type, an implicit conversion is conducted if necessary.
- You supply an additional operand with a tag whose data type is greater than the data type of the instruction:
  The data type of the instruction is changed to the larger data type. An implicit conversion is performed, if necessary, for operands that deviate from the newly set data type of the instruction.

Each change in the data type of an operand can result in a change of the data type of the instruction. Other operands may possibly be implicitly converted as a result. Operands for which an implicit conversion is performed are marked with a gray square.

Note

Please also observe the information on data type conversion for your device and, in particular, the notes on the IEC check.

See also: Data type conversion (Page 2091)

See also

Defining the data type of an instruction (Page 1591)

Defining the data type of an instruction

Introduction

Some instructions can be executed with several different data types. When you insert such instructions into your program, you must specify the data type for these instructions at the actual point in the program.

Specifying the data type by means of the drop-down list

To define the data type of an instruction using the drop-down list, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Click the triangle in the upper corner of the drop-down list.
   The drop-down list will open to display the data types valid for the instruction.
3. Select a data type from the drop-down list.
   The selected data type is displayed.
4. If the instruction has two drop-down lists, select the data type for the instruction inputs in the left-hand drop-down list and the data type for the instruction outputs in the right-hand drop-down list.

Specifying data type by assigning tags

To define the data type of an instruction by assigning tags, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. At an input or output, specify a valid tag, the data type of which is to be applied as the instruction data type.
   The data type of the tag is displayed in the drop-down list.
3. Enter a valid tag at an input and a valid tag at an output if data types need to be defined for both the inputs and outputs of the instruction. The tag specified at the input determines the data type of the inputs; the tag specified at the output determines the data type of the outputs of the instruction.

Automatically specifying the data type of mathematical instructions

To automatically specify the data type for mathematical instructions, follow these steps:

1. Insert the mathematical instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Select the "Auto" entry from the drop-down list.
3. Enter a valid tag at an input or output.
   The data type of the tag is applied as data type of the instruction. The entry in the drop-down list changes to "Auto (<Data type>)".

See also: Selecting a data type (Page 1590)

See also

Selecting a data type (Page 1590)

Using favorites in LAD

Adding LAD elements to Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing LAD elements from Favorites (Page 1594)

Overview of the program editor (Page 1529)

Inserting LAD elements using favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing LAD elements from Favorites (Page 1594)

Overview of the program editor (Page 1529)

Removing LAD elements from Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding LAD elements to Favorites (Page 1593)

Inserting LAD elements using favorites (Page 1593)

Overview of the program editor (Page 1529)

Insert block calls in LAD

Inserting block calls using a drag-and-drop operation

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree. If you call function blocks from other function blocks, you can either call them as single-instance or multi-instance blocks. If a function block is called as single instance, it will store its data in a data block of its own. If a function block is called as multi-instance, it will store its data in the instance data block of the calling function block.

Requirement

- A network is available.
- The block that is to be called is available.

Inserting a call of a function (FC)

To insert a call of a function (FC) into a network using a drag-and-drop operation, follow these steps:

1. Drag the function from the project tree to the required network.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Drag the function block from the project tree to the required network.
   The "Call options" dialog opens.
2. Enter in the dialog whether you wish to call the block as single or multi-instance.
   - If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block.
   - If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
3. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You can then assign the parameters.

See also: Auto-Hotspot

Note

If when calling a function block you specify an instance data block that does not exist, it will be created. If you have called a function block as a multi-instance, this will be entered as a static tag in the interface.

See also

Updating block calls in LAD (Page 1596)

Changing the instance type (Page 1598)

Single instances (Page 1427)

Multi-instances (Page 1427)

Updating block calls in LAD

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have the following options for updating the block calls:

- Explicit updating of all inconsistent block calls in the program editor.
  The inconsistent block calls within the open block are updated. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are removed if they are not connected.
  - Renamed parameters get the new parameter names.
- Explicit updating of a block call in the program editor.
  The "Interface update" dialog is displayed. In this dialog, you have the option of changing the connection of the operands of the new interface. The inconsistent call of this block is then updated. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are removed if they are not connected.
  - Renamed parameters get the new parameter names.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated. Note that when you call functions (FCs) before the next compilation process, all new formal parameters must be supplied with actual parameters.

Updating all inconsistent block calls in the program editor

To open all block calls in a block, follow these steps:

1. Open the calling block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

Updating a specific block call in the program editor

To update a specific block call in the program editor, follow these steps:

1. Open the calling block in the program editor.
2. Right-click on the block call that you want to update.
3. Select the "Update" command in the shortcut menu.
   The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
4. If necessary, change the connection of the operands. To do this, you have the following options:
   - You can use either a drag-and-drop operation or a cut/copy-and-paste operation to move the operand from the old interface to the new interface.
   - You can delete an operand.
   - You can rename an operand.
   - You can specify a new operand via autocompletion.
5. Click "OK" to update the block call. If you want to cancel the update, click "Cancel".

Note

Note that the "Update block call" command is only available provided you did not previously update all block calls in the editor with the "Update inconsistent block calls" command.

Update block calls during compilation

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Inserting block calls using a drag-and-drop operation (Page 1595)

Changing the instance type (Page 1598)

Changing the block call

You have the option of changing the called block for a block call. But keep in mind that no new instance data blocks are created, for example, when changing from a function (FC) to a function block (FB).

Procedure

To change the called block of a block call, follow these steps:

1. Click on the name of the called block within the block call and press the <F2> key. Or double-click the name of the called block.
   A text box opens, and the name of the currently called block is selected.
2. Enter the name of the block you want to call or select a block in the autocompletion.
3. If you want to call an FB, create a new instance data block, if necessary, and specify it as operand.

Changing the instance type

Instance type

There are two ways of calling function blocks:

- As a single instance
- As a multiple instance

See also: Auto-Hotspot

You can modify a defined instance type at any time.

Requirement

The user program contains a block call.

Procedure

To change the instance type of a function block, follow these steps:

1. Open the code block and select the block call.
2. Select the "Change instance" command in the shortcut menu.
   The "Call options" dialog opens.
3. Click the "Single instance" or "Multi instance" button.
   - If you select the "Single instance" instance type, enter a name for the data block that is to be assigned to the function block.
   - If you select "Multiple instance" as the instance type, enter in the "Name in the interface" text field the name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".

Note

The previous single and multiple instances will not be deleted automatically.

See also

Inserting block calls using a drag-and-drop operation (Page 1595)

Updating block calls in LAD (Page 1596)

Inserting complex LAD instructions

Using the "Calculate" instruction

Requirement

A network is available.

Procedure

Proceed as follows to use the "Calculate" instruction:

1. Open the "Instructions" task card.
2. Navigate to "Math functions > CALCULATE" in the "Basic instructions" pane.
3. Use drag-and-drop to move the element to the desired place in the network.
   The instruction "Calculate" will be inserted for the data type with a placeholder expression and question mark.
4. Enter the data type for the calculation.
5. Enter the operands for the calculation.

   Note

   The calculation is run with the inputs of the "Calculate" instruction. If you want to use constants you must also insert appropriate inputs for them.

6. Click on the "Edit 'Calculate' instruction" button to replace the placeholder expression with the correct expression.
   The "Edit 'Calculate' instruction" dialog will open.
7. Enter the required expression in the "OUT:= " text box.

   Note

   In the "Example" area you can find an example of a valid expression and possible instructions that you can use.

   To determine a value with the help of Pythagoras' theorem, for example, enter "OUT := SQRT (SQR (IN1) + SQR (IN2))".

8. Confirm your entry with "OK".

See also

CALCULATE: Calculate (Page 2300)

Using free-form comments

Basic information on using free-form comments in LAD

Introduction

Free-form comments allow you to add comments to the source code for graphic programming languages similar to line comments for textual languages.

Free-form comments can be used for the following elements:

- Boxes
- Coils

See also

Inserting free-form comments (Page 1600)

Editing free-form comments (Page 1601)

Deleting free-form comments (Page 1602)

Inserting free-form comments

Requirement

A network with instructions is available.

Procedure

To insert a free comment on an instruction, proceed as follows:

1. If necessary, activate the "Free-form comments on/off" button in the toolbar.
2. Right-click on the instruction for which you want to insert a free-form comment.
3. Select the "Insert comment" command in the shortcut menu.
   A comment box with a standard comment opens. The comment box is connected by an arrow to the corresponding instruction.
4. Enter the required comment in the comment box.

See also

Basic information on using free-form comments in LAD (Page 1600)

Editing free-form comments (Page 1601)

Deleting free-form comments (Page 1602)

Editing free-form comments

Introduction

Free-form comments can be edited as follows:

- Changing the comment text
- Changing the position and size of the comment box
- Attaching a comment to another element
- Showing and hiding free comments

Changing the comment text

To change the text of free-form comments, follow these steps:

1. Click on the comment box.
2. Enter the desired text.

Changing the position of the comment box

To change the positioning of the comment box, follow the steps below:

1. Left-click the comment box and keep the mouse button pressed.
2. Drag the comment box to the desired location.

Changing the size of the comment box

To change the size of the comment box, follow the steps below:

1. Click on the comment box.
2. Drag the comment box on the move handle in the lower right corner to the desired size.

Attaching a comment to another element

To attach a free-form comment to another element, follow these steps:

1. Left-click the point of the arrow that links the comment box with the instruction and keep the mouse button pressed.
2. Drag the arrow to the element to which you want to attach the comment. Possible insertion points are marked with a green square.
3. Release the mouse button.

Showing and hiding free comments

To show or hide a free-form comments, follow these steps:

1. Click the "Free-form comment on/off" button in the toolbar.

See also

Basic information on using free-form comments in LAD (Page 1600)

Inserting free-form comments (Page 1600)

Deleting free-form comments (Page 1602)

Deleting free-form comments

Procedure

To delete a free-form comment, proceed as follows:

1. Right-click on the free-form comment that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on using free-form comments in LAD (Page 1600)

Inserting free-form comments (Page 1600)

Editing free-form comments (Page 1601)

Editing LAD elements

Selecting LAD elements

You can select several individual elements or all elements in a network.

Requirement

LAD elements are available

Selecting several individual LAD elements

To select several individual LAD elements, follow these steps:

1. Press and hold down the <Ctrl> key.
2. Click on all the LAD elements you wish to select.
3. Now release the <Ctrl> key.

Selecting all LAD elements in a network

To select all LAD elements in a network, follow these steps:

1. Go to the network whose elements you wish to select.
2. Select the "Select all" command in the "Edit" menu or press <Ctrl A>.

See also

Copying LAD elements (Page 1603)
Cutting LAD elements (Page 1604)
Pasting an LAD element from the clipboard (Page 1604)
Replacing LAD elements (Page 1605)
Inserting additional inputs and outputs in LAD elements (Page 1606)
Removing inputs and outputs (Page 1607)
Enabling and disabling the EN/ENO mechanism (Page 1608)
Deleting LAD elements (Page 1609)

Copying LAD elements

Requirement

An LAD element is available.

Procedure

To copy a LAD element, follow these steps:

1. Right-click the LAD element that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

The LAD element will be copied and saved to the clipboard.

See also

Selecting LAD elements (Page 1602)
Cutting LAD elements (Page 1604)
Pasting an LAD element from the clipboard (Page 1604)
Replacing LAD elements (Page 1605)
Inserting additional inputs and outputs in LAD elements (Page 1606)
Removing inputs and outputs (Page 1607)
Enabling and disabling the EN/ENO mechanism (Page 1608)
Deleting LAD elements (Page 1609)

Cutting LAD elements

Requirement

An LAD element is available.

Cutting

To cut a LAD element, follow these steps:

1. Right-click the LAD element that you want to cut.
2. Select "Cut" in the shortcut menu.

Result

The LAD element will be cut and saved to the clipboard.

See also

Selecting LAD elements (Page 1602)
Copying LAD elements (Page 1603)
Pasting an LAD element from the clipboard (Page 1604)
Replacing LAD elements (Page 1605)
Inserting additional inputs and outputs in LAD elements (Page 1606)
Removing inputs and outputs (Page 1607)
Enabling and disabling the EN/ENO mechanism (Page 1608)
Deleting LAD elements (Page 1609)

Pasting an LAD element from the clipboard

Requirement

An LAD element is available.

Procedure

To paste an LAD element from the clipboard, follow these steps:

1. Copy a LAD element or cut a LAD element.
2. Right-click the point in the network where you want to paste the element.
3. Select "Paste" in the shortcut menu.

See also

Selecting LAD elements (Page 1602)

Copying LAD elements (Page 1603)

Cutting LAD elements (Page 1604)

Replacing LAD elements (Page 1605)

Inserting additional inputs and outputs in LAD elements (Page 1606)

Removing inputs and outputs (Page 1607)

Enabling and disabling the EN/ENO mechanism (Page 1608)

Deleting LAD elements (Page 1609)

Replacing LAD elements

You can easily exchange LAD elements with other LAD elements of the same type. This has the advantage that the parameters are retained and need not be entered again. For example, you can exchange normally open contacts and normally closed contacts or RS FlipFlop and SR FlipFlop.

Requirements

A network with at least one LAD element is present.

Procedure

To replace an LAD element with another LAD element, follow these steps:

1. Select the LAD element that you want to replace.
2. Position the cursor over the triangle in the top right-hand corner of the LAD element. A drop-down list is displayed.
3. From the drop-down list, select the LAD element that you want to use to replace the existing LAD element.

See also

Selecting LAD elements (Page 1602)

Copying LAD elements (Page 1603)

Cutting LAD elements (Page 1604)

Pasting an LAD element from the clipboard (Page 1604)

Inserting additional inputs and outputs in LAD elements (Page 1606)

Removing inputs and outputs (Page 1607)

Enabling and disabling the EN/ENO mechanism (Page 1608)

Deleting LAD elements (Page 1609)

Inserting additional inputs and outputs in LAD elements

Introduction

You can expand LAD elements which execute commutative arithmetic instructions by adding additional inputs. Such elements are, for example, the instructions "Add" (ADD) and "Multiply" (MUL). You can expand the MOVE and DEMUX instruction boxes by adding additional outputs.

Requirement

An LAD element is available that permits the insertion of additional inputs and outputs.

Inserting an additional input

To add an additional input to the box of a LAD element, follow these steps:

1. Right-click on an existing input of the LAD element.
2. Select "Insert input" in the shortcut menu.
   An additional input is added to the box of the LAD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.
   An additional input is added to the box of the LAD element.

Inserting an additional output

To add an additional output to the box of a LAD element, follow these steps:

1. Right-click on an existing output of the LAD element.
2. Select "Insert output" from the shortcut menu.
   An additional output is added to the box of the LAD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.
   An additional output is added to the box of the LAD element.

See also

Selecting LAD elements (Page 1602)

Copying LAD elements (Page 1603)

Cutting LAD elements (Page 1604)

Pasting an LAD element from the clipboard (Page 1604)

Replacing LAD elements (Page 1605)

Removing inputs and outputs (Page 1607)

Enabling and disabling the EN/ENO mechanism (Page 1608)

Deleting LAD elements (Page 1609)

Removing inputs and outputs

Introduction

Inputs and outputs which you have added to an instruction can be removed.

Requirement

An LAD element is available to which you have added additional inputs and outputs.

Remove input

To remove an input, follow these steps:

1. Select the input that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The input of the LAD element is removed.

Remove output

To remove an output, follow these steps:

1. Select the output that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The output of the LAD element will be removed.

See also

Selecting LAD elements (Page 1602)

Copying LAD elements (Page 1603)

Cutting LAD elements (Page 1604)

Pasting an LAD element from the clipboard (Page 1604)

Replacing LAD elements (Page 1605)

Inserting additional inputs and outputs in LAD elements (Page 1606)

Enabling and disabling the EN/ENO mechanism (Page 1608)

Deleting LAD elements (Page 1609)

Enabling and disabling the EN/ENO mechanism

In LAD and FBD, certain instructions have an enable output ENO and thus use the EN/ENO mechanism. This allows you to query runtime errors in instructions and react to them. In order to increase the performance of the CPU, the EN/ENO mechanism is disabled in the default setting. This means that you are not initially able to react to runtime errors of the instruction using the ENO value. However, you can enable the EN/ENO mechanism again at any time, if required.

You can enable the EN/ENO mechanism individually for each instruction in order to generate the ENO. If you enable the EN/ENO mechanism for an instruction, other instructions that you subsequently add to your program are also inserted with the EN/ENO mechanism enabled. You can disable the EN/ENO mechanism again at any time if you do not want to use the evaluation of ENO for an instruction. Further instructions that you subsequently add to your program will then be inserted without the EN/ENO mechanism.

See also: Basics of the EN/ENO mechanism (Page 1473)

Activating the EN/ENO mechanism

Proceed as follows to activate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to activate the EN/ENO mechanism.
2. Select the "Generate ENO" command from the shortcut menu.
   The ENO value is again generated for the instruction. Other instructions are inserted with the enable output.

Deactivating the EN/ENO mechanism

Proceed as follows to deactivate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to deactivate the EN/ENO mechanism.
2. Select the "Do not generate ENO" command from the shortcut menu.
   The ENO value is no longer generated for the instruction. Other instructions are inserted without enable output.

See also

Selecting LAD elements (Page 1602)

Copying LAD elements (Page 1603)

Cutting LAD elements (Page 1604)

Pasting an LAD element from the clipboard (Page 1604)

Replacing LAD elements (Page 1605)

Inserting additional inputs and outputs in LAD elements (Page 1606)

Removing inputs and outputs (Page 1607)

Deleting LAD elements (Page 1609)

Deleting LAD elements

Requirement

An LAD element is available.

Procedure

To delete a LAD element, follow these steps:

1. Right-click the LAD element that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Selecting LAD elements (Page 1602)

Copying LAD elements (Page 1603)

Cutting LAD elements (Page 1604)

Pasting an LAD element from the clipboard (Page 1604)

Replacing LAD elements (Page 1605)

Inserting additional inputs and outputs in LAD elements (Page 1606)

Removing inputs and outputs (Page 1607)

Enabling and disabling the EN/ENO mechanism (Page 1608)

Inserting operands into LAD instructions

Inserting operands

The character strings "<???>", "<??.?>" and "..." are inserted as placeholders for the parameters when an LAD element is inserted. The "<???>" and "<??.?>" strings displayed in red indicate parameters that need to be connected. The "..." string displayed in black indicates parameters that may be connected. "<??.?>" stands for Boolean placeholders.

Note

If you position the cursor over the placeholder, the expected data type will be displayed.

Requirement

An LAD element is available.

Procedure

To connect the parameters of a LAD element, follow these steps:

1. Double-click the placeholder of the parameter.
   An entry field opens, and the placeholder is selected.
2. Enter the appropriate parameter.

   Note

   If you enter the absolute address of a parameter that has already been defined, this absolute address will be changed to the symbolic name of the parameter as soon as the input is confirmed. If you have not yet defined the parameter, a new tag with this absolute address and the default name "Tag_<n>" will be entered in the PLC tag table. When you confirm your input, the absolute address will be replaced with the symbolic name "Tag_<n>".

3. Confirm the parameter with the Enter key.
4. If you have not yet defined the parameter, you can define it directly in the program editor using the shortcut menu.
   See also:
   Declaring PLC tags in the program editor (Page 1487)
   Declaring local tags in the program editor (Page 1558)

Or drag from it the PLC tag table:

1. In the project tree, select the "PLC tags" folder or open the PLC tag table.
2. If you have opened the PLC tag table, drag the symbol from the first column of the desired tag to the appropriate place in your program. If you have not opened the PLC tag table yet, open the detail view now. Drag the desired tag from the detail view to the appropriate place in your program.

Or drag from it the block interface:

1. Open the block interface.
2. Drag the required operand from the block interface to the instruction window.

Result

- If the syntax is error-free, the displayed parameter is black. The editor then jumps to the next placeholder.
- If there is an error in the syntax, the cursor stays in the entry field and a corresponding error message is displayed in the status line. If you press the Enter key again, the entry field is closed and the faulty entry is displayed in red italics.

Wiring hidden parameters

Introduction

Depending on the CPU used, you can use complex instructions in your program that are included with the TIA Portal. These instructions can contain parameters that are declared as hidden.

If an instruction contains hidden parameters, the instruction box has a small arrow on the lower edge. You can recognize hidden parameters by their white font.

You can show and wire the hidden parameters at any time.

Showing or hiding hidden parameters

To show or hide hidden parameters, follow these steps:

1. Click on the down arrow at the bottom edge of the instruction box to show hidden parameters.
2. Click on the up arrow at the bottom edge of the instruction box to hide hidden parameters.

Wiring hidden parameters

To wire parameters, follow these steps:

1. Wire the hidden parameters like normally visible parameters.
   The hidden parameter is transformed into a visible parameter.

See also

Using libraries (Page 470)

Displaying or hiding variable information

Introduction

You can display the following information about the tags to be used in the Program editor:

- Name of the tag
- Address of the tag
- Simple or hierarchical comments for tag documentation

The information is taken from the block interface for local tags and DB tags and from the PLC tag table for tags that are valid CPU-wide.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all the blocks currently opened and opened in future is shown.

You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual blocks that are open.

When you select the display of tag information with hierarchical comments, the comments of the higher structure levels are also displayed for structured tags. The comments are shown in brackets after the tag comment; the comments of the individual levels are separated by a dot. If there is no comment for a tag on a structure level, it is not displayed. This is indicated by two successive dots.

Displaying or hiding tag information for all the blocks

Proceed as follows to display or hide the tag information for all the blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to display the tag information, select the "Show" option or the "Tag information with hierarchical comments" option in the drop-down list, depending on whether you want to display simple or hierarchical comments.
4. If you want to hide the tag information, select the option "Collapse" in the "Tag information" drop-down list.
   The tag information is displayed or hidden for all the blocks. When you open additional blocks, the tag information is displayed or hidden depending on the selected setting.

Displaying or hiding tag information for an opened block

Proceed as follows to display or hide the tag information for an opened block:

1. If you want to display the tag information, select the "Show tag information" option or the "Tag information with hierarchical comments" option in the "Shows the tag information" drop-down list, depending on whether you want to display simple or hierarchical comments.
2. If you want to hide the tag information, select the option "Hide tag information" in the "Shows the tag information" drop-down list.
   The tag information is displayed or hidden.

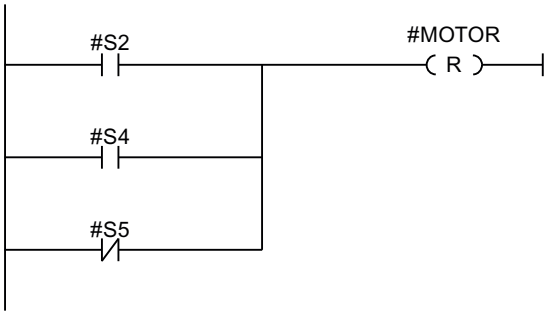
Branches in LAD

Basic information on branches in LAD

Definition

You use branches to program parallel circuits with the Ladder Logic (LAD) programming language. Branches are inserted in the main rung. You can insert several contacts into the branch and thus achieve a parallel circuit of series connections. This allows you to program complex ladder logic.

The figure below shows an example of the use of branches:



MOTOR carries signal 1, if one of the following conditions is fulfilled:

- Signal 1 is pending on S2 or S4
- Signal 0 is pending on S5.

See also

Rules for branches in LAD (Page 1613)

Inserting branches into the LAD network (Page 1614)

Closing branches in the LAD network (Page 1614)

Deleting branches in LAD networks (Page 1615)

Rules for branches in LAD

Rules

The following rules apply to simultaneous branches:

- A simultaneous branch can only be inserted if the main branch already contains an LAD element.
- Simultaneous branches are opened downwards or are connected directly to the power rail. They are terminated upwards.
- Simultaneous branches are opened after the selected LAD element.
- Simultaneous branches are terminated after the selected LAD element.
- To delete a simultaneous branch, you must delete all LAD elements of this branch. When the last LAD element is removed from the branch, the rest of the branch is also removed.

See also

Basic information on branches in LAD (Page 1612)

Inserting branches into the LAD network (Page 1614)

Deleting branches in LAD networks (Page 1615)

Closing branches in the LAD network (Page 1614)

Inserting branches into the LAD network

You can create several branches in a network.

Requirement

- A network is available.
- The network contains elements.

Procedure

To insert a new branch in a network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Open branches" in the "Simple instructions" palette.
3. Use drag-and-drop to move the element to the desired place in the network.
   If you want to connect the new branch directly to the power rail, drag the element to the power rail.

See also

Basic information on branches in LAD (Page 1612)

Rules for branches in LAD (Page 1613)

Deleting branches in LAD networks (Page 1615)

Closing branches in the LAD network

Branches must be closed again at suitable places. If necessary, branches will be arranged so that they do not cross each other.

Requirement

A branch is available.

Procedure

To close an open branch, follow these steps:

1. Select the open branch.
2. Press and hold down the left mouse button.
   A dashed line will appear as soon as the cursor is moved.
3. Drag the dashed line to a suitable place on the network. Permissible connections are indicated by green lines.
4. Release the left mouse button.

See also

Basic information on branches in LAD (Page 1612)

Rules for branches in LAD (Page 1613)

Deleting branches in LAD networks

Requirement

A branch is available.

Procedure

To delete a branch, follow these steps:

1. Select the connection line that links the branch to the main branch.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on branches in LAD (Page 1612)

Rules for branches in LAD (Page 1613)

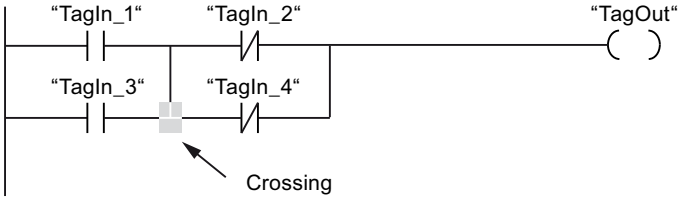
Inserting branches into the LAD network (Page 1614)

Crossings in LAD

Basic information on crossings in LAD

Definition

A crossing is a place in a LAD network where one branch is closed and at the same time another branch is opened.



"TagOut" receives signal 1, if the following two conditions are met:

- "TagIn_1" or "TagIn_3" has signal 1
- "TagIn_2" or "TagIn_4" has signal 0

Inserting crossings

You can insert crossings in a LAD network by creating connections between the main branch and an additional branch or between different branches.

Requirements

A branch is available.

Procedure

To insert a new crossing in an LAD network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Open branches" in the "Simple instructions" palette.
3. Drag the element behind the existing branch.
4. Insert any element into the open branch.
5. Click the arrow of the open branch after the inserted element.
6. Hold down the left mouse button and drag the dashed connecting line to the main branch.
7. Release the left mouse button.

See also

Rearranging crossings (Page 1616)

Deleting crossings (Page 1617)

Inserting branches into the LAD network (Page 1614)

Rearranging crossings

Requirement

A crossing is available.

Procedure

To rearrange a connection, follow these steps:

1. Select the connection line that defines the crossings in the respective branches.
2. Select the "Delete" command in the shortcut menu.
3. Open the "Instructions" task card.
4. Navigate to "General > Open branches" in the "Simple instructions" palette.
5. Use a drag-and-drop operation to move the element to the place in the network where you want to insert the new crossing.
6. Click on the arrow for the open branch.

7. Hold down the left mouse button and drag the dashed connecting line to the subsidiary branch in which you wish to insert the new crossing.
8. Release the left mouse button.

See also

Inserting crossings (Page 1616)

Deleting crossings (Page 1617)

Deleting crossings

Requirement

A crossing is available.

Procedure

To delete a crossing, follow these steps:

1. Select the connection line that defines the crossings in the respective branches.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting crossings (Page 1616)

Rearranging crossings (Page 1616)

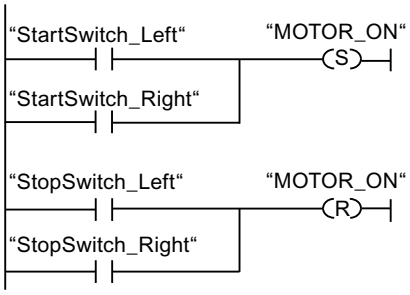
Rungs in LAD

Basic information on rungs in LAD

Using rungs

The program is mapped in one or more networks. A network contains a power rail on the left where one or more rungs originate. The binary signal scans are arranged in the form of contacts on the rungs. The serial arrangement of the elements on a rung creates a series connection; arrangement on simultaneous branches creates a parallel connection. A rung is closed by a coil or a box in which the result of logic operation will be written.

The figure below shows an example of the use of several rungs within a network:



Rules

Remember the following rules when using several rungs:

- Connections are not permitted between rungs.
- Only one jump instruction is permissible per network. The positioning rules for jump instructions remain valid.

Running rungs

Rungs and networks are executed from top to bottom and from left to right. This means that the first instruction in the first rung of the first network is processed first. All instructions of this rung are then processed. After this come all other rungs of the first network. The next network is processed only after all rungs have first been run.

Differences between branches and rungs

The difference between branches and rungs is that the rungs are independent branches that can also stand in a different network. Branches, on the other hand, permit the programming of a parallel connection.

See also

Insert rung (Page 1618)

Deleting a rung (Page 1619)

Insert rung

Requirement

- A block is open.
- A network is available.

Procedure

To insert a new rung in a network, proceed as follows:

1. Insert any coil on the power rail.
   A new rung will be inserted and the coil positioned at the end of the rung.
2. Insert additional instructions in the new rung.

See also

Basic information on rungs in LAD (Page 1617)

Deleting a rung (Page 1619)

Deleting a rung

Requirement

A rung is available.

Procedure

To delete a rung, proceed as follows:

1. Hold down the left mouse button and draw a frame around the rung. At the same time, make sure that you select all instructions. Alternatively, you can hold down the <Shift> key and select the first the last instruction of the rung.
2. Right-click on one of the instructions in the rung.
3. Select the "Delete" command in the shortcut menu.

See also

Basic information on rungs in LAD (Page 1617)

Insert rung (Page 1618)

Creating FBD programs

Basic information on FBD

FBD programming language

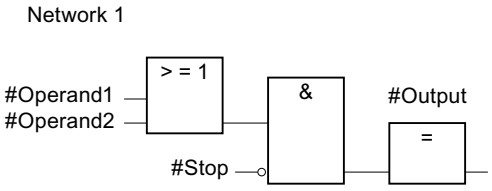
Overview of the Function Block Diagram (FBD) programming language

FBD is a graphical programming language. The representation is based on electronic circuit systems.

The program is mapped in one or more networks. A network contains one or more logic operation paths. The binary signal scans are linked by boxes. The representation of the logic is based on the graphical logic symbols used in Boolean algebra.

Example of networks in FBD

The following figure shows an FBD network with AND and OR boxes and an assignment:



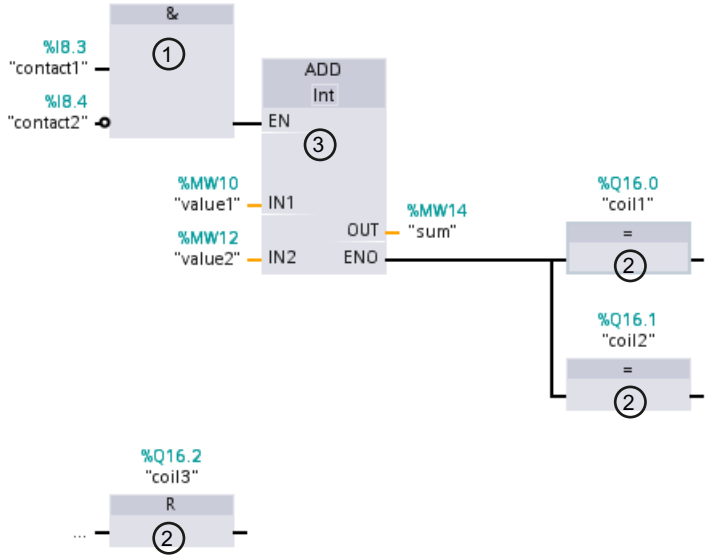
Overview of the FBD elements

FBD elements

An FBD program consists of separate elements that are linked by means of a binary signal flow. Most program elements must be supplied with tags.

A FBD network is programmed from left to right.

For example, the following figure shows elements of an FBD network:



1) Binary function

2) Standard box

3) Complex box

Binary functions

You can use binary functions to query binary operands and to combine their signal states. The following operations are examples of binary functions: "AND operation", "OR operation" and "EXCLUSIVE OR operation".

Standard boxes:

You can use standard boxes to control binary operands, perform RLO edge detection or execute jump functions in the program. Standard boxes generally have only one single input.

Complex boxes

Complex boxes represent program elements with complex functions. The empty box is an exception. You can use the empty box as a placeholder in which you can select the required instruction.

The following types of boxes are available to you in an FBD program:

- Complex boxes without EN/ENO mechanism:
  A box is executed independently of the signal state at the box inputs. The error status of the processing cannot be queried.
- Complex boxes with EN/ENO mechanism:
  A box is only executed if the enable input "EN" has the signal state "1". If the box is processed correctly, the "ENO" enable output has signal state "1". If an error occurs during processing, the "ENO" output is reset.
  If the EN enable input is not interconnected, the box is always executed.

Calls of code block are also shown in the network as complex boxes with EN/ENO mechanism.

Settings for FBD

Overview of the settings for FBD

Overview

The following table shows the settings that you can make:

Group	Setting	Description
Font	Font size	Font size in program editor
View	Layout	Compact or wide Changes the vertical spacing between operands and other objects (such as operand and contact). The change becomes visible once the block is reopened.
	With absolute information	Additional display of the absolute addresses

Group	Setting	Description
Operand field	Maximum width	Maximum number of characters that can be entered horizontally in the operand field. This setting recalculates the layout of the networks.
	Maximum height	Maximum number of characters that can be entered vertically in the operand field. This setting recalculates the layout of the networks.

See also

Changing the settings (Page 1622)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for FBD (Page 1621)

Working with networks

Using networks

Function

The user program is created in the block within networks. For a code block to be programmed, it must contain at least one network. To achieve a better overview of the user program, you can also subdivide your program into several networks.

See also

Inserting network title (Page 1626)

Entering a network comment (Page 1627)

Navigating networks (Page 1628)

Inserting networks

Requirement

A block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Result

A new empty network is inserted into the block.

See also

Inserting network title (Page 1626)

Entering a network comment (Page 1627)

Navigating networks (Page 1628)

Selecting networks

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.
   The first and last networks and all those in between are selected.

See also

Inserting networks (Page 1623)

Inserting network title (Page 1626)

Entering a network comment (Page 1627)

Navigating networks (Page 1628)

Copying and pasting networks

Copied networks can be pasted within the block or in another block. Networks that were created in LAD or FBD can also be inserted in blocks of the respective other programming language.

Requirement

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Inserting networks (Page 1623)

Selecting networks (Page 1623)

Inserting network title (Page 1626)

Entering a network comment (Page 1627)

Navigating networks (Page 1628)

Deleting networks

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting networks (Page 1623)

Selecting networks (Page 1623)

Copying and pasting networks (Page 1624)

Inserting network title (Page 1626)

Entering a network comment (Page 1627)

Navigating networks (Page 1628)

Expanding and collapsing networks

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Inserting networks (Page 1623)

Selecting networks (Page 1623)

Copying and pasting networks (Page 1624)

Deleting networks (Page 1625)

Inserting network title (Page 1626)

Entering a network comment (Page 1627)

Navigating networks (Page 1628)

Inserting network title

The network title is the header of a network. The length of the network title is limited to one line. You can enter the title manually or set it automatically. When you set it automatically, you can do this for individual networks or use the settings to specify that the network title is always set automatically.

For automatic insertion of the network title, the comment of the operand in one of the following instructions in the network is evaluated:

- Assignment
- Set output
- Reset output

The instruction that is listed first in the network is used.

The network title is only inserted automatically when the following conditions are fulfilled:

- The network does not have a title yet.
- The operand of the instruction used for the comment has a comment.

Note

Note the following restrictions for automatic insertion of the network title:

- The network title is not adapted if you change the comment of the operand at a later time.
- The network title is not adapted if you change the operand of the instruction.
- The network title is only set by the writing instructions listed above.
- If the operand is of the data type array, the comment of the array is used and not the comments of the array elements.
- Comments of invalid operands are not taken into consideration.

Entering the network title manually

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

Setting the network title automatically

To specify that the network titles are always set automatically, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Select the "Set network title automatically" check box in the "Additional settings" group.
   The network titles are set automatically as of this time if the conditions listed above are fulfilled.

To set an individual network title automatically, follow these steps:

1. Right-click "Network <Number of the network>" in the title bar of a network.
2. Select the "Set network title automatically" command in the shortcut menu.
   The title of the selected network is set based on the comment of the operand if the conditions listed above are fulfilled.

See also

Using networks (Page 1622)

Inserting networks (Page 1623)

Selecting networks (Page 1623)

Copying and pasting networks (Page 1624)

Deleting networks (Page 1625)

Expanding and collapsing networks (Page 1625)

Entering a network comment (Page 1627)

Navigating networks (Page 1628)

Entering a network comment

You can use network comments to provide comments on the program contents of individual networks. For example, you can indicate the function of the network or draw attention to special characteristics.

Requirement

A network is available.

Procedure

To enter a network comment, follow these steps:

1. Click on the right arrow before the network title.
2. If the comment area is not visible, click "Network comments on/off" in the toolbar.
   The comment area is displayed.

3. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
4. Enter the network comment.

See also

Using networks (Page 1622)

Inserting networks (Page 1623)

Selecting networks (Page 1623)

Copying and pasting networks (Page 1624)

Deleting networks (Page 1625)

Expanding and collapsing networks (Page 1625)

Inserting network title (Page 1626)

Navigating networks (Page 1628)

Navigating networks

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click in the code area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.
   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 1622)

Inserting networks (Page 1623)

Selecting networks (Page 1623)

Copying and pasting networks (Page 1624)

Deleting networks (Page 1625)
Expanding and collapsing networks (Page 1625)
Inserting network title (Page 1626)
Entering a network comment (Page 1627)

Inserting FBD elements

Rules for the use of FBD elements

Rules

Note the following rules when inserting FBD elements:

- An FBD network can consist of several elements. All elements of a logic path must be linked to each other according to IEC 61131-3.
- Standard boxes (flip flops, counters, timers, math operations, etc.) can be added as output to boxes with binary logic operations (for example, AND, OR). Comparison boxes are excluded from this rule.
- Only Boolean inputs in an instruction can be combined with preceding logic operations.
- Only the bottom Boolean output in an instruction can be combined with an additional logic operation.
- Enable input EN or enable output ENO can be connected to boxes, but this is not mandatory.
- Constants (for example, TRUE or FALSE) cannot be assigned to binary logic operations. Instead, use tags of the BOOL data type.
- Only one jump instruction can be inserted in each network.
- Only one jump label can be inserted in each network.
- Instructions for positive or negative RLO edge detection may not be arranged right at the left of the network as this requires a prior logic operation.

Placement rules for S7-1200/1500 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
SET_BF	Set bit field	No
RESET_BF	Reset bit field	No
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
JMP_LIST	Define jump list	No

Instruction		Preceding logic operation required
Mnemonics	**Name**	
SWITCH	Jump distributor	No
RET	Return	No

Placement rules for S7-300/400 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	**Name**	
S	Set output	Yes
R	Reset output	Yes
SP	Start pulse timer	Yes
SE	Start extended pulse timer	Yes
SD	Start on-delay timer	Yes
SS	Start retentive on-delay timer	Yes
SF	Start off-delay timer	Yes
SC	Set counter value	Yes
CU	Count up	Yes
CD	Count down	Yes
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
RET	Return	No
OPN	Open global data block	No
OPNI	Open instance data block	No
CALL	Call block	No
SAVE	Save RLO in BR bit	No
MCRA	Enable MCR range	No
MCRD	Disable MCR range	No
MCR<	Open MCR ranges	No
MCR>	Close MCR ranges	No

Inserting FBD elements using the "Instructions" task card

Requirement

A network is available.

Procedure

To insert FBD elements into a network using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to the FBD element that you want to insert.
3. Use drag-and-drop to move the element to the desired place in the network.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multiple instance", these are located in the block interface in the "Static" section.

Or:

1. Select the point in the network at which you want to insert the element.
2. Open the "Instructions" task card.
3. Double-click on the element you want to insert.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multiple instance", these are located in the block interface in the "Static" section.

Result

The selected FBD element is inserted with dummy entries for the parameters.

See also

Rules for the use of FBD elements (Page 1629)

Inserting FBD elements using an empty box

Requirement

A network is available.

Procedure

To insert FBD elements into a network using an empty box, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Empty box" in the "Basic instructions" palette.
3. Use a drag-and-drop operation to move the "Empty box" element to the desired place in the network.

4. Position the cursor over the triangle in the top right-hand corner of the empty box.
   A drop-down list is displayed.
5. Select the desired FBD element from the drop-down list.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multi-instance", these are located in the block interface in the "Static" section.

Result

The empty box is changed to the respective FBD element. Placeholders are inserted for the parameters.

Selecting the data type of an FBD element

Selecting a data type

Introduction

Some instructions can be executed with several different data types. If you use one of these instructions in the program, you have to specify a valid data type for the instruction at the specific point in the program. For some instructions, you have to select the data types for the inputs and outputs separately.

Note

The valid data type (BOOL) for the tags on the enable input EN and the enable output ENO is predefined by the system and cannot be changed.

The valid data types for an instruction are listed in the instruction drop-down list. You specify the data type of the instruction by selecting an entry from the drop-down list. If the data type of an operand differs from the data type of the instruction and cannot be converted implicitly, the operand is displayed in red and a rollout with the corresponding error message appears.

Data type selection of mathematical instructions

Some mathematical instructions provide you with the option of having the data type automatically set corresponding to the data types of the operand. In the drop-down list for data type selection, these instructions have the entry "Auto" in addition to the actual data types. If you select this entry and then allocate the first operand, the data type of the operand is selected as data type for the instruction. The entry in the drop-down list changes to "Auto (<Data type>)", e.g. "Auto (Real)". If you allocate additional operands, the automatically set data type of the instruction is adjusted according to the following criteria:

- You supply all other operands with tags of the same data type:
  The data type of the instruction is not changed.
- You supply all other operands with tags whose data type is smaller than the data type of the instruction:
  The data type of the instruction is not changed. For the operand with the smaller data type, an implicit conversion is conducted if necessary.
- You supply an additional operand with a tag whose data type is greater than the data type of the instruction:
  The data type of the instruction is changed to the larger data type. An implicit conversion is performed, if necessary, for operands that deviate from the newly set data type of the instruction.

Each change in the data type of an operand can result in a change of the data type of the instruction. Other operands may possibly be implicitly converted as a result. Operands for which an implicit conversion is performed are marked with a gray square.

Note

Please also observe the information on data type conversion for your device and, in particular, the notes on the IEC check.

See also: Data type conversion (Page 2091)

See also

Defining the data type of an instruction (Page 1633)

Defining the data type of an instruction

Introduction

Some instructions can be executed with several different data types. When you insert such instructions into your program, you must specify the data type for these instructions at the actual point in the program.

Specifying the data type by means of the drop-down list

To define the data type of an instruction using the drop-down list, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Click the triangle in the upper corner of the drop-down list.
   The drop-down list will open to display the data types valid for the instruction.
3. Select a data type from the drop-down list.
   The selected data type is displayed.
4. If the instruction has two drop-down lists, select the data type for the instruction inputs in the left-hand drop-down list and the data type for the instruction outputs in the right-hand drop-down list.

Specifying data type by assigning tags

To define the data type of an instruction by assigning tags, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. At an input or output, specify a valid tag, the data type of which is to be applied as the instruction data type.
   The data type of the tag is displayed in the drop-down list.
3. Enter a valid tag at an input and a valid tag at an output if data types need to be defined for both the inputs and outputs of the instruction. The tag specified at the input determines the data type of the inputs; the tag specified at the output determines the data type of the outputs of the instruction.

Automatically specifying the data type of mathematical instructions

To automatically specify the data type for mathematical instructions, follow these steps:

1. Insert the mathematical instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Select the "Auto" entry from the drop-down list.
3. Enter a valid tag at an input or output.
   The data type of the tag is applied as data type of the instruction. The entry in the drop-down list changes to "Auto (<Data type>)".

See also: Selecting a data type (Page 1632)

See also

Selecting a data type (Page 1632)

Using favorites in FBD

Adding FBD elements to Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing FBD elements from Favorites (Page 1636)

Overview of the program editor (Page 1529)

Inserting FBD elements using favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 1529)

Removing FBD elements from Favorites (Page 1636)

Removing FBD elements from Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding FBD elements to Favorites (Page 1635)

Inserting FBD elements using favorites (Page 1635)

Overview of the program editor (Page 1529)

Inserting block calls in FBD

Inserting block calls using a drag-and-drop operation

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree. If you call function blocks from other function blocks, you can either call them as single-instance or multi-instance blocks. If a function block is called as single instance, it will store its data in a data block of its own. If a function block is called as multi-instance, it will store its data in the instance data block of the calling function block.

Requirement

- A network is available.
- The block that is to be called is available.

Inserting a call of a function (FC)

To insert a call of a function (FC) into a network using a drag-and-drop operation, follow these steps:

1. Drag the function from the project tree to the required network.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Drag the function block from the project tree to the required network.
   The "Call options" dialog opens.
2. Enter in the dialog whether you wish to call the block as single or multi-instance.
   - If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block.
   - If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
3. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You can then assign the parameters.

See also: Auto-Hotspot

Note

If when calling a function block you specify an instance data block that does not exist, it will be created. If you have called a function block as a multi-instance, this will be entered as a static tag in the interface.

See also

Updating block calls in FBD (Page 1638)

Changing the instance type (Page 1640)

Single instances (Page 1427)

Multi-instances (Page 1427)

Updating block calls in FBD

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have the following options for updating the block calls:

- Explicit updating of all inconsistent block calls in the program editor.
  The inconsistent block calls within the open block are updated. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are removed if they are not connected.
  - Renamed parameters get the new parameter names.
- Explicit updating of a block call in the program editor.
  The "Interface update" dialog is displayed. In this dialog, you have the option of changing the connection of the operands of the new interface. The inconsistent call of this block is then updated. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are removed if they are not connected.
  - Renamed parameters get the new parameter names.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated. Note that when you call functions (FCs) before the next compilation process, all new formal parameters must be supplied with actual parameters.

Updating all inconsistent block calls in the program editor

To open all block calls in a block, follow these steps:

1. Open the calling block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

Updating a specific block call in the program editor

To update a specific block call in the program editor, follow these steps:

1. Open the calling block in the program editor.
2. Right-click on the block call that you want to update.
3. Select the "Update" command in the shortcut menu.
   The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
4. If necessary, change the connection of the operands. To do this, you have the following options:
   - You can use either a drag-and-drop operation or a cut/copy-and-paste operation to move the operand from the old interface to the new interface.
   - You can delete an operand.
   - You can rename an operand.
   - You can specify a new operand via autocompletion.
5. Click "OK" to update the block call. If you want to cancel the update, click "Cancel".

Note

Note that the "Update block call" command is only available provided you did not previously update all block calls in the editor with the "Update inconsistent block calls" command.

Update block calls during compilation

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Inserting block calls using a drag-and-drop operation (Page 1637)

Changing the instance type (Page 1640)

Changing the block call

You have the option of changing the called block for a block call. But keep in mind that no new instance data blocks are created, for example, when changing from a function (FC) to a function block (FB).

Procedure

To change the called block of a block call, follow these steps:

1. Click on the name of the called block within the block call and press the <F2> key. Or double-click the name of the called block.
   A text box opens, and the name of the currently called block is selected.
2. Enter the name of the block you want to call or select a block in the autocompletion.
3. If you want to call an FB, create a new instance data block, if necessary, and specify it as operand.

Changing the instance type

Instance type

There are two ways of calling function blocks:

- As a single instance
- As a multiple instance

See also: Auto-Hotspot

You can modify a defined instance type at any time.

Requirement

The user program contains a block call.

Procedure

To change the instance type of a function block, follow these steps:

1. Open the code block and select the block call.
2. Select the "Change instance" command in the shortcut menu.
   The "Call options" dialog opens.
3. Click the "Single instance" or "Multi instance" button.
   - If you select the "Single instance" instance type, enter a name for the data block that is to be assigned to the function block.
   - If you select "Multiple instance" as the instance type, enter in the "Name in the interface" text field the name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".

Note

The previous single and multiple instances will not be deleted automatically.

See also

Inserting block calls using a drag-and-drop operation (Page 1637)

Updating block calls in FBD (Page 1638)

Inserting complex FBD instructions

Using the "Calculate" instruction

Requirement

A network is available.

Procedure

Proceed as follows to use the "Calculate" instruction:

1. Open the "Instructions" task card.
2. Navigate to "Math functions > CALCULATE" in the "Basic instructions" pane.
3. Use drag-and-drop to move the element to the desired place in the network.
   The instruction "Calculate" will be inserted for the data type with a placeholder expression and question mark.
4. Enter the data type for the calculation.
5. Enter the operands for the calculation.

   Note

   The calculation is run with the inputs of the "Calculate" instruction. If you want to use constants you must also insert appropriate inputs for them.

6. Click on the "Edit 'Calculate' instruction" button to replace the placeholder expression with the correct expression.
   The "Edit 'Calculate' instruction" dialog will open.
7. Enter the required expression in the "OUT:= " text box.

   Note

   In the "Example" area you can find an example of a valid expression and possible instructions that you can use.

   To determine a value with the help of Pythagoras' theorem, for example, enter "OUT := SQRT (SQR (IN1) + SQR (IN2))".

8. Confirm your entry with "OK".

See also

CALCULATE: Calculate (Page 2575)

Using free-form comments

Basic information on using free comments in FBD

Introduction

Free-form comments allow you to add comments to the source code for graphic programming languages similar to line comments for textual languages.

Free-form comments can be used for all non-binary boxes.

See also

Inserting free-form comments (Page 1642)

Editing free-form comments (Page 1643)

Deleting free-form comments (Page 1644)

Inserting free-form comments

Requirement

A network with instructions is available.

Procedure

To insert a free comment on an instruction, proceed as follows:

1. If necessary, activate the "Free-form comments on/off" button in the toolbar.
2. Right-click on the instruction for which you want to insert a free-form comment.
3. Select the "Insert comment" command in the shortcut menu.
   A comment box with a standard comment opens. The comment box is connected by an arrow to the corresponding instruction.
4. Enter the required comment in the comment box.

See also

Basic information on using free comments in FBD (Page 1642)

Editing free-form comments (Page 1643)

Deleting free-form comments (Page 1644)

Editing free-form comments

Introduction

Free-form comments can be edited as follows:

- Changing the comment text
- Changing the position and size of the comment box
- Attaching a comment to another element
- Showing and hiding free comments

Changing the comment text

To change the text of free-form comments, follow these steps:

1. Click on the comment box.
2. Enter the desired text.

Changing the position of the comment box

To change the positioning of the comment box, follow the steps below:

1. Left-click the comment box and keep the mouse button pressed.
2. Drag the comment box to the desired location.

Changing the size of the comment box

To change the size of the comment box, follow the steps below:

1. Click on the comment box.
2. Drag the comment box on the move handle in the lower right corner to the desired size.

Attaching a comment to another element

To attach a free-form comment to another element, follow these steps:

1. Left-click the point of the arrow that links the comment box with the instruction and keep the mouse button pressed.
2. Drag the arrow to the element to which you want to attach the comment. Possible insertion points are marked with a green square.
3. Release the mouse button.

Showing and hiding free comments

To show or hide a free-form comments, follow these steps:

1. Click the "Free-form comment on/off" button in the toolbar.

See also

Basic information on using free comments in FBD (Page 1642)
Inserting free-form comments (Page 1642)
Deleting free-form comments (Page 1644)

Deleting free-form comments

Procedure

To delete a free-form comment, proceed as follows:

1. Right-click on the free-form comment that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on using free comments in FBD (Page 1642)
Inserting free-form comments (Page 1642)
Editing free-form comments (Page 1643)

Editing FBD elements

Selecting FBD elements

You can select several individual elements or all elements in a network.

Requirement

FBD elements are available

Selecting several individual FBD elements

To select several individual FBD elements, follow these steps:

1. Press and hold down the <Ctrl> key.
2. Click on all the FBD elements you wish to select.
3. Now release the <Ctrl> key.

Selecting all FBD elements in a network

To select all FBD elements in a network, follow these steps:

1. Go to the network whose elements you wish to select.
2. Select the "Select all" command in the "Edit" menu or press <Ctrl+A>.

See also

Copying FBD elements (Page 1645)
Cutting FBD elements (Page 1646)
Pasting an FBD element from the clipboard (Page 1646)
Replacing FBD elements (Page 1647)
Adding additional inputs and outputs to FBD elements (Page 1648)
Removing instruction inputs and outputs (Page 1649)
Enabling and disabling the EN/ENO mechanism (Page 1650)
Deleting FBD elements (Page 1651)

Copying FBD elements

Requirement

An FBD element is available.

Procedure

To copy an FBD element, follow these steps:

1. Right-click the FBD element that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

The FBD element will be copied and saved to the clipboard.

See also

Selecting FBD elements (Page 1644)
Cutting FBD elements (Page 1646)
Pasting an FBD element from the clipboard (Page 1646)
Replacing FBD elements (Page 1647)
Adding additional inputs and outputs to FBD elements (Page 1648)
Removing instruction inputs and outputs (Page 1649)
Enabling and disabling the EN/ENO mechanism (Page 1650)
Deleting FBD elements (Page 1651)

Cutting FBD elements

Requirement

An FBD element is available.

Cutting

To cut an FBD element, follow these steps:

1. Right-click the FBD element that you want to cut.
2. Select "Cut" in the shortcut menu.

Result

The FBD element will be cut and saved to the clipboard.

See also

Selecting FBD elements (Page 1644)
Copying FBD elements (Page 1645)
Pasting an FBD element from the clipboard (Page 1646)
Replacing FBD elements (Page 1647)
Adding additional inputs and outputs to FBD elements (Page 1648)
Removing instruction inputs and outputs (Page 1649)
Enabling and disabling the EN/ENO mechanism (Page 1650)
Deleting FBD elements (Page 1651)

Pasting an FBD element from the clipboard

Requirement

An FBD element is available.

Procedure

To paste an FBD element from the clipboard, follow these steps:

1. Copy an FBD element or cut an FBD element.
2. Right-click the point in the network where you want to paste the element.
3. Select "Paste" in the shortcut menu.

See also

Selecting FBD elements (Page 1644)

Copying FBD elements (Page 1645)

Cutting FBD elements (Page 1646)

Replacing FBD elements (Page 1647)

Adding additional inputs and outputs to FBD elements (Page 1648)

Removing instruction inputs and outputs (Page 1649)

Enabling and disabling the EN/ENO mechanism (Page 1650)

Deleting FBD elements (Page 1651)

Replacing FBD elements

You can easily exchange FBD elements with other FBD elements of the same type. This has the advantage that the parameters are retained and need not be entered again. For example, you can exchange OR and AND, RS-FlipFlop and SR-FlipFlop, comparison functions or jump instructions.

Requirements

A network with at least one FBD element is present.

Procedure

To replace an FBD element with another FBD element, follow these steps:

1. Select the FBD element that you want to replace.
   If elements compatible with the selected FBD element are available, a triangle will appear in the upper right-hand corner of the element.
2. Position the cursor above the triangle of the FBD element.
   A drop-down list is displayed.
3. From the drop-down list, select the FBD element that you want to use to replace the existing FBD element.

See also

Selecting FBD elements (Page 1644)

Copying FBD elements (Page 1645)

Cutting FBD elements (Page 1646)

Pasting an FBD element from the clipboard (Page 1646)

Adding additional inputs and outputs to FBD elements (Page 1648)

Removing instruction inputs and outputs (Page 1649)

Enabling and disabling the EN/ENO mechanism (Page 1650)

Deleting FBD elements (Page 1651)

Adding additional inputs and outputs to FBD elements

Introduction

You can expand several FBD elements with additional inputs that execute arithmetic or binary operations. Such elements are, for example, the instructions "Add" (ADD), "Multiply" (MUL), AND or OR. You can expand the "MOVE value" (MOVE) and "Demultiplex" (DEMUX) instruction boxes by adding additional outputs.

The name of the new inputs and outputs is comprised of the type of inserted element and a consecutive number. The name of a new input is may be "IN2"; the name of a new output may be "OUT2".

Requirements

An FBD element is available that permits the insertion of additional inputs and outputs.

Inserting an additional input

To add an additional input to the box of an FBD element, follow these steps:

1. Right-click on an existing input of the FBD element.
2. Select "Insert input" in the shortcut menu.
   An additional input is added to the box of the FBD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.
   An additional input is added to the box of the FBD element.

Inserting an additional output

To add an additional output to the box of an FBD element, follow these steps:

1. Right-click on an existing output of the FBD element.
2. Select "Insert output" from the shortcut menu.
   An additional output is added to the box of the FBD element.

Or:

1. Click on the yellow star symbol beside the last output of the instruction box.
   An additional output is added to the box of the FBD element.

See also

Selecting FBD elements (Page 1644)

Copying FBD elements (Page 1645)

Cutting FBD elements (Page 1646)
Pasting an FBD element from the clipboard (Page 1646)
Replacing FBD elements (Page 1647)
Removing instruction inputs and outputs (Page 1649)
Enabling and disabling the EN/ENO mechanism (Page 1650)
Deleting FBD elements (Page 1651)

Removing instruction inputs and outputs

Introduction

Inputs and outputs which you have added to an instruction can be removed.

Requirement

An FBD element is available, which you have expanded with additional inputs or outputs.

Remove input

To remove an input, follow these steps:

1. Select the input that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The input of the FBD element is removed.

Remove output

To remove an output, follow these steps:

1. Select the output that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The output of the FBD element will be removed.

See also

Selecting FBD elements (Page 1644)
Copying FBD elements (Page 1645)
Cutting FBD elements (Page 1646)
Pasting an FBD element from the clipboard (Page 1646)
Replacing FBD elements (Page 1647)
Adding additional inputs and outputs to FBD elements (Page 1648)

Enabling and disabling the EN/ENO mechanism (Page 1650)

Deleting FBD elements (Page 1651)

Enabling and disabling the EN/ENO mechanism

In LAD and FBD, certain instructions have an enable output ENO and thus use the EN/ENO mechanism. This allows you to query runtime errors in instructions and react to them. In order to increase the performance of the CPU, the EN/ENO mechanism is disabled in the default setting. This means that you are not initially able to react to runtime errors of the instruction using the ENO value. However, you can enable the EN/ENO mechanism again at any time, if required.

You can enable the EN/ENO mechanism individually for each instruction in order to generate the ENO. If you enable the EN/ENO mechanism for an instruction, other instructions that you subsequently add to your program are also inserted with the EN/ENO mechanism enabled. You can disable the EN/ENO mechanism again at any time if you do not want to use the evaluation of ENO for an instruction. Further instructions that you subsequently add to your program will then be inserted without the EN/ENO mechanism.

See also: Basics of the EN/ENO mechanism (Page 1473)

Activating the EN/ENO mechanism

Proceed as follows to activate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to activate the EN/ENO mechanism.
2. Select the "Generate ENO" command from the shortcut menu.
   The ENO value is again generated for the instruction. Other instructions are inserted with the enable output.

Deactivating the EN/ENO mechanism

Proceed as follows to deactivate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to deactivate the EN/ENO mechanism.
2. Select the "Do not generate ENO" command from the shortcut menu.
   The ENO value is no longer generated for the instruction. Other instructions are inserted without enable output.

See also

Selecting FBD elements (Page 1644)

Copying FBD elements (Page 1645)

Cutting FBD elements (Page 1646)

Pasting an FBD element from the clipboard (Page 1646)

Replacing FBD elements (Page 1647)

Adding additional inputs and outputs to FBD elements (Page 1648)
Removing instruction inputs and outputs (Page 1649)
Deleting FBD elements (Page 1651)

Deleting FBD elements

Requirement

An FBD element is available.

Procedure

To delete an FBD element, follow these steps:

1. Right-click the FBD element that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Selecting FBD elements (Page 1644)
Copying FBD elements (Page 1645)
Cutting FBD elements (Page 1646)
Pasting an FBD element from the clipboard (Page 1646)
Replacing FBD elements (Page 1647)
Adding additional inputs and outputs to FBD elements (Page 1648)
Removing instruction inputs and outputs (Page 1649)
Enabling and disabling the EN/ENO mechanism (Page 1650)

Inserting operands in FBD instructions

Inserting operands

The character strings "<???>", "<??.?>" and "..." are inserted as placeholders for the parameters when a FBD element is inserted. The "<???>" and "<??.?>" strings displayed in red indicate parameters that need to be connected. The "..." string displayed in black indicates parameters that may be connected. "<??.?>" stands for Boolean placeholders.

Note

To display the available data types in a tooltip, move the cursor over the placeholder.

Requirement

An FBD element is available.

Procedure

To connect the parameters of an FBD element, follow these steps:

1. Click the placeholder of the parameter.
   An input field is opened.
2. Enter the corresponding parameters, for example a PLC tag, a local tag or a constant.

   Note

   If you enter the absolute address of a parameter that has already been defined, this absolute address will be changed to the symbolic name of the parameter as soon as the input is confirmed. If you have not yet defined the parameter, a new tag with this absolute address and the default name "Tag_1" will be entered in the PLC tag table. When you confirm your input, the absolute address will be replaced with the symbolic name "Tag_1".

3. Confirm the parameter with the Enter key.
4. If you have not yet defined the parameter, you can define it directly in the program editor using the shortcut menu.
   See also: "Declaring PLC tags in the program editor (Page 1487)".

Or drag from it the PLC tag table:

1. In the project tree, select the "PLC tags" folder and open the PLC tag table.
2. If you have opened the PLC tag table, drag the desired tag to the corresponding location in your program. If you have not opened the PLC tag table yet, open the detail view now. Drag the desired tag from the detail view to the appropriate place in your program.

Or drag from it the block interface:

1. Open the block interface.
2. Drag the desired operand from the block interface to the corresponding location in your program.

Result

- If the syntax is error-free, the displayed parameter is black.
- If there is an error in the syntax, the cursor stays in the input field and a corresponding error message is displayed in the inspector window in the "Info > Syntax" register.

Wiring hidden parameters

Introduction

Depending on the CPU used, you can use complex instructions in your program that are included with the TIA Portal. These instructions can contain parameters that are declared as hidden.

If an instruction contains hidden parameters, the instruction box has a small arrow on the lower edge. You can recognize hidden parameters by their white font.

You can show and wire the hidden parameters at any time.

Showing or hiding hidden parameters

To show or hide hidden parameters, follow these steps:

1. Click on the down arrow at the bottom edge of the instruction box to show hidden parameters.
2. Click on the up arrow at the bottom edge of the instruction box to hide hidden parameters.

Wiring hidden parameters

To wire parameters, follow these steps:

1. Wire the hidden parameters like normally visible parameters.
   The hidden parameter is transformed into a visible parameter.

See also

Using libraries (Page 470)

Displaying or hiding variable information

Introduction

You can display the following information about the tags to be used in the Program editor:

- Name of the tag
- Address of the tag
- Simple or hierarchical comments for tag documentation

The information is taken from the block interface for local tags and DB tags and from the PLC tag table for tags that are valid CPU-wide.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all the blocks currently opened and opened in future is shown.

You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual blocks that are open.

When you select the display of tag information with hierarchical comments, the comments of the higher structure levels are also displayed for structured tags. The comments are shown in brackets after the tag comment; the comments of the individual levels are separated by a dot. If there is no comment for a tag on a structure level, it is not displayed. This is indicated by two successive dots.

Displaying or hiding tag information for all the blocks

Proceed as follows to display or hide the tag information for all the blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to display the tag information, select the "Show" option or the "Tag information with hierarchical comments" option in the drop-down list, depending on whether you want to display simple or hierarchical comments.
4. If you want to hide the tag information, select the option "Collapse" in the "Tag information" drop-down list.
   The tag information is displayed or hidden for all the blocks. When you open additional blocks, the tag information is displayed or hidden depending on the selected setting.

Displaying or hiding tag information for an opened block

Proceed as follows to display or hide the tag information for an opened block:

1. If you want to display the tag information, select the "Show tag information" option or the "Tag information with hierarchical comments" option in the "Shows the tag information" drop-down list, depending on whether you want to display simple or hierarchical comments.
2. If you want to hide the tag information, select the option "Hide tag information" in the "Shows the tag information" drop-down list.
   The tag information is displayed or hidden.

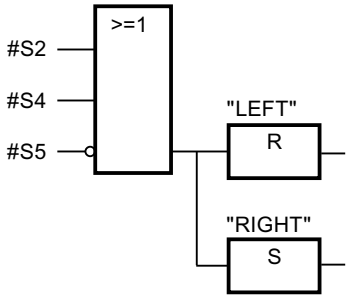
Branches in FBD

Basic information on branches in FBD

Definition

You can use the Function Block Diagram (FBD) programming language to program parallel branches. This is done using branches that are inserted between the boxes. You can insert additional boxes within the branch and in this way build up complex function block diagrams.

The figure below shows an example of the use of branches:



See also

Rules for branches in FBD (Page 1655)

Inserting branches in FBD networks (Page 1655)

Deleting branches in FBD networks (Page 1656)

Rules for branches in FBD

Rules

The following rules apply to the use of branches in FBD:

- Branches are opened downward.
- Branches can be inserted only between FBD elements.
- To delete a branch, you must delete all FBD elements, including the branch itself.
- If you delete the connection between two branches, the FBD elements of the interrupted branch will be positioned freely in the network.

See also

Basic information on branches in FBD (Page 1654)

Inserting branches in FBD networks (Page 1655)

Deleting branches in FBD networks (Page 1656)

Inserting branches in FBD networks

Requirement

A network is available.

Procedure

To insert a new branch in a network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Branch" in the "Basic instructions" palette.
3. Drag the element from the "Elements" pane to the a required location on a connection line between two boxes.

See also

Rules for branches in FBD (Page 1655)

Basic information on branches in FBD (Page 1654)

Deleting branches in FBD networks (Page 1656)

Deleting branches in FBD networks

Requirement

A branch is available.

Procedure

To delete a branch, follow these steps:

1. Select the connection line that links the branch to the main branch.
2. Select the "Delete" command in the shortcut menu.

Result

The branch is now deleted. Boxes connected to the deleted branch are placed freely within the network.

See also

Rules for branches in FBD (Page 1655)

Basic information on branches in FBD (Page 1654)

Inserting branches in FBD networks (Page 1655)

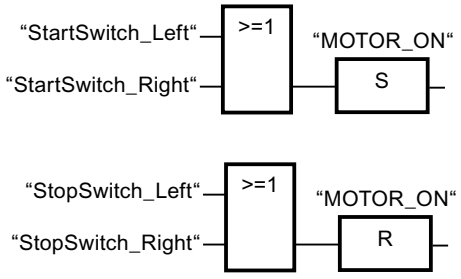
Logic paths in FBD

Basic information on logic paths in FBD

Use of logic paths

The user program will be mapped in one or more networks. The networks can contain one or more logic paths on which the binary signals are arranged in the form of boxes.

The following figure shows an example of the use of several logic paths within a network:



Rules

Remember the following rules when using logic paths:

- Connections are not permitted between logic paths.
- Only one jump instruction is permissible per network. The positioning rules for jump instructions remain valid.

Executing logic paths

Logic paths are executed from top to bottom and from left to right. This means that the first instruction in the first logic path of the first network is executed first. All instructions of this logic path are then executed. After this come all other logic paths of the first network. The next network is executed only after all logic paths have first been executed.

When jumps are used the regular execution of the logic paths is circumvented and the instruction is executed at the jump destination.

Differences between branches and logic paths

The difference between branches and logic paths is that the logic paths are independent branches that can also stand in a different network. Branches, on the other hand, permit the programming of a parallel connection and have a common preceding logic operation.

See also

Inserting a logic path (Page 1658)

Deleting a logic operation path (Page 1658)

Inserting a logic path

Requirement

- A block is open.
- A network is available.

Procedure

To insert a new logic path in a network, follow these steps:

1. Insert any instruction in a network in such a way that it has no connection to existing instructions.
   A new logic path is inserted.
2. Insert an assignment at the end of the new logic path.
3. Insert additional instructions in the new logic path.

See also

Basic information on logic paths in FBD (Page 1657)

Deleting a logic operation path (Page 1658)

Deleting a logic operation path

Requirement

A logic path is available.

Procedure

To delete a logic path, proceed as follows:

1. Hold down the left mouse button and draw a frame around the logic path. At the same time, make sure that you select all instructions of the logic path. Alternatively, you can hold down the <Shift> key and select the first the last instruction of the logic path.
2. Right-click on one of the instructions in the logic path.
3. Select the "Delete" command in the shortcut menu.

See also

Basic information on logic paths in FBD (Page 1657)

Inserting a logic path (Page 1658)

Creating SCL programs

Basics of SCL

Programming language SCL

Programming language SCL

SCL (Structured Control Language) is a high-level programming language based on PASCAL. The language is based on DIN EN 61131-3 (international IEC 1131-3).

The standard standardizes programming languages for programmable logic controllers. The SCL programming language fulfills the PLCopen Basis Level of ST language (Structured Text) defined in this standard.

Language elements

SCL also contains higher programming languages in addition to the typical elements of the PLC, such as inputs, outputs, timers or memory bits.

- Expressions
- Value assignments
- Operators

Program control

SCL provides convenient instructions for controlling the program allowing you, for example, to create program branches, loops or jumps.

Application

SCL is therefore particularly suitable for the following areas of application:

- Data management
- Process optimization
- Recipe management
- Mathematical / statistical tasks

Expressions

Description

Expressions are calculated during the runtime of the program and return a value. An expression consists of operands (such as constants, tags or function calls) and optionally out of operators (such as *, /, + or -). Expressions can be linked together or nested within each other by operators.

Evaluation order

The evaluation of the expression occurs in a specific order that is defined by the following factors:

- Priority of the operators involved
- Left-to-right order
- Brackets

Types of expressions

The following expression types are available depending on the operator:

- Arithmetic expressions
  Arithmetic expressions consist of either a numerical value or combine two values or expressions with arithmetic operators.
- Relational expressions
  Relational expressions compare the values of two operands and yield a Boolean value. The result is TRUE if the comparison is true, and FALSE if it is not met.
- Logical expressions
  Logical expressions combine two operands with logical operators (AND, OR, XOR) or negating operands (NOT).

How expressions are used

You can use the result of an expression in different ways:

- As a value assignment for a tag
- As as a condition for a control instruction
- As a parameter for a calling a block or instruction

See also

Operators and operator precedence (Page 1665)

Arithmetic expressions (Page 1660)

Relational expressions (Page 1663)

Logical expressions (Page 1665)

Arithmetic expressions

Description

Arithmetic expressions consist of either a numerical value or combine two values or expressions with arithmetic operators.

Arithmetic operators can process the data types that are allowed in the CPU in use. If two operands are involved in the operation, the data type of the result is determined based on the following criteria:

- If both operands are integers with sign and have different lengths, the result receives the data type of the longer integer (e. g. INT + DINT = DINT).
- If both operands are integers without sign and have different lengths, the result receives the data type of the longer integer (e. g. USINT + UDINT = UDINT).
- If one operand is an integer with sign and the other integer is an operand without sign, the result receives the next larger data type with sign that covers the integer without sign (e. g. SINT + USINT = INT).
  You can only execute an operation with such operands if the IEC check is not set.
- If one operand is an integer and the other operand is a floating-point number, the result receives the data type of the floating-point number (e. g. INT + REAL = REAL).
- If both operands are floating-point numbers and have different lengths, the result receives the data type of the longer floating-point number (e. g. REAL + LREAL = LREAL).
- The data type of the result of an operation that involves operands of the data type groups "Times" and "Date and time" can be found in the table in section "Data types of arithmetic expressions".
  You cannot use data types of the data type groups "Times" and "Date and time" when the IEC check is set.

Data types of arithmetic expressions

The following table shows the data types you can use in arithmetic expressions:

Operation	Operator	1st Operand	2nd Operand	Result
Power	**	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
Unary plus	+	Integer/floating-point number TIME, LTIME	-	Integer/floating-point number TIME, LTIME
Unary minus	-	Integer/floating-point number TIME, LTIME	-	Integer/floating-point number TIME, LTIME
Multiplication	*	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME, LTIME	Integer	TIME, LTIME
Division	/	Integer/floating-point number	Integer/floating-point number (not equal 0)	Integer/floating-point number
		TIME, LTIME	Integer	TIME, LTIME
Modulo function	MOD	Integer	Integer	Integer

Operation	Operator	1st Operand	2nd Operand	Result
Addition	+	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME	TIME	TIME
		TIME	DINT	TIME
		LTIME	TIME, LTIME	LTIME
		LTIME	LINT	LTIME
		TOD	TIME	TOD
		TOD	DINT	TOD
		LTOD	TIME, LTIME	LTOD
		LTOD	LINT	LTOD
		DATE	LTOD	DTL
		DATE	TOD	• S7-300/400: DT • S7-1200/1500: DTL
		DT	TIME	DT
		LDT	TIME, LTIME	LDT
		DTL	TIME, LTIME	DTL
Subtraction	-	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME	TIME	TIME
		TIME	DINT	TIME
		LTIME [1)]	TIME, LTIME	LTIME
		LTIME	LINT	LTIME
		TOD	TIME	TOD
			DINT	TOD
		LTOD	TIME, LTIME	LTOD
		LTOD	LINT	LTOD
		DATE	DATE	• S7-300/400/1200: TIME • S7-1500: LTIME
		DT	TIME	DT
		LDT	TIME, LTIME	LDT
		DTL	TIME, LTIME	DTL
		DTL	DTL	• S7-1200: TIME • S7-1500: LTIME
[1)] Combinations between nanoseconds and milliseconds are not possible within expressions.				

For additional information on valid data types, refer to "See also".

Example

The following example shows an arithmetic expression:

```
SCL
"MyTag1":= "MyTag2" * "MyTag3";
```

See also

Expressions (Page 1659)

Operators and operator precedence (Page 1665)

Relational expressions

Description

Relational expressions compare the values of two operands and yield a Boolean value. The result is TRUE if the comparison is true, and FALSE if it is not met.

Relational operators can process the data types that are allowed in the CPU in use. The data type of the result always is BOOL.

Note the following rules when forming relational expressions:

- All tags are comparable within the following data type groups:
  - Integers/floating-point numbers
  - Binary numbers
  - String
- With the following data types/data groups, only tags of the same type can be compared:
  - TIME, LTIME
  - Date and time
- The comparison of strings is based on the ASCII character set. The length of the tags and the numerical value of each ASCII character are used for the comparison.
- S5TIME tags are not permitted as comparison operands. An explicit conversion from S5TIME to TIME or LTIME is necessary.

Data types of relational expressions

The following table shows the data types/data type groups you can use in relational expressions:

Operation	Operator	1st Operand	2nd Operand	Result
Compare for equal, not equal	=, <>	Integer/floating-point number	Integer/floating-point number	BOOL
		Binary number	Binary number	BOOL
		String	String	BOOL
		TIME, LTIME	TIME, LTIME	BOOL
		Date and time	Date and time	BOOL
Compare for less than, less than-equal to, greater than, greater than or equal to	<, <=, >, >=	Integer/floating-point number	Integer/floating-point number	BOOL
		Bit strings (S7-1200/1500 only)	Bit strings (S7-1200/1500 only)	BOOL
		String	String	BOOL
		TIME, LTIME	TIME, LTIME	BOOL
		Date and time	Date and time	BOOL

For additional information on valid data types, refer to "See also".

Example

The following example shows a relational expression:

SCL

```
IF a > b THEN c:= a;
IF A > 20 AND B < 20 THEN C:= TRUE;
IF A<>(B AND C) THEN C:= FALSE;
```

Note

The comparison for STRING and DT are executed internally in the S7-300/400 by extended instructions. The following operands are not permitted for these functions:

- Parameter of a FC
- In-out parameter of an FB of type STRUCT or ARRAY

See also

Expressions (Page 1659)

Operators and operator precedence (Page 1665)

Logical expressions

Description

Logical expressions combine two operands with logical operators AND OR XOR or negating operands NOT.

Logical operators can process the data types that are allowed in the CPU in use. The result of a logical expression is of BOOL data type, if both operands are of BOOL data type. If at least one of both operands is a bit string, then the result is also a bit string and is determined by the type of the highest operand. For example, when you link a BYTE type operand to a WORD type operand, the result is type WORD.

To link a BOOL type operand with a bit string, you must first explicitly convert it to a bit string.

Data types of logical expressions

The following table shows the data types you can use in logical expressions:

Operation	Operator	1st Operand	2nd Operand	Result
Negation	NOT	BOOL	-	BOOL
AND logic operation	AND or &	BOOL	BOOL	BOOL
		Bit string	Bit string	Bit string
OR logic operation	OR	BOOL	BOOL	BOOL
		Bit string	Bit string	Bit string
EXCLUSIVE OR logic operation	XOR	BOOL	BOOL	BOOL
		Bit string	Bit string	Bit string

Example

The following example shows a logical expression:

```
SCL
IF "MyTag1" AND NOT "MyTag2" THEN c:=a;
MyTag:=ALPHA OR BETA;
```

See also

Expressions (Page 1659)

Operators and operator precedence (Page 1665)

Operators and operator precedence

Operators and their order of evaluation

Expressions can be linked together or nested within each other by operators.

The order of evaluation for expressions depends on the precedence of operators and brackets. The following basic rules apply:

- Arithmetic operators are evaluated before relational operators and relational operators are evaluated before logical operators.
- Operators with no precedence are evaluated according to their occurrence from left to right.
- Operations in brackets are evaluated first.

The following table provides an overview of the operators and their precedence:

Operator	Operation	Precedence
Arithmetic expressions		
+ (Page 1660)	Unary plus	2
- (Page 1660)	Unary minus	2
** (Page 1660)	Power	3
* (Page 1660)	Multiplication	4
/ (Page 1660)	Division	4
MOD (Page 1660)	Modulo function	4
+ (Page 1660)	Addition	5
- (Page 1660)	Subtraction	5
Relational expressions		
< (Page 1663)	Less than	6
> (Page 1663)	Greater than	6
<= (Page 1663)	Less than or equal	6
>= (Page 1663)	Greater than or equal	6
= (Page 1663)	Equal	7
<> (Page 1663)	Not equal	7
Logical expressions		
NOT (Page 1665)	Negation	3
AND (Page 1665) or & (Page 1665)	Boolean AND	8
XOR (Page 1665)	Exclusive OR	9
OR (Page 1665)	Boolean OR	10
Miscellaneous operations		
( ) (Page 1659)	Brackets	1
:= (Page 1666)	Assignment	11

Value assignments

Definition

You can use a value assignment to assign the value of an expression to a tag. On the left side of the assignment is the tag that takes the value of the expression on the right.

The name of a function can also be specified as an expression. The function is called by the value assignment and returns its function value to the tag on the left.

The data type of value assignment is defined by the data type of the tag on the left. The data type of the expression on the right must match this type.

For additional information on compatibility and conversion of data types, refer to "See also".

Value assignments for STRUCT data type or PLC data types

An entire structure can be assigned to another if the structures are identically organized and the data types as well as the names of the structural components match.

You can assign a tag, an expression or another structural element to an individual structural element.

Value assignments for the ARRAY data type

An entire ARRAY can be assigned to another ARRAY if both the data types of the ARRAY elements as well as the ARRAY limits match.

You can assign a tag, an expression or another ARRAY element to an individual ARRAY element.

Value assignments for the STRING data type

An entire STRING can be assigned to another STRING.

You can assign another STRING element to an individual STRING element.

Value assignment for data type WSTRING (S7-1200/1500)

An entire WSTRING can be assigned to another WSTRING.

You can assign another WSTRING element to an individual WSTRING element.

Value assignments for the ANY data type

You can assign a tag with the ANY data type only to the following objects:

- Input parameters or temporary local data of FBs that also have the data type ANY.
- Temporary local data of FCs that also have the data type ANY.

Note that you can only point to memory areas with "standard" access mode with the ANY pointer.

Value assignments for the POINTER data type

You cannot use POINTER in value assignments in SCL.

Examples

The following table shows examples for value assignments:

SCL	
`"MyTag1" := "MyTag2";`	`(* Assignment of a tag*)`
`"MyTag1" := "MyTag2" * "MyTag3";`	`(* Assignment of an expression*)`
`"MyTag" := "MyFC"();`	`(* Call for a function that assigns its function value to the "MyTag" tag*)`
`#MyStruct.MyStructElement := "MyTag";`	`(* Assignment of a tag to a structure element*)`
`#MyArray[2] := "MyTag";`	`(* Assignment of a tag to an ARRAY element*)`
`"MyTag" := #MyArray[1,4];`	`(* Assignment of an ARRAY element to a tag*)`
`#MyString[2] := #MyOtherString[5];`	`(* Assignment of a STRING element to another STRING element*)`

See also

Operators and operator precedence (Page 1665)

Settings for SCL

Overview of the settings for SCL

Overview

The following tables show the settings you can make for SCL:

Editor settings

Group	Setting	Description
View	Keyword highlighting	Notation used to represent the keywords of the programming language. You can choose between uppercase and lowercase letters or a notation corresponding to the conventions of the Pascal programming language.

Default settings for new blocks

If you create new blocks, the following settings are set as default values. You can change these in the block properties at a later point in time.

Group	Setting	Description
Compile	Create extended status information	Allows all tags in a block to be monitored. The memory requirements of the program and execution times increase, however, with this option.
	Check ARRAY limits[1)]	Checks at runtime whether array indices are within the declared range for an ARRAY. If an array index exceeds the permissible range, the enable output ENO of the block is set to "0".
	Set ENO automatically	Checks at runtime whether errors occur in the processing of certain instructions. If a runtime error occurs, the enable output ENO of the block is set to "0".
[1)]For CPUs of the S7-300/400 series: When the ARRAY limits are violated, the enable output ENO is set to FALSE. For CPUs of the S7-1200/1500 series: When the ARRAY limits are violated, the enable output ENO is not set to FALSE. See "Addressing structured variables (Page 1459)" for error query options.		

See also

Changing the settings (Page 1669)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for SCL (Page 1668)

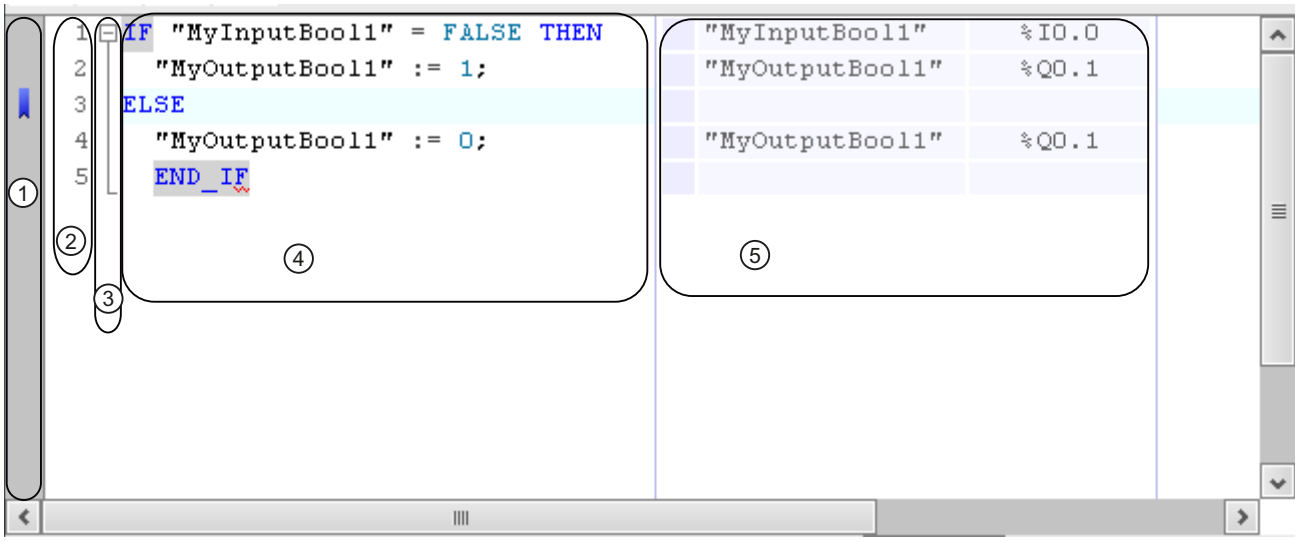
The programming window of SCL

Overview of the programming window

Function

The programming window is the work area, where you enter the SCL program.

The following figure shows the programming window of SCL:



The programming window consists of the following sections:

Section	Meaning
① Sidebar	You can set bookmarks and breakpoints in the sidebar.
② Line numbers	The line numbers are displayed to the left of the program code.
③ Outline view	The outline view highlights related code sections.
④ Code area	You edit the SCL program in the code area.
⑤ Display of the absolute operands	This table shows the assignment of symbolic operands to absolute addresses.

See also

Customizing the programming window (Page 1671)

Formatting SCL code (Page 1672)

Expanding and collapsing sections of code (Page 1673)

Customizing the programming window

Introduction

You can customize the appearance of the programming window and the program code in the following way:

- By setting the font, size and color
- By setting the tab spacing
- By displaying the line numbers
- By showing or hiding the absolute operands

Setting the font, size and color

To set the font, size and color, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Script/text editors" group.
3. Select the desired font and font size or choose a font color for the individual language elements.

Setting the tab spacing

To provide a better overview of the program, lines are indented according to syntax. Define the depth of indentation with the tab spacing.

To set the tab spacing, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Script/text editors" group.
3. Set the tab spacing.

Show line numbers

To display the line numbers, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Script/text editors" group.
3. Select the "Show line numbers" option.

Show or hide the absolute operands

You can show the assignment of symbolic and absolute operands in a table next to the program code, if required.

To hide or show the display of the absolute operands, follow these steps:

1. Click the "Absolute/symbolic operands" icon in the toolbar.
   The display of the absolute operands appears.
2. To move the display, click the table and drag it to the desired position while holding down the mouse button.
3. To change the width of the table, click on the right or left table border and drag it to the right or left while holding down the mouse button.

See also

Overview of the programming window (Page 1670)

Formatting SCL code (Page 1672)

Expanding and collapsing sections of code (Page 1673)

Formatting SCL code

Introduction

To make the program clearer, you can indent or outdent individual lines manually or format code sections. Note the following information about formatting code sections:

- The type of formatting is based on the general settings for indents, but at least the line or the section is always indented. If you selected the setting "Smart", unnecessary spaces within the SCL instruction are also removed.
- Only syntactically correct code sections can be formatted.
- If you place the insertion point in the first or last line of an instruction for program control, for example in an IF instruction in the line with the "IF", the entire instruction is formatted.
- If you select text, only the selected text is formatted.

Indenting or outdenting lines

To indent or outdent individual lines, follow these steps:

1. Click on the line you want to indent or outdent.
2. Press the "Indent text", "Outdent text" button into the toolbar of the programming editor.

Note

You can set the width of the indent in "Options > Settings".

Formatting code sections

To format code sections, follow these steps:

1. Select the text that you want to format or place the insertion point in the appropriate line.
2. Select the "Format selected text automatically" button into the toolbar of the programming editor.

See also

Overview of the programming window (Page 1670)

Customizing the programming window (Page 1671)

Expanding and collapsing sections of code (Page 1673)

Overview of the script and text editor settings (Page 304)

Expanding and collapsing sections of code

Introduction

SCL instructions can span several lines. Examples for this are program control instructions or block calls.

Such instructions that belong together are identified as follows:

- An outline view between the display line number and the program code marks the entire code section.
- When you select the opening keyword, the closing keyword is automatically highlighted.

To improve clarity, you can expand or collapse sections of code that belong together in the outline display. The selected outline display is retained when you close the block or the project so that, the next time you open the block, the sections of code are displayed in exactly the same way as they were when you closed it.

Procedure

To expand or collapse the code section, follow these steps:

1. Click the minus sign in the outline view.
   The code section closes.
2. Click the plus sign in the outline view.
   The code section opens.

See also

Overview of the programming window (Page 1670)

Customizing the programming window (Page 1671)

Formatting SCL code (Page 1672)

Using bookmarks

Basics of bookmarks

Function

You can use bookmarks to mark program locations in extensive programs so that you can find them quickly later if they need revising. Bookmarks are displayed in the sidebar of the programming window. You can navigate between multiple bookmarks within a block using menu commands.

Bookmarks are saved with the project and are therefore available for anyone who wants to edit the block. However, they are not loaded to a device.

Bookmarks are not evaluated when blocks are compared.

See also

Setting bookmarks (Page 1674)

Navigating between bookmarks (Page 1675)

Deleting bookmarks (Page 1675)

Setting bookmarks

Requirement

The SCL block is open.

Procedure

To set a bookmark, follow these steps:

1. Right-click on the desired line in the sidebar.
2. Select the "Bookmarks > Set" command in the shortcut menu.

Or:

1. Click on the line in which you want to place the bookmark.
2. Click the "Set/delete bookmark" button in the toolbar.

Or:

1. Hold down the <Ctrl> key.
2. Click on the line in the sidebar in which you want to place the bookmark.

Result

A bookmark is placed in the program code.

See also

Basics of bookmarks (Page 1674)

Navigating between bookmarks (Page 1675)

Deleting bookmarks (Page 1675)

Navigating between bookmarks

Requirement

Several bookmarks are set in a block.

Procedure

To navigate between bookmarks, follow these steps:

1. Set the insertion cursor in the program code.
2. In the "Edit" menu, select the "Go to > Next bookmark" or "Go to > Previous bookmark" command.

Or:

1. Set the insertion cursor in the program code.
2. In the toolbar of the programming editor, click the "Go to next bookmark", "Go to previous bookmark" button.

Or:

1. Click in the sidebar.
2. Select the "Bookmarks > Next" or "Bookmarks > Previous" command in the shortcut menu.

Result

The line with the bookmark is highlighted.

See also

Basics of bookmarks (Page 1674)

Setting bookmarks (Page 1674)

Deleting bookmarks (Page 1675)

Deleting bookmarks

You can delete individual bookmarks or all bookmarks from the block or the CPU.

Deleting individual bookmarks

To delete an individual bookmark, follow these steps:

1. Right-click in the sidebar on the line in which you want to delete the bookmark.
2. Select the "Bookmarks > Remove" command in the shortcut menu.

Or:

1. Click on the line in which you want to delete the bookmark.
2. In the "Edit" menu, select the "Bookmarks > Remove" command.

Or:

1. Click on the line in which you want to delete the bookmark.
2. Click the "Set/delete bookmark" button in the toolbar.

Deleting all bookmarks from the block

To delete all bookmarks from the block, follow these steps:

1. Right-click in the sidebar.
2. Select the "Bookmarks > Delete all from block" command in the shortcut menu.

Or:

1. In the "Edit" menu, select the "Bookmarks > Delete all from block" command.

See also

Basics of bookmarks (Page 1674)
Setting bookmarks (Page 1674)
Navigating between bookmarks (Page 1675)

Entering SCL instructions

Rules for SCL instructions

Instructions in SCL

SCL recognizes the following types of instructions:

- Value assignments
  Value assignments are used to assign a tag a constant value, the result of an expression or the value of another tag.
- Instructions for program control
  Instructions for program control are used to implement program branches, loops or jumps.

- Additional instructions from the "Instructions" task card
  The "Instructions" task card offers a wide selection of standard instructions that you can use in your SCL program.
- Block calls
  Block calls are used to call up subroutines that have been placed in other blocks and to further process their results.

Rules

You need to observe the following rules when entering SCL instructions:

- Instructions can span several lines.
- Each instruction ends with a semicolon (;).
- No distinction is made between upper and lower case.
- Comments serve only for documentation of the program. They do not affect the program execution.

Examples

The following examples shows the various types of instructions:

```
SCL
// Example of a value assignment
"MyTag":= 0;
// Example of a block call
"MyDB"."MyFB" (ParamInput:= 10);
// Example of a program control instruction
WHILE "Counter" < 10 DO
    "MyTAG" := "MyTag" + 2;
END_WHILE;
```

Entering SCL instructions manually

Requirement

An SCL block is open.

Procedure

To enter SCL instructions, follow these steps:

1. Enter the syntax of the instruction using the keyboard.
   You are supported by the auto-complete function when performing this task. It offers all the instructions and operands that are allowed at the current location.
2. Select the required instruction or the desired operand from the auto-complete function.
   If you select an instruction that requires specification of operands, placeholders for the operands are inserted into the program. The placeholders for the operands are highlighted in yellow. The first placeholder is selected.

3. Replace this placeholder with an operand.
4. Use the <TAB> key to navigate to all other placeholders and replace them with operands.

Note

You can also drag-and-drop a defined operand from the PLC tag table or from the block interface into the program. To replace an operand that has already been inserted, hover the mouse pointer briefly over the operand to be replaced before releasing the mouse button. This selects the operand and when you release the mouse button it is replaced by the new operand.

Result

The instruction is inserted.

The programming editor performs a syntax check. Incorrect entries are displayed in red and italics. In addition, you also receive a detailed error message in the inspector window.

See also

Using autocompletion in textual programming languages (Page 1548)

Data type conversion for S7-1200: (Page 2091)

Expanding and reducing the parameter list (Page 1693)

Inserting SCL instructions using the "Instructions" task card

The "Instructions" task card offers a wide selection of instructions that you can use in your SCL program. The SCL-specific instructions for program control are available in the "Instructions" task card.

Requirement

An SCL block is open.

Procedure

To insert SCL instructions into a program using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. To insert the instruction, select one of the following steps:
   - Navigate to the SCL instruction you want to insert and drag-and-drop it to the required line in the program code. The insertion location is highlighted by a green rectangle.
   - Select the location in the program code where you want to insert the instruction and then double-click on the instruction you want to insert.

   The instruction is inserted in the program. The placeholders for the operands are highlighted in yellow. The first placeholder is selected.

3. Replace this placeholder with an operand. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
4. Use the <TAB> key to navigate to all other placeholders and replace them with operands.

Result

The instruction is inserted.

The programming editor performs a syntax check. Incorrect entries are displayed in red and italics. In addition, you also receive a detailed error message in the inspector window.

See also

Using autocompletion in textual programming languages (Page 1548)

Data type conversion for S7-1200: (Page 2091)

Expanding and reducing the parameter list (Page 1693)

Defining the data type of an SCL instruction

Basic information on the data types of SCL instructions

Introduction

The SCL instructions that you employ for block programming use specific data types to calculate function values. Certain SCL instructions only support the use of a specific data type. You cannot change the data type for these instructions. However, most of the SCL instructions support the use of different data types. We differentiate between the following two types of such instructions:

- Instructions for which the data type of the function value is determined by the data type of the input parameters. This is the case for most instructions.
- Instructions with default data type. The instructions listed in the following table are of this type.

You will have to change the default data type if this is incompatible with the data type of the input parameter used. You can always change the data type based on the following syntax:
`_<data type>`

SCL instructions with default data type

The following table lists the SCL instructions with default data types:

Instruction	Default data type
CEIL	DINT
DECO	DWORD
FLOOR	DINT
NORM_X	REAL

Instruction	Default data type
PEEK	BYTE
SCALE_X	INT
TRUNC	DINT
CONCAT	STRING

See also

Changing the data type of an SCL instruction (Page 1680)

Example for changing the data type of an SCL instruction (Page 1681)

Changing the data type of an SCL instruction

Procedure

Proceed as follows to insert an SCL instruction and change its data type:

1. Insert the instruction at the required point in the program using drag-and-drop.
2. Specify the operands for the instruction.
   The data type of the function value is specified based on the input parameters, or the default data type of the instruction is used.
3. Append the "_<data type>" string to the instruction name.
   "<data type>" represents the data type you need for the instruction.

See also

Basic information on the data types of SCL instructions (Page 1679)

Example for changing the data type of an SCL instruction (Page 1681)

Modifying the data types of IEC timers and IEC counters

IEC timers and IEC counters are internal system function blocks and require an instance data block. You can create the instance data blocks either as single or multi-instance. The data type of the instance data block is determined according to the associated instruction. For CPUs of the S7-1200 and S7-1500 series, you can, however execute the instructions with different data types, depending on your requirements.

If the newly set data type of the instance data block does not match the data type of the input parameter, an implicit conversion takes place if possible. If the conversion is not possible, you will receive an error message.

Procedure

To change the data type of an IEC timer or IEC-counter instance data block, proceed as follows:

1. Open the block in which you call the IEC timer or IEC counter.
   Depending on the instance type of the instance data block, there is a green-bordered box before (multi-instance) or after (single instance) the name of the instance data block.
2. Click the green-bordered box.
   A drop-down list box with the valid data types for the instance data block is opened.
3. Select the desired data type.

Example for changing the data type of an SCL instruction

Changing the default data type of the "Decode" instruction (DECO)

Data type DWORD is set as default if you insert the "Decode" instruction in the program.

```
"Tag_Result" := DECO(IN := "Tag_Value");
```

Modify the program code as follows to convert the data type from DWORD to BYTE:

```
"Tag_Result_BYTE" := DECO_BYTE(IN := "Tag_Value");
```

See also

Basic information on the data types of SCL instructions (Page 1679)

Changing the data type of an SCL instruction (Page 1680)

Displaying or hiding tag information

Introduction

Regardless of whether the operands are represented in absolute or symbolic form, you can show and hide simple or hierarchical comments used to document global tags. This information is taken from the PLC tag table.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all blocks currently opened and opened in future is shown.

You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual ones that you have opened.

If you select the display of tag information with hierarchical comments, the comments of the higher structure levels of structured tags will also be displayed. The display is in brackets after the comment of the tags; the comments of the individual levels are separated by a period. If there is no comment at a structure level for a tag, it is omitted in the display and this is recognizable because there are two periods.

Displaying or hiding tag information for all blocks

Follow the steps below to display or hide the tag information for all blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to show the tag information, either select the "Expand" option in the "Tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
4. If you want to hide the tag information, select the "Collapse" option in the "Tag information" drop-down list.
   The tag information is displayed or hidden for all blocks. When you open further blocks, the tag information is displayed or hidden depending on the selected setting.

Displaying or hiding tag information for an opened block

Follow the steps below to display or hide the tag information for an opened block:

1. If you want to show the tag information, either select the "Show tag information" option in the "Shows the tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
2. If you want to hide the tag information, select the "Hide tag information" option in the "Hides tag information" drop-down list.
   The tag information is displayed or hidden.

Using Favorites in SCL

Adding SCL instructions to the Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 1529)

Inserting SCL instructions using Favorites (Page 1683)

Removing SCL instructions from the Favorites (Page 1684)

Inserting SCL instructions using Favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 1529)

Adding SCL instructions to the Favorites (Page 1682)

Removing SCL instructions from the Favorites (Page 1684)

Removing SCL instructions from the Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 1529)

Adding SCL instructions to the Favorites (Page 1682)

Inserting SCL instructions using Favorites (Page 1683)

Insert block calls in SCL

Basic information on the block call in SCL

Calling function blocks

Syntax of a call

The following syntax is used to call a function block as a single or multi-instance:

- Single instance:
  - If the function block originates from the project:
    `<DBName> (Parameter list)`
  - If the function block originates from the "Instructions" task card:
    `<DB name>.<Instruction name> (Parameter list)`
- Multi-instance
  `<#Instance name> (Parameter list)`

Calling as single instance or multi-instance

Function blocks can be called either as a single instance or a multi-instance.

- Calling as a single instance
  The called function block stores its data in a data block of its own.
- Calling as a multi-instance
  The called function block stores its data in the instance data block of the calling function block.

For additional information on the types of calls, refer to "See also".

Parameter list

If you call another code block from a SCL block, you can supply the formal parameters of the called block with actual parameters.

The specification of the parameters has the form of a value assignment. This value assignment enables you to assign values (actual parameters) to the parameters you have defined in the called block.

The formal parameters of the called code block are listed in brackets directly after the call. Input and in-out parameters have the assignment identifier ":=", output parameters have the assignment identifier "=>". A placeholder placed after the parameter shows the required data type and the type of the parameter.

Rules for supplying parameters

The following rules apply to supplying parameters:

- Constants, tags and expressions can be used as actual parameters.
- The assignment order is not of importance.
- The data types of formal and actual parameters must match.
- The individual assignments are separated by commas.
- If the called block has only one parameter, it is sufficient to specify the actual parameter in the brackets. The formal parameter need not be specified.

See also

Manually inserting block calls (Page 1689)

Inserting block calls with drag-and-drop (Page 1690)

Examples for calling a function block in SCL (Page 1687)

Calling functions

Syntax of a call

The following syntax is used to call a function:

```
<Function name> (Parameter list);     //Standard call
```

```
<Operand>:=<Function name> (Parameter list); // Call in an expression
```

Function value

Functions that provide a return value can be used in any expression in place of an operand. For this reason, the return value is also known as the "function value" in SCL.

The call options of functions depend on whether the function returns a function value to the calling block.

The function value is defined in the RET_VAL parameter. If the RET_VAL parameter is of the VOID data type, then the function will not return a value to the calling block. If the RET_VAL parameter has another data type, then the function returns a function value of this data type.

In SCL, all data types are permitted for the RET_VAL parameter except ANY, ARRAY, STRUCT and VARIANT, as well as the parameter types TIMER and COUNTER.

Call options

There are two possibilities for calling functions in SCL:

- Standard call for functions with and without a function value
  With a standard call, the results of the function is made available as an output and in-out parameter.
- Call in an expression for functions with a function value
  Functions that return a function value can be used in any expression in place of an operand, for example, a value assignment.
  The function calculates the function value, which has the same name as the function and returns it to the calling block. There the value replaces the function call.
  Following the call, the results of the function is made available as a function value or as an output and in-out parameter.

Parameter list

If you call another code block from a SCL block, you need to supply the formal parameters of the called block with actual parameters.

The specification of the parameters has the form of a value assignment. This value assignment enables you to assign values (actual parameters) to the parameters you have defined in the called block.

The formal parameters of the called code block are listed in brackets directly after the call. Input and in-out parameters have the assignment identifier ":=", output parameters have the assignment identifier "=>". A gray placeholder placed after the parameter shows the required data type and the type of the parameter.

Rules for supplying parameters

The following rules apply to supplying parameters to functions:

- All parameters of the function must be supplied.
- The assignment order is not of importance.

- Constants, tags and expressions can be used as actual parameters.
- The data types of formal and actual parameters must match.
- The individual assignments are separated by commas.
- If the called block has only one parameter, it is sufficient to specify the actual parameter in the brackets. The formal parameter need not be specified.
- When you call functions in SCL, you cannot use the release mechanism via EN. Use an IF statement instead to call functions conditionally.

See also

Manually inserting block calls (Page 1689)

Inserting block calls with drag-and-drop (Page 1690)

Examples for calling functions in SCL (Page 1688)

Examples for calling a function block in SCL

Calling as a single instance

The following example shows the call of an FB as a single instance:

```
SCL
// Call as a single instance
"MyDB" (MyInput:=10, MyInout:= "Tag1");
```

Result

After the call is executed, the value determined for the "MyInout" in/out parameter is available in "Tag1" in the "MyDB" data block.

Calling as a multi-instance

The following example shows the call of an FB as a multi-instance:

```
SCL
// Call as a multi-instance
"MyFB" (MyInput:= 10, MyInout:= "Tag1");
```

Result

After the "MyFB" block is executed, the value determined for the "MyInout" in-out parameter is made available in "Tag1" in the data block of the calling code block.

See also

Calling function blocks (Page 1684)

Manually inserting block calls (Page 1689)

Inserting block calls with drag-and-drop (Page 1690)

Examples for calling functions in SCL

Standard call

The following example shows a standard function call:

SCL

```
// Standard function call
"MyFC" (MyInput := 10, MyInOut := "Tag1");
```

Result

After the "MyFC" block is executed, the value determined for the "MyInOut" in/out parameter is available in "Tag1" in the calling block and needs to be further processed there.

Call in a value assignment

The following example shows a function call in a value assignment:

SCL

```
(*Call in a value assignment, a function value was defined for "MyFC" *)
#MyOperand := "MyFC" (MyInput1 := 3, MyInput2 := 2, MyInput3 := 8.9,
MyInOut := "Tag1");
```

Result

The function value of "MyFC" is transferred to "#MyOperand".

Call in an arithmetic expression

The following example shows a function call in an arithmetic expression:

SCL

```
(*Call in a mathematical expression, a function value was defined for
"MyFC"  *)
#MyOperand := "Tag2" + "MyFC" (MyInput1 := 3, MyInput2 := 2, MyInput3 :=
8.9);
```

Result

The function value of "MyFC" will be added to "Tag2" and the result will be transferred to "MyOperand".

See also

Calling functions (Page 1685)

Manually inserting block calls (Page 1689)

Inserting block calls with drag-and-drop (Page 1690)

Manually inserting block calls

You can insert calls for functions (FCs) and function blocks (FBs).

Inserting a call for a function (FC)

Proceed as follows to insert a function call:

1. Enter the function name.
2. Confirm your entry with the Return key.
   The syntax for the function call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
3. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
4. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Enter the name of the function block.
2. Confirm your entry with the Return key.
   The "Call options" dialog opens.
3. In the dialog, specify whether you want to call the block as a single or multi-instance.
   - If you click the "Single instance" button, in the "Name" field enter a name for the data block to be assigned to the call.
   - If you click the "Multi-instance" button, in the "Name in the interface" field enter a name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".
   The syntax for the function block call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.

5. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
6. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Result

The block call is inserted.

If you specify an instance data block that does not exist when calling a function block, it is created.

See also

Updating block calls (Page 1691)

Expanding and reducing the parameter list (Page 1693)

Using autocompletion in textual programming languages (Page 1548)

Inserting block calls with drag-and-drop

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree.

Requirement

The function to be called (FC) or the function block (FB) to be called is present.

Inserting a call for a function (FC)

To insert a function call using drag-and-drop, follow these steps:

1. Drag the function from the project tree into the program.
   The syntax for the function call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
2. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
3. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Inserting a call for a function block (FB)

To insert a call for a function block (FB) using drag-and-drop, follow these steps:

1. Drag the function block from the project tree and drop it into the program.
   The "Call options" dialog opens.
2. In the dialog, specify whether you want to call the block as a single or multi-instance.
   - If you click the "Single instance" button, in the "Name" field enter a name for the data block to be assigned to the call.
   - If you click the "Multi-instance" button, in the "Name in the interface" field enter a name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
3. Confirm your entries with "OK".
   The syntax for the function block call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
4. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
5. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Result

The block call is inserted.

If you specify an instance data block that does not exist when calling a function block, it is created.

See also

Updating block calls (Page 1691)

Expanding and reducing the parameter list (Page 1693)

Using autocompletion in textual programming languages (Page 1548)

Updating block calls

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have the following options for updating the block calls:

- Explicit updating of all inconsistent block calls in the programming editor.
  The inconsistent block calls within the open block are updated. The following actions are carried out in the process:
  - New parameters are added. Please note, however, that the parameters are hidden for function blocks (FBs) and the parameters are supplied via the corresponding instance data block (DB). If required, you can show the parameters using the shortcut menu command "Show all parameters".
  - Deleted parameters are not removed. If necessary, expand the parameter list to remove deleted parameters manually.
  - Renamed parameters get the new parameter names.
- Explicit updating of a block call in the programming editor.
  The inconsistent call of this block is updated at all call locations. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are not removed. If necessary, expand the parameter list to remove deleted parameters manually.
  - Renamed parameters get the new parameter names.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated. Make sure that you manually remove deleted parameters before the compilation process and supply all new formal parameters with actual parameters when you call functions.

Updating all inconsistent block calls in the programming editor

To update all block calls in a block, follow these steps:

1. Open the calling block in the programming editor.
2. Click "Update inconsistent block calls" in the toolbar.
   All inconsistent calls are updated. If necessary, supply new formal parameters of functions (FCs) with actual parameters.

Updating a specific block call in the programming editor

To update a specific block call in the programming editor, follow these steps:

1. Open the calling block in the programming editor.
2. Right-click on the block call that you want to update.
3. Select the "Update block call" command in the shortcut menu.
4. If parameters were added, enter the values for the new block parameters.

Note

Note that the "Update block call" command is only available as long as you did not previously update all block calls in the editor with the "Update inconsistent block calls" command.

Updating block calls during compilation

To implicitly update all block calls and uses of PLC data types during compilation, follow these steps:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the "Compile > Software (rebuild all blocks)" command in the shortcut menu.

See also

Manually inserting block calls (Page 1689)

Inserting block calls with drag-and-drop (Page 1690)

Expanding and reducing the parameter list (Page 1693)

Expanding and reducing the parameter list

In SCL, if you call blocks or insert instructions that are system-internal function blocks, the syntax and the parameter list with the placeholders for the actual parameters are inserted in the SCL program. To make the program code easier to read, the unused optional parameters are removed from the parameter list when you edit other instructions. You can restore these at any time. You can also explicitly reduce the parameter list when you have finished assigning the parameters.

Expanding the parameter list

To expand the parameter list, follow these steps:

1. Right-click in the block call or the instruction.
2. Select the "Expand parameter list" command from the shortcut menu or press the key combination <Ctrl+Shift+Space bar>.
   The parameter list is displayed in full again.

Reducing the parameter list

To reduce the parameter list, follow these steps:

1. Right-click in the block call or the instruction.
2. Select the "Reduce parameter list" command from the shortcut menu or press the key combination <Ctrl+Shift+Space bar>.
   All unused optional parameters are hidden.

See also

Entering SCL instructions manually (Page 1677)

Inserting SCL instructions using the "Instructions" task card (Page 1678)

Manually inserting block calls (Page 1689)

Inserting block calls with drag-and-drop (Page 1690)

Inserting comments

Commenting program code

You have various options for commenting SCL programs:

- Line comment
  A line comment starts with "//" and extends to the end of the line.
- Comment section
  A comment section is started with "(* and completed by "*)". It can span several lines.

Inserting a line comment

To insert line comments, follow these steps:

1. Type "//" at the position where you want to place the comment. This does not have to be the beginning of the line.
2. Enter the comment text.

Inserting a comment section

To insert a comment section, follow these steps:

1. Type "(*" at the position where you want to place the comment. This does not have to be the beginning of the line.
2. Enter the comment text.
3. Complete the comment with "*)".

Disabling one or more lines with comments

To disable program code with comments, follow these steps:

1. Select the code lines you want to comment out.
2. Click the "Comment selection" button in the editor.
   "//" is inserted at the beginning of the line in the selected lines. The code that follows is interpreted as a comment. If lines already containing a line comment are disabled, "//" is inserted as well. If these lines are enabled again, the original comments are retained.

Enabling comment lines

To enable lines that have been commented out to be enabled as code again, proceed as follows:

1. Select the code lines you want to enable.
2. Click the "Remove comment" button in the editor.
   The "//" mark for line comments at the beginning of the line is removed.

Example

The following code contains comment sections and line comments

```
(*******************************************************************************
  A description of the instructions that follow can be placed here
*******************************************************************************)
IF "MyVal1" > 0 THEN //No division by 0
    "MyReal" := "MyVal2" (* input value *) / "MyVal1" (* measured value *);
END_IF;
//Data type conversion
"MyInt" := REAL_TO_INT("MyReal");
```

Editing SCL instructions

Selecting instructions

You can select individual instructions or all instructions of a block.

Requirement

An SCL block is open.

Selecting individual instructions

To select individual instructions, follow these steps:

1. Set the insertion mark before the first character that you want to select.
2. Press and hold down the left mouse button.
3. Move the cursor to a position after the last character that you want to select.
4. Release the left mouse button.

Selecting all the instructions of a program

To select all instructions, follow these steps:

1. In the "Edit" menu, select the "Select All" command or use the keyboard shortcut <Ctrl+A>.

Note

When you select the opening keyword of an instructing, the closing keyword is automatically highlighted.

Copying, cutting and pasting instructions

Copying an instruction

To copy an instruction, follow these steps:

1. Select the instruction you want to copy.
2. Select "Copy" in the shortcut menu.

Cutting an instruction

To cut an instruction, follow these steps:

1. Select the instruction you want to cut.
2. Select the "Cut" command in the shortcut menu.

Inserting an instruction from the clipboard

To insert an instruction from the clipboard, follow these steps:

1. Copy or cut an instruction.
2. Click on the position at which you want to insert the instruction.
3. Select "Paste" in the shortcut menu.

Deleting instructions

Requirement

An SCL block is open.

Procedure

To delete an instruction, follow these steps:

1. Select the instruction you want to delete.
2. Select the "Delete" command in the shortcut menu.

Eliminating syntax errors in the program

Basic information on syntax errors

Syntax errors

Below are some examples of syntax errors:

- Missing separators or the use of too many separators
- Incorrect keyword spelling
- Incorrect jump label spelling/notation
- Notation which does not match the set mnemonics (for example, "I2.3" instead of "E2.3")
- The use of key words as operands

Identification of syntax errors

Syntax errors are underlined in red or appear in red type.

This identification allows you to recognise incorrect inputs at a glance and jump from error to error to eliminate them. Syntax errors are also listed in the "Info" tab of the inspector window with an error message.

See also

Finding syntax errors in the program (Page 1697)

Finding syntax errors in the program

Procedure

To find syntax errors in the program, follow these steps:

1. Select the position in the program in which you wish to look for errors.
2. Click "Go to next error" in the toolbar.
   The first error after the position you have selected will be marked.

You can use "Go to next error" and "Go to previous error" in the toolbar to find and correct all errors in the block.

Or:

1. Open the error list in the inspector window with "Info > Syntax".
   All syntax errors are listed in the table with a short description of the error.
2. If there are any errors, click on the blue question mark next to the error text to obtain information on eliminating the problem.
3. Double-click the error you want to correct.
   The corresponding error is highlighted.

See also

Basic information on syntax errors (Page 1697)

Changing the programming language

Rules for changing the programming language

Rules

Observe the following rules if you want to change the programming language for a block:

- All CPU series:
  - You can only change the programming language of entire blocks. The programming language cannot be changed for individual networks.
  - You cannot switch blocks programmed in the programming languages SCL or GRAPH. In GRAPH blocks, however, you can change between LAD and FBD as network languages.
- S7-300/400:
  - You can only change between the programming languages LAD, FBD and STL.
  - You can create networks within a block using another programming language and then copy them into the desired block.
  - If the language of individual networks of the block cannot be changed, these networks is displayed in their original language.
- S7-1200/1500:
  - You can change between the programming languages LAD and FBD.
- S7-1500:
  - You can create STL networks within the LAD and FBD blocks. However, you cannot copy between STL and LAD/FBD.

Change the programming language

Procedure

To change the programming language, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.
3. Select the "General" entry in the area navigation.
4. Select the new programming language in the "Language" drop-down list.
5. Confirm your selection with "OK".

See also

Rules for changing the programming language (Page 1698)

Handling program execution errors

Basics of error handling

Introduction

Program execution errors are programming or I/O access errors. You have a number of different options for responding to program execution errors depending on the CPU used.

Handling program execution errors in S7-300/400

You can program the program execution error OB (OB 85) for S7-300/400 CPUs. If a program execution error occurs and you do not use the program execution error OB, the CPU will switch to "STOP" mode.

You will find additional information about the program execution error OB in the description of the mode of operation of S7-300/400 CPUs.

Handling program execution errors in S7-1200/1500

You can select the type of error handling for CPUs of the S7-1200 and S7-1500 series. You have the following two options:

- Use the CPU's global troubleshooting:
  - S7-1200: The CPU generates a diagnostic buffer entry and remains in "RUN" mode.
  - S7-1500: You can program the programming error OB (OB 121) and the I/O access error OB (OB 122) for S7-1500 CPUs. If no programming error OB exists in the CPU, the CPU switches to "STOP" mode when a programming error occurs. In the event of an I/O access error, the CPU always remains in "RUN" mode, regardless of whether the I/O access error OB is present.
    Please note, however, that an existing I/O access or programming error OB is not called synchronously to the error. Therefore, depending on the selected priority, the execution of I/O access or programming error OBs may be delayed instead of taking place immediately when the error occurs. If other errors occur before execution of the I/O access or programming error OB is complete, no further I/O access or programming error OB is called. If you want to prevent I/O access or programming error OBs from being discarded, set the priority correspondingly high.
    You can use the enable output ENO to detect I/O access and programming errors for the instructions "Read field" (FieldRead), "Write field" (FieldWrite) , "Read memory address" (PEEK) and "Write memory address" (POKE).
    You can find more information about these error OBs in the description of the mode of operation of S7-1500 CPUs.
- You use separate local error handling. Local error handling is error handling within a block. Local error handling has the following advantages:
  - The error information is stored in the system memory, which you can query and evaluate.
  - You can use the error information to program a response in the block to the error that has occurred.
  - Programmed error evaluation and error reactions do not interrupt the program cycle.
  - The system performance is not unnecessarily burdened by the local error handling. If no errors occur, programmed error analyses and reactions are not executed.
    Local error handling applies only to blocks for which it has been set explicitly. If local error handling is set for a block, no global error handling is conducted for errors in this block.

Note

Note the following information:

- All memory access errors and I/O access errors must be captured either by global or local error handling.
- If the parameters of an instruction do not cause any memory access errors, you can query the associated ENO.

See also

GET_ERROR: Get error locally (Page 2694)

GET_ERR_ID: Get error ID locally (Page 2697)

GET_ERROR: Get error locally (Page 2414)

GET_ERR_ID: Get error ID locally (Page 2417)

Querying and fixing errors in the program code (Page 228)

Local error handling

Principles of local error handling

Introduction

Local error handling makes it possible to query the occurrence of errors within a block and evaluate the associated error information. You can set local error handling for organization blocks (OBs), function blocks (FBs), and functions (FCs). If local error handling is enabled, the system reaction is ignored.

Local error handling applies only to blocks for which it has been set explicitly. The local error handling setting is not assumed by a calling block, nor is it transferred to called blocks. For higher-level blocks and lower-level blocks, the system settings still apply provided dedicated error handling has not been programmed for these blocks.

General procedure for local error handling

When errors occur while a block is being executed with local error handling, a predefined response is initiated based on the following error types:

- Write errors: These errors are ignored, and program execution simply continues.
- Read errors: Program execution continues with the substitute value "0".
- Execution errors: Execution of the instruction is aborted. Program execution resumes with the next instruction.

Information about the first error that occurs is stored in the system memory. This information can be queried and output with an instruction (GET_ERROR or GET_ERR_ID). Error information is output in a format that can undergo additional processing. You can use additional instructions to analyze error information and program a reaction to the error based.

When information about the first error is queried, the error memory space in the system memory is enabled. Then, when additional errors occur, information about the next error is output.

Instructions for local error handling

You can use the following instructions for local error handling:

- GET_ERROR: Get error locally
- GET_ERR_ID: Get error ID locally

The instructions differ in the amount of error information that is output with each one.

For additional information on the instructions, refer to "See also".

See also

GET_ERROR: Get error locally (Page 2694)

GET_ERR_ID: Get error ID locally (Page 2417)

GET_ERR_ID: Get error ID locally (Page 2697)

GET_ERROR: Get error locally (Page 2414)

Error output priorities

Overview of the priorities

In local error handling, information about the first error that occurred is displayed. If multiple errors occur at the same time while an instruction is being executed, these errors are displayed according to their priority. The following table shows the priority of different types of errors.

Priority	Error type
1	Error in the program code
2	Missing reference
3	Invalid range
4	DB does not exist
5	Operands are not compatible
6	Width of specified area is not sufficient
7	Timers or counters do not exist
8	No write access to a DB
9	I/O error
10	Instruction does not exist
11	Block does not exist
12	Invalid nesting depth

The highest priority is 1 and the lowest priority is 12.

See also

GET_ERROR: Get error locally (Page 2694)

GET_ERR_ID: Get error ID locally (Page 2417)

GET_ERR_ID: Get error ID locally (Page 2697)

GET_ERROR: Get error locally (Page 2414)

Enabling local error handling for a block

Introduction

Local error handling is enabled for a block if you insert one of the following instructions in a network.

- GET_ERROR: Get error locally
- GET_ERR_ID: Get error ID locally

For additional information on the instructions, refer to "See also".

If local error handling is enabled for a block, the system reactions for this block are ignored.

Requirement

- The block is open.
- Die "Instructions" task card is open.

Procedure

To enable local error handling for a block, proceed as follows:

1. Navigate to the "Basic instructions" pane of the "Instructions" task card.
2. Open the "Program Control" folder.
3. Drag the instruction "Get error locally" (GET_ERROR) or "Get error ID locally" (GET_ERR_ID) to the required network.

Result

Local error handling is enabled for the open block. The "Handle errors within block" check box is selected in the Inspector window under "Properties > Attributes". This setting cannot be edited in the Inspector window. Local error handling can be deactivated by deleting the inserted instructions on local error handling.

See also

GET_ERROR: Get error locally (Page 2694)

GET_ERR_ID: Get error ID locally (Page 2417)

GET_ERR_ID: Get error ID locally (Page 2697)

GET_ERROR: Get error locally (Page 2414)

11.1.4.3 Programming data blocks

Basic principles for programming of data blocks

A data block (DB) is used to save the values that are written during execution of the program.

In contrast to the code block, the data block contains only tag declarations. It contains no networks or instructions. The tag declarations define the structure of the data block.

Types of data blocks

There are two types of data blocks:

- Global data blocks
  The global data block is not assigned to a code block. You can access the values of a global data block from any code block. A global data block contains only static tags.
  The structure of the global data block can be freely defined. In the declaration table for data blocks, you declare the data elements that are to contained in the global data block.
- Instance data blocks
  The instance data block is assigned directly to a function block (FB). The structure of an instance data block cannot be freely defined, but is instead determined by the interface declaration of the function block. The instance data block contains exactly those block parameters and tags that are declared there.
  However, you can define instance-specific values in the instance data block, for example, start values for the declared tags.

ARRAY data blocks (S7-1500)

ARRAY data blocks are global data blocks that consist of an ARRAY. This ARRAY can be based on any data type. For example, an ARRAY of a PLC data type (UDT) is possible. The DB contains no other elements besides the ARRAY. Because of their flat structure, ARRAY data blocks facilitate access to the ARRAY elements and their transfer to called blocks.

The "Move operations" section of the "Instructions" task card offers options for addressing of ARRAY DBs.

PLC data types as a template for global data blocks

PLC data types can be used as templates for the creation of global data blocks with identical data structures. You create the structure as PLC data type only once and then generate the required data blocks by assigning the PLC data type.

System data types as a template for global data blocks

System data types can also be used as templates for creating global data blocks with identical data structure. System data types already have a pre-defined structure. You insert the system data type in the program only once and then generate additional data blocks with an identical structure by assigning the system data type.

Access modes

There are two different modes of accessing data values in data blocks:

- Data blocks with optimized access (only S7-1200)
  Data blocks with optimized access have no fixed defined structure. In the declaration, the data elements are assigned only a symbolic name and no fixed address within the block. You access the data values in these block via symbolic names.
  The "Optimized block access" attribute is always enabled for ARRAY data blocks.
- Data blocks with standard access (all CPU families)
  Data blocks with standard access have a fixed structure. In the declaration, the data elements are assigned both a symbolic name and a fixed address within the block. You can access the data values in these blocks via symbolic names or the address.
  ARRAY data blocks with standard access are not possible.

Retentivity of data values

To prevent data loss in the event of power failure, you can store the data values in a retentive memory area.

See also

Creating data blocks (Page 1508)

Global data blocks (DB) (Page 1416)

Instance data blocks (Page 1417)

Structure of the declaration table for data blocks

Structure of the declaration table for data blocks

The figure below shows the structure of the declaration table for data blocks. The display will vary depending on type of block and type of access.

	Name	Data type	Start value	Retain	Visible in HMI	Comment
	Input					
	MyInput1	Bool	false	☑	☑	
	Output					
	MyOutput1	Byte	0	☐	☑	
	InOut					
	Static					

Display of instance-specific values

In instance data blocks, you can apply the already defined values from the interface of the assigned function block or define instance-specific start values. Values that are applied from the function block cannot be edited. You can replace the grayed-out values with instance-specific values. Values that were already changed instance specific are not grayed out.

Meaning of the columns

The following table shows the meaning of the individual columns. You can show or hide the columns as required. The number of columns displayed varies depending on the CPU series.

Column	Explanation
	Symbol you can click to move or copy the tag. You can, for example, drag-and-drop the tag into a program and use it there as operand.
Name	Name of the tags.
Data type	Data type of the tags.
Offset	Relative address of the tags. The column is only visible in data blocks with standard access.
Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
Start value	Value that the tag should assume at startup. The default values defined in a code block are used as start values during the creation of the data block. You can then replace these adopted values with instance-specific start values. Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.
Monitor value	Current data value in the CPU. This column only appears if an online connection is available and you click "Monitor".
Snapshot	Shows values that were loaded from the device.
Retentivity	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off.
Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
Accessible from HMI	Shows whether HMI can access this tag during runtime.
Setting value	Setting values are the values that will probably have to be fine tuned during commissioning. After commissioning, the values of these tags can be transferred to the offline program as start values and stored there.
Comment	Comment to document the tags.

See also

Creating data blocks (Page 1508)

Basic information on start values (Page 1713)

Creating data blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a data block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Select the type of the data block. You have the following options available to you:
   - To create a global data block, select the list entry "Global DB".
   - To create an ARRAY data block, select the "ARRAY DB" entry in the list.
   - To create an instance data block, select the function block to which you want to assign the instance data block from the list. The list contains only the function blocks that were previously created for the CPU.
   - To create a data block based on a PLC data type, select the PLC data type from the list. The list contains only the PLC data types that were previously created for the CPU.
   - To create a data block based on a system data type, select the system data type from the list. The list contains only those system data types that have already been inserted to program blocks in the CPU.
4. Enter a name for the data block.
5. Enter the properties of the new data block.
6. If you have selected an ARRAY DB as the data block type, enter the ARRAY data type and the high limit for the ARRAY.
   You can change the high limit for the ARRAY at any time in the property window of the created block. The ARRAY data type cannot be changed subsequently.
7. To enter additional properties of the new data block, click "Additional information".
   An area with further input fields is displayed.
8. Enter all the properties you require.
9. Activate the "Add new and open" check box if the block does not open as soon as it is created.
10. Confirm your entry with "OK".

Result

The new data block is created. You can find the data block in the project tree in the "Program blocks" folder.

See also

Instance data blocks (Page 1417)

Global data blocks (DB) (Page 1416)

Overview of block properties (Page 1518)

Updating data blocks

Introduction

Changes in the interface of a function block or a PLC data type can lead to the corresponding data blocks becoming inconsistent. These inconsistencies are marked in red in the declaration table and at the call point of the block. To remedy these inconsistencies, the data blocks must be updated.

You have three options to update block calls:

- Explicit updating in the declaration table for data blocks.
  The data block is updated. Changes from the interface of the assigned function block and changes to the used PLC data types are applied.
- Explicit updating in the program editor.
  The block calls in the open block will be updated. The associated instance data block is also adjusted in the process.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types and the corresponding instance data blocks are updated.

Explicit Updating in the Declaration Table for Data Blocks

To explicitly update an individual data block, follow these steps:

1. Open the data block.
2. Select "Update interface" in the shortcut menu.

Explicit Updating in the Program Editor

To update all block calls or a specific call within a block, follow these steps:

1. Open the block in the program editor.
2. Right-click on the instruction with the block call.
3. Select the "Update" command in the shortcut menu.
4. The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
5. If you want to update the block call, click "OK". To cancel the update, click "Cancel".

Implicit Updating during Compilation

To implicitly update all block calls and uses of PLC data types as well as the instance data blocks during the compiling, follow these steps:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Changing the properties of tags in instance data blocks (Page 1720)

Extending data blocks

Description

In order to enable the editing of PLC programs that have already been commissioned and that are running without error on a system, CPUs of the S7-1500 series and most CPUs of the S7-1200 V4 and higher series support the option of extending global data blocks during runtime.

You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Principle of operation

Each data block is always assigned a default memory reserve. The memory reserve is not used initially. Activate the memory reserve if you decide on loading interface changes after having compiled and downloaded the block. All tags that you subsequently declare will be saved to the memory reserve. A subsequent download has no impact on the values of tags that have already been loaded.

If you decide to review your program at a later time while the plant is not in operation, you are also provided an option of reworking the memory layout of individual or several blocks in a single pass. With this action, you move all tags from the reserve area to the regular area. The memory reserve is now cleared and made available for further interface extensions.

Requirements

This "Download without reinitialization" function is available if the following requirements are met:

- The project is in the "TIA Portal V12" format or a higher version.
- You are working with a CPU that supports "Download without reinitialization".
- The blocks were created in LAD, FBD, STL, or SCL.
- The blocks were created by the user, i.e. they are not included with the blocks delivered in your package.
- These blocks are assigned the optimized access attribute.

Basic steps

Perform the following steps if you want to extend the data block and then load the block without re-initialization.

1. All blocks have a default memory reserve of 100 bytes. You can adapt this memory reserve to suit your requirements.
2. Activate the memory reserve.
3. Extend the block interface.
4. Compile the block.
5. Download the block to the CPU as usual.

Reference

For more information on the various steps, refer to chapter "Loading blocks (S7-1200/1500) ".

Creating a data structure for global data blocks

Declaring tags of elementary data type

Requirement

A global data block is open.

Note

You cannot change the structure of instance data blocks and of data blocks based on a PLC data type directly, since the structures of these blocks are defined by the respective function block or the PLC data type.

The type of the data block is entered in the block properties.

Procedure

To declare a tag of the elementary data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. In the "Data type" column, click the button for the data type selection.
   A list of the permissible data types is opened.
3. Select the desired data type.
4. Optional: Change the properties of the tags that are displayed in the other columns.
5. Repeat steps 1 to 4 for all tags that are to be declared.

See also

Displaying and editing block properties (Page 1523)

Declaring tags of the ARRAY data type (Page 1711)

Declaring tags of STRUCT data type (Page 1712)

Editing tables (Page 347)

Declaring tags of the ARRAY data type

Requirement

A global data block is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the "Array" data type in the "Data type" column. You will be supported by autocompletion in this step.
   The "Array" dialog opens.
3. In the "Data type" text box, specify the data type of the array elements.
4. In the "ARRAY limits" text box, specify the high and low limit for each dimension.
   Example of a one-dimensional ARRAY:
   `[0..3]`
   Example of a three-dimensional ARRAY:
   `[0..3, 0..15, 0..33]`
5. Confirm your entry.
6. Optional: Change the properties of the tags that are displayed in the other columns.

The tag is created but remains collapsed. To expand the ARRAY, click the triangle in front of the tag. Not that you cannot expand very large ARRAYs for reasons of clarity.

Entering start values of ARRAY elements

To set default start values for the individual elements of an ARRAY, follow these steps:

1. Click the triangle in front of the ARRAY data type tags.
   The ARRAY opens and the individual ARRAY elements are shown in separate rows.
2. Enter the required value in the "Start value" column.

See also

Array (Page 1941)

Declaring tags of STRUCT data type

Requirement

A global data block is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.
   An empty, indented row is inserted after the new tag.
3. Insert the first structural element in the first empty row.
   An additional empty row is inserted after the element.
4. Select a data type for the structure element.
5. Optional: Change the properties of the structural element that is displayed in the other columns of the block interface.
6. Repeat the step 4 to 7 for all additional structure elements.
   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
7. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

Enter start values of structure elements

To set default start values for the individual elements of a structure, follow these steps:

1. Click the triangle in front of the STRUCT data type tags.
   The structure opens and the individual structure elements are shown in separate rows.
2. Enter the required value in the "Start value" column.

Note

S7-1500: A maximum of 252 structures in one data block

A maximum of 252 structures is permitted in one data block. If you need additional structures, you must restructure your program. You can, for example, create the structures in several global data blocks.

See also

STRUCT (Page 1945)

Declaring tags based on a PLC data type

Requirements

- A global data block is open.
- A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a PLC data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the PLC data type in the "Data type" column. You will be supported by autocompletion during input.
3. Optional: Change the properties of the tags that are displayed in the other columns of the table.

Result

The tag is created.

See also

Layout of the block interface (Page 1551)

Define start values

Basic information on start values

Definition of "Start value"

The start value of a tag is a value defined by you which the tag assumes after a CPU startup.

The retentive tags have a special status. Their values take the defined start value only after a "cold restart". After a "warm restart", they retain their values and are not reset to the start value.

Definition of "Default value"

The structure of the data blocks can be derived from higher-level elements.

- An instance data block is based, for example, on the interface of a higher-level code block.
- A global data block can be based on a predefined PLC data type.

In this case you can define a default value for each tag in the higher-level element. These default values are used as start values during the creation of the data block. You can then replace these values with instance-specific start values in the data block.

Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.

See also

Define start values (Page 1714)

Structure of the declaration table for data blocks (Page 1705)

Declaring local tags and constants in the block interface (Page 1557)

Applying values from the online program as start values (Page 1733)

Define start values

Define start values

To define the start values for the tags of a data block, follow these steps:

1. Open the data block.
   The "Default value" column shows the default values that were defined for the tags in the interface of a higher-level code block or in a PLC data type.
2. Click the "Expanded mode" button to show all elements of structured data types.
3. Enter the desired start values in the "Start value" column. The value must match the data type of the tag and should not exceed the range of the data type.
   The start values are defined. The tag takes the defined value at startup, provided it was not declared as retentive.

Resetting a tag to the default value

To reset a tag for which you have defined a start value to the default value, follow these steps:

1. Select a modified value in the table.
2. Delete the value.
   The default value is entered. The default value is displayed.

Resetting all tags to the default value

To reset to the default value all tags for which you have defined an start value, follow these steps:

1. Select the "Reset start values" icon in the toolbar.
   The default values are transferred to the "Start value" column. Write-protected start values are not overwritten.

See also

Basic information on start values (Page 1713)

Applying values from the online program as start values (Page 1733)

Loading changed values

Introduction

To apply the changed start values from the offline program to the online program, you must load the changes. The following cases must be distinguished:

- Loading changed start values of non-retentive tags
- Loading changed start values of retentive tags
- Loading changed start values of setting values

Requirement

The start values in the offline program were changed.

Procedure

To load changed start values of non-retentive tags, follow these steps:

1. Select the blocks to be loaded in the project tree.
2. Select the "Download to device > Software (only changes)" command from the shortcut menu.
   The blocks are compiled and loaded.
   The start values of the newly defined tags are placed in the load memory of the CPU. The program runs with the new start values at the next transition from STOP to RUN.

To load changed start values of retentive tags, follow these steps:

1. Select the blocks to be loaded in the project tree.
2. Select the "Compile > Software (rebuild all blocks)" command in the shortcut menu.
3. In the "Online" menu, select the "Download and reset PLC program" command.
   The online blocks are deleted and replaced with the new blocks. This reinitializes all tags, including the retentive tags.

Information about loading changed setting values and general information about loading can be found under "See also".

See also

Initializing setting values in the online program (Page 1732)

Setting retentivity

Retentivity of tags in data blocks

Retentive behavior

To prevent data loss in the event of power failure, you can mark the data as retentive. This data is stored in a retentive memory area. The options for setting the retentivity depend on the type of data block and the type of block access that is set.

See also

Setting retentivity in an instance data block (Page 1716)

Setting retentivity in a global data block (Page 1717)

Setting retentivity in an instance data block

Introduction

In an instance data block, the editability of the retentive behavior depends on the type of access of the higher-level function block:

- Function block with standard access
  You can define the instance data both as retentive or non-retentive. Individual retentivity settings are not possible for individual tags.
- Function block with optimized access
  In the instance data block, you can define the retentivity settings of the tags that are selected in the block interface with "Set in IDB". With these tags also, you cannot individually set the retentive behavior for each tag. The retentivity setting has an impact on all tags that are selected in the block interface with "Set in IDB".

Setting Retentivity for Standard Access

To centrally set the retentivity of all tags in the data block with standard access, follow these steps:

1. Open the instance data block.
2. Select the check box in the "Retain" column of a tag.
   All tags are defined as retentive.
3. To reset the retentivity setting for all tags, clear the check box in the "Retain" column of a tag.
   All tags will be defined as non-retentive.

Setting Retentivity for Optimized Access

To set the retentive behavior of the tags that are selected with "Set in IDB" in data blocks with optimized access, follow these steps:

1. Open the instance data block.
2. Select the check box in the "Retain" column of a tag.
   All tags selected with "Set in IDB" in the data block interface are defined as retentive.
3. To reset the retentivity setting for the tags, clear the check box in the "Retain" column of a tag.
   All tags selected with "Set in IDB" in the data block interface will be defined as non-retentive.

See also

Basics of block access (Page 1419)

Retentivity of tags in data blocks (Page 1716)

Setting retentivity in a global data block

Introduction

In a global data block, the editability of the retentive behavior depends on the type of access:

- Global data block with standard access
  You can define the data both as retentive or non-retentive. Individual retentivity settings are not possible for individual tags.
- Global data block with optimized access
  You can individually define the retentivity settings of the tags. For tags with structured data types, retentivity settings are transferred for all tag elements.

Setting Retentivity for Standard Access

To centrally set the retentivity of all tags in the data block with standard access, follow these steps:

1. Open the global data block.
2. Select the check box in the "Retain" column of a tag.
   All tags are defined as retentive.
3. To reset the retentivity setting for all tags, clear the check box in the "Retain" column of a tag.
   All tags are defined as non-retentive.

Setting Retentivity for Optimized Access

To individually set the retentivity of all tags in data blocks with optimized access, follow these steps:

1. Open the global data block.
2. In the "Retain" column, select the check box for the tags for which you want to set a retentive behavior.
   The selected tag is defined as retentive.
3. To reset the retentivity setting for the tags, clear the check box in the "Retain" column of a tag.
   All selected tags are defined as non-retentive.

See also

Basics of block access (Page 1419)

Retentivity of tags in data blocks (Page 1716)

Editing the properties of tags in data blocks

Properties of the tags in data blocks

Properties

The following table provides an overview of the properties of tags in data blocks:

Group	Property	Description
General	Name	Name of the tags.
	Data type	Data type of the tags.
	Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
	Start value	Value that the tag should assume at CPU startup. The default values defined in a code block are used as start values during the creation of the data block. You can then replace these adopted values with instance-specific start values. Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.
	Comment	Comment on the tag.

Group	Property	Description
Attributes	Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This attribute is only available in the interface of the function block with optimized access.
	Accessible from HMI	Indicates whether the tag can be used in HMI. When the attribute is set, you have read or write access to the tag from the HMI. When the attribute is not set, you cannot access the tag from the HMI. Please note, however, that you cannot implement general access protection for the tag with the "Accessible from HMI" attribute. Read or write access from other applications is possible even if the attribute is not enabled.
	Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
	Hidden parameter	Indicates whether the tag should be hidden for the block call. This is only possible if you have specified a valid predefined actual parameter beforehand.
	Predefined actual parameter	Defines a parameter that is to be used as actual parameter during the block call.
	Visible	Indicates whether a parameter is visible in CFC.
	Configurable	Indicates whether a parameter is configurable in CFC.
	For test	Indicates whether a parameter is registered for the CFC test mode.
	Interconnectable	Indicates whether a parameter is interconnectable in CFC.
	Enable tag readback	Indicates whether a parameter is relevant for the "Read back chart" function in CFC.
	Enumeration texts	Assigns a parameter to an enumeration in CFC.
	Engineering unit	Assigns a parameter to a unit in CFC.
	Low limit	Defines the low limit for the parameter in CFC.
	High limit	Defines the high limit for the parameter in CFC.

See also

Changing the properties of tags in instance data blocks (Page 1720)

Changing the properties of tags in global data blocks (Page 1721)

Changing the properties of tags in instance data blocks

Instance-specific tag properties

Two options are available for defining the tag properties:

- The tag properties are applied from the interface of the assigned function block.
  Properties that are applied from the function block are displayed grayed out out in the columns of the declaration table. The "Name" and "Data type" properties are always applied.
- You define instance-specific properties.
  You can change some properties instance specific. Changeable values are, for example, "Comment" or "Visible in HMI". Properties that were changed instance specific are not grayed out in the columns of the declaration table. The instance-specific changes are retained, even if the interface of the higher-level function block is changed and the instance data blocks are subsequently updated.

Editing properties of an element in the declaration table

To edit the properties of an element, follow these steps:

1. Open the instance data block.
2. Select the required element in the table.
3. Change the entries in the columns.

Editing properties of several elements in the declaration table

You can also simultaneously set or reset the "Retain", "Visible in HMI", "Accessible in HMI" and "Setpoint" columns for one or more selected elements.

To change one of these properties for several elements, follow these steps:

1. Open the data block.
2. Hold down the CTRL key.
3. In the required column, select each of the table cells whose value you want to change.
4. Select the "Set <property>" or "Reset <property>" command in the shortcut menu.

Editing properties in the properties window

To edit the properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.
   The properties window opens. It shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

Reset individual properties to the default value.

To reset individual tag properties to the value that was defined as default in the function block, follow these steps:

1. Select an instance-specific, modified value in the table.
2. Delete the value.
   The instance-specific value will be deleted and the default value from the interface of the function block entered. The default value is displayed grayed out.

See also

Updating data blocks (Page 1708)

Properties of the tags in data blocks (Page 1718)

Changing the properties of tags in global data blocks

Introduction

Two options are available for defining the tag properties:

- The tag properties are applied from the PLC data type.
  Properties that are applied from the PLC data type are shown grayed out in the columns of the declaration table. The "Name" and "Data type" properties are always applied.
- You define specific properties.
  You can change some properties in the global data block. Changeable values are, for example, "Comment" or "Visible in HMI". Properties that were changed are not grayed out in the columns of the declaration table. The changes are retained, even if the PLC data type changes and the global data block is subsequently updated.

Editing properties of an element in the declaration table

To edit the properties of an element, follow these steps:

1. Open the global data block.
2. Select the required element in the table.
3. Change the entries in the columns.

Editing properties of several elements in the declaration table

You can also simultaneously set or reset the "Retain", "Visible in HMI", "Accessible in HMI" and "Setpoint" columns for one or more selected elements.

To change one of these properties for several elements, follow these steps:

1. Open the data block.
2. Hold down the CTRL key.

3. In the required column, select each of the table cells whose value you want to change.
4. Select the "Set <property>" or "Reset <property>" command in the shortcut menu.

Editing properties in the properties window

To edit the properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.
   The properties window opens. It shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

Reset individual properties to the default value.

To reset individual tag properties to the value that was defined as default in the PLC data type, follow these steps:

1. Select a modified value in the table.
2. Delete the value.
   The default value from the PLC data type is entered. The default value is displayed grayed out.

See also

Properties of the tags in data blocks (Page 1718)

Editing the declaration table for data blocks

Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

See also

Editing tables (Page 347)

Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

See also

Editing tables (Page 347)

Deleting tags

Requirements

A global data block is open.

Procedure

To delete a tag, follow these steps:

1. Select the row with the tag to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

Note

You cannot directly change the structure of instance data blocks and of global data blocks based on a PLC data type, since the structures of these blocks are defined in the higher-level object.

The type of the data block is entered in the block properties.

See also: Displaying and editing block properties (Page 1523)

See also

Editing tables (Page 347)

Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.
5. To hide or show several columns, click "More" and activate or deactivate the check box of the corresponding columns in the "Show/Hide" dialog.

Editing tags with external editors

To edit individual tags in external editors outside the TIA Portal, you can export or import these tags using copy and paste. However, you cannot copy structured tags to an editor.

Requirement

The data block and an external editor are opened.

Procedure

To export and re-import individual tags by drag-and-drop operation, follow these steps:

1. Select one or more tags.
2. Select "Copy" in the shortcut menu.
3. Switch to the external editor and paste the copied tags.
4. Edit the tags as required.
5. Copy the tags in the external editor.
6. Switch back to the declaration table.
7. Select "Paste" in the shortcut menu.

Monitoring and modifying tags in data blocks

Functions for monitoring and modifying tags in data blocks

Overview of functions

The data block editor offers different options for monitoring and modifying tags. These functions directly access the actual values of the tags in the online program. Actual values are the values which the tags have at the current time during program execution in the CPU work memory.

The following table provides an overview of the functions for monitoring and modifying. Detailed descriptions of the individual functions can be found in the following chapters.

Button	Function	Description	S7-300/400	S7-1200/1500
	Monitor tags online (Page 1726)	Displays the actual values which the tags currently have in the CPU.	X	X
-	Modify individual actual values (Page 1726)	Modifies individual tags immediately and once only to specific values in the declaration table. The CPU then uses these values as actual values in the online program.	X	X

Button	Function	Description	S7-300/400	S7-1200/1500
	Create a snapshot of the actual values (Page 1727)	Saves the actual values present at the current time as snapshot. The snapshot always captures the actual values of all tags of the data block.	X	X
	Overwrite actual values with a snapshot (Page 1728)	Overwrites the actual values of all tags of the data block with a snapshot. The CPU then uses these values as actual values in the online program.	-	S7-1200 V4.1 and higher S7-1500 V1.7 and higher

Additional functions

You also have the option of specifically calibrating individual values during commissioning.

See also: Auto-Hotspot

Monitor tags

You can monitor the current values of the tags in the CPU directly in the declaration table.

Requirement

- An online connection is available.
- The data block has been loaded to the CPU.
- The program execution is active (CPU in "RUN").
- The data block is open.

Procedure

To monitor the tags, follow these steps:

1. Start monitoring by clicking the "Monitor all" button.
   The additional "Monitor value" column is displayed in the table. This shows the current data values.
   See also: Structure of the declaration table for data blocks (Page 1705)
2. End the monitoring by clicking the "Monitor all" button again.

Modify tags

You can modify an individual tag in the data block to a specific value. The CPU then uses this values as actual value in the online program.

Danger

Danger when changing tag values

Changing the tag values while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors!

Make sure that no dangerous situations can arise before you modify the tags.

Requirement

- An online connection to the CPU is available.
- The data block whose tags you wish to modify is identical offline and online.
- The data block is open.

Procedure

To modify an individual tag in the data block, follow these steps:

1. Start monitoring by clicking the "Monitor all" button.
   The additional "Monitor value" column is displayed in the table. This shows the current data values.
2. Select the tag to be modified.
3. Select "Modify operand" in the shortcut menu.
   The "Modify operand" dialog opens.
4. Enter the required value in the "Control word" text box.
   If there is a snapshot, the value is already entered as default.
5. Confirm your entry with "OK".

Result

The tag will have the specified value once when you execute the modify job. The job is executed immediately and is not tied to the next cycle control point.

See also: Introduction to modifying tags (Page 1873)

Create a snapshot of the actual values

You can store the actual values of one or more data blocks as snapshot. Actual values are the values which the tags have at the current time during program processing in the CPU work memory.

You have the following options for creating a snapshot:

- Creating a snapshot of an open data block
- Creating a snapshot of several selected data blocks

A snapshot is also created automatically when you download a block or a program from the device.

Caution
Creating the snapshot

The values in the snapshot may originate in several cycles.

Requirement

- An online connection to the CPU is available.
- The data blocks for which you want to create a snapshot are identical offline and online.

Procedure

To create a snapshot of an open data block, follow these steps:

1. Open the data block.
2. Click "Snapshot of the monitored values".

To create a snapshot of several selected data blocks, follow these steps:

1. Select the blocks in the project tree.
   You can select the blocks individually or select devices, groups or folders in the project tree which include the data blocks.
2. In the shortcut menu, select "Snapshot of the monitored values" or select the menu command "Options > Snapshot of the monitored values".

Result

- The latest monitored values will be applied in the "Snapshot" column.
- An alarm is shown in the Inspector window after the operation is complete.
- The time stamp of the snapshot is displayed above the declaration table.

Note

If you subsequently change the structure of the data block, the display of the current values gets lost. The "Snapshot" column will then be empty.

Overwrite actual values with a snapshot

You can overwrite the actual values of all tags with a snapshot. The values from the snapshot are written directly to the CPU work memory. The CPU then uses these values as actual values in the online program.

Danger

Danger when changing tag values

Changing the tag values while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors!

- Make sure that the plant is in a safe state before you overwrite the actual values.
- Make sure that the program does not read or write the affected data during transmission.
- You may want to use the "Modify tag" function in the watch table or in the DB editor as an alternative.

Dependencies on the CPU mode

You can execute this function in "RUN" mode as well as "STOP" mode. The table below shows the reactions of the CPU in the different modes:

Action	System reaction	Consequences for the online program
Overwrite actual values in "RUN" mode	The values of all DB tags are overwritten in the current program. No distinction is made between retentive and non-retentive values.	Changing the actual values can result in inconsistencies between the program and the actual process. If the transmitted amount of data is too large, the values may be transmitted in several cycles. If the program accesses tags before all values are completely transmitted, there is a risk that inconsistent value combinations may be created and processed. Even copying the values of elementary data types may take several cycles. These values are potentially invalid until they have been completely transmitted. Dangerous states may occur if the program accesses these values before they have been completely transmitted.
Overwrite actual values in "STOP" mode	Only the actual values of the retentive tags are overwritten by the snapshot. Non-retentive tags are initialized with their start values during the transition from STOP to RUN. The values from the snapshot are not taken into consideration.	Because only the data from the snapshot are transmitted, there is a risk that inconsistent value combinations may be created and processed.

Requirement

- You are using an S7-1200 V4.1 or higher or an S7-1500 V1.7 or higher.
- An online connection to the CPU is available.
- The data blocks whose actual values you wish to overwrite are identical offline and online.
- A snapshot of the data block exists.

Procedure

To overwrite the actual values of a block with a snapshot, follow these steps:

1. Open the data block.
2. Click "Copy all values from the snapshot to the actual values of the CPU".

Result

The actual values in the online program are overwritten with the start values from the snapshot.

Setting data values during commissioning

Basic information on adjusting data values during commissioning

Introduction

During commissioning of a plant, data values have to be frequently adjusted in order to optimally adapt the program to the general operating conditions on site. The declaration table for data blocks offers some functions for this purpose.

To use the function, first define specific tags as "Setting values" in the program. Setting values are the values that will probably have to be fine tuned during commissioning.

The following table provides an overview of the functions for tuning values during commissioning. Detailed descriptions of the individual functions can be found in the following chapters.

Button	Function	Description
	Initialize setting values in "RUN" mode (Page 1732)	This function enables you to change the values of individual tags online to quickly determine the optimum tag values.
	Apply values from the online program as start values to the offline program	When you have determined the optimum tag values, you can apply these as start values in the offline program. This allows you to ensure that the program starts with the optimized values the next time it is loaded. You can either apply all values or only the setting values.

Additional functions

There are still some more general functions for monitoring and modifying the data block.

See also: Functions for monitoring and modifying tags in data blocks (Page 1725)

See also

Marking data as values that can be set (Page 1731)

Applying values from the online program as start values (Page 1733)

Marking data as values that can be set

You can mark specific tags in the program as "Setting values". Setting values are the values that will probably have to be fine tuned during commissioning.

Rules

You can mark tags as "Setting value" in the following block types:

- In function blocks (FB), but only in the "Static" section
- in global data blocks (DB)
- in PLC data types (UDT)
  In the case of PLC data types (UDT), however, the setting is only effective, if the UDT is used in the "Static" section of a function block or data block.

It is not possible to define setting values in the following block types:

- In data blocks based on a PLC data type, and in instance data blocks. These inherit the setting from the higher-level FB or UDT.
- You cannot mark tags as a "Setting value" in ARRAY data blocks.
- You also cannot mark tags as a "Setting value" at the call point of a multi-instance. You have to make the setting in the interface of the function block that is called as multiple instance.
- You cannot change the "Setting value" marking in know-how-protected blocks. To do so, you must first remove the know-how protection.

Requirement

A function block, a global data block or a PLC data type (UDT) is open.

Procedure

To mark a tag as "Setting value", follow these steps:

1. Select a tag from the "Static" section.
2. Select the check box in the "Setting value" column.
   - You cannot define the higher-level element of a structure or a PLC data type as "Setting value". You have to make the setting for the lower-level elements individually.
   - In the case of ARRAYs, you can only mark the higher-level element as "Setting value". The lower-level elements inherit the setting.
   - For ARRAYs of STUCT, you can only mark the elements below the first structure as setting values. The elements of other structures inherit the setting.

Result

The tags are marked as setting values. During commissioning, these tags can be initialized online. You do not need to set the CPU to "STOP" mode; it can remain in "RUN". In addition, the current tag values can be transferred as start values to the offline program and saved there.

Initializing setting values in the online program

Basics on initializing setting values

You can initialize all tags marked as a "Setting value" with new values in the online program. At the same time, the start values will be loaded from the offline program to the online program. The CPU remains in "RUN" mode. All tags that are marked as a setting value are initialized once at the next cycle control point. This applies both to retentive and non-retentive tags. The program execution is then continued with the new tag values.

Danger

Danger when changing tag values

Changing the tag values while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors!

Make sure that no dangerous situations can arise before you re-initialize the setting values.

Requirement

- An online connection to the CPU exists.
- The structure of the data block is identical offline and online.
- One or more tags are marked as a "Setting value".

Procedure

To initialize all setting values of the data block, follow these steps:

1. Open a global data block or an instance data block.
2. Enter the desired values in the "Start value" column. The start values must correspond to the indicated data type.
3. Click the "Initialize setting values" button.

Result

The setting values in the online program are initialized with the start values from the offline program at the next cycle control point.

The maximum number of tags that can be initialized is dependent on the CPU. If too many setting values are marked, an alarm informs you about this. In this case, you can insert the tags in a watch table and initialize them using the "Modify" function in the watch table.

Alternatively, you can also load the entire data block. For more information on this, refer to "See also".

See also

Loading changed values (Page 1715)

Applying values from the online program as start values

In order to apply tag values from the online program to the offline program as start values, first create a snapshot of the tag values from the online program. You can then apply them to the offline program. Note that the values from the snapshot are always copied. There is no check to determine whether all values originate from the same cycle.

Write-protected start values are not overwritten.

You have the following basic options for applying the values:

- Applying the values of an open data block
  You can apply all values or only the values of the tags marked as a "setting value" as start values in an open data block.
- Applying the values of multiple blocks in the project tree
  You can either apply all setting values or all retentive values as start values in the project tree.

Requirement

- An online connection to the CPU is available.
- As least one data block has been loaded to the CPU.

Procedure

In order to apply all values or only the values of the tags marked as a "setting value" in a data block, follow these steps:

1. Open the data block.
2. Start monitoring by clicking the "Monitor all" button.
   The "Monitor value" column is displayed in the table. This shows the current data values.
3. On the toolbar, click "Snapshot of monitored values".
   The latest monitored values will be applied in the "Snapshot" column.
4. Click one of the following buttons on the toolbar:
   – "Apply setting values from the snapshot as start values"
   – "Apply all values from the snapshot as start values"

The values from the "Snapshot" column are applied to the "Start value" column.

To apply the monitored values of multiple data blocks in the project tree, follow these steps:

1. Select the blocks in the project tree.
2. Select the "Snapshot of the monitored values" command in the shortcut menu.
   The current monitored values of all selected blocks will be applied in the "Snapshot" column.
   An alarm is shown in the Inspector window after the operation is complete.
3. Then select one of the following commands in the shortcut menu:
   - "Apply snapshot values as start values > All values"
   - "Apply snapshot values as start values > Only setpoints"
   - "Apply snapshot values as start values > Only retain values"

The values from the "Snapshot" column are applied to the "Start value" column.

Result

The new start values are stored in the offline program.

Note

Applying values of individual tags

You can also transfer the values of individual tags that were not marked as a setting value beforehand from the "Snapshot" column to the "Start values" column. Use the "Copy" and "Paste" commands from the shortcut menu to copy the values and insert them in the "Start value" column. Note that only the values that are currently located in the visible area of the table are copied.

See also

Basic information on start values (Page 1713)

Define start values (Page 1714)

11.1.4.4 Programming PLC data types

Basics of PLC data types

Description

PLC data types are data structures that you define and that can be used multiple times within the program. The structure of a PLC is made up of several components, each of which can contain different data types. You define the type of components during the declaration of the PLC data type.

You can create up to 65534 PLC data types for a CPU of the S7-1200 or S7-1500 series. Each of these PLC data types can include up to 252 components.

PLC data types can be used for the following applications:

- PLC data types can be used as data types for variables in the variable declaration of logic blocks or in data blocks.
- PLC data types can be used as templates for the creation of global data blocks with identical data structures.
- PLC data types can be used in S7-1200 and S7-1500 as a template for the creation of structured PLC tags.

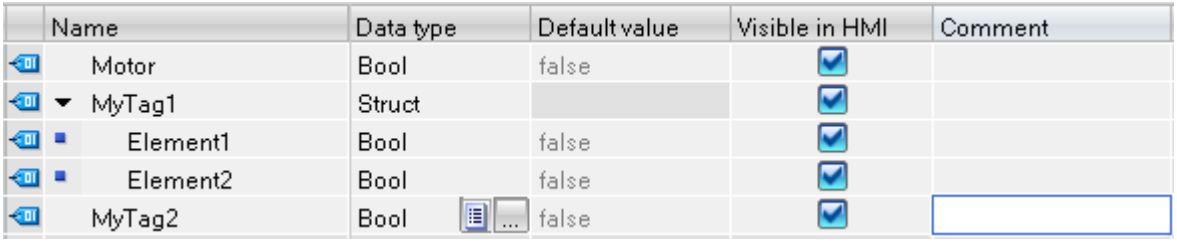
See also

Creating PLC data types (Page 1736)

Structure of the declaration table for PLC data types

Structure of the declaration table for PLC data types

The figure below shows the structure of the declaration table for PLC data types.



Meaning of the columns

The following table shows the meaning of the individual columns. You can show or hide the columns as required. The number of columns displayed varies depending on the CPU series.

Column	Explanation
	Symbol you can click to move or copy the tag.
Name	Name of the tags.
Data type	Data type of the tags.
Default value	Value with which you predefine the tag in the declaration of the PLC data type. Specification of the default value is optional. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL.
Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
Accessible from HMI	Shows whether HMI can access this tag during runtime.

Column	Explanation
Setting value	Setting values are the values that will probably have to be fine tuned during commissioning. After commissioning, the values of these tags can be transferred to the offline program as start values and stored there.
Comment	Comment to document the tags.

See also

Creating PLC data types (Page 1736)

Show and hide table columns (Page 1744)

Creating PLC data types

Requirement

The "PLC data types" folder opens in the project tree.

Procedure

To create a PLC data type, proceed as follows:

1. In the "PLC data types" folder, click the "Add new data type" command.
   A new declaration table for creating a PLC data type will be created and opened.
2. Select the PLC data type and select the "Rename" command in the shortcut menu.
3. Enter the name of the PLC data type.

Result

The new PLC data type is created. You can find the PLC data type in the project tree in the "PLC data types" folder.

See also

Structure of the declaration table for PLC data types (Page 1735)

Basics of PLC data types (Page 1734)

Delete PLC data types

Requirement

The PLC data type you want to delete is not open.

Procedure

To delete a PLC data type, follow these steps:

1. In the project tree, open the "PLC data types" folder.
2. Select the PLC data type to be deleted. You can also select several PLC data types by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last data type.
3. Select the "Delete" command in the shortcut menu.

Note

If you delete a PLC data type, the blocks that use the data type will become inconsistent. These inconsistencies are marked in red in the block used. To remedy these inconsistencies, the data blocks have to be updated.

See also:

Updating the block interface (Page 1564)

Updating data blocks (Page 1708)

Renumbering PLC data types

For performance reasons, PLC data types are processed internally with numbers. If there are number conflicts, these are resolved automatically. But this is not possible for a PLC data type which is used by a know-how protected block. The block must be recompiled when you change the number of the PLC data type which results in a password prompt for the know-how protected block. You can bypass this step by setting up a separate numbering scheme for your PLC data types. Use numbers greater than 5000.

Procedure

To change the default number of a PLC data type, follow these steps:

1. Open the project library in the "Libraries" task card.
2. Drag the compilable PLC data type to the "Types" folder.
   The "Add type" dialog opens.
3. Enter the properties of the new type.
4. Click "OK" to confirm.
5. Right-click the PLC data type in the project library and select the "Edit type" command from the shortcut menu.
6. Click "OK" to confirm your selection of the instance.
   The library view opens.
7. Close the library view.
   The extension "in testing" is now added to the name of the PLC data type.
8. Right-click the PLC data type and select the "Properties" command from the shortcut menu.

9. Select the "General" group in the area navigation.
   You can now edit the number of the PLC data type.
10. Change the number of the PLC data type.
11. Click "OK" to confirm.
12. Right-click the PLC data type in the project library and select the "Release version" command from the shortcut menu.
    The PLC data type has a new number. The assigned number is retained even if the type of the PLC data type is revoked.

Programming the structure of PLC data types

Declaring tags of elementary data type

Requirement

A PLC data type is open.

Procedure

To declare a tag, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
3. Optional: Change the properties of the tags that are displayed in the other columns.
4. Repeat steps 1 to 3 for all tags that are to be declared.

See also

Editing tables (Page 347)

Declaring tags of the ARRAY data type

Requirement

A PLC data type is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the "Array" data type in the "Data type" column. You will be supported by autocompletion in this step.
   The "Array" dialog opens.
3. In the "Data type" text box, specify the data type of the array elements.
4. In the "ARRAY limits" text box, specify the high and low limit for each dimension.
   Example of a one-dimensional ARRAY:
   `[0..3]`
   Example of a three-dimensional ARRAY:
   `[0..3, 0..15, 0..33]`
5. Confirm your entry.
6. Optional: Change the properties of the tags that are displayed in the other columns.

Note

You cannot define specific default values for ARRAY elements. You can, however, assign them start values at the usage point in the data block.

See also

Array (Page 1941)

Structure of the declaration table for PLC data types (Page 1735)

Declaring tags of STRUCT data type

Requirements

A PLC data type is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.
   An empty, indented row is inserted after the new tag.
3. Insert the first structural element in the first empty row.
   An additional empty row is inserted after the element.
4. Select a data type for the structure element.

5. Optional: Change the properties of the structural element that is displayed in the other columns.
6. Repeat steps 3 to 5 for all additional structure elements.
   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
7. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

See also

STRUCT (Page 1945)

Structure of the declaration table for PLC data types (Page 1735)

Declaring tags based on a different PLC data type

Requirements

- A global data block is open.
- A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a different PLC data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the PLC data type in the "Data type" column. You will be supported by autocompletion during input.

Result

The tag is created.

Note

You define the default values of tags within a PLC data type when the PLC data type is created. You cannot change these values at the point of use of the PLC data type.

See also

Basics of PLC data types (Page 1734)

Structure of the declaration table for PLC data types (Page 1735)

Editing tag properties in PLC data types

Properties of tags in PLC data types

Properties

The following table gives an overview of tag properties in PLC data types:

Group	Property	Description
General	Name	Name of the tags.
	Data type	Data type of the tags.
	Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
	Start value	Not relevant in PLC data types
	Comment	Comment on the tag.
Attributes	Retain	Not relevant in PLC data types
	Visible	Indicates whether a parameter is visible in CFC.
	Configurable	Indicates whether a parameter is configurable in CFC.
	For test	Indicates whether a parameter is registered for the CFC test mode.
	Interconnectable	Indicates whether a parameter is interconnectable in CFC.

See also

Changing the properties of tags in PLC data types (Page 1741)
Basics of PLC data types (Page 1734)
Structure of the declaration table for PLC data types (Page 1735)

Changing the properties of tags in PLC data types

Editing general properties in the declaration table

To edit the general properties of one or more tags, follow these steps:

1. Open the PLC data type.
2. Change the entries in the columns.

Editing detailed properties in the properties window

To edit the detailed properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.
   The inspector window shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

See also

Updating the block interface (Page 1564)

Updating data blocks (Page 1708)

Editing the declaration table for PLC data types

Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

Deleting tags

Procedure

Follow the steps below to delete elements:

1. Select the row with the element to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

See also

Updating the block interface (Page 1564)

Updating data blocks (Page 1708)

Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.
5. To hide or show several columns, click "More" and activate or deactivate the check box of the corresponding columns in the "Show/Hide" dialog.

11.1.4.5 Using external source files

Basics of using external source files

Function

The textual programming languages STL and SCL allow you to enter the program code in any ASCII editor and save it as an external source file. This enables you to perform a range of tasks, for example:

- Declaring tags
- Specify block properties
- Programming blocks

You can import these source files to your project and use them to generate blocks. You can generate a number of different blocks from one source file. Observe the following special features when generating blocks from a source file:

- A block that exists under the same name in the project will be overwritten.
- If a block was programmed with its absolute block number instead of a symbolic name in the source file and this number is already assigned by a block in the project, the new generated block is initially assigned the next free symbolic name.

- If you have not explicitly defined the access mode for a block in the external source file, the block access mode is set depending on the CPU series used:
  - Blocks generated for a CPU of the S7-1200/1500 series are assigned "optimized" access mode by default.
  - Blocks generated for a CPU of the S7-300/400 product range are assigned "standard" access mode by default.

  Organization blocks are the exception in this case, as they are always assigned the "standard" access mode by default, regardless of the CPU series. You have the option of changing the block access mode manually.
- It is possible that not all comments from the source file will be applied in the block.
- If you use absolute addressing in the external source file, a symbolic tag is created for each absolute address during the generation of the block. The names of these tags are made up of "Tag_" and a time stamp. This may result in relatively long tag names, which you can change manually if required.
- If you are using instructions in a different version in the external source file than in the target device, this may result in compilation errors. Correct the respective instructions in this case and start the compilation process once again. You can also select a different version for the target device.

You also have the option of saving existing blocks as external source files.

See also

Rules for programming external source files (Page 1745)

Saving blocks as external source files (Page 1746)

Inserting external source files (Page 1748)

Opening and editing external source files (Page 1748)

Generating blocks from external source files (Page 1749)

Rules for programming external source files

An external source file basically consists of continuous text. To compile the source into blocks, certain structures and syntax rules must however be adhered to.

Syntax rules

The syntax of the instructions in external source files is very similar to that in the creation of user programs in the program editor with STL or SCL. Note, however, the following additional syntax rules:

- Block call
  When calling a block, transfer the parameters in the defined order in the ASCII editor. If you do not, the comment assignments for these lines may not match.
  Enter the parameters in brackets. The individual parameters are separated by a comma.
- Upper or lower case
  The program editor generally disregards upper or lower case. Jump labels are an exception to this. Character string entries are also case-sensitive ("STRING" data type). Keywords are displayed in upper case. For compilation purposes, however, case is disregarded; you can therefore specify keywords in upper or lower case or a mixture of the two.
- Semicolon
  Mark the end of every instruction and every tag declaration with a semicolon. You can enter several instructions per line.
- Forward slashes
  Begin every comment with two forward slashes (//) and end the comment with the <Enter> key.
- Use of String constants
  To avoid compilation errors when using String constants, enter the text in the language of the target project. You can explicitly use the data type WSTRING for CPUs of the S7-1200/1500 series by using the prefix "WString#":
  `Operand := WString#'<String constant>';`

See also

Basics of using external source files (Page 1744)

Saving blocks as external source files (Page 1746)

Inserting external source files (Page 1748)

Opening and editing external source files (Page 1748)

Generating blocks from external source files (Page 1749)

Saving blocks as external source files

You have the following options for saving STL and SCL blocks as external source files:

- Copying a block as text
- Generating external source file from blocks

Copying a block as text

To copy a block as text and save it to an external source file, follow these steps:

1. In the project tree, right-click on the block that you want to save in an external source file.
2. Select the "Copy as text" command in the shortcut menu.

3. Open an external text editor.
4. Paste the copied text from the clipboard.
5. Save the file with one of the following file name extensions:
   - ".scl", if you want to generate a source file for an SCL block
   - ".stl", if you want to generate a source file for an STL block
   - ".DB" if you wish to generate a source file for a data block
   - ".UDT", if you want to generate a source file for a PLC data type

Generating external source file from blocks

To generate external source files from STL or SCL blocks, follow these steps:

1. In the project tree or in the overview window, select the STL or SCL blocks from which you want to generate an external source file.
2. Select the "Generate source from blocks" command from the shortcut menu.
   The "Generate source from blocks" dialog opens.
3. Specify a path and a name for the external source file.
4. Click "OK".

You can also generate an opened STL or SCL block as external source file. To do this, follow these steps:

1. Click on the "Generate source from block" button in the program editor.
   The "Generate source from blocks" dialog opens.
2. Specify a path and a name for the external source file.
3. Click "OK".

Result

The block has been saved as an external source file. You can include this source file in a project in the TIA Portal and use it to generate other blocks. However, please note that you can use STL source files only in S7-300/400/1500 CPUs.

See also

Basics of using external source files (Page 1744)

Rules for programming external source files (Page 1745)

Inserting external source files (Page 1748)

Opening and editing external source files (Page 1748)

Generating blocks from external source files (Page 1749)

Inserting external source files

Requirement

- An external source file is available and complies with the syntax and structure rules.
- The "External source files" folder is open in the project tree.

Procedure

Follow these steps to insert an external source file:

1. Double-click on the "Add new external file" command.
   The "Open" dialog box is opened.
2. Navigate to and select existing external source files.
3. Confirm your selection with "Open".

Result

The new source file will be added to the "External source files" folder.

See also

Basics of using external source files (Page 1744)

Rules for programming external source files (Page 1745)

Saving blocks as external source files (Page 1746)

Opening and editing external source files (Page 1748)

Generating blocks from external source files (Page 1749)

Opening and editing external source files

By linking the files with the file name extensions "stl" and "scl" to an editor you will be able to open and edit external source files with these formats directly. Use Notepad as editor because other text editors may not let you open several sources at the same time.

This means you do not need to insert the external source files again after editing.

Linking files with the file name extensions "stl" and "scl" file types to an editor

Proceed as follows to link files with the file name extensions "stl" and "scl" to an editor:

1. Open Windows Explorer.
2. Right-click on an STL file.
3. Select "Properties" in the shortcut menu.
   The "Properties" dialog box opens.
4. Click "Change" in the "File type" area on the "General" tab.
   The "Open with" dialog box opens.

5. Select the text editor you want to link to the "stl" file type.
6. Confirm your selection with "OK".
7. Close the "Properties" dialog with "OK".
8. Repeat steps 2 to 7 with an SCL file.

Opening and editing an external source file

To open an external source file, follow these steps:

1. Open the "External source files" folder in the project tree.
2. Double-click on the external source file you want to open.
   The external source file will open in the linked editor and can be edited.

See also

Basics of using external source files (Page 1744)

Rules for programming external source files (Page 1745)

Saving blocks as external source files (Page 1746)

Inserting external source files (Page 1748)

Generating blocks from external source files (Page 1749)

Generating blocks from external source files

Requirement

- The "External source files" folder is open in the project tree.
- An external source file is available.

Procedure

To generate blocks from an external source file, follow these steps:

1. Open the external source file from which you wish to generate blocks.
2. Select the "Generate blocks from source" command in the "Edit" menu.
3. A prompt will appear telling you any existing blocks will be overwritten.
4. Confirm the safety prompt with "Yes".

Result

The external source file blocks will be generated and inserted in the "Program blocks" folder in the project tree. In the event of errors, information about the errors which have occurred will be displayed in the inspector window. This information, however, relates to the external source file and not to the block generated.

See also

Basics of using external source files (Page 1744)

Rules for programming external source files (Page 1745)

Saving blocks as external source files (Page 1746)

Inserting external source files (Page 1748)

Opening and editing external source files (Page 1748)

11.1.5 Comparing PLC programs

11.1.5.1 Basic information on comparing PLC programs

Introduction to comparing PLC programs

Function

You can compare the following objects of a PLC program in order to detect any differences:

- Code blocks with other code blocks
- Data blocks with other data blocks
- PLC tags of a PLC tag table with the PLC tags of another PLC tag table
- PLC data types with other PLC data types

Types and levels of comparison

Two different basic types of comparison can be used:

- Online/offline comparison:
  The objects in the project are compared with the objects of the corresponding device. An online connection to the device is necessary for this comparison.
- Offline/offline comparison:
  The objects of two devices either within a project or from different projects or libraries are compared. No online connection is required for this comparison.

Please note that you cannot carry out an unlimited number of comparisons at the same time, but only one comparison per comparison type (online/offline or offline/offline).

You can choose between the following levels of comparison depending on how in-depth an object comparison you require:

- Compare editor
- Detailed comparison

When you start a comparison, you will first receive an overview in the compare editor. For some comparison objects, you can then start a detailed comparison in which the objects

compared will be opened side-by-side, each in its own program editor instance. Any differences will be highlighted.

The table below gives an overview of the types and levels of comparison you can apply for each object:

Object	Online/offline		Offline/offline	
Compare editor	Detailed comparison	Compare editor	Detailed comparison	
LAD block	X	X	X	X
FBD block	X	X	X	X
STL block[1]	X	X	X	X
SCL block	X	X[3] [4]	X	X
GRAPH block[2]	X	X[4]	X	X[5]
Global data block	X	X	X	X
Instance data block	X	X	X	X
PLC tags	-	-	X	X
PLC data type	X[4]	X[4]	X	X

Legend:

X: available
-: not available

[1]: STL is not available for S7-1200

[2]: GRAPH is not available for S7-1200

[3]: not for S7-1200 prior to version 2.0

[4]: not for S7-300/400

[5]: only comparable with a GRAPH block from the same CPU family

Note

Please note the following:

- You cannot perform a detailed comparison for know-how protected blocks.
- If the detail comparison detects differences only with respect to the data types of local tags, with offline being an interrupt data type (C_ALARM C_ALARM_S C_ALARM_8 C_ALARM_8P C_ALARM_T C_AR_SEND C_NOTIFY C_NOTIFY_8P) and online a DWORD, this difference is not marked as such.
- You cannot run a detailed comparison for types and master copies from libraries.

See also

Basics of project data comparison (Page 402)

Comparison of code blocks (Page 1752)

Comparison of data blocks (Page 1753)

Comparing PLC tags and PLC data types (Page 1753)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Comparison of code blocks

Introduction

The blocks to be compared in a code block comparison are assigned for comparison on the basis of the following criteria:

- Online/offline comparison: Addresses, e.g. FB100
- Offline/offline comparison: Symbolic names of the blocks

The comparison involves an evaluation of the block time stamps. The results are displayed as an overview in the comparison editor. You can then use actions to define what is to be done about the differences. You can also start detailed comparisons for the individual blocks. The versions of a block compared are opened beside each other and the differences are highlighted.

For the comparison of code blocks, both the block interfaces and the individual networks are compared. Any differing tag names are also determined. All comments and other block attributes are excluded from an online/offline comparison.

If the block interface changes, the time stamp of the code block interface will also change. This change means a change in the time stamp of the program code. The first step in comparing block interfaces is therefore a comparison of the program code time stamps. If these time stamps are the same, it is assumed that the interfaces are the same. If the time stamps of the interfaces differ, the next step is to compare the data types of the interfaces, section by section. Multiple instances and PLC data types are included in the comparison. If the data types in the sections are the same, the start values of the tags are compared. All differences are displayed.

When networks are compared, first inserted or deleted networks are detected. Then the other networks are compared. Instructions are the same if the operator and operand are the same. The first difference in each instruction is displayed. However, several differences per network can be displayed.

See also

Introduction to comparing PLC programs (Page 1750)

Comparison of data blocks (Page 1753)

Comparing PLC tags and PLC data types (Page 1753)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Comparison of data blocks

Introduction

The blocks to be compared in a data block comparison are assigned for comparison on the basis of the following criteria:

- Online/offline comparison: Addresses, e.g. DB100
- Offline/offline comparison: Symbolic names of the blocks

The first step in data block comparison is comparing the time stamps of the data block. If these time stamps are the same, it is assumed that the data structures are the same. If the time stamps differ, the structures are then compared until the first difference is found. If the data structures in the sections are the same, the initial values of the tags are then compared. All differences are displayed. Any differing tag names are also determined. Comments and PLC data type structures used in the data block are not included in the comparison.

See also

Introduction to comparing PLC programs (Page 1750)

Comparison of code blocks (Page 1752)

Comparing PLC tags and PLC data types (Page 1753)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Comparing PLC tags and PLC data types

Introduction

The device PLC tag tables and the device PLC data types will also be shown in the comparison editor if you carry out an offline/offline comparison. The PLC tag tables and the PLC data types will be matched by name and you will receive the following information:

- Status: A symbol shows whether the PLC tags/PLC data types are identical or differ.
- Missing PLC tag tables / PLC data types: You can see at a glance whether the PLC tag tables / PLC data types are available in both devices.

You obtain the following information with an online/offline comparison of CPUs of the S7-1200/1500 series:

- PLC tags: A symbol shows whether the PLC tags are identical or differ. Because PLC tag tables are not downloaded to the device during loading, they cannot be displayed during an online/offline comparison.
- PLC data types: You receive the status symbol for each PLC data type. You can see at a glance whether the PLC data types are available in both devices.

See also

Introduction to comparing PLC programs (Page 1750)

Comparison of code blocks (Page 1752)

Comparison of data blocks (Page 1753)

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

11.1.5.2 Comparing blocks

Comparing blocks in the compare editor

You have the following options for comparing blocks in the compare editor:

- Online/offline comparison
  The blocks in the project are compared with the blocks of the selected device.
- Automatic offline/offline comparison
  All blocks of the selected devices are compared offline.
- Manual offline/offline comparison
  The selected blocks of the devices are compared offline.

Carrying out an online/offline comparison of blocks

To perform an online/offline comparison, follow these steps:

1. In the project tree, select a device that allows online/offline comparison.
2. Select the "Compare > Offline/online" command in the shortcut menu.
3. If you have not already established an online connection to this device, the "Go online" dialog opens. In this case, set all the necessary parameters for the connection and click "Connect".
   The online connection is established and the compare editor opens.
4. Open the "Program blocks" folder.
   You can identify the status based on the symbols in the status and action area. You can define certain actions depending on the status of the objects. Note, however, that you can only perform actions in one direction in a synchronization action.

Carrying out an automatic offline/offline comparison of blocks

To perform an automatic offline/offline comparison of blocks, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.

3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. Open the "Program blocks" folder.
   You can identify the status of the objects based on the symbols in the status and action area. You can define certain actions depending on the status of the objects. When you select an object, the object's properties and the corresponding object of the assigned device are clearly shown in the properties comparison.

You can drag any other device to the drop area at any time to perform further comparisons.

Carrying out a manual offline/offline comparison of blocks

To perform a manual offline/offline comparison of blocks, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. In the status and action area, click on the button for switching between automatic and manual comparison.
5. Select the objects that you want to compare.
   The properties comparison is displayed. You can identify the status of the objects based on the symbols. You can define certain actions depending on the status of the objects.

You can drag any other device to the drop area at any time to perform further comparisons.

See also

Introduction to comparing PLC programs (Page 1750)

Using the compare editor (Page 405)

Comparing PLC tags (Page 1778)

Comparing PLC data types (Page 1780)

Performing detailed block comparisons

Start a detailed comparison for LAD/FBD/STL/SCL blocks

You can start a detailed comparison for blocks. The versions of a block compared are opened beside each other and the differences highlighted.

Note

Please note the following:

- For blocks that are created in the programming language SCL, the detail comparison is not available for S7-1200 series CPUs with a version older than 2.0.
- S7-1500: It is possible that another user can carry out a loading process on the CPU through joint parallel working on a CPU. If the currently compared block is changed or deleted by this loading process, the detailed comparison is terminated and a message is displayed. In this case, re-start the detailed comparison, if required.

Starting detailed comparisons using the compare editor

To start a detailed comparison for a block using the compare editor, follow these steps:

1. First, perform an online/offline or an offline/offline comparison.
   The compare editor opens.
2. In the compare editor, select the block for which you want to perform a detailed comparison.
3. Click "Start detailed comparison" in the toolbar.

Starting detailed comparisons in the program editor

For the comparison type online/offline, you can start the detailed comparison directly in the program editor. To do this, follow these steps:

1. Open the block for which you wish to carry out a detailed comparison.
2. Establish an online connection.
   See also: Go online and Go offline

   Note

   Please note that the block must be available online in order for you to be able to start the detailed comparison for the block within the programming editor.

3. Click "Detailed comparison" in the toolbar.
4. Confirm the dialog for closing the block with "Yes".

Result

One instance of the program editor will be opened for each version of the block compared and the two instances are displayed side by side. Any differences will be highlighted.

See also

Carrying out offline/offline comparisons (Page 404)

Carrying out an online/offline comparison (Page 403)

Using the compare editor (Page 405)

Visualization of the comparison result for LAD/FBD (Page 1759)

Visualization of the comparison result for STL (Page 1763)

Visualization of the comparison result for SCL (Page 1766)

Navigating in the detailed comparison (Page 1775)

Changing blocks during the detailed comparison (Page 1776)

Updating comparison results (Page 1778)

Start detailed comparison for GRAPH blocks

You can start a detailed comparison for GRAPH blocks. The versions of a block compared are opened beside each other and the differences highlighted.

The following comparison methods are available for GRAPH blocks:

- Compare sequence
  Complete sequencers are compared with each other in this mode. A detailed comparison for a GRAPH block always starts in the "Compare sequence" comparison mode. This means the comparison starts at the beginning of the sequencer and the differences between the sequencers are displayed. If there are structural differences between the sequencers, the comparison results are only displayed up to the first structural difference.
- Compare selection
  You can use the "Compare selection" comparison mode to compare any areas with each other. This means you can compare sections of the sequencer which are downstream from a structural difference in the sequence.

You can toggle at random between both comparison modes.

Note

S7-1500: It is possible that another user can carry out a loading process on the CPU through joint parallel working on a CPU. If the currently compared block is changed or deleted by this loading process, the detailed comparison is terminated and a message is displayed. In this case, re-start the detailed comparison, if required.

Procedure

To start a detailed comparison, follow these steps:

1. First, perform an online/offline or an offline/offline comparison between the two devices. The compare editor opens.
2. In the compare editor, select the block for which you want to perform a detailed comparison.

   Note

   The manual comparison is also available for an offline/offline comparison. It lets you select any block in the comparison editor for the comparison.

3. Click "Start detailed comparison" in the toolbar.
   One instance of the program editor is opened for each version of the blocks compared and the two instances are displayed side by side. Any differences will be highlighted. The comparison takes place in the "Compare sequence" mode.
4. To compare specific areas within the displayed sequencers, select the step from which you want to start the comparison in each sequencer.
5. Click "Comparison mode" in the toolbar.
   The comparison mode changes to "Compare selection" and the comparison result is updated. The sequencers are compared as of the selected steps. The compared substrings are highlighted.
6. To compare other areas, select the steps which are to serve as starting point for the comparison.
7. Click "Update the comparison result" in the toolbar.

   Note

   When you change the comparison mode, you are also updating the comparison results at the same time. If you stay in "Compare selection" mode, you must manually update the comparison results every time you change the starting point.

8. To compare the complete sequencers once again, click "Comparison mode" in the toolbar. Each time you click "Comparison mode" you are changing the comparison mode. You can also open the drop-down list with the arrow and select the required mode from the list.

See also

Carrying out offline/offline comparisons (Page 404)

Carrying out an online/offline comparison (Page 403)

Using the compare editor (Page 405)

Visualization of the comparison result for GRAPH (Page 1769)

Navigating in the detailed comparison (Page 1775)

Changing blocks during the detailed comparison (Page 1776)

Updating comparison results (Page 1778)

Visualization of the comparison result

Visualization of the comparison result for LAD/FBD

Introduction

The detailed comparison allows you to identify the exact places where versions of a block differ. The following color coding allows you to find these places as quickly as possible:

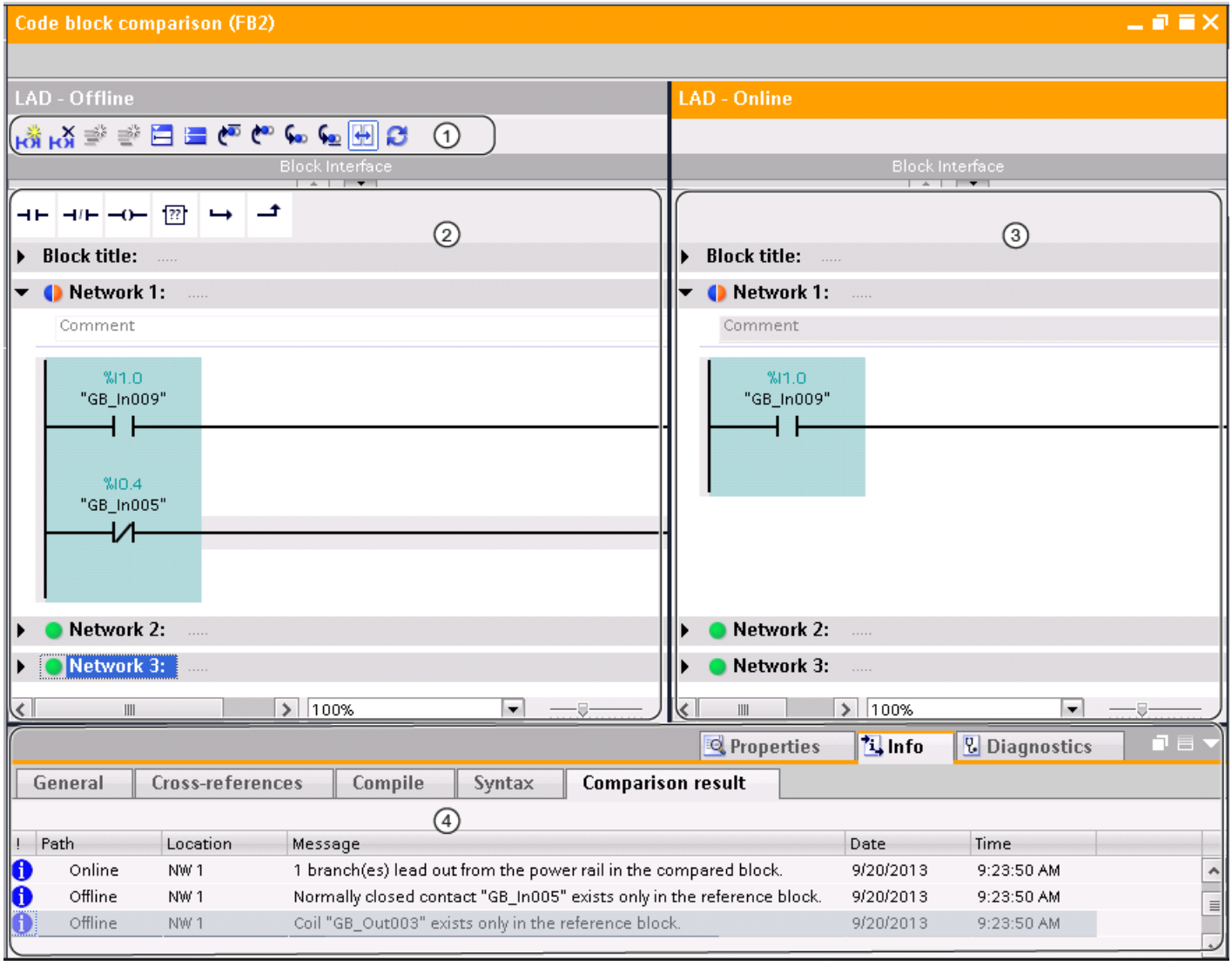
- The lines where there are differences are highlighted in gray.
- Differing operands and instructions are highlighted in green.
- If the number of networks differs, pseudo-networks are added to allow the display of identical networks to be synchronized. These pseudo-networks are highlighted in gray and contain the text "No corresponding network found" in the title bar of the network. Pseudo-networks cannot be edited.

For the sake of clarity, not all the differences are highlighted but rather the first difference of an operation in each case. For example, if all the inputs in an instruction with multiple inputs are different in the offline and online versions of the block, only the first input is highlighted as a difference. You can resolve this difference and update the comparison list. The next input will then be highlighted as a difference.

The number of differences highlighted within a network therefore depends on the number of instructions.

Structure of the detailed comparison

The following figure shows an example of the online/offline detailed comparison for the LAD programming language:



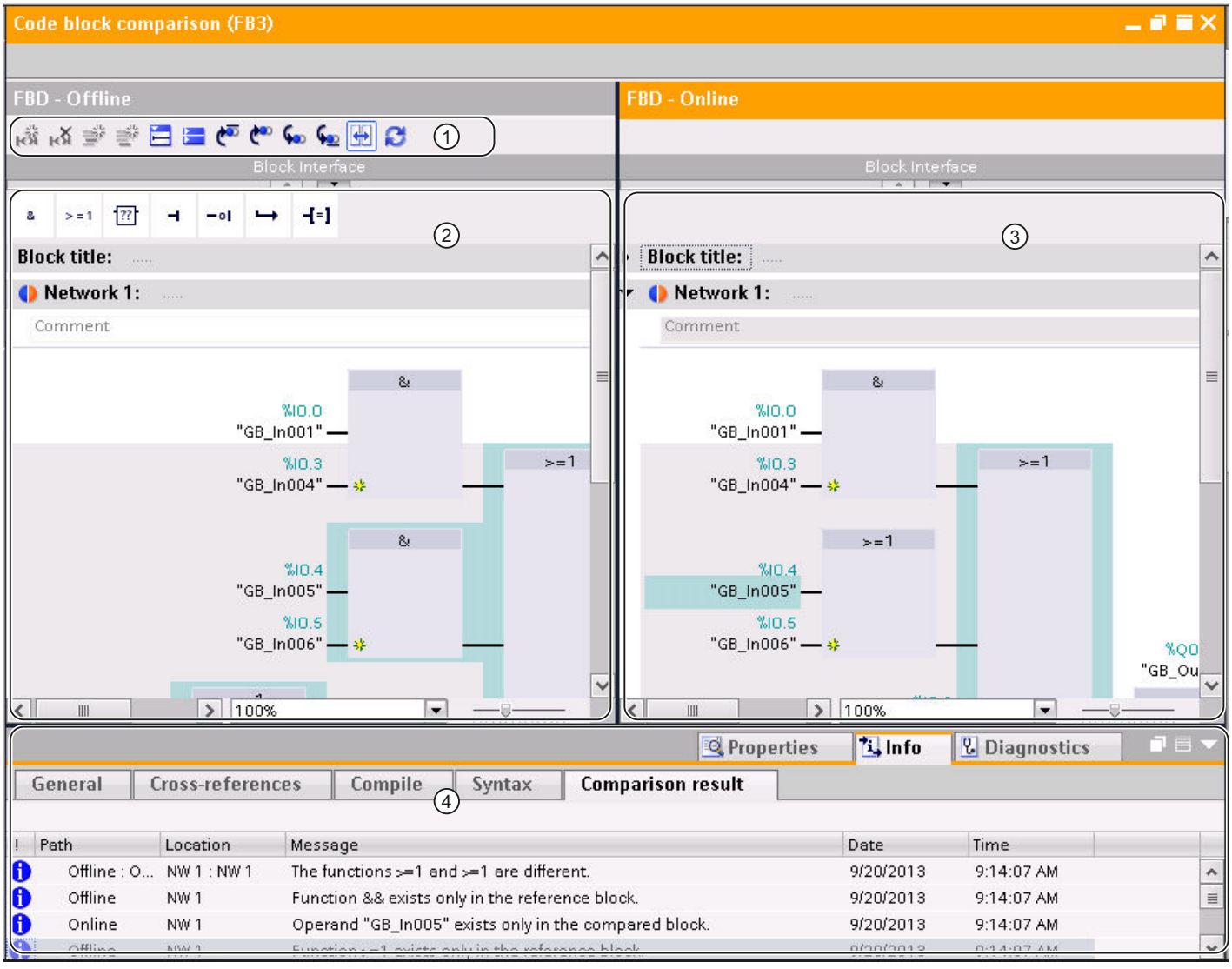
① Toolbar of the detailed comparison for LAD

② Reference block

③ Compared block

④ Comparison result in the Inspector window

The following figure shows an example of the online/offline detailed comparison for the FBD programming language:



① Toolbar of the detailed comparison for FBD

② Reference block

③ Compared block

④ Comparison result in the Inspector window

Note

Display of the symbolic names of the online version of the block is only possible for S7-1200 and S7-1500.

Toolbar of the detailed comparison

With the toolbar, you can access the following functions:

- General functions
  - Insert network
  - Delete network
  - Insert row
  - Add row
  - Open all networks
  - Close all networks
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

Reference block

The reference block is displayed in the left window. In an online/offline comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an online/offline comparison, the compared block is the online version of the block.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 1756)

Navigating in the detailed comparison (Page 1775)

Changing blocks during the detailed comparison (Page 1776)

Updating comparison results (Page 1778)

Visualization of the comparison result for STL

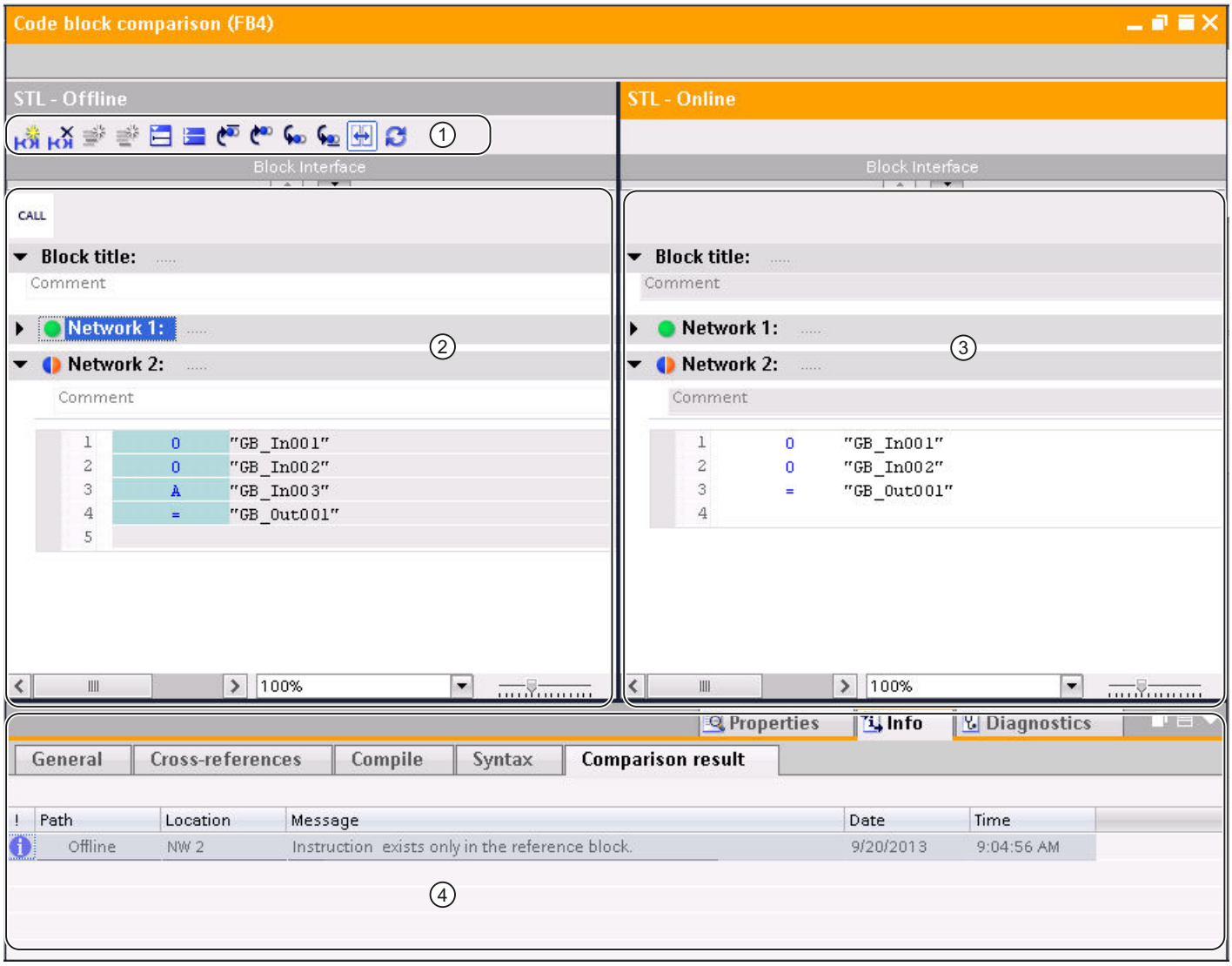
Introduction

The detailed comparison allows you to identify the exact places where versions of a block differ. The following color coding allows you to find these places as quickly as possible:

- The lines where there are differences are highlighted in gray.
- Differing operands and instructions are highlighted in green.
- If the number of networks differs, pseudo-networks are added to allow the display of identical networks to be synchronized. These pseudo-networks are highlighted in gray and contain the text "No corresponding network found" in the title bar of the network. Pseudo-networks cannot be edited.

Structure of the detailed comparison

The following figure shows an example of the online/offline detailed comparison for the STL programming language:



① Toolbar of the detailed comparison for STL

② Reference block

③ Compared block

④ Comparison result in the Inspector window

Note

The display of the symbolic names of the online version of the block is only possible for S7-1500.

Toolbar of the detailed comparison

With the toolbar, you can access the following functions:

- General functions
  - Insert network
  - Delete network
  - Insert row
  - Add row
  - Open all networks
  - Close all networks
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

Reference block

The reference block is displayed in the left window. In an online/offline comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an online/offline comparison, the compared block is the online version of the block.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 1756)

Navigating in the detailed comparison (Page 1775)

Changing blocks during the detailed comparison (Page 1776)

Updating comparison results (Page 1778)

Visualization of the comparison result for SCL

Introduction

The detailed comparison allows you to identify the exact places where versions of a block differ. The following color coding allows you to find these places as quickly as possible:

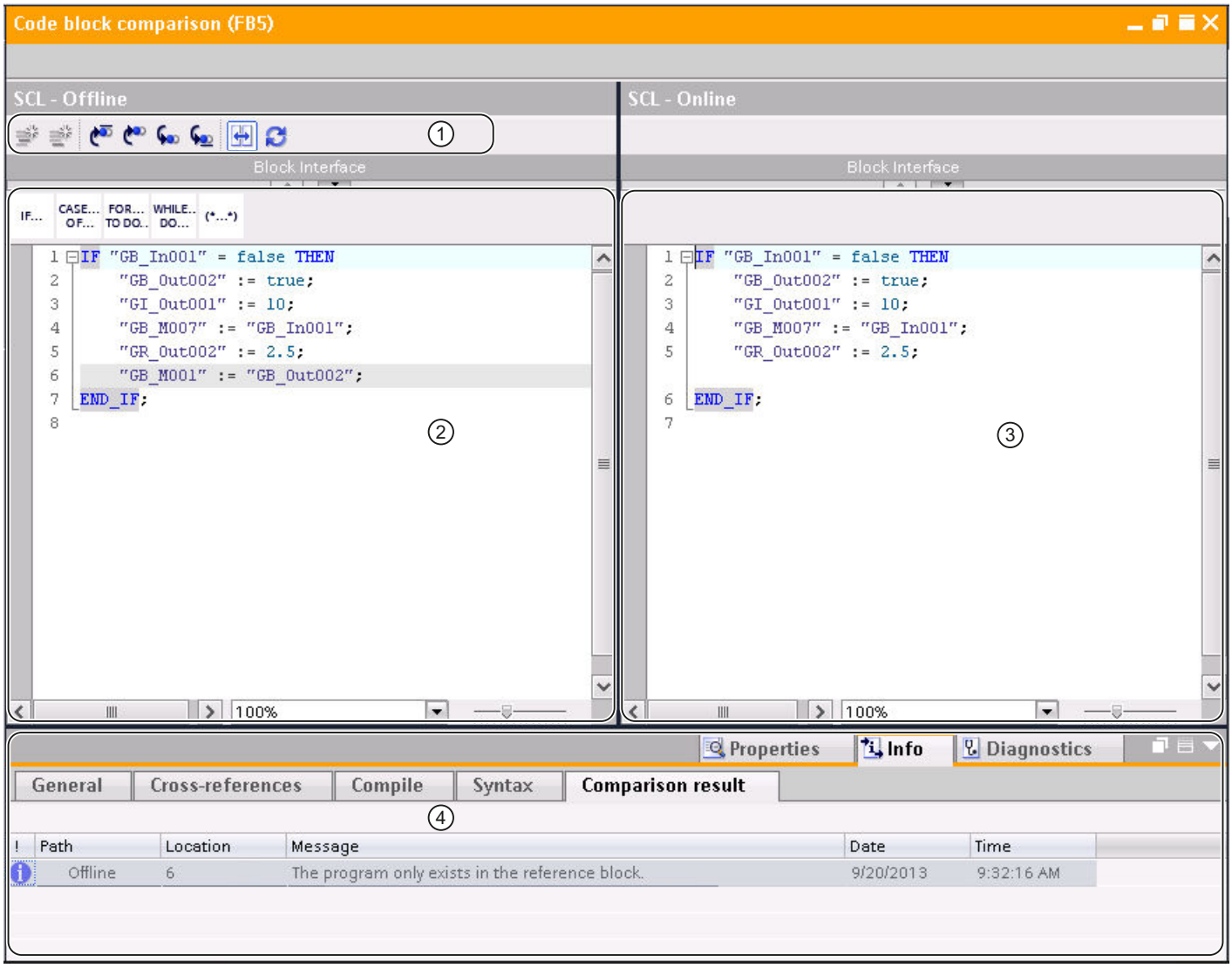
- The lines where there are differences are highlighted in gray.
- Differing operands and instructions are highlighted in green.

Note

The online/offline detailed comparison is not available for the CPU families S7-300/400 and for S7-1200 with a version less than 2.0.

Structure of the detailed comparison

The following figure shows an example of the online/offline detailed comparison for the SCL programming language:



① Toolbar of the detailed comparison for SCL

② Reference block

③ Compared block

④ Comparison result in the Inspector window

Note

The display of the symbolic name of the online version of the block is only possible for S7-1200 and S7-1500.

Toolbar of the detailed comparison

With the toolbar, you can access the following functions:

- General functions
  - Insert row
  - Add row
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

Reference block

The reference block is displayed in the left window. In an online/offline comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an online/offline comparison, the compared block is the online version of the block.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 1756)

Navigating in the detailed comparison (Page 1775)

Changing blocks during the detailed comparison (Page 1776)

Updating comparison results (Page 1778)

Visualization of the comparison result for GRAPH

Introduction

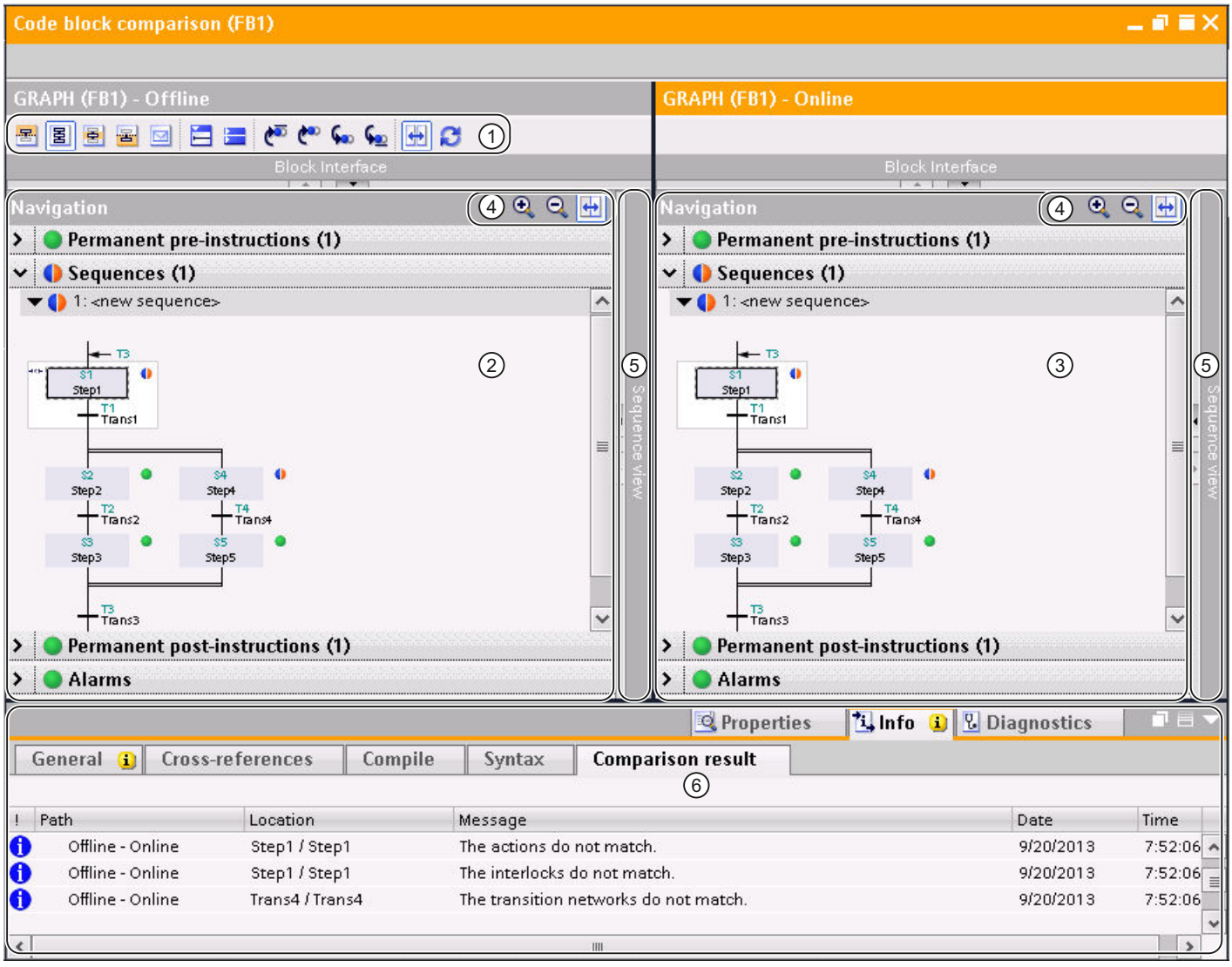
The detailed comparison allows you to identify the exact places where versions of a block differ. When you start a detailed comparison for a GRAPH block, navigation is opened first. Use the dividers to toggle between the navigation and the currently set view. You can select other views with the toolbar of the detailed comparison.

The result of the comparison is indicated by the comparison symbols.

See also: Overview of the comparison editor (Page 405)

Structure of the detailed comparison

The following figure shows an example for the navigation view with an online/offline detailed comparison for the GRAPH programming language:



① Toolbar of the detailed comparison for GRAPH

② Reference block

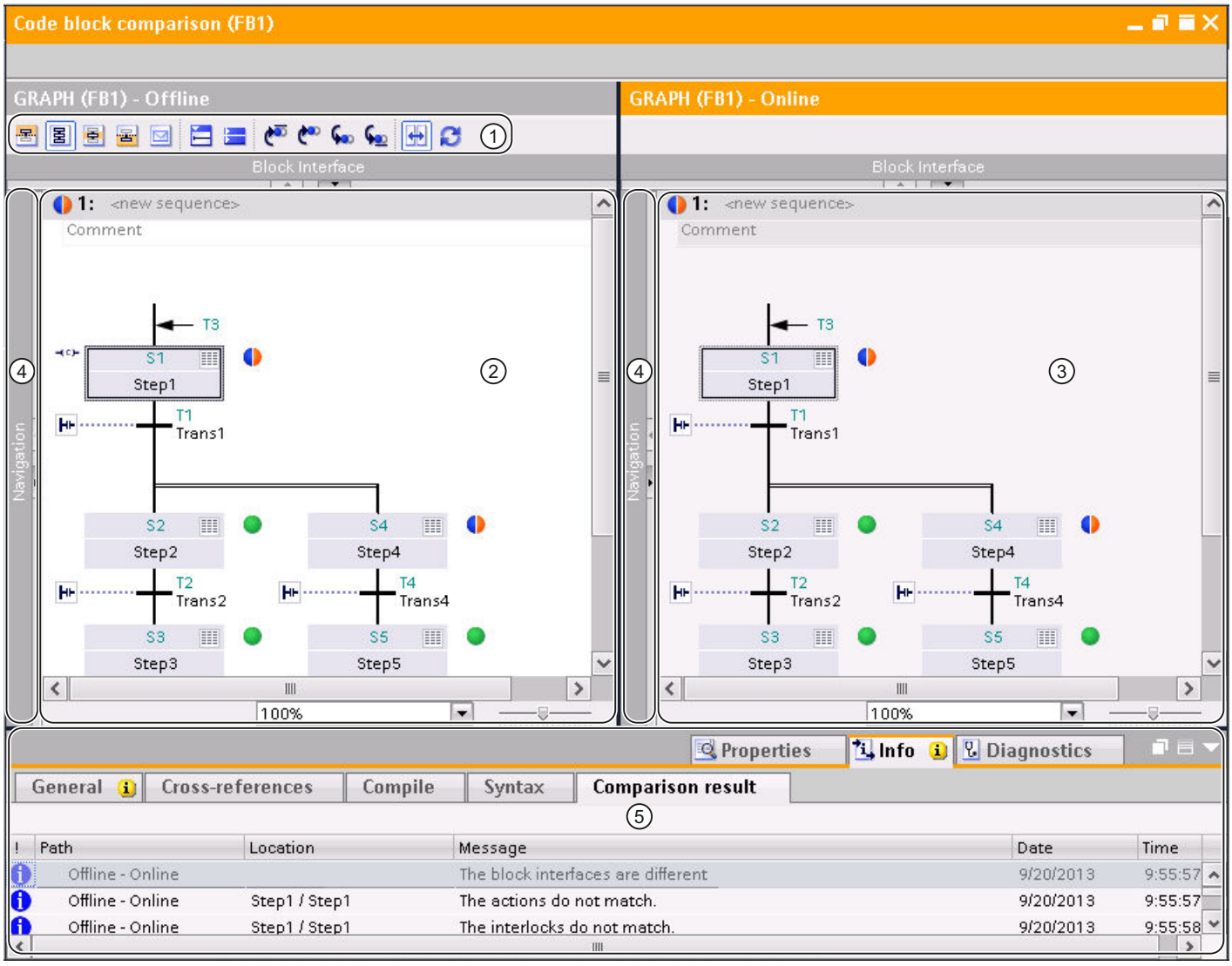
③ Compared block

④ Navigation toolbar

⑤ Dividers

⑥ Comparison result in the Inspector window

The following figure shows an example for the sequence view with an online/offline detailed comparison for the GRAPH programming language:



① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

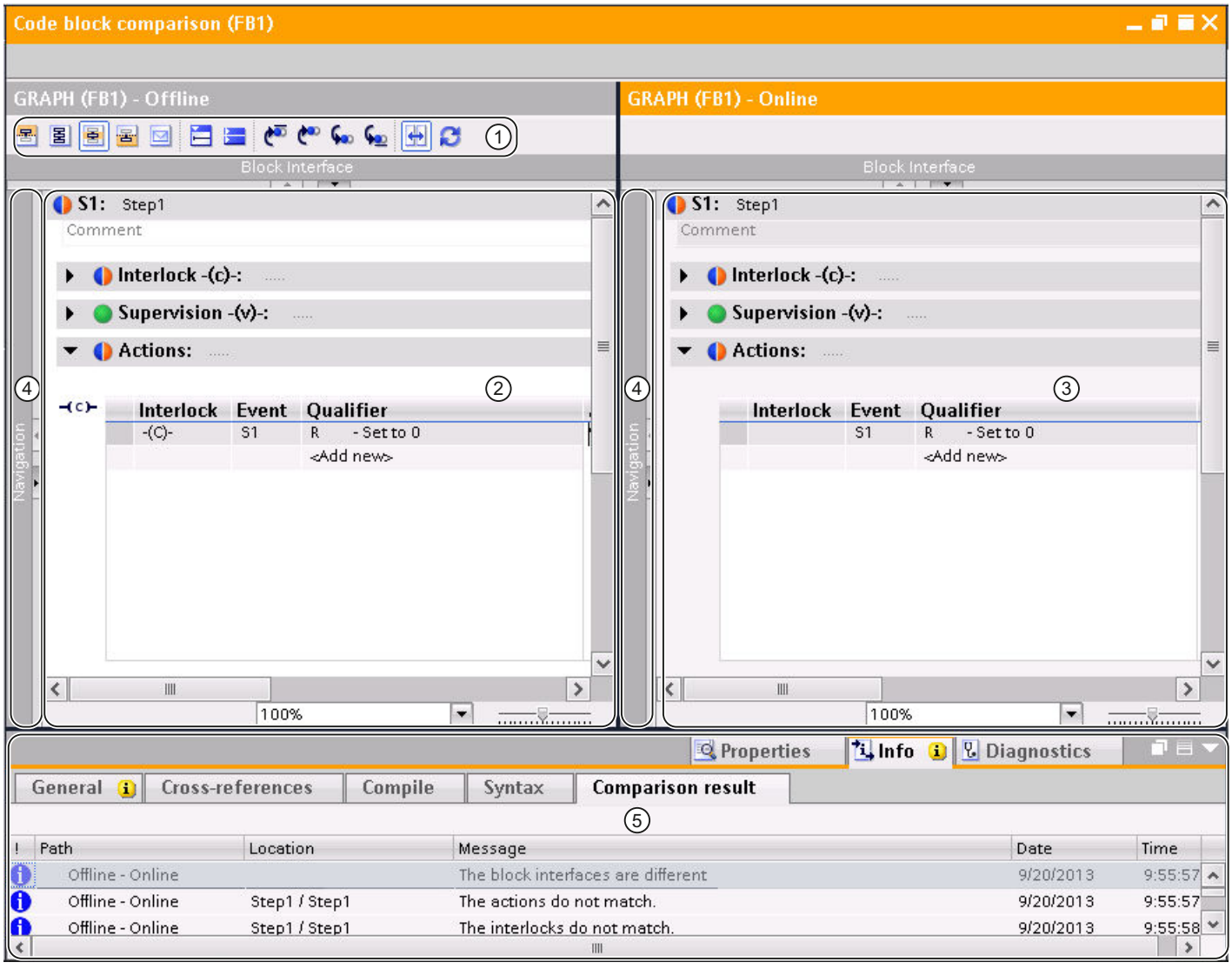
④ Dividers

⑤ Comparison result in the Inspector window

Note

If there are structural differences between the blocks, the comparison results are only displayed up to the first structural difference in the sequence view.

The following figure shows an example for the single step view with an online/offline detailed comparison for the GRAPH programming language:



① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

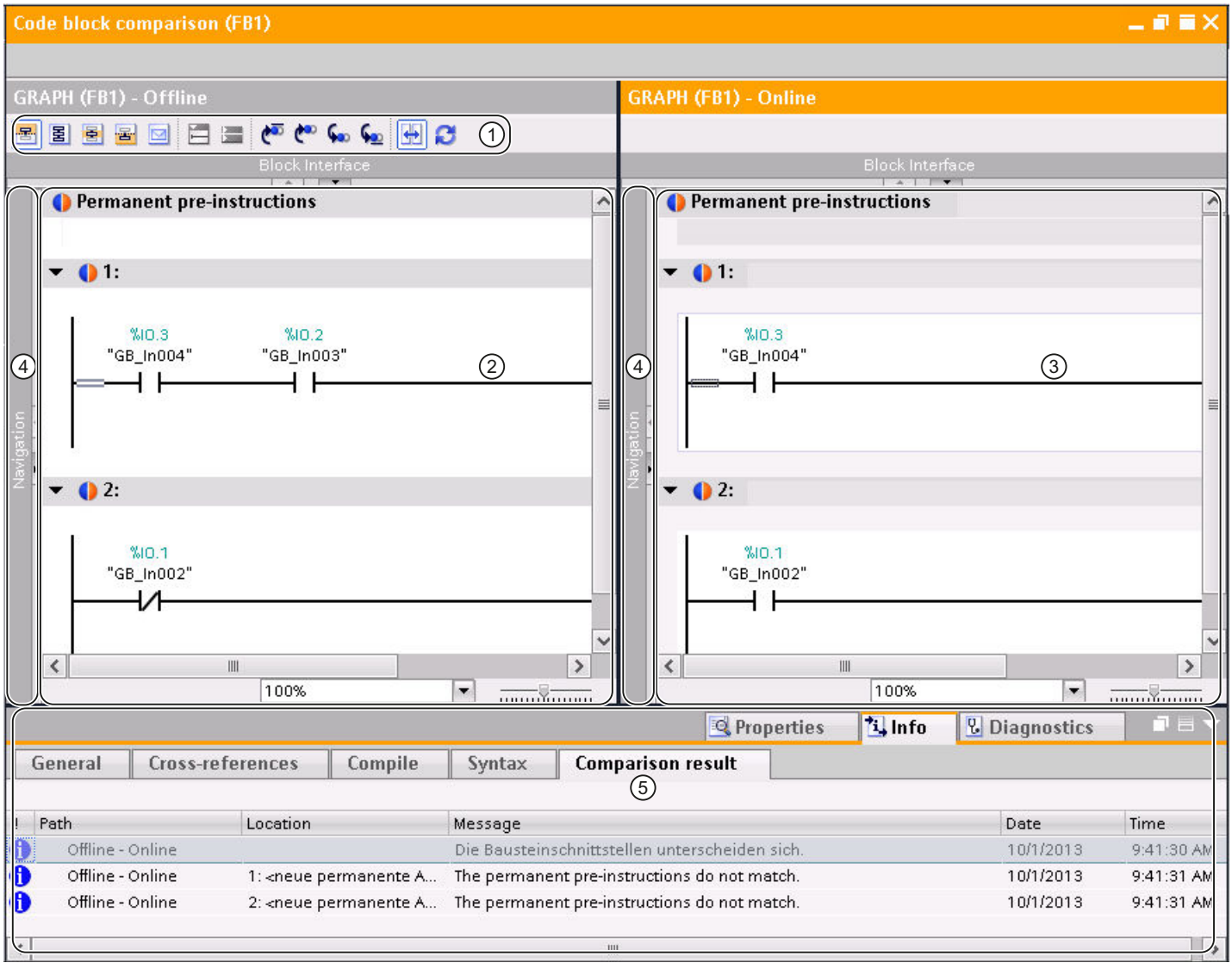
④ Dividers

⑤ Comparison result in the Inspector window

Note

The result of the comparison refers to the complete network Differences within the networks are not identified.

The following figure shows an example for the view of permanent instructions with an online/ offline detailed comparison for the GRAPH programming language:



① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

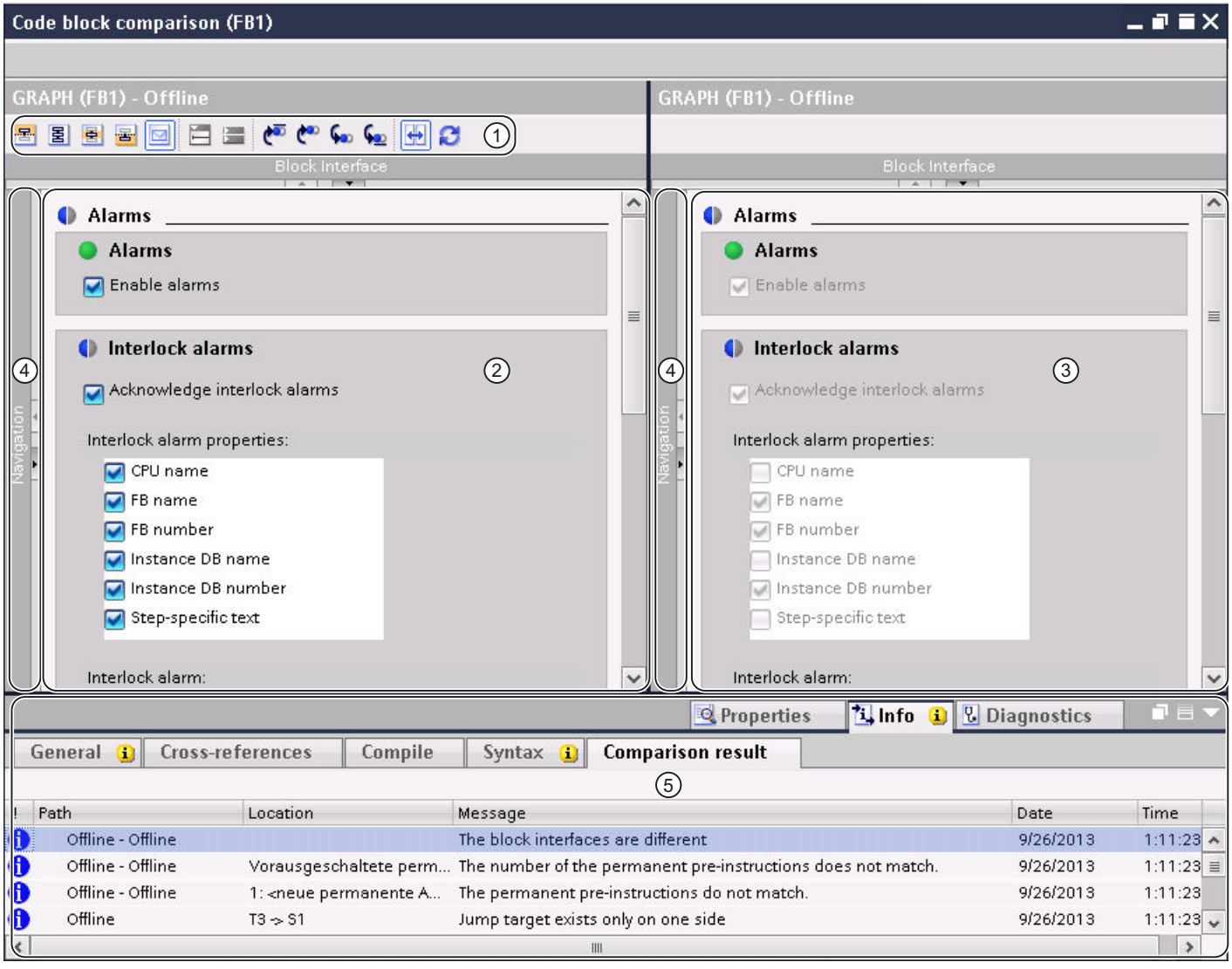
④ Dividers

⑤ Comparison result in the Inspector window

Note

The result of the comparison refers to the complete network Differences within the networks are not identified.

The following figure shows an example for the alarm view with an online/offline detailed comparison for the GRAPH programming language:



① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

④ Dividers

⑤ Comparison result in the Inspector window

Toolbars

With the toolbar of the detailed comparison, you can access the following functions:

- General functions
  - Change to permanent pre-instructions
  - Change to sequence view
  - Change to single step view
  - Change to permanent post-instructions
  - Change to alarm view
  - Open all networks
  - Close all networks
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

The navigation has its own toolbar with the following functions:

- Zoom in or zoom out of elements within the navigation
- Synchronize navigation

Reference block

The reference block is displayed in the left window. In an online/offline comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an online/offline comparison, the compared block is the online version of the block.

Dividers

You can click the dividers to toggle quickly between the navigation and the current view.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 1756)

Start detailed comparison for GRAPH blocks (Page 1757)

Navigating in the detailed comparison (Page 1775)

Changing blocks during the detailed comparison (Page 1776)

Updating comparison results (Page 1778)

Navigating in the detailed comparison

Requirement

You have run a detailed comparison.

Navigate to the differences

To navigate to a difference between the two blocks, follow these steps:

1. Open the list of results for the detailed comparison under "Info > Comparison result" in the Inspector window.
2. Double-click a difference.
   The difference is selected in both editors.

Or:

1. Click one of the following navigation buttons on the toolbar:
   - Position on first difference
     Navigates to the first difference in the block, and displays the difference in both editors.
   - Position on previous difference
     Navigates to the previous difference starting from the current position, and displays the difference in both editors.
   - Position on next difference
     Navigates to the next difference starting from the current position, and displays the difference in both editors.
   - Position on last difference
     Navigates to the last difference in the block, and displays the difference in both editors.

Switching off/on the synchronization of the vertical scrolling between the editors

The scrolling for both editors is synchronized to ensure that the corresponding networks are visible parallel to each other during vertical scrolling. You can switch this mode off and on. To do this, follow these steps:

1. To switch off synchronized scrolling, click the "Synchronize scrolling between editors" button in the toolbar.
2. To switch on synchronized scrolling again, click the "Synchronize scrolling between editors" button one more time in the toolbar.

See also

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 1756)

Start detailed comparison for GRAPH blocks (Page 1757)

Visualization of the comparison result for LAD/FBD (Page 1759)

Visualization of the comparison result for STL (Page 1763)

Visualization of the comparison result for SCL (Page 1766)

Visualization of the comparison result for GRAPH (Page 1769)

Changing blocks during the detailed comparison (Page 1776)

Updating comparison results (Page 1778)

Changing blocks during the detailed comparison

While you are carrying out the detailed comparison, you can make changes to the blocks that are being compared. Remember the following:

- Online/offline comparison: You can change only the offline block.
- Offline/offline comparison: You can change only the offline block in the left-hand area.

After a block change, it may be necessary to manually update the comparison result in the comparison editor to ensure that the comparison status is displayed correctly. You can then specify actions to synchronize the objects.

Note

You cannot change SCL blocks manually. However, you can apply changes from one block to the other block. In this regard, note the following points:

- You cannot apply any changes in an online block.
- You can only apply changes in an offline block if this is not write-protected. This is the case, for example, if the blocks of the detailed comparison come from different CPUs. It is then also possible to apply the changes to the block in the right-hand area.

Changing LAD, FBD or STL blocks

To change LAD, FBD or STL blocks, follow these steps:

1. Change the block in the left-hand area according to your requirements.
2. If necessary, click on "Update comparison results" in the toolbar.

Changing GRAPH blocks

To change a GRAPH block, follow these steps:

1. Switch to sequence view by clicking "Sequence view" between the two blocks.
2. Change the block in the left-hand area according to your requirements.
3. If necessary, click on "Update comparison results" in the toolbar.

Changing SCL blocks

To apply a change from one block to another one, follow these steps:

1. In the sidebar of the block from which you want to apply a change to another block, click on the arrow in the corresponding line.
   The line is inserted in the other block and the arrow buttons are removed.

   Note

   The colors of the arrows have the following meanings:
   - Gray: The changes cannot be applied to the other block, as the other block is either an online block or write-protected.
   - Blue: The changes are applied from an offline block to the other block.
   - Orange: The changes are applied from an online block to the other block.

2. If necessary, click on "Update comparison results" in the toolbar.

See also

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 1756)

Start detailed comparison for GRAPH blocks (Page 1757)

Visualization of the comparison result for LAD/FBD (Page 1759)

Visualization of the comparison result for STL (Page 1763)

Visualization of the comparison result for SCL (Page 1766)

Visualization of the comparison result for GRAPH (Page 1769)

Navigating in the detailed comparison (Page 1775)

Updating comparison results (Page 1778)

Updating comparison results

As soon as you change an object, the comparison results are no longer valid and must be updated.

Requirement

You have run a detailed comparison.

Procedure

To update the comparison results, follow these steps:

1. Click "Update the comparison result" in the toolbar.

See also

Carrying out an online/offline comparison (Page 403)

Carrying out offline/offline comparisons (Page 404)

Using the compare editor (Page 405)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 1756)

Start detailed comparison for GRAPH blocks (Page 1757)

Visualization of the comparison result for LAD/FBD (Page 1759)

Visualization of the comparison result for STL (Page 1763)

Visualization of the comparison result for SCL (Page 1766)

Visualization of the comparison result for GRAPH (Page 1769)

Navigating in the detailed comparison (Page 1775)

Changing blocks during the detailed comparison (Page 1776)

11.1.5.3 Comparing PLC tags

You have the following options for comparing PLC tags:

- Automatic offline/offline comparison in the compare editor
  The PLC tag tables of the selected devices are compared offline.
- Manual offline/offline comparison in the compare editor
  The selected PLC tag tables of the devices are compared offline.
- Detailed comparison
  Use the detailed comparison to determine differences within the PLC tag tables.

Performing automatic offline/offline comparison in the compare editor

To perform an automatic offline/offline comparison of PLC tag tables, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. Open the "PLC tags" folder.
   You can identify the status of the PLC tag tables based on the symbols in the status and action area. You can define certain actions depending on the status.

You can drag any other device to the drop area at any time to perform further comparisons.

Performing manual offline/offline comparison in the compare editor

To perform a manual offline/offline comparison of PLC tag tables, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. In the status and action area, click on the button for switching between automatic and manual comparison.
5. Select the PLC tag tables that you want to compare.
   The properties comparison is displayed. You can identify the status based on the symbols.

You can drag any other device to the drop area at any time to perform further comparisons.

Running a detailed comparison

To start a detailed comparison for a PLC tag table, follow these steps:

1. Perform an automatic or manual offline/offline comparison.
2. For an automatic offline/offline comparison in the compare editor, select the PLC tag table for which you want to run a detailed comparison. Note that two PLC tag tables must be selected for comparison for a manual offline/offline comparison.
3. Click the "Start detailed comparison" button in the toolbar.
   A separate compare editor opens. All existing PLC tags of the selected PLC tag tables are displayed depending on the settings of the compare editor. User and system constants are not shown, however. You can identify the status of the PLC tags based on the symbols. You can define certain actions depending on the status of the PLC tags.

See also

Introduction to comparing PLC programs (Page 1750)

Using the compare editor (Page 405)

Comparing PLC data types (Page 1780)

11.1.5.4 Comparing PLC data types

You have the following options for comparing PLC data types:

- Online/offline comparison (S7-1200/1500 only)
  The PLC data types in the project are compared with the PLC data types of the selected device.
- Automatic offline/offline comparison in the compare editor
  The PLC data types of the selected devices are compared offline.
- Manual offline/offline comparison in the compare editor
  The selected PLC data types of the devices are compared offline.
- Detailed comparison
  Use the detailed comparison to determine differences between PLC data types.

Performing online/offline comparison of PLC data types

To perform an online/offline comparison, follow these steps:

1. In the project tree, select a device that allows online/offline comparison.
2. Select the "Compare > Offline/online" command in the shortcut menu.
   If you have not already established an online connection to this device, the "Go online" dialog opens. In this case, set all the necessary parameters for the connection and click "Connect".
   The online connection is established and the compare editor opens.
3. Open the "PLC data types" folder.
   You can identify the status based on the symbols in the status and action area. When you select an object, the properties of the PLC data type and the corresponding PLC data type of the assigned device are displayed in the properties comparison.

Performing automatic offline/offline comparison in the compare editor

To perform an automatic offline/offline comparison of PLC tag tables, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. Open the "PLC data types" folder.
   You can identify the status of the PLC tag tables based on the symbols in the status and action area. You can define certain actions depending on the status.

You can drag any other device to the drop area at any time to perform further comparisons.

Performing manual offline/offline comparison in the compare editor

To perform a manual offline/offline comparison of PLC data types, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. In the status and action area, click on the button for switching between automatic and manual comparison.
5. Select the PLC data types that you want to compare.
   The properties comparison is displayed. You can identify the status based on the symbols.

You can drag any other device to the drop area at any time to perform further comparisons.

Running a detailed comparison

To start a detailed comparison for a PLC data type, follow these steps:

1. Perform an offline/offline comparison. You can also perform an online/offline comparison for CPUs of the S7-1200/1500 series.
2. For an automatic offline/offline comparison in the compare editor, select the PLC data type for which you want to run a detailed comparison. Note that two PLC data types must be selected for comparison with a manual offline/offline comparison.
3. Click the "Start detailed comparison" button in the toolbar.
   The two PLC data types are opened next to each other so that you can easily identify the differences.

See also

Introduction to comparing PLC programs (Page 1750)

Using the compare editor (Page 405)

Comparing PLC tags (Page 1778)

11.1.6 Compiling and downloading blocks

11.1.6.1 Compiling blocks

Basic information on compiling blocks

Introduction

A user program must first be compiled before the CPU can execute it. You need to recompile your program each time you make a change.

The following procedures take place during compilation:

- The user program is checked for syntax errors.
- Unneeded instructions are removed from the user program.
- All the block calls within the compiled blocks are checked. In case of changes to the interface of called blocks, errors will be shown in the "Compilation" tab of the information window. You have to correct these errors first.
- The blocks must be numbered uniquely in the user program. If more than one block has the same number, the blocks with number conflicts are renumbered automatically during compilation. A block will not be renumbered in the following cases:
  - The block was selected either individually or as part of a multiselection for the compilation.
  - The block is know-how protected.
  - The number assignment is set to "manual" in the properties of the block.

  Number conflicts that cannot be resolved by automatic renumbering must be corrected manually. Note the messages in the Inspector window for this. You can only correct know-how-protected blocks manually if you know the password since the blocks must then be recompiled.
- Finally, the user program is compiled into a code that can be read by the CPU.

Compilation methods

You can start compilation in the following windows or editors:

- Compiling blocks in the project tree
  Serves to compile individual blocks or the simultaneous compilation of one or several blocks in the "Program blocks" folder.
- Compiling blocks in the program editor
  This is intended for compilation of a single open block.
- Compiling blocks in the call or dependency structure
  Used to compile individual blocks.
  See also: Call structure (Page 1818), Dependency structure (Page 1824)

Compilation options

If you are compiling blocks in project tree, you have further options:

- Software (changes only)
  All program changes of the selected blocks are compiled. If you have selected a block folder, all program changes to the bocks contained in the folder are compiled.
- Software (compile all blocks)
  All blocks are compiled. This is recommended for the first compilation and after major revisions.
- Software (reset memory reserve)
  All tags declared in the reserve area of the interface of selected blocks are moved to the standard area of the interface. Memory reserve is now available for further interface extensions.

Note

This option is only available for CPUs of the S7-1500 and S7-1200 V4 and higher series.

Consistency check

Changing the interfaces of blocks called or PLC data types used can result in inconsistencies between calling blocks and called blocks or between the PLC data types and the global data blocks which use these PLC data types.

To avoid such inconsistencies in the user program, the system performs an automatic consistency check before each compilation process. The time stamps are compared and compilation is then either carried out or canceled depending on the results of the comparison.

- The calling block can only be compiled if the time stamps of the interfaces of the called blocks are older than those of the calling block.
- A global data block based on a PLC data type can only be compiled correctly if the time stamp of the global data block is newer than the time stamp of the PLC data type used.
- The instance data block can only be compiled correctly if the interface time stamps for the interface of the instance data block are identical to those of the assigned function block.

If the compilation process is cancelled, an alarm is displayed in the inspector window. Update the block calls in the relevant blocks and the PLC data types in the global data blocks and then restart compilation. The consistency check also finds know-how protected blocks which cannot be compiled. The corresponding messages will also be shown in the inspector window.

If you start loading immediately instead of first compiling, the blocks selected will be automatically compiled and the block call and global data blocks implicitly updated. Please note the following differences between the CPU families:

- S7-1200/1500: All blocks affected are loaded to ensure no inconsistencies can arise.
- S7-300/400: Only the block selected is loaded.

See also

Compiling blocks in the project tree (Page 1784)

Compiling blocks in the program editor (Page 1785)

Correcting compilation errors (Page 1786)

Block time stamps (Page 1520)

Updating block calls in LAD (Page 1596)

Updating block calls in FBD (Page 1638)

Compiling project data (Page 392)

Compiling blocks in the project tree

You can compile one block, multiple blocks or all of the blocks in the project tree.

For CPUs of the S7-1500 and S7-1200 V4 series, you can also reset the memory layout of blocks with memory reserve by running a compilation. For more information on memory reserve, refer to chapter "Loading blocks (S7-1200/1500) > Loading block changes without reinitialization".

Requirement

The project tree is open.

Compiling one or more blocks in the project tree

To compile multiple blocks in the project tree, follow these steps:

1. Open the "Program blocks" folder in project tree.
2. Select the blocks you want to compile.
3. Select the "Compile > Software (only changes)" command from the shortcut menu.

Compiling all blocks in the project tree

To compile all blocks in the "Program blocks" folder in project tree, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. You can select one of two different options for the compilation:
   - If you want to compile only the changes since the last compilation, select the "Compile > Software (only changes)" command in the shortcut menu.
   - If you want to compile all blocks completely, select the "Compile > Software (compile all blocks)" command in the shortcut menu.

Resetting memory layout (S7-1500/S7-1200 V4)

Proceed as follows to reset the memory layout of blocks:

1. Select the "Program blocks" folder, or specific blocks in this folder.
2. Select the "Compile > Software (Reset memory reserve)" command from the shortcut menu.

Result

The code for the blocks will be generated if the consistency check has been successful. Instance data blocks generated by the system which are no longer needed will be deleted.

The message under "Info > Compilation" in the inspector window reports whether the compilation was successful.

See also

Basic information on compiling blocks (Page 1782)

Compiling blocks in the program editor (Page 1785)

Correcting compilation errors (Page 1786)

Finding syntax errors in the program (Page 1697)

Compiling blocks in the program editor

Note

Note that the block is recompiled even if you have not made any changes and the time stamp of the block is modified.

Requirement

The block to be compiled is open.

Procedure

To compile a block in the program editor, follow these steps:

1. Right-click in the instruction window of the programming editor.
2. Select the "Compile" command in the shortcut menu.

Result

The code for the block is generated. Instance data blocks generated by the system which are no longer needed will be deleted.

The message under "Info > Compilation" in the inspector window reports whether the compilation was successful.

See also

Basic information on compiling blocks (Page 1782)

Compiling blocks in the project tree (Page 1784)

Correcting compilation errors (Page 1786)

Correcting compilation errors

In the Inspector window in "Info > Compile", you can see whether any compilation was successful or whether errors were detected in the program. If errors occur, you will need to correct them and then start the compilation again.

Procedure

To correct errors following compilation, follow these steps:

1. Open the error list in the Inspector window with "Info > Compile".
2. If there is one, click on the blue question mark next to the error text for information on remedying errors.
3. Double-click the error you want to correct.
   The corresponding error is highlighted.
4. Correct the error.
5. Restart compilation.

See also

Basic information on compiling blocks (Page 1782)

Compiling blocks in the program editor (Page 1785)

Compiling blocks in the project tree (Page 1784)

11.1.6.2 Downloading blocks for S7-1200/1500

Introduction to downloading blocks

Downloading blocks to device

So that the CPU can execute the user program, the program must first be compiled and then downloaded to the device. Loading into the device can be started through different methods:

- By using the commands in the "Online" menu
- By using the the context menu
- By using the "Load" button in the toolbar

Depending on the selected object (such as individual blocks in the programming editor, the complete block folder or the connected CPU in the project navigation) and the command respectively selected, you can load the following components:

- Hardware and software (only changes)
- Hardware configuration
- Software (only changes)
- Software (all)

During the loading operation, all information that is required for the reconstruction of the program, including symbolic information such as the names and comments for code and data blocks, is also loaded in the current project language. If you change the project language, you must therefore re-load the program.

The symbolic information is not loaded to the work memory, but rather to the load memory.

After the data has been loaded from a device, the symbolic information is available again in your program, which increases the readability of your program code. Please note, however, that loading to and from a device is not a substitute for storing data in an offline project, as watch tables or multi-language capability of projects cannot be reproduced by loading to and from a device.

After loading from a device, you can only display all data from know-how-protected blocks by entering the correct password.

Note

No valid hardware configuration available

If no valid hardware configuration was found on the CPU while loading the software, the existing hardware is loaded as well during a "Download to device".

Please note that in this case hardware components are also loaded during the download, even though you executed the command for loading software!

Note

To avoid inconsistencies between calling and called blocks, all blocks affected are compiled and loaded after each global change, such as a change in the block interface.

Note

S7-1200 Version 1.0

If you download an element of your project to the CPU, for example a program block, a data block or the hardware configuration, the CPU runs a cold restart the next time it changes to RUN mode. Apart from deleting the inputs, initializing the outputs and deleting the non-retentive memory, cold restart also deletes the retentive memory areas. All subsequent changes from STOP to RUN are warm restarts in which the retentive memory is not deleted.

Note

S7-1500

The load memory of S7-1500 series CPUs is on the SIMATIC memory card. Therefore, a SIMATIC memory card absolutely must be inserted in order to operate the CPU.

Uploading blocks from device

You can load the blocks of a device to your project. This is necessary, for example, if you want to edit blocks that only exist in this device. You have the option of loading either all available blocks (organization blocks, function blocks, functions and data blocks) and global PLC tags or individual blocks to the project.

Uploading blocks from or downloading blocks to a memory card

Memory cards are plug-in cards used with an S7-1200 series CPU, for example, to replace the load memory of a device. In the case of S7-1500 series CPUs, they contain the load memory. Only the SIMATIC memory card from Siemens can be used for devices of the S7-1200 and S7-1500 series.

To use a memory card as load memory, you must download the user program or individual blocks to a memory card. You can just as well upload blocks from a memory card back into the project.

Note

S7-1200

Note the following when uploading to or downloading from a memory card:

- If the CPU contains no previous program and you insert an empty memory card in the CPU the program will be loaded from the PG/PC to the memory card and not to the CPU.
- If you insert an empty memory card prior to the startup of the CPU, the program that is on the CPU will be transferred automatically to the memory card. The program on the CPU will then be deleted.
- If you insert a memory card with a program in the CPU prior to the startup of the CPU and the CPU already contains a program, the program on the memory card will be executed and not the program on the CPU. The program on the CPU will be deleted.

Loading GRAPH function blocks

If you load a GRAPH function block together with its instance data block, the processing of the sequencer starts over at the initial step. As a result, problems may occur when synchronizing the sequencer with the process. You can avoid these problems by switching off the sequencer before loading.

Downloading changes without reinitialization

It often proves necessary to edit or expand a PLC program that was already commissioned and that is running on the plant without error. Such operations should be performed without causing any major interruptions of current operations.

S7-1500 therefore offers the option of extending the interfaces of function or data blocks during runtime and loading the modified blocks without setting the CPU to STOP or affecting the value of tags that are already loaded. This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Effects of a load operation on the tag values of a data blocks

When data blocks are downloaded to a device in STOP mode, the next transition of the device to RUN affects the current tag values as follows:

- Tags not marked as being retentive retain their defined start values.
- Retentive tags of the S7-1200 only retain their values if the following conditions are met:
  - You loaded the data block by means of "Download to device > Software (changes only)".
  - You made no changes to the DB structure.

  Otherwise the retentive tags will also retain their defined start values.
- Retentive tags of the S7-1500 only retain their values if the following conditions are met:
  - You loaded the data block by means of "Download to device > Software (changes only)".
  - You made no changes to the structure of the data block or modified it within the memory reserve.

  Otherwise the retentive tags will also retain their defined start values.

Loading blocks with synchronization

In team engineering, it is possible for several users to work on one project with several engineering systems at the same time and access one S7-1500 CPU. To ensure consistency within the shared project, it is necessary to synchronize the changed data prior to loading so that nothing gets overwritten unintentionally.

If differences are determined between the online and offline data management within the shared project during loading that were caused by a different engineering system, automatic synchronization of the data to be loaded is offered during loading.

In this case, the "Synchronization" dialog displays the data to be synchronized with the current status (online-offline comparison) and the possible actions.

The following options are available for synchronization:

Application case	Recommendation	Synchronization
One or more blocks on the CPU (online) are more recent than in the engineering system (offline).	These blocks should be downloaded from the CPU to the engineering system before loading.	Automatic synchronization is possible: The blocks in the engineering system are updated prior to loading.
One or more new blocks have been created and exist only in the CPU (online).	These blocks should be downloaded from the CPU to the engineering system before loading.	Automatic synchronization is possible: The new blocks are added prior to the download in the engineering system.
One or more blocks on the CPU have been deleted.	The blocks should also be deleted prior to the download in the engineering system.	Automatic synchronization is not possible. The blocks deleted on the CPU should be manually deleted in the offline project in the engineering system.
One or more blocks on the CPU and in the engineering system are different. This is the case when a different user has changed blocks to which you have also made corrections and has already downloaded them to the CPU.	These blocks with competing changes must be adapted manually. You decide in this case which changes you are going to accept. If the blocks on the CPU are to be retained, you should accept these blocks prior to download from the CPU to your engineering system. If the blocks that you have changed are to be applied, you can continue with the download without synchronization.	Automatic synchronization is not possible: The affected blocks on the CPU or in the engineering system must be adapted manually. One of the existing block versions (online or offline) will be overwritten in the process.
There are differences in the hardware configuration on the CPU (online) and in the engineering system (offline).	Differences in the hardware configuration must be adapted manually. You decide in this case which hardware configuration you are going to accept. If the existing hardware configuration on the CPU is to be retained, you should apply these in your engineering system prior to loading. If you want to apply the changed hardware configuration, you can continue with the download without synchronization.	Automatic synchronization is not possible: The hardware configuration must be adapted manually. One of the existing hardware configurations (online or offline) will be overwritten in the process.

You can use the "Force download to device" command to download blocks without synchronization, if desired.

Downloading blocks in the "RUN" operating mode to the device

Basics on downloading blocks in the "RUN" operating mode

When you download modified blocks to the device, it is not always necessary to switch the device to the "STOP" operating mode. Prior to a download operation, the Engineering System checks whether the device must be stopped before downloading. The result of this check is displayed in the "Load preview" dialog.

If it is necessary to change to the "STOP" operating mode, you cannot continue the download process until you have stopped the CPU.

If the requirements are met, you can also download a modified program or program parts to a CPU in "RUN" operating mode.

The quantity structures may be exceeded for very complex programs which prevents a download in "RUN". In this case you must first create the prerequisites for a download in "RUN".

Tips:

- Use a memory card with sufficient capacity.
- Select a CPU with sufficient work memory.
- If necessary, reduce the number of downloaded objects (blocks, constants, PLC tags, data types).
  If you cannot download all objects at once, proceed in several steps and download a smaller number of objects in each step.

Note

Actual parameters are not overwritten by a download process in the "RUN" operating mode. Changes to the actual parameters will not become effective until the next time you change the operating mode from "STOP" to "RUN".

Downloading changes in "RUN"

The table below shows which program and configuration changes can be downloaded in "RUN" operating mode, sorted by CPU family and taking into consideration the firmware versions of the CPUs.

Explanations on the table:

- "RUN": Change can be downloaded in "STOP" operating mode as well as in "RUN" operating mode to the CPU.
- "RUN (< 57)": The CPU can integrate up to 56 new or modified objects/blocks in one program cycle. If you download more objects/blocks, they are integrated in several successive program cycles. If you want to download all objects/blocks consistently, you must set the CPU to "STOP" operating mode. This number depends on the setting for S7-300 CPUs with configuration option "Process mode/test mode".
- "RUN (Init)": Change can be downloaded in "RUN" operating mode; downloaded data blocks are re-initialized.
- "STOP": Change can only be downloaded in "STOP" operating mode.
- "STOP (Reset)": Change can only be downloaded in "STOP" operating mode; all data including retentive data is reset.

	S7-300	S7-400	S7-1200 V4.0 and higher	S7-1500
Action/type of change	**Download possible in mode ...**			
Modified properties of HW components. This includes changes to comments in the HW configuration.	STOP	STOP	STOP	STOP
Added HW components	STOP	STOP	STOP	STOP
New/revised text lists (messages)	**RUN**	**RUN**	STOP	**RUN (V1.1 and higher)**
Revised comments (new, revised, deleted) with the exception of comments in the HW configuration	-	-	**RUN**	**RUN**
Number of blocks downloaded at the same time	**RUN (<17)**	**RUN (<57)**	**RUN (<21)**	**RUN (all)**
Download PLC program to the device and reset	STOP (Reset)	STOP (Reset)	STOP (Reset)	STOP (Reset)
New OB	**RUN**	**RUN**	STOP	**RUN**
Modified OB: Code changes	**RUN**	**RUN**	**RUN**	**RUN**
OB with modified properties (e.g., cycle time change)	**RUN**	**RUN**	STOP	**RUN**
Deleted OB	**RUN**	**RUN**	STOP	**RUN**
New FB/FC/DB/User Data Type (UDT)	**RUN**	**RUN**	**RUN**	**RUN**
Deleted FB/FC/DB/User Data Type (UDT)	**RUN**	**RUN**	**RUN**	**RUN**
Revised FB/FC: Code change	**RUN**	**RUN**	**RUN**	**RUN**
Revised FB/FC: Interface change*	STOP	STOP	**RUN**	**RUN**
Modified DB: Modified property ("Only store in load memory" attribute changed)	STOP	STOP	**RUN (Init)**	**RUN (Init)**
Modified DB (memory reserve not enabled): Name/type of tags modified, tags added or deleted **	**RUN (Init)**	**RUN (Init)**	**RUN (Init)**	**RUN (Init)**
Modified DB (memory reserve enabled): New tag added**	-	-	**RUN**	**RUN**
Modified User Data Type (UDT)	STOP	STOP	**RUN (Init)**	**RUN (Init)**
Add new PLC tags (timer, counter, bit memory)	**RUN**	**RUN**	**RUN**	**RUN**
Modified retentivity settings (timer, counter, bit memory, DB area)	STOP	STOP	STOP	STOP
Motion Control technology objects: Changes to MC Servo cycle clock, change from free-running to cyclical (and vice versa). Changes to the HW interface of the TO	-	-	-	STOP

* If the interface change results in structural changes at the instance DB, see "Modified DB...".

** For the effect of download of data block changes to the data block content, see section "Downloading data blocks to the CPU"

Download of changes in "RUN" with older CPU firmware versions

The table below shows which changes you can download in "RUN" operating mode, in particular, for which older CPU firmware versions.

	S7-1200 V1.0 - 2.1	S7-1200 V2.2 - V3.0
Action/type of change	**Download possible in mode ...**	
Modified properties of HW components. This includes changes to comments in the HW configuration.	STOP	STOP
Added HW components	STOP	STOP
New/revised text lists (messages)	STOP	STOP
Revised comments (new, revised, deleted) with the exception of comments in the HW configuration	STOP	**RUN**
Number of blocks downloaded at the same time	STOP	**RUN (<11)**
Download PLC program to the device and reset	STOP (Reset)	STOP (Reset)
New OB	STOP	STOP
Modified OB: Code changes	STOP	**RUN**
OB with modified properties (e.g., cycle time change)	STOP	STOP
Deleted OB	STOP	STOP
New FB/FC/DB/User Data Type (UDT)	STOP	**RUN**
Deleted FB/FC/DB/User Data Type (UDT)	STOP	**RUN**
Revised FB/FC: Code change	STOP	**RUN**
Revised FB/FC: Interface change	STOP	STOP
Modified DB: Modified property ("Only store in load memory" attribute changed)	STOP	STOP
Modified DB (memory reserve not enabled): Name/type of tags modified, tags added or deleted	STOP	STOP
Modified User Data Type (UDT)	STOP	STOP
Add new PLC tags (timer, counter, bit memory)	STOP	STOP
Modified retentivity settings (timer, counter, bit memory, DB area)	STOP	STOP

Downloading data blocks to the CPU

Depending on the conditions, the download of new or modified data blocks has an effect on the actual values in the data block:

Download new data blocks	Actual values in the new data blocks are set to start values.
Download structurally modified data blocks (memory reserve not enabled)	Actual values of added tags in the structurally modified data blocks are set to start values.

Download structurally modified data blocks (memory reserve enabled)	The following actual values are retained: • Actual values of tags outside the memory reserve • Actual values of tags that were not modified within the memory reserve Actual values of added tags within the memory reserve are set to start values.
Download of simply modified data blocks (no structural modification)	Actual values are retained.

Additional information

Additional information on download of block extensions without re-initialization and download of modified values for data blocks is available at "See also".

See also

Downloading blocks from program editor to device (Page 1794)

Downloading blocks from the project tree to the device (Page 1796)

Downloading project data to a device (Page 396)

Downloading blocks from program editor to device

Requirement

The block to be downloaded is open.

Procedure

To download a block from the program editor to the device, follow these steps:

1. Right-click in the instruction window of the programming editor.
2. Select the "Download to device" command in the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load".
   - You can save your preferred connection parameters as default under "Options > Settings > Online & Diagnostics". When you first load with new connection parameters, a query is automatically displayed whether you want to store it as a default setting. Click "Yes", if you want to save the current connection parameters as default.
   - You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command. You can also open the "Extended download to device" dialog explicitly via the "Online" menu.
     See also: Establishing and terminating an online connection
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays alarms and proposes actions necessary for loading.
3. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as downloading becomes possible, the "Load" button is enabled.

   Note

   Actions

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   Note

   To avoid inconsistencies between calling and called blocks, download all affected blocks each time you make global changes, such as changes in the block interface. Select the "Consistent download" action.

4. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
5. If you want to start the modules again directly after downloading, select the "Start all" check box.
6. To close the "Load results" dialog box, click "Finish".

Result

The code for the block will be downloaded to the device. If the changes affect additional blocks, these will be compiled and also downloaded to the device. Blocks that only exist online in the device are deleted. Existing CPU data blocks are retained, however. Inconsistencies between the blocks in the user program are avoided by loading all blocks affected and deleting the unneeded blocks in the device.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Downloading blocks from the project tree to the device (Page 1796)

Downloading project data to a device (Page 396)

Downloading blocks in the "RUN" operating mode to the device (Page 1790)

Downloading blocks from the project tree to the device

In the project tree you can download one block, multiple blocks or all blocks to a device.

Downloading one or more blocks from the project tree to the device

To download one block or multiple blocks to the device from the project tree, follow these steps:

1. Open the "Program blocks" folder in project tree.
2. Select the blocks you want to download.
3. Select the "Download to device > Software (only changes)" command from the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load".
   - You can save your preferred connection parameters as default under "Options > Settings > Online & Diagnostics". You will be prompted as such when you download new connection parameters for the first time. Select "Yes", if you want to save the current connection parameters as default.
   - You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command. You can also open the "Extended download to device" dialog explicitly via the "Online" menu.
     See also: Go online and Go offline
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays alarms and proposes actions necessary for loading.

4. Check the alarms and, where necessary, enable the actions in the "Action" column.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   As soon as downloading becomes possible, the "Load" button is enabled.

5. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
6. If you want to start the modules again directly after downloading, select the "Start all" check box.
7. To close the "Load results" dialog box, click "Finish".

Downloading blocks from the project tree to the device

To download all blocks in the "Program blocks" folder to the device from the project tree, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. Select the "Download to device" submenu in the shortcut menu.
3. If you only want to download the changes since the last download, select the "Software (only changes)" option. If all blocks are to be fully loaded and all values are to be reset to their start values, select "Download PLC program to the device and reset".
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load". You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command. You can also open the "Extended download to device" dialog with the "Online" menu.
     See also: Go online and Go offline
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   As soon as downloading becomes possible, the "Load" button is enabled.

5. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
6. If you want to start the modules again directly after downloading, select the "Start all" check box.
7. To close the "Load results" dialog box, click "Finish".

Result

The code for the blocks is downloaded to the device. If the changes affect additional blocks, these will be compiled and also downloaded to the device. Blocks that only exist online in the device are deleted. Inconsistencies between the blocks in the user program are avoided by loading all blocks affected and deleting the unneeded blocks in the device.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Downloading blocks from program editor to device (Page 1794)

Downloading project data to a device (Page 396)

Downloading blocks in the "RUN" operating mode to the device (Page 1790)

Uploading blocks from device

You can load either all blocks or individual blocks from a device into your project.

Note

Please note the following:

- Please note that when you load individual blocks, no tags or other required blocks to which you may refer are loaded together with the individual blocks. During the loading operation, reference to tags and blocks are reassigned where possible based on the names. After the loading operation, check whether these assignments are correct.
- When uploading from a device to an empty project, any existing folder structures for blocks and objects from libraries may not be downloaded.
- S7-1500: When GRAPH function blocks are loaded from a device to your project, the step-specific alarm texts for the interlock and supervision alarms are not loaded.

Requirement

The online and offline versions of a block to be loaded are different or the blocks only exists online.

Uploading all blocks from a device

To upload all blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to upload the blocks.
   See also: Go online and Go offline
2. In the project tree, select the device folder from which you want to upload blocks.
3. In the "Online" menu, select the "Upload from device" command.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   The "Upload from device" button will be enabled as soon as uploading becomes possible.
5. Click on the "Upload from device" button.
   The load is executed.

Uploading individual blocks from a device

To upload individual blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to upload the blocks.
   See also: Go online and Go offline
2. In the project tree, select the blocks that you want to upload from the device.
3. In the "Online" menu, select the "Upload from device" command.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   The "Upload from device" button will be enabled as soon as uploading becomes possible.
5. Click on the "Upload from device" button.
   The load is executed.

Result

The blocks will be uploaded from the device to the project. You can edit them as normal, recompile them and download them to the device again.

Downloading blocks to a memory card

Requirement

- The memory card is marked as a program card.
- The "Program blocks" folder of the memory card is open.

Procedure

To download blocks to a memory card, follow these steps:

1. Open the "Program blocks" folder of the device in the project tree.
2. Select the blocks you want to download to the memory card.
3. Drag the blocks in project tree to the "Program blocks" folder of the memory card. You can also copy the blocks and add them to the memory card.
   If necessary, the blocks are compiled. The "Load preview" dialog then opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
5. As soon as downloading becomes possible, the "Load" button is enabled.
6. Click the "Load" button.
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
7. Click "Finish".

Or:

1. In the project tree, select the blocks that you want to upload.
2. Select the "Card reader/USB memory > Write to memory card" command in the "Project" menu.
   The "Select memory card" dialog opens.
3. Select a memory card which is compatible with the CPU.
   A button with a green check mark is activated at the bottom of the dialog.
4. Select the button with the green check mark.
   If necessary, the project data are compiled. The "Load preview" dialog then opens. This dialog displays alarms and proposes actions necessary for loading.
5. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as downloading becomes possible, the "Load" button is enabled.
6. Click the "Load" button.
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
7. Click "Finish".

Result

The blocks are downloaded to the memory card. If the changes affect additional blocks, these will also be downloaded to the memory card. Blocks that exist only on the memory card are deleted. Inconsistencies between the blocks in the user program are avoided by downloading all affected blocks and the deleting of the non-required blocks on the memory card.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Uploading blocks from a memory card (Page 1801)

Accessing memory cards (Page 468)

Uploading blocks from a memory card

You can only upload all blocks from one memory card back into your project.

Requirement

The memory card is displayed.

See also: Accessing memory cards (Page 468)

Procedure

To upload blocks from a memory card to your project, follow these steps:

1. In the project tree, drag the folder of the memory card to the folder of the device in the project. You can also copy the memory card and insert it in the device.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for loading.
2. Check the alarms and, where necessary, enable the actions in the "Action" column.
   The "Upload from device" button will be enabled as soon as uploading becomes possible.
3. Click on the "Upload from device" button.

See also

Downloading blocks to a memory card (Page 1799)

Switching off the sequencer prior to loading a GRAPH DB

You can specify the switching off of the sequencer prior to loading an instance data block either globally or during the load operation.

Globally switching off the sequencer

To switch off the sequencer globally for each loading operation of an instance data block, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Select the "Turn off sequence before downloading DB" check box.
   For future loading operations, the sequencer is switched off prior to loading of the instance data block.

Switching off the sequencer during the loading operation

To switch off the sequencer during the loading operation, follow these steps:

1. Load the GRAPH function block to the device.
   During the loading operation, the "Load preview" dialog opens. This dialog displays alarms and suggests the required actions for loading. If the instance data block must be loaded together with the GRAPH function block, the "Load preview" dialog suggests the action "Turn off sequence before downloading the DB".
2. Select the "Turn off sequence before downloading DB" check box.

11.1.7 Protecting blocks

11.1.7.1 Protecting blocks

Introduction

You can use a password to protect one or more blocks of the OB, FB, FC type and global data blocks from unauthorized access. Note the following particularities:

- You can not manually protect instance data blocks; they depend on the know-how protection of the assigned FB. This means that when you create an instance data block for a know-how protected FB, the instance data block also receives this know-how protection. This is independent of whether you explicitly create the instance data block or if it is created by a block call.
- The know-how protection acts as write protection in global blocks and in instance data blocks.
- You cannot provide ARRAY data blocks with know-how protection.

If a block is know-how protected, only the following data is readable without the correct password:

- Interface parameters Input, Output, InOut, Return, Static
- Block title
- Block comment

- Block properties
- Tags of global data blocks without specification of the location of use

The following actions can be performed with a know-how protected block:

- Copying and deleting
- Calling in a program
- Online/offline comparison
- Downloading

The code of the block, on the other hand, is protected from unauthorized reading and modification. For S7-1200/1500 CPUs, you can also set up copy protection which binds execution of the block to the CPU or the memory card with the defined serial number.

Note

Please note the following:

- S7-1200 version 1.0 and S7-300/400 (only GRAPH and SCL blocks): If you download a know-how-protected block to a device, no restore information is loaded along with it. This means that you cannot open a know-how-protected block again even with the correct password if you upload it from the device.
- Only the non-protected data is compared in offline-online comparison of know-how protected blocks.
- You will no longer be able to access the block if you do not have the password.
- If you add a know-how-protected block to a library, the master copy created will also be know-how protected.
- For S7-1500, you can select the "Block can be used as know-how protected library item" check box in the block properties to obtain the information on whether or not the block can be used as protected library item. For this purpose, the block cannot use any tags from the operand areas Output (Q), Input (I), Bit memory (M), Timer function (T) or Count function (C), and cannot access data blocks.
- Cross references to used tags, bit memories, inputs and outputs in know-how protected blocks are not displayed even after the correct password is entered..
- If you change the number of a block, the loadable binary component of the block is out of date. This means that the block must be recompiled before loading it to a device. For know-how-protected blocks, this is only possible with the correct password. Keep this in mind particularly if you want to copy a know-how-protected block to another device in which there is already a block with the same number.
- Always pass on a project that includes know-how-protected blocks as a project archive or library archive. In this way, you ensure that the know-how protection cannot be bypassed.
- If you wish to assign know-how protection to several blocks using multiple selection, no instance data block can be included in the selection. Otherwise, it will not be possible to set know-how protection.
- You cannot change the know-how protection settings for an open, know-how protected block.

See also

Setting up and removing block copy protection (Page 1804)

Setting up block know-how protection (Page 1805)

Opening know-how protected blocks (Page 1806)

Printing know-how protected blocks (Page 1807)

Removing block know-how protection (Page 1808)

Changing a password (Page 1808)

Creating compressed project archive (Page 387)

Archiving global libraries (Page 493)

11.1.7.2 Setting up and removing block copy protection

For S7-1200 /1500 CPUs, you can set up copy protection which binds execution of the block to a specific CPU or a specific memory card. The block can then only be executed if it is in the device with the set serial number.

It is important that you also know-how-protect any block for which you have set up copy protection. If you do not, anyone can reset the copy protection.

Note

S7-1500 and S7-1200 V2.2 and higher: If you download a copy protected block to a device that does not match the specified serial number, the entire download operation will be rejected. This means that blocks without copy protection, too, will not be downloaded.

Setting up copy protection

To set up copy protection for a block, follow these steps:

1. Open the block you wish to copy-protect.
2. Open the "Properties" tab in the inspector window.
3. Select "Protection" in the area navigation in the inspector window.
4. Select either "Bind to serial number of the CPU" or "Bind to serial number of the memory card" from the drop-down list in the "Copy protection" area.
5. Enter the serial number of the CPU or the memory card for a S7-1500 CPU. You can either enter the serial number directly for a S7-1200 CPU or enable the option "Serial number inserted when downloading to a device or memory card" if the serial number is to be inserted automatically during loading.
6. You can now set up the know-how protection for the block in the "Know-how protection" area, if the block does not already have know-how protection.

Removing copy protection

To remove copy protection, follow these steps:

1. Open the block for which you wish to remove copy protection.
2. Open the "Properties" tab in the inspector window.
3. Select "Protection" in the area navigation in the inspector window.
4. Select "No binding" in the drop-down list in the "Copy protection" area.

See also

Protecting blocks (Page 1802)
Setting up block know-how protection (Page 1805)
Opening know-how protected blocks (Page 1806)
Printing know-how protected blocks (Page 1807)
Removing block know-how protection (Page 1808)
Changing a password (Page 1808)

11.1.7.3 Setting up block know-how protection

You can set up know-how protection for blocks in the devices in your project.

Procedure

To set up block know-how protection, follow these steps:

1. Select the blocks with no know-how protection which you want to protect.
2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.
3. Click "Define".
   The "Define password" dialog box opens.
4. Enter a password in the "New" field.
5. Enter the same password in the "Confirm" field.
6. Confirm your entries with "OK".
7. Close the "Know-how protection" dialog by clicking on "OK".

Result

The blocks selected will be know-how-protected. Know-how protected blocks are marked with a lock in the project tree. The password entered is valid for all blocks selected.

See also

Protecting blocks (Page 1802)

Setting up and removing block copy protection (Page 1804)

Opening know-how protected blocks (Page 1806)

Printing know-how protected blocks (Page 1807)

Removing block know-how protection (Page 1808)

Changing a password (Page 1808)

11.1.7.4 Opening know-how protected blocks

You can only open multiple know-how protected blocks at once if they are protected with the same password.

Procedure

To open a know-how protected block, follow these steps:

1. Double-click on the block you wish to open.
   The "Access protection" dialog will open.
2. Enter the password for the know-how protected block.
3. Confirm your entry with "OK".

Result

The know-how protected block will open provided you have entered the correct password. However, the block will remain know-how protected. If you copy the block or add it to a library, for example, the copies will also be know-how protected.

Once you have opened the block, you can edit the program code and the block interface of the block for as long as the block or TIA Portal is open. The password must be entered again the next time the block is opened. If you close the "Access protection" dialog with "Cancel", the block will open but the block code will not be displayed and you will not be able to edit the block.

See also

Protecting blocks (Page 1802)

Setting up and removing block copy protection (Page 1804)

Setting up block know-how protection (Page 1805)

Printing know-how protected blocks (Page 1807)

Removing block know-how protection (Page 1808)

Changing a password (Page 1808)

11.1.7.5 Printing know-how protected blocks

You can only print complete know-how protected blocks if they have been opened with the correct password. If you print a closed block or if the block was not opened with the correct password, only the non-protected block data will be printed.

Procedure

To print a know-how protected block in full, follow these steps:

1. Open the know-how protected block you wish to print.
   See also: Opening know-how protected blocks (Page 1806)
2. Select the "Print" command in the "Project" menu.
   The "Print" dialog will open.
3. Select the printer in the "Name" field.
4. Click "Advanced" to modify the Windows printer settings.
5. Select the documentation information set in the "Document information" drop-down list that you want to use for the frame layout.
6. Under "Print objects/area" select whether you want to print all objects or the complete area, or only a selection.
7. Under "Properties" select the print scope.
   - Select "All" to print the complete block.
   - Choose "Visible" to print all the information within the block that is visible on the screen.
   - Select "Compact" to print a shortened form of the block.
8. Click "Preview" to generate a print preview in advance.
   A print preview is created in the work area.
9. Click "Print" to start the printout.

See also

Printing project contents (Page 419)

Protecting blocks (Page 1802)

Setting up and removing block copy protection (Page 1804)

Setting up block know-how protection (Page 1805)

Removing block know-how protection (Page 1808)

Changing a password (Page 1808)

11.1.7.6 Changing a password

Procedure

To change the password, follow these steps:

1. Select the know-how protected blocks for which you want to change the password.

   Note

   You can only change the password for several blocks at once if all blocks selected have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.
3. Click the "Change" button.
4. Enter the old password in the "Old" field.
5. Enter the new password in the "New" field.
6. Enter the new password again in the "Confirm" field.
7. Confirm your entries with "OK".
8. Close the "Know-how protection" dialog by clicking on "OK".

See also

Protecting blocks (Page 1802)

Setting up and removing block copy protection (Page 1804)

Setting up block know-how protection (Page 1805)

Opening know-how protected blocks (Page 1806)

Printing know-how protected blocks (Page 1807)

Removing block know-how protection (Page 1808)

11.1.7.7 Removing block know-how protection

Procedure

To remove block know-how protection, follow these steps:

1. Select the blocks for which you want to remove know-how protection.

   Note

   You can only remove know-how protection for several blocks at once if all blocks selected have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.

3. Deactivate the check box "Hide code (know-how protection)".
4. Enter the password.
5. Confirm your entries with "OK".

Result

Know-how protection will be disabled for the blocks selected.

See also

Protecting blocks (Page 1802)
Setting up and removing block copy protection (Page 1804)
Setting up block know-how protection (Page 1805)
Opening know-how protected blocks (Page 1806)
Printing know-how protected blocks (Page 1807)
Changing a password (Page 1808)

11.2 Displaying program information

11.2.1 Overview of available program information

Program information

The program information of a user program contains the view specified in the following table.

View	Application
Assignment list (Page 1810)	Provides an overview of the address bits for the I, Q, and M memory areas already allocated within the user program. Also indicates if an address has been allocated by access from an S7 program or if the address has been assigned to a SIMATIC S7 module.
Call structure (Page 1818)	Shows the call structure of the blocks within the user program and provides an overview of the blocks used and their relationships.

View	Application
Dependency structure (Page 1824)	Shows the list of blocks used in the user program. A block is shown at the first level and blocks that call or use this block are indented below it. In contrast to the call structure, instance blocks are listed separately.
Resources (Page 1830)	Shows the hardware resources of the CPU for objects (OB, FC, FB, DB, user-defined data types and PLC tags), for CPU memory areas and for the existing I/O modules.

Displaying several views simultaneously

You can generate and display several views for one or more user programs to facilitate testing and changing your user program.

Displaying multiple views, for example, enables you to:

- Display all program information for a user program next to one another
- Compare different user programs

11.2.2 Displaying an assignment list

11.2.2.1 Introduction to the assignment list

Program information in the assignment list

The assignment list shows if an address has been allocated by access from an S7 program or if the address has been assigned to a SIMATIC S7 module. It is therefore an important basis for locating errors or changes in the user program.

In the assignment list, you have a CPU-specific overview of which bit is used in which byte of the memory areas listed below:

- Input (I)
- Output (O)
- Bit memory (M)
- Timer (T)
- Counter (C)
- I/O (P)

Display of the assignment list

The assignment list of inputs, outputs, and bit memory is displayed in several separate work windows.

Filters

You can filter the display within the assignment list. You can use predefined filters or create your own.

Displaying cross-reference information

You have the option of displaying cross-reference information for selected addresses in the assignment list.

You can display the cross-references for a selected address in the Inspector window using the "Cross-reference information" shortcut menu command. The command "Tools > Cross-references" allows you to also open the cross-reference list for the selected object.

Displaying the PLC tag table

You can open the PLC tag table from the assignment list and edit the properties of the tags used.

To do this select an address of the assignment list and select the "Open editor" command in the shortcut menu.

Enabling the display of retentivity

You can enable and disable the display of the retentive state of bit memory by selecting the "Hide/show retain area" toolbar button.

See also

Symbols in the assignment list (Page 1812)

Layout of the assignment list (Page 1811)

11.2.2.2 Layout of the assignment list

Layout of the assignment list

Depending on the CPU, the assignment list is displayed in several work windows with the following operands.

For S7-300/400 CPUs:

- Inputs
- Outputs
- Bit memory
- Timers
- Counters

For S7-1200 CPUs:

- Inputs
- Outputs
- Bit memory

Displaying inputs, outputs, bit memory, timers and counters

It shows all operands used and their assignment in the S7 program.

For all displayed operands, each line in the assignment list is dedicated to a byte of the memory area, in which the corresponding eight bits from 7 to 0 are labeled according to their access. In conclusion, a "bar" indicates if access is made by a byte (B), word (W) or double word (D).

You can find an explanation of the symbols in the assignment list here. (Page 1812)

See also

Introduction to the assignment list (Page 1810)

11.2.2.3 Symbols in the assignment list

Meaning of the symbols in the assignment list

The following table shows the meaning of the symbols in the assignment list:

Symbol	Meaning
	Indicates the address assignment in the selected state.
	Indicates the address assignment in the non-selected state.
	Indicates that a pointer start address and a tag address access the same address range and that they are selected.
	Indicates that a pointer start address and a tag address access the same address range and that they are not selected.
	Indicates the pointer assignment in the selected state.
	Indicates the pointer assignment in the non-selected state.
	Indicates that the byte is in use with byte access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
	Indicates that the byte is in use with byte access and the corresponding tag is not selected.
	Indicates that the byte is in use with word access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.

Symbol	Meaning
	Indicates that the byte is in use with word access and the corresponding tag is not selected.
	Indicates that the byte is in use with double word access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
	Indicates that the byte is in use with double word access and the corresponding tag is not selected.
Background color: gray	Indicates that a byte is in use with byte, word or double word access and that the address is also in use by the hardware. The gray background color indicates overlapping memory access.
Background color: yellow	Indicates that the address is not in use by the hardware.
	Indicates that the memory area has been defined as system memory.
	Indicates that the memory area has been defined as clock memory.

See also

Layout of the assignment list (Page 1811)

Introduction to the assignment list (Page 1810)

11.2.2.4 Displaying an assignment list

Requirement

A project has been created with programmed blocks.

Procedure

Proceed as follows to display the assignment list:

1. Select the "Program blocks" folder or one or more of the blocks it contains.
2. Select the "Assignment list" command in the "Tools" menu.

Result

The assignment list for the selected program is displayed.

View options in the assignment list

Refer to view respective view options that are set to display the desired information in the assignment list.

See also

Setting the view options for the assignment list (Page 1814)

Layout of the assignment list (Page 1811)

11.2.2.5 Setting the view options for the assignment list

Introduction

The following view options are available for the assignment list:

- Used addresses:
  When this check box is activated, the addresses, I/Os and pointers used in the program are displayed.
- Free hardware addresses:
  When this check box is activated, only the free hardware addresses are displayed.

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Procedure

Proceed as follows to set the view options for the assignment list:

1. Click on the arrow of the symbol ("View options") in the task bar.
   The view options for the assignment list are opened. Check marks are set in front of the activated view options.
2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the desired information is displayed in the assignment list.

11.2.2.6 Filter options in the assignment list

Filter settings

You can define your own filter settings for the assignment list. The following options are available for defining filters:

- Display all addresses of the address areas specified.
- Display of single, defined addresses from the selected address area, for example, "0" and "200".
- Display of complete areas from the selected address area, for example, "0 - 256".

The following table provides an overview of all available options:

Selection in the	Selection	Symbol	Meaning
Address area	All CPU-dependent displayed addresses (I, O, M, T, C) can be activated as they are by default, or individual address areas can be activated.	Check box is activated	Only the activated address areas (I, O, M, T, C) are shown in the assignment list.
Filter area	Show assignment for all addresses	*	Displays the assignment of all addresses of the enabled address areas (I, Q, M).
	Show assignment for selected addresses, for example, for the inputs "IB 0" and "IB 256"	0;256 Separate individual addresses and areas by a semicolon.	Assignments of selected addresses for the activated address areas (I) are shown.
	Show assignment for selected areas, for example, for the inputs "IB 0 to IB 100" and "IB 200 to IB 256".	0-100;200-256 Contiguous areas should be connected by a hyphen.	Assignments of selected areas for the activated address areas (I) are shown.

11.2.2.7 Defining filters for assignment list

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Defining filter

Proceed as follows to define a filter for the assignment list:

1. Click on the symbol ("Filter") in the task bar.
   The "Assignment List Filter" dialog opens.
2. Click on the symbol ("Create new filter") in the task bar.
   A new filter is created with the name "Filter_1". The check boxes for all addresses (inputs, outputs, memory bits, timers and counters) are activated by default for the filter.

3. If you want to change the name of the filter, click on the drop-down list in the task bar and enter a new filter name.
4. Deactivate the check boxes of addresses that are not to be affected by the filter.
5. Enter one of the following options in the filter area of the activated address:
   - Show all addresses used = "*"
   - Show single, defined addresses, for example, IB 0" and IB 25 = "0.25". Individual addresses and address areas are separated by commas or semicolons.
   - Show complete address areas, for example, IB 0 to IB 256 = "0-256". Complete address areas should be connected by a hyphen.
6. Confirm your entries with "OK".
   The newly defined filter is shown in the task bar of the assignment list under the specified name.

Delete filter

Proceed as follows to delete a filter:

1. Click on the symbol ("Filter") in the task bar.
   The filter dialog for the assignment list opens.
2. In the drop-down list of the task bar, select the filter you want to delete.
3. Click on the symbol ("Delete selected filter") in the task bar.
   The selected filter is deleted.

See also

Filter options in the assignment list (Page 1815)

Displaying an assignment list (Page 1813)

Introduction to the assignment list (Page 1810)

11.2.2.8 Filtering an assignment list

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Procedure

1. Click on the arrow on the drop-down list.
   The available filter are displayed.
2. Select the desired filter.

Result

The assignment list is filtered according to the settings of the selected filter.

Note

The filter settings are saved when the project is closed.

11.2.2.9 Defining retentive memory areas for bit memories

Introduction

In the assignment list you can define the width of the retentive memory area for bit memories. The content of tags which are addressed in retentive memory is retained after power off and at the STOP to RUN transition after power on.

The display of retentive bit memories can be enabled and disabled in the assignment list. If their display is enabled, retentive bit memories are identified by an icon in the "Address" column.

Requirement

The assignment list is open.

Procedure

Proceed as follows to define the width of the retentive memory area for bit memories:

1. Click "Retain" in the toolbar.
   The "Retain memory" dialog will open.
2. Starting at the count of 0, define the width of the retentive memory area by entering the last byte of this area in the input field. Watch out for any addresses of tags already assigned to the retentive area.
3. Load the block to the target system. Select the "Program blocks" folder in the Project tree and select the "Download to device" submenu in the shortcut menu.

Result

The width of the retentive memory area is defined. If enabled in the assignment list, an icon will indicate the retentive state of all tags in the "Address" column.

11.2.2.10 Enabling the display of retentive bit memories

Introduction

In the assignment list you can enable and disable the display of retentive bit memories. The retentive bit memories are identified by means of an icon in the "Address" column if the display of retentivity is enabled.

Requirement

The assignment list is open.

Procedure

Proceed as follows to enable and disable the display of retentive bit memories:

1. Click "Display/hide retentivity" in the toolbar.

Result

The retentive tags are identified by means of an icon in the "Address" column of the bit memory area if the display of retentivity is enabled. The icons in the "Address" column are hidden if the display of retentivity is disabled.

11.2.3 Displaying the call structure

11.2.3.1 Introduction to the call structure

Call structure

The call structure describes the call hierarchy of the block within an S7 program.

It provides an overview of:

- The blocks used
- Jumps to the places of use of the blocks
- Relationships between blocks
- Local data requirements of the blocks
- Status of the blocks

Information in the call structure

Displaying the call stucture provides you with a list of the blocks used in the user program. The first level of the call structure is highlighted in color and shows the blocks that are not called by any other block in the program. Organization blocks are always shown on the first level of the call structure. Functions, function blocks and data blocks are only shown on the first level if they are not called by an organization block. When a block calls other blocks or functions, they are listed indented under the calling block. Instructions and blocks are shown in the call structure only if they are called by a block.

View options

The following view options are available for the call structure:

- Show conflicts only:
  When this check box is activated, only the conflicts within the call structure are displayed.
- Group multiple calls together:
  When this check box is activated, several block calls are grouped together. The number of block calls is displayed in the "Call frequency" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Displaying the block calls

You can display the block calls in a block by clicking on the arrow in front of the block title. To display the call information of all blocks, click on the "Expand list" icon in the toolbar.

You can hide the total overview by clicking the "Collapse list" icon.

Displaying cross-reference information

You can display the cross-reference information for a block in the Inspector window by right-clicking on the relevant block and selecting the "Cross-reference information" command from the shortcut menu.

To open the "Cross-references" view, click the "Cross-references" command in the shortcut menu.

Displaying blocks in the program editor

You can open the program editor and edit blocks there from the call structure.

To do this select the required block in the call structure and select the "Open editor" command in the shortcut menu.

Displaying deleted blocks

The rows belonging to deleted blocks are identified by an icon.

Note

Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Symbols in the call structure (Page 1820)

11.2.3.2 Symbols in the call structure

Meaning of the symbols in the call structure

The following table shows the meaning of the symbols in the call structure:

Symbol	Meaning
	Indicates an organization block (OB).
	Indicates a function block (FB).
	Indicates a function (FC).
	Indicates a data block (DB).
	Indicates that the block is declared as a multiinstance.
	The object has an interface dependency to an object connected to the left.
	Indicates that the block needs to be compiled again.
	Indicates that the data block needs to be compiled again.
	Indicates that the object is not available.
	Indicates that the interface causes a time stamp conflict.
	Indicates that the variable causes a time stamp conflict.
	Indicates that the block is not called directly or indirectly from an OB.
	Indicates that an object has know-how protection.

Symbol	Meaning
	Indicates that the block is normally called recursively.
	Indicates that a tag declaration in the interface has a recursive dependency: • Scenario 1: FB1 calls FB2 and this then calls FB1. The instance data blocks of these FBs have a recursion in the interface. • Scenario 2: A multiple instance FB uses the instance DB of its parent FB as a global DB.

11.2.3.3 Layout of the call structure

Layout of the call structure

The view of the call structure consists of the following columns:

Column	Content/meaning
Call structure	Shows an overview of the blocks called If the viewing option "Group multiple calls together" is enabled, several block calls are grouped together and the "Number of calls" column is displayed.
Call type (!)	Shows the type of call, for example recursive block call.
Address	Shows the absolute address of the block. With a function block, the absolute address of the corresponding instance data block is also shown.
Call frequency	Indicates the number of multiple calls of blocks.
Details	Shows the network or interface of the calling block. All information are offered as a link in this column. With this link, you can jump to the location of the block call in the program editor. If the viewing option "Group multiple calls together" option is enabled, the calls are grouped together and are available as links in a drop-down list.
Local data (in path)	Indicates the local data requirement of the full path. Blocks with optimized access have higher local data requirements because the information for the symbolic addressing is stored with them. Please note that any existing local data can only be displayed or updated after compiling a block.
Local data (for blocks)	Show the local data requirements of the block. Blocks with optimized access have higher local data requirements because the information for the symbolic addressing is stored with them. Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Symbols in the call structure (Page 1820)

Introducing the consistency check in the call structure (Page 1823)

11.2.3.4 Displaying the call structure

Requirement

A project has been created with blocks.

Procedure

Proceed as follows to display the call structure:

1. Select the "Program blocks" folder or one or more of the blocks it contains.
2. Select the "Call structure" command in the "Tools" menu.

Result

The call structure for the selected program is displayed.

Note

Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Setting the view options for the call structure (Page 1822)

11.2.3.5 Setting the view options for the call structure

Introduction

The following view options are available for the call structure:

- Show conflicts only:
  Only the blocks causing conflicts within the call structure are displayed if this check box is activated.
  The following blocks cause conflicts:
  - Blocks executing any calls with older or newer code time stamps.
  - Blocks calling a block with modified interface.
  - Blocks using a tag with modified address and/or data type.
  - Block called neither directly, nor indirectly by an OB.
  - Blocks calling a block which no longer exists.
- Group multiple calls together:
  When this viewing option is enabled, several block calls and data block accesses are grouped together. The number of block calls is displayed in the "Call frequency" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Requirement

- A project has been created with programmed blocks.
- The call structure is open.

Procedure

Proceed as follows to set the view options for the call structure:

1. Click on the arrow of the symbol ("View options") in the task bar.
   The view options for the call structure opens. Check marks are set in front of the activated view options.
2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the required information is displayed in the call structure.

11.2.3.6 Introducing the consistency check in the call structure

Consistency check

Changing the time stamp of a block during or after the program is generated can lead to time stamp conflicts, which in turn cause inconsistencies among the blocks that are calling and being called.

Using the consistency check

The "Consistency check" function is used to visualize inconsistencies when time stamp conflicts occur. Whe the consistency check is performed, the inconsistent blocks are shown in the call structure and marked with the correspoinding symbols.

- Most time stamp and interface conflicts can be rectified by recompiling the blocks.
- If compilation fails to clear up inconsistencies you can use the link in the "Details" column to go to the source of the problem in the program editor and manually eliminate any inconsistencies.
- The blocks marked in red must be recompiled.

See also

Symbols in the call structure (Page 1820)

11.2.3.7 Checking block consistency in the call structure

Requirement

- A project has been created with programmed blocks.
- The call structure is open.

Procedure

Proceed as follows to check the block consistency:

1. Click on the symbol ("Consistency check") in the task bar.
   The block consistency is checked. Blocks found to be inconsistent are marked accordingly by a symbol.
2. If a block is inconsistent, click on the arrow in front of the block title in the call structure.
   The inconsistent blocks are displayed. The exact problem locations are listed as links in the "Details" column.
3. Click on the respective link in the "Details" column to jump to the location in the block requiring correction.
4. Check and correct the inconsistencies in the blocks.
5. Recompile the blocks by selecting the required blocks and clicking on the command "Compile" in the shortcut menu.
6. Download the corrected blocks to the target system by clicking the command "Download to device" in the shortcut menu.

Result

The block consistency is checked. The inconsistencies in the blocks are corrected. The corrected blocks are loaded to the target system.

See also

Symbols in the call structure (Page 1820)

11.2.4 Displaying the dependency structure

11.2.4.1 Introduction to the dependency structure

Introduction

The dependency structure shows the dependencies each block has to other blocks in the program.

Information in the dependency structure

Displaying the dependency structure provides you with a list of the blocks used in the user program. A block is shown at the far left and blocks that call or use this block are indented below it.

The dependency structure also shows the status of the individual blocks using symbols.

Objects causing a time stamp conflict and perhaps leading to an inconsistency in the program are marked with various symbols.

The dependency structure is an extension of the cross-reference list for objects.

View options

The following view options are available for the dependency structure:

- Show conflicts only:
  When this check box is activated, only the conflicts within the dependency structure are displayed.
- Group multiple calls together:
  When this check box is activated, several block calls are grouped together. The number of block calls is shown numerically in the "Dependency structure" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Displaying the dependency structure

Clicking on the arrow in front of the block title displays the blocks that call or use this block. To display the dependencies of all blocks,

click the "Expand list" icon in the toolbar.

You can hide the total overview by clicking the "Collapse list" icon.

Displaying cross-reference information

You can display the cross-reference information for a block in the Inspector window by right-clicking on the respective block and selecting the "Display Usage" command from the shortcut menu.

Displaying blocks in the program editor

You can open the program editor and edit blocks there from the dependency structure. To do this select the required block in the dependency structure and select the "Open editor" command in the shortcut menu.

11.2.4.2 Layout of the dependency structure

Layout of the dependency structure

The view of the dependency structure consists of the following columns:

Column	Content/meaning
Dependency	It indicates the dependencies between each block and the other blocks in the program.
Call type (!)	Shows the type of call, for example recursive block call.
Address	Shows the absolute address of the block.
Call frequency	Indicates the number of multiple calls of blocks.
Details	Shows the network or interface of the called block. All information are offered as a link in this column. With this link, you can jump to the location of the block call in the program editor. If the viewing option "Group multiple calls together" option is enabled, the calls are grouped together and are available as links in a drop-down list.

See also

Symbols in the dependency structure (Page 1826)

11.2.4.3 Symbols in the dependency structure

Meaning of the symbols in the dependency structure

The following table shows the meaning of the symbols in the dependency structure:

Symbol	Meaning
	Indicates an organization block (OB).
	Indicates a function block (FB).
	Indicates a function (FC).
	Indicates a data block (DB).
	The object has an interface dependency to an object connected to the left.
	Indicates that the block needs to be compiled again.
	Indicates that the data block needs to be compiled again.
	Indicates that there is an inconsistency with this object.

Symbol	Meaning
	Indicates that an object has know-how protection.
	Indicates that a tag declaration in the interface has a recursive dependency: • Scenario 1: FB1 calls FB2 and this then calls FB1. The instance data blocks of these FBs have a recursion in the interface. • Scenario 2: A multiple instance FB uses the instance DB of its parent FB as a global DB.

11.2.4.4 Displaying the dependency structure

Requirement

A project has been created with programmed blocks.

Procedure

Proceed as follows to display the dependency structure:

1. Select the block folder or one or more of the blocks contained therein.
2. Select the "Dependency structure" command in the "Tools" menu.

Result

The dependency structure for the selected program is displayed.

See also

Setting the view options for the dependency structure (Page 1828)

11.2.4.5 Setting the view options for the dependency structure

Introduction

The following view options are available for the dependency structure:

- Show conflicts only:
  When this check box is activated, only the conflicts within the dependency structure are displayed.
  The following blocks cause conflicts:
  - Blocks executing any calls with older or newer code time stamps.
  - Blocks called by a block with modified interface.
  - Blocks using a tag with modified address and/or data type.
  - Block called neither directly, nor indirectly by an OB.
- Group multiple calls together:
  When this check box is activated, several block calls are grouped together. The number of block calls is shown in the relevant column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Requirement

- A project has been created with programmed blocks.
- The dependency structure is open.

Procedure

Proceed as follows to set the view options for the dependency structure:

1. Click on the arrow of the symbol ("View options") in the task bar.
   The view options for the dependency structure are opened. Check marks are set in front of the activated view options.
2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the required information is displayed in the dependency structure.

11.2.4.6 Introducing the consistency check in the dependency structure

Consistency check

Changing the time stamp of a block during or after the program is generated can lead to time stamp conflicts, which in turn cause inconsistencies among the blocks that are calling and being called.

Using the consistency check

The "Consistency check" function is used to visualize inconsistencies. Whe the consistency check is performed, the inconsistent blocks are shown in the dependency structure and marked with the correspoinding symbols.

- Most time stamp and interface conflicts can be rectified by recompiling the blocks.
- If compilation fails to clear up inconsistencies you can use the link in the "Details" column to go to the source of the problem in the program editor and manually eliminate any inconsistencies.
- The blocks marked in red must be recompiled.

See also

Layout of the dependency structure (Page 1826)

Symbols in the dependency structure (Page 1826)

11.2.4.7 Checking block consistency in the dependency structure

Requirement

- A project has been created with programmed blocks.
- The dependency structure is open.

Procedure

Proceed as follows to check the block consistency:

1. Click on the symbol ("Consistency check") in the task bar.
   The block consistency is checked. Blocks found to be inconsistent are marked accordingly by a symbol.
2. If a block is inconsistent, click on the arrow in front of the block title in the dependency structure.
   The inconsistent blocks are displayed. The exact problem locations are listed as links in the "Details" column.
3. Check and correct the inconsistencies in the blocks.
4. Recompile the blocks by selecting the required blocks and clicking on the command "Compile" in the shortcut menu.
5. Download the corrected blocks to the target system by clicking the command "Download to device" in the shortcut menu.

Result

The block consistency is checked. The inconsistencies in the blocks are corrected. The corrected blocks are loaded to the target system.

See also

Symbols in the dependency structure (Page 1826)

11.2.5 Displaying CPU resources

11.2.5.1 Introducing resources

Introduction

The "Resources" tab indicates the hardware resources of the configured CPU for:

- the used programming objects,
- the assignment of the different memory areas within the CPU and
- the assigned inputs and outputs of the existing input and output modules.

Information provided in the "Resources" tab

The resources tab provides an overview of the hardware resources. The display in this tab depends on the CPU which you are using. The following information is displayed:

- the programming objects used in the CPU (e.g. OB, FC, FB, DB, data types and PLC tags)
- the memory areas available on the CPU (load memory, work memory - divided into code work memory and data work memory depending on the CPU -, retentive memory), their maximum size and utilization by the programming objects stated above
- the I/O of modules which can be configured for the CPU (I/O modules, digital input modules, digital output modules, analog input modules, and analog output modules), including the I/O already in use.

Display of the maximum available load memory

The maximum size of available load memory can be selected from a drop-down list box in the "Total" row of the "Load memory" column.

Display of the maximum available work memory

The maximum size of available work memory is displayed in the "Work memory" column or in the "Code work memory" and "Data work memory" columns in the "Total" row.

Display of the maximum available retentive memory

The maximum size of available retentive memory can be selected from a drop-down list box in the "Total" row of the "Retentive memory" column.

Note

Retentive memory data

All bit memories and data blocks specified as retentive will be integrated in the calculation of the retentive data.

Updating the display in the "Resources" tab

Click the "Update view" toolbar button to update the display of objects.

Benefits of the display in the "Resources" tab

The "Resources" tab of the program information dialog provides a detailed list of all objects and of the corresponding memory area used.

The tab also indicates shortage of resources and helps to avoid such states.

Blocks which are not compiled can be identified as their size is indicated by a question mark.

See also

Layout of the "Resources" tab (Page 1832)

Displaying resources (Page 1833)

Selecting the maximum load memory available (Page 1834)

11.2.5.2 Layout of the "Resources" tab

Layout of the "Resources" tab in the program information

The view of the "Resources" tab consists of the following columns:

Column	Content/meaning
Objects	The "Details" area provides an overview of the programming objects available in the CPU, including their memory assignments.
Load memory	Displays the maximum load memory resources of the CPU as a percentage and as absolute value. The values displayed under "Total" provide information on the maximum memory available in the load memory. The values displayed under "Used" provide information on the memory actually used in the load memory. If a value is displayed in red, the available memory capacity has been exceeded.
Work memory or code and data work memory	Displays the maximum work memory resources of the CPU as a percentage and as absolute value. The work memory depends on the CPU and is divided into "Code work memory" and "Data work memory" for a CPU from the S7-400 or S7-1500 series, for example. The values displayed under "Total" provide information on the maximum memory available in the work memory. The values displayed under "Used" provide information on the memory space actually used in the work memory. If a value is displayed in red, the available memory capacity has been exceeded.
Retentive memory	Displays the maximum resources for retentive memory in the CPU as a percentage and as absolute value. The values displayed under "Total" provide information on the maximum memory available in the retentive memory. The values displayed under "Used" provide information on the memory actually used in the retentive memory. If a value is displayed in red, the available memory capacity has been exceeded.

Column	Content/meaning
I/O	Displays the I/Os which are available on the CPU, including their module-specific availability in the next columns. The values displayed at "Configured" provide information about the maximum number of I/O available. The values displayed under "Used" provide information on the actually used inputs and outputs.
DI / DQ / AI / AQ	Displays the number of configured and used inputs/outputs: DI = Digital inputs DQ = Digital outputs AI = Analog inputs AQ = Analog outputs The values displayed at "Configured" provide information about the maximum number of I/O available. The values displayed under "Used" provide information on the actually used inputs and outputs.

See also

Displaying resources (Page 1833)

Selecting the maximum load memory available (Page 1834)

Introducing resources (Page 1830)

11.2.5.3 Displaying resources

Requirement

A project with programmed blocks has been created.

Procedure

Proceed as follows to display the resources of the respective CPU memory areas:

1. Select the block folder below the relevant CPU, or one or several of the blocks contained therein.
2. Select the "Resources" command in the "Tools" menu.

Result

The memory resources of the assigned CPU are displayed.

11.2.5.4 Selecting the maximum load memory available

Requirement

A project with programmed blocks has been created.

Procedure

Proceed as follows to display the available maximum of load memory resources:

1. Select the block folder below the relevant CPU, or one or several of the blocks contained therein.
2. Select the "Resources" command in the "Tools" menu.
3. In the dialog that is displayed, open the drop-down list in the "Total" field of the "Load memory" column by clicking the icon.
4. Select a corresponding value for the CPU used by clicking it in the drop-down list box.

Result

The "Total" field displays the selected maximum memory resources.

Note

Display of maximum memory resources

If a value is displayed in red for the maximum memory resources, the available memory capacity has been exceeded.

In this case, adapt the memory capacity as described above.

11.3 Displaying cross-references

11.3.1 General information about cross references

Introduction

The cross-reference list provides an overview of the use of operands and tags within the user program.

Uses of cross-references

The cross-reference list offers you the following advantages:

- When creating and changing a program, you retain an overview of the operands, tags and block calls you have used.
- From the cross-references, you can jump directly to the point of use of operands and tags.
- During a program test or when troubleshooting, you are informed of the following:
  - Which operand is processed by which command in which block.
  - Which tag is used in which picture.
  - Which block is called by which other block.
  - Cross-reference information for subordinate and higher-level structures.
- As part of the project documentation, the cross-references provide a comprehensive overview of all operands, memory areas, blocks, tags and pictures used.

See also

Structure of the cross-reference list (Page 1835)

Displaying the cross-reference list (Page 1836)

Displaying cross-references in the Inspector window (Page 1838)

11.3.2 Structure of the cross-reference list

Views of the cross-reference list

There are two views of the cross-reference list. The difference between the two views is in the objects displayed in the first column:

- Used by:
  Display of the referenced objects. Here, you can see where the object is used.
- Used:
  Display of the referencing objects. Here, you can see the users of the object.

The assigned tool tips provide additional information about each object.

Structure of the cross-reference list

The cross-reference list has the following structure:

Column	Content/meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Number	Number of uses
Point of use	Each point of use, for example, network

Column	Content/meaning
Property	Special properties of referenced objects, for example, the tag names in multi-instance declarations.
as	Shows additional information about the object, e.g., that an instance DB is used as template or as multiple instance.
Access	Type of access, whether access to the operand is read access (R) and/or write access (W).
Address	Address of the operand
Type	Information on the type and language used to create the object
Path	Path of object in project tree

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

Settings in the cross-reference list

You can make the following settings using the buttons in the toolbar of the cross-reference list:

- Update cross-reference list
  Updates the current cross-reference list.
- Making settings for the cross-reference list
  Here, you select check boxes to specify whether all used, all unused, all defined or all undefined objects will be displayed. If the "Undefined objects" option is enabled, references to previously deleted objects are also displayed.
- Collapse entries
  Reduces the entries in the current cross-reference list by closing the lower-level objects.
- Expand entries
  Expands the entries in the current cross-reference list by opening the low-level objects.

Sorting in the cross-reference list

You can sort the entries in the "Object" column, including other product-specific columns, in ascending or descending order. To do this, click on the relevant column title.

See also

General information about cross references (Page 1834)

Displaying the cross-reference list (Page 1836)

11.3.3 Displaying the cross-reference list

Requirements

You have created a project.

Introduction

There are several ways of displaying cross-references depending on whether you are in the Portal view or in the Project view and which object you have selected in the project tree.

In the Portal view, you can only display cross-references for the entire CPU; in the Project view, you can, for example, display cross-references for the following objects:

- "PLC tags" folder
- "PLC data types" folder
- "Program blocks" folder
- "Tags and connections" folder
- Individual tags
- Individual PLC data types
- Individual blocks
- Technological objects
- Watch tables

Displaying cross-references

Proceed as follows to display cross-references:

1. Select the required action in the Portal view, for example "PLC programming" and the "Show cross-references" command or select one of the objects listed above in the Project view and select the "Cross-references" command in the "Tools" menu.
   The cross-reference list is displayed.
2. Click the "Used by" button to display where the objects shown in the cross-reference list are used.
3. Click the "Uses" button to view the users of the objects displayed in the cross-reference list.
4. You can perform the following actions using the buttons in the toolbar:
   - Update cross-reference list
   - Making settings for the cross-reference list
   - Collapse entries
   - Expand entries
5. You can sort the entries in the "Object" and "Address" columns in ascending or descending order by clicking on the relevant column title.
6. To go to the point of use of the object, click on the displayed link.

See also

General information about cross references (Page 1834)

Structure of the cross-reference list (Page 1835)

11.3.4 Displaying cross-references in the Inspector window

Introduction

The Inspector window displays cross-reference information about an object you have selected in the "Info > Cross-references" tab. This tab displays the instances where a selected object is being used and the other objects using it.

The Inspector window also includes blocks which are only available online in the cross-references.

You can use the "Show overlapping access..." shortcut menu command to also have overlapping access displayed for specific objects.

Note

Displaying "Overlapping accesses"

Please note that you can only have overlapping accesses displayed online and solely for structures such as for structured variables or structured elements in data blocks.

Structure

The Inspector window displays the cross-reference information in tabular format. Each column contains specific and detailed information on the selected object and its application. The table below shows the additional information listed in the "Info > Cross-reference" tab:

Column	Meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Number	Number of uses
Point of use	Each location of use, for example, network
Property	Special properties of referenced objects, for example, the tag name in multi-instance declarations
as	Shows additional information about the object, e.g., that an instance DB is used as template or as multiple instance.
Access	Access mode Shows whether the operand is accessed by a read (R) and/or write (W) operation.
Address	Address of the operand
Monitor value	This column will only be displayed when the program editor is open.
Type	Information about the type and language used to create the object
Path	Path of object in project tree

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

11.4 Testing the user program

11.4.1 Basics of testing the user program

Functions

You have the option of testing the running of your user program on the device. You can then monitor signal states and values of tags and can assign values to tags to simulate certain situations in the running of the program.

Requirement

There must be an executable program loaded on the device.

Test options

The following test options are available:

- Testing with program status
  The program status allows you to monitor the running of the program. You can display the values of operands and the results of logic operations (RLO) allowing you to recognize and fix logical errors in your program.
- Testing in single step mode (S7-300/400 only)
  You can test blocks you created in STL or SCL in the single step mode. You do this by setting breakpoints in the program code at which program execution stops. You can then continue to run the program one step at a time. Within a CPU, you can test either with program status or in single step mode. You cannot, however, use both test options at the same time within a CPU.
- Testing with the watch table
  With the watch table, you can monitor and modify the current values of individual tags in the user program or on a CPU. You can assign values to individual tags for testing and run the program in a variety of different situations. You can also assign fixed values to the I/O outputs of a CPU in STOP mode, for example to check the wiring.
- Testing with the force table
  With the force table, you can monitor and force the current values of individual tags in the user program or on a CPU. When you force, you overwrite individual tags with specified values. This allows you to test your user program and run through various situations. When forcing, make sure that you keep to the necessary safety measures for forcing (Page 1899)!

See also

Introduction to testing with program status (Page 1840)

Introduction to testing with the watch table (Page 1854)

Introduction for testing with the force table (Page 1881)

11.4.2 Testing with program status

11.4.2.1 Introduction to testing with program status

Program Status function

If you display the program status, you can monitor the execution of the program. This provides you with an overview of the values of the individual operands and the results of the logic operations and you can check whether the components of the automation system are correctly controlled.

The display of the program execution in the program status can differ slightly, depending on the CPU family used.

Testing with program status for S7-300/400

During testing with program status, the CPU cycle time can become extended in test mode because the recording of all test data can deviate from the duration of the programmed instructions due to the CPU capacity and therefore not run in realtime.

During the execution of the following test functions an alarm indicating the danger of a time-out is displayed once for each online session.

- During testing with call conditions
- During testing with breakpoints

You can only perform these test functions after you have acknowledged the alarm.

Note

With older CPUs from the S7-300/400 CPU family, you will need to change the operating response using the hardware configuration and then download the hardware configuration to the device. You have the option to set the "Process operation" or "Test operation" operating response.

Testing with program status for S7-1500

When you run the "Test with program status" function with a CPU from the S7-1500 family, the cycle time may increase notably while monitoring loops, which may result in the CPU changing to STOP mode.

Note

To avoid a possible STOP in the CPU , ensure that no programmed loops are displayed in the active monitoring window during "Testing with program status". Alternatively you can also increase the maximum permissible cycle time of the CPU.

Restrictions with the "Program status" function

The monitoring of loops can increase the cycle time significantly, depending in each case on the number of tags to be monitored and on the actual number of loops processed.

To ensure that the cycle time is influenced as little as possible, the "Program status" function is restricted as follows:

- The status display of a programmed loop is stopped at the return point.

Warning

Testing with program status

A test with the "Program status" function can cause serious damage to property or injury to persons if there are functional disturbances or program errors.

Make sure that no dangerous situations can arise before you conduct a test with the "Program status" function.

11.4.2.2 Setting the call environment

Basics of the call environment

Function

You can define the call environment for blocks and for breakpoints. With this, you specify the conditions for recording the program status of a block or for stopping program execution at a breakpoint.

Specification of the call environment for blocks

You can alternatively activate one of the following options to specify the call environment:

- No condition defined
  If no other option was selected this option is the default.
- Instance data block
  The program status of a function block will then only be recorded when you call the function block with the selected instance data block.
- Call path
  The program status of a block will then only be recorded when you call the block with a specific block or from a specific path.
- Manually adapted call path
  You can enter the desired call environment manually in this field. The "Transfer to "Adjusted manually"" button is used to transfer the content selected under "Call environment", which you can edit further if required.
  The program status of a block will then only be recorded when you call the block with a specific block or from a specific path.

Specification of the call environment for breakpoints

You can also specify a separate condition for each breakpoint.

If you do not specify the call environment, the program status of a block call is selected at random and recorded within the call structure and program execution is always interrupted at the relevant breakpoint. You should therefore always specify the call environment if you want to display the program status for a specific call.

See also

Setting the call environment of the block (Page 1842)

Setting the call environment of the block

By setting the call environment, you can specify when the program status of a block is recorded.

The section "Setting the call environment for breakpoints" describes how to set the call environment for breakpoints.

Requirement

- The block is open.

Specifying the call environment

To set the call environment, follow these steps:

1. Open the "Testing" task card.
2. Click the "Change" button in the "Call environment" pane.
   The "Call environment of the block" dialog opens.
3. Select the condition you want to apply.
   See also: Basics of the call environment (Page 1841)
4. Confirm your selection with "OK".

Result

The selected call environment is displayed in the "Testing" task card within the "Call environment" pane. The program status is now carried out in accordance with the set call environment.

Changing the call environment

Proceed as follows to change the call environment:

1. Open the "Testing" task card.
   If a call environment is already set, this is displayed in the "Call environment" pane.
2. Click the "Change" button in the "Call environment" pane.
   The "Call environment of the block" dialog opens.
3. Select the condition you want to apply.
   See also: Basics of the call environment (Page 1841)
4. Confirm your selection with "OK".

Result

The selected call environment is displayed in the "Testing" task card within the "Call environment" pane. The program status is now carried out in accordance with the set call environment.

See also

Basics of the call environment (Page 1841)

Introduction to testing with program status (Page 1840)

11.4.2.3 Switching test with program status on/off

You can monitor all blocks by switching on the program status of the block. This function is available to you for all code blocks, regardless of the programming language used. For blocks that were programmed with LAD, FBD or SCL you can also enable the program status from a specific position or for a specific selection. You can switch on the program status for an open block directly, or open a block from the calling block and view the program status.

Note

Note the following:

- The resources for testing with program status are limited. If there are not enough resources for the current test, earlier tests will be terminated.
- It is possible that another user can carry out a loading process on the selected CPU through joint parallel working on a CPU. It is therefore possible in the following cases that you can either not start the test with program status or that a running test is terminated:
  - Through the loading process the block for which you want to start or already have carried out the test with program status is loaded.
  - You use an instance data block as the call environment for the test with program status and the block changes structurally through the loading process, e.g. by renumbering.
  - You use a call path as the test condition for the test with program status and a block within the call path changes through the loading process.

  If a running test is terminated, a corresponding message is displayed in the Inspector window.

Requirement

- Code block: The code of the offline block is identical with the code of the online block. In this case the "Code" time stamps of the blocks are identical.
- Data block: The structure of the offline block is identical with the structure of the online block. In this case the "Interface" time stamps of the blocks are identical.

Switching the program status on or off directly in the block

To switch the program status for a block on or off directly in the block, follow these steps:

1. Open the block for which you wish to switch on the program status.
2. Click the "Monitoring on/off" button in the toolbar.
   If you have not already established an online connection, the "Go online" dialog opens. In this dialog, you can establish an online connection.
   See also: Establishing and terminating an online connection
3. Click the "Monitoring on/off" button in the toolbar again to switch the program status off.

Switching on or off program status starting at a specific point in a network

To start the program status for LAD and FBD at a specific point, follow these steps:

1. Open the block for which you wish to switch on the program status.
2. Click the "Monitoring on/off" button in the toolbar.
3. Right-click on the tag you want program status to start from.
4. Select "Modify > "Monitor from here" in the shortcut menu.
5. Click the "Monitoring on/off" button in the toolbar again to switch the program status off.

Switching on or off program status for selected tags

To start the program status for LAD and FBD for selected tags, follow these steps:

1. Open the block for which you wish to switch on the program status.
2. Click the "Monitoring on/off" button in the toolbar.
3. Select the tags for which you want to start the program status.
4. Select "Modify > Monitor selection" in the shortcut menu.
5. Click the "Monitoring on/off" button in the toolbar again to switch the program status off.

Switching on program status from the calling block

To switch on the program status for a block from the calling block (e.g. OB1), follow these steps:

1. Open the calling block.
2. Right-click on the block call.
3. Select the command "Open and monitor" in the shortcut menu.
   The block will open in the program editor. An online connection is established and the program status is displayed.

Result

If you enable the display of the program status, an online connection is established and the program status is displayed. When you turn off the display of the program status, you can terminate the online connection at the same time.

The call path of the block is shown under the block interface. If necessary, you can change the call environment in the "Call environment" section at the right-hand edge at "Test" and "Options". In the case of CPUs of the S7-1200/1500 series the "Call hierarchy" is displayed additionally on the right-hand edge. You can open the calling block by clicking the link.

11.4.2.4 Editing blocks during the program test

If you edit blocks while the test with program status is still running, online monitoring will be interrupted and you will be able to edit the block offline. If the block is not available offline in the project, you will first have to load it from the device to the project. After editing the block, you will also have to compile and download it again.

Procedure

To edit blocks while the test with program status is still running, follow these steps:

1. Edit the block as necessary.
   The test with program status is interrupted and the block is switched offline assuming it exists offline.
2. If the block does not exist offline, load it to the project from the device.
3. Compile the block.
   See also: Auto-Hotspot
4. Download the block to the device.
   See also: Auto-Hotspot

Result

The block now contains your modifications both online and offline. The online connection is re-established and testing with program status continues.

11.4.2.5 Modifying tags in the program status

While testing with the program status, you have the option of modifying tags to the following values once and immediately:

- Modify to 1
  Modifies tags of the "Bool" data type to the value "True".
- Modify to 0
  Modifies tags of the "Bool" data type to the value "False"
- Modify operand
  You can enter a modify value for tags that do not belong to the "Bool" data type.

Note that you cannot modify peripheral inputs, for example, via TagName:P.

Procedure

To modify tags during testing with the program status, proceed as follows:

1. Right-click on the tag you want to modify.
2. Select one of the following commands in the shortcut menu:
   - "Modify > Modify to 1"
   - "Modify > Modify to 0"
   - "Modify > Modify operand"
3. If you select "Modify operand", the "Modify operand" dialog opens. Enter the value you require in the "Modify value" box and confirm with "OK".

11.4.2.6 Switching display formats in the program status

Introduction

The display formats for tags are generally displayed in "integer" form. In the program status, you have the option of switching the current display format by means of the shortcut menu. The possible display formats for a tag are offered in a list. This is useful, for example, when you need a hexadecimal display in order to search for a hexadecimal error code.

Procedure

To switch the display format, follow these steps:

1. Open the desired block in the programming editor.
2. Switch on the program status by clicking "Monitoring on/off" in the toolbar.
   If you have not already established an online connection, the "Go online" dialog opens. In this dialog, you can establish an online connection.
3. Select the tags for which you want to start the program status.
4. Select "Modify > Monitor selection" in the shortcut menu to start monitoring this tag.
5. Select the desired tag at the corresponding block output and then select the desired display format in the shortcut menu, for example, "Modify > Display format > Hexadecimal".

Result

The display format for the selected tag is shown in hexadecimal form.

Note

Switching the display format in the program status

Please note that it is not possible to switch the display format for unconnected outputs, as no monitoring value is output in this case.

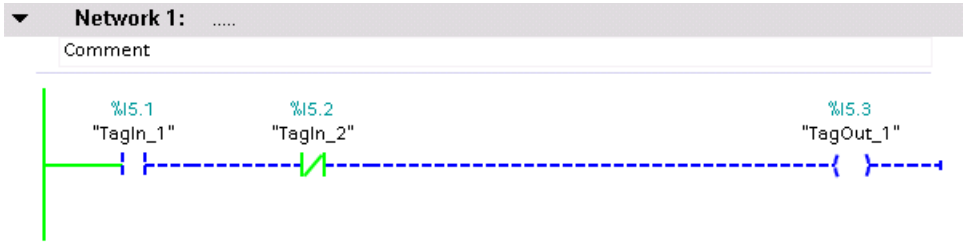
11.4.2.7 Examples of program status display

Program status display for LAD programs

Displays in program status

The display of the program status is updated cyclically.

The following figure shows an example of the program status display for LAD:



Representation of the program status

You can recognize the status of individual instructions and lines of a network quickly based on the color and type of lines and symbols. The following table shows the relationship between representation and status:

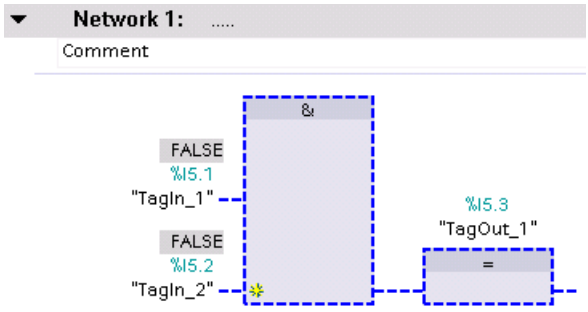
Representation	Status
Green solid	Satisfied
Blue dashed	Not satisfied
Gray solid	Unknown or not executed
Black	Not interconnected
Parameter in a frame with a saturation of 100 %	Value is current
Parameter in a frame with a saturation of 50 %	Value originates from an earlier cycle. The point in the program was not executed in the current cycle.

Program status display for FBD programs

Displays in program status

The display of the program status is updated cyclically.

The following figure shows an example of the program status display for FBD:



Representation of the program status

You can recognize the status of individual instructions and lines of a network quickly based on the color and type of lines and symbols. The following table shows the relationship between representation and status:

Representation	Status
Green solid	Satisfied
Blue dashed	Not satisfied
Gray solid	Unknown or not executed
Black	Not interconnected
Parameter in a frame with a saturation of 100 %	Value is current
Parameter in a frame with a saturation of 50 %	Value originates from an earlier cycle. The point in the program was not executed in the current cycle.

The values of the operands are displayed above the relevant operand name in a gray box.

Note

Program status display for outputs which are not interconnected

Please note that a monitor value cannot be displayed for outputs which are not interconnected.

Program status display for STL programs

Displays in program status

The display of the program status is updated cyclically and shown in tables. The tables are shown directly next to the STL program. You can read the program status for each line of the program. The display depends on the CPU in use (S7-300, S7-400, S7-1200 or S7-1500).

The tables to be displayed in the program status always contain the following information:

- RLO
  The "RLO" column shows the result of logic operation for each line of program. You can recognize the value of the RLO based on the background color of the table cell. Here, green means an RLO of 1 and lilac an RLO of 0.
- Value
  The current value of the operand is shown in the "Value" column.
- Extra
  The "Extra" column shows additional information depending on the particular operation, for example, relevant status bits for mathematical instructions, time or count values for timers and counters or memory addresses for indirect addressing.

The following figure shows an example of the program status display of a CPU S7-300 under STL:

Network 1: .....

Comment

			RLO	Value	Extra
1	OPN "DB1"	%DB1	0	DB1	
2	L %DBW2	%DBW2	0	16#4D2	DB1
3	L %DBW4	%DBW4	0	16#3333	DB1
4	+I		0	14341	OS=0,OV=0,A0=0,A1=1
5	T "Tag_15"	%MW4	0	14341	
6					
7	L 3.0	3.0	0	3.0	
8	T "Tag_24"	%MD10	0	3.0	
9	L 5.0	5.0	0	5.0	
10	T "Tag_52"	%MD20	0	5.0	
11	L MD ["Tag_29"]	%MD26	0	16#40400000	P#10.0
12	LAR1 P#M16.0	P#M16.0	0	P#M16.0	
13	L MD [AR1 , P#4.0]	P#4.0	0	16#40A00000	P#M20.0
14	+R		0	8.0	OS=0,OV=0,A0=0,A1=1
15	T "Tag_17"	%MD30	0	8.0	
16					
17	OPN DB ["Tag_48"]	%MW14	0	DB1	
18	L DBW ["Tag_38"]	%MD60	0	16#3333	P#4.0
19	L W#16#7777	W#16#7777	0	16#7777	
20	XOW		0	16#4444	
21	T "Tag_47"	%MW70	0	16#4444	
22					
23	A "Tag_1"	%I0.0	1	1	
24	L S5T#55S	S5T#55S	1	S5T#55S	
25	SE "Tag_49"	%T5	1	S5T#47S600MS	
26	A "Tag_49"	%T5	1	1	S5T#47S600MS
27	= "Tag_3"	%Q0.0	1	1	
28					
29	A "Tag_55"	%I0.2	1	1	
30	CU "Tag_23"	%C1	1	C#556	
31	A "Tag_23"	%C1	1	1	C#556
32	= "Tag_3"	%Q0.0	1	1	
33					
34	A "Tag_31"	%M10.0	0	0	
35	AN M ["Tag_51"]	%MD64	0	0	P#5.1
36	O DBX ["Tag_38"]	%MD60	1	1	P#4.0
37	LAR2 P#M10.0	P#M10.0	1	P#M10.0	
38	X [AR2 , P#3.4]	P#3.4	1	0	P#M13.4
39	= Q ["Tag_51"]	%MD64	1	1	P#5.1
40					

Program status display for SCL programs

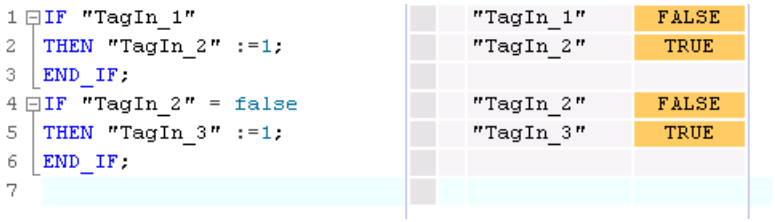
Displays in program status

The display of the program status is updated cyclically and shown in a table. The table is displayed immediately beside the SCL program and you can see the program status for each line of the program. The table contains the following information:

- Tag names
- Value

You can move the table to the left or right at any time.

The following figure shows an example of the program status display for SCL:



In the first column, you can see the name of the tag for which the current value is being displayed. If the line includes the "IF", "WHILE" or "REPEAT" instruction, the result of the instruction is displayed in the line as "True" or "False". If the line contains more than one tag, the value of the first tag is displayed. In both cases, all tags of these lines are displayed with their values in a separate list as soon as you select a line. If you place the cursor in a tag in the program code, this is shown in bold face in the list. You can also display the other tags of a line explicitly by clicking the arrow right located in front of lines containing more than one tag.

If the code of the line is not executed, the tag name is displayed in the values table in gray text.

The current values of the tags are displayed in the last column. If no values can be displayed for a tag, the line has a yellow background and three question marks are shown. In this case, select the "Create extended status information" check box in the properties of the block and download the block to the device again. All values are then displayed.

Program status display for GRAPH programs

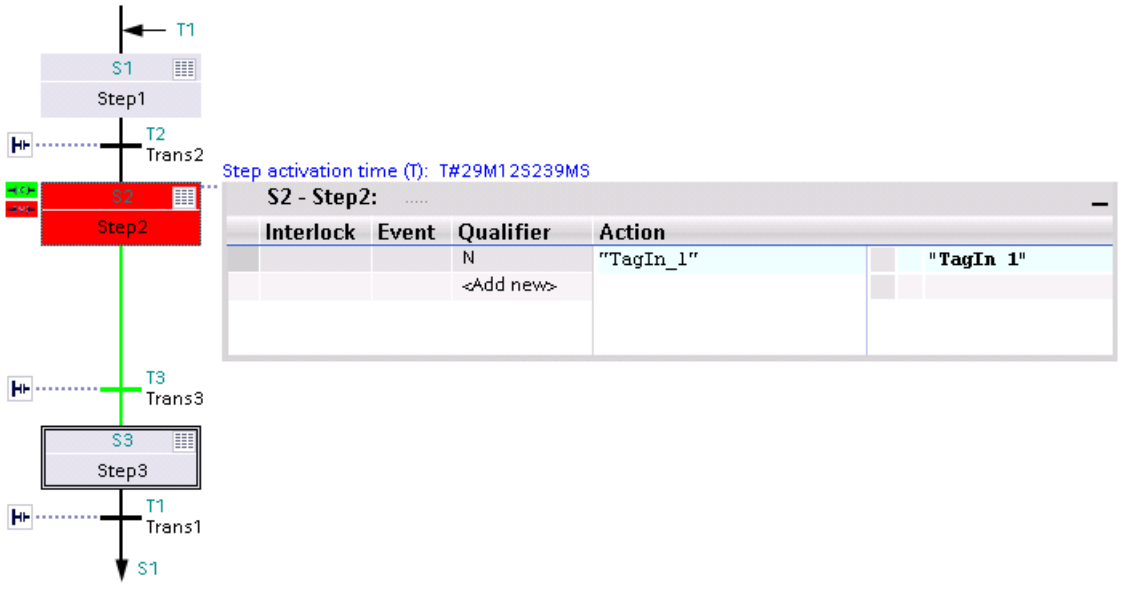
Displays in program status

For GRAPH programs you can have the program status displayed in sequence view and in single step view and for the permanent instructions. The program status display of permanent instructions corresponds to the program status display for LAB/FBD programs. The display of the program status is updated cyclically.

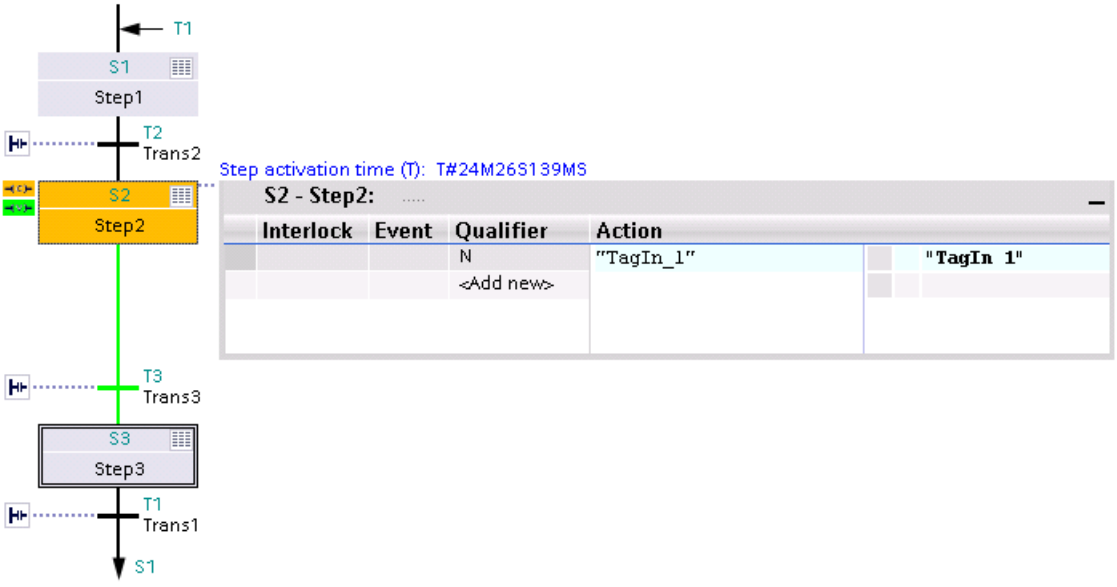
The following table shows the relationship between representation and status:

Representation	Area	Status
Green	Step, sequencer	There are no faults.
	Condition	The transition is fulfilled.
Red	Step, sequencer	There is a supervision error.
Yellow	Step, sequencer	There is an interlock error.
Black	Condition	The transition is not fulfilled.

The following figures show examples of the display for the program status in the sequence view:

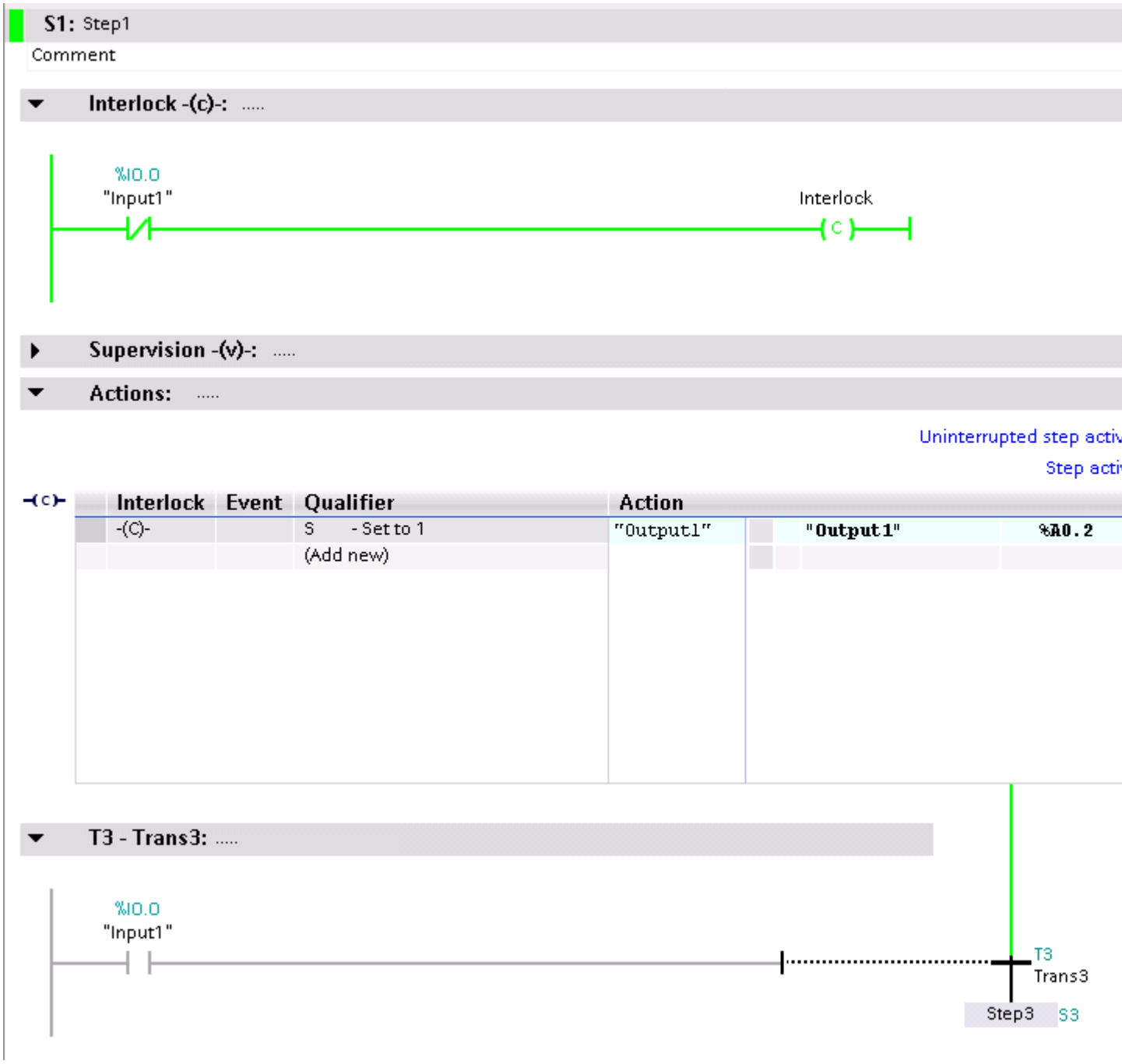


Step 2 contains a supervision error. The transition for switching to step 3 is fulfilled.



Step 2 contains an interlock error. The transition for switching to step 3 is fulfilled.

The figure below shows an example of the display for the program status in the single step view of a S7-300:



The step does not contain an interlock error. The subsequent transition is not satisfied.

11.4.3 Testing with the watch table

11.4.3.1 Introduction to testing with the watch table

Overview

The following functions are available in the watch table:

- **Monitoring tags**
  This allows the current values of the individual tags of a user program or a CPU to be displayed on the programming device or PC.
- **Monitoring tags**
  You can use this function to assign fixed values to the individual tags of a user program or CPU. Modifying is also possible with Test with program status .
- **"Enable peripheral outputs" and "Modify now"**
  These two functions enable you to assign fixed values to individual peripheral outputs of a CPU in the STOP mode. You can also use them to check your wiring.

Monitoring and modifying tags

The following tags can be monitored and modified:

- Inputs, outputs, and bit memories
- Contents of data blocks
- Content of UDTs
- I/O

Team engineering in the watch table

As of TIA Portal V13 SP1, within the framework of team engineering with an S7-1500 CPU with firmware version >= 1.7, several Engineering Systems can access the CPU online simultaneously, for example to monitor and modify tags and download blocks. If you use this function, make sure that you keep to the requirements and rules that apply to team engineering as explained in the information system in "Using Team Engineering" in the section "Shared commissioning of projects".

Possible applications

The advantage of the watch table is that a variety of test environments can be stored. This enables you to reproduce tests during commissioning or for service and maintenance purposes.

See also

Creating and editing watch tables (Page 1858)

Layout of the watch table (Page 1855)

Basic mode and expanded mode in the watch table (Page 1856)

Icons in the watch table (Page 1857)

11.4.3.2 Layout of the watch table

Introduction

A watch table contains the tags you defined for the entire CPU. A "Watch and force tables" folder is automatically generated for each CPU created in the project. You create a new watch table in this folder by selecting the "Add new Watch table" command.

Layout of the watch table

The columns displayed in the watch table depend on the mode you are working in: basic mode or expanded mode.

The following additional columns are shown in expanded mode:

- Monitor with trigger
- Modify with trigger

The names of the columns can also be changed dynamically based on the action.

Meaning of the columns

The following table shows the meaning of the individual columns in basic mode and expanded mode:

Mode	Column	Meaning
Basic mode		Identifier column
	Name	Name of the inserted tag
	Address	Address of the inserted tag
	Display format	Selected display format
	Monitor value	Values of the tags, depending on the selected display format.
	Modify value	Value with which the tag is modified.
		Select the tag to be modified by clicking the corresponding check box.
	Comment	Comment for documentation of the tags
The following additional columns are shown in expanded mode:	Monitor with trigger	Display of selected monitoring mode
	Modify with trigger	Display of selected modify mode

See also

Icons in the watch table (Page 1857)

11.4.3.3 Basic mode and expanded mode in the watch table

Difference between basic mode and expanded mode in the watch table

Depending on the mode specified, the watch table displays different columns and column headings that can be used to perform different actions.

You will find a detailed list of the columns in Layout of the watch table (Page 1855).

Switching between basic mode and expanded mode

You have the following options of toggling between the basic and expanded mode:

- Click the icon "Show/hide advanced setting columns". Click this icon again to return to the basic mode.
  Or:
- In the "Online" menu, select the "Expanded mode" check box. Deactivate this check box to return to the basic mode.

Functionality in expanded mode

The following functionality is only possible in expanded mode:

- Monitor with trigger
- Modify with trigger
- Enable peripheral outputs
- Monitor peripheral inputs
- Modify peripheral outputs

Notice

Danger of a time-out while monitoring peripheral inputs and controlling peripheral outputs

Note that the monitoring of peripheral inputs and the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

See also

Setting the monitoring and modify mode (Page 1869)

11.4.3.4 Icons in the watch table

Meaning of the icons

The following table shows the meaning of the icons in the watch table:

Icon	Meaning
	Identifies a table inside the project tree as a watch table.
	Shows information in the identifier column.
	Inserts a row before the selected row.
	Inserts a row after the selected row.
	Modifies the addresses of all selected tags immediately and once. This command is executed once and as quickly as possible without reference to a defined trigger point in the user program.
	Modifies the addresses of all selected tags with reference to a defined trigger point in the user program.
	Disables the command output disable of the peripheral outputs. You can then modify the peripheral outputs when the CPU is in STOP mode.
	Displays all columns of expanded mode. If you click this icon again, the columns of expanded mode will be hidden.
	Displays all modify columns. If you click this icon again, the modify columns will be hidden.
	Starts monitoring of the visible tags in the active watch table. The default setting for the monitoring mode in basic mode is "permanent". In expanded mode, you can set defined trigger points for the monitoring of tags.
	Starts monitoring of the visible tags in the active watch table. This command is executed immediately and the tags are monitored once.
	Displays the check box for the selection of tags to be modified.
	Indicates that the value of the selected tag has been modified to "1".
	Indicates that the value of the selected tag has been modified to "0".
	Indicates that the address is being used multiple times.
	Indicates that the substitute value is being used. Substitute values are values that are output to the process in case of signal output module faults or are used instead of a process value in the user program in case of signal input module faults. The substitute values can be assigned by the user (e.g., retain old value).
	Indicates that the address is blocked because it is already being modified.
	Indicates that the address cannot be modified.
	Indicates that the address cannot be monitored.
	Indicates that an address is being forced.
	Indicates that an address is being partly forced.

Icon	Meaning
	Indicates that an associated I/O address is being fully or partly forced.
	Indicates that an address cannot be fully forced. Example: It is indeed possible to force the address QW0:P, but it is not possible to force the address QD0:P since this address area is eventually not available on the CPU.
	Indicates that a syntax error occurred.
	Indicates that the address is selected but at the moment e.g. has not yet been modified.

See also

Layout of the watch table (Page 1855)

11.4.3.5 Creating and editing watch tables

Creating a watch table

Introduction

The watch table allows you to monitor and modify tags in the user program. Once you have created a watch table, you can save it, duplicate it, and print it and use it again and again to monitor and modify tags.

Requirement

A project is open.

Procedure

To create a watch table, follow these steps:

1. Click "Project view" in the status bar.
   The project view is displayed.
2. In the project tree, double-click the CPU for which you want to create a watch table.
3. Double-click the "Watch and force tables" folder and then the "Add new watch table" command.
   A new watch table is added.
4. In the "Name" column or in the "Address" column, enter the name or the absolute address for the tags that you want to monitor or modify.
5. You can select a display format from the drop-down list in the "Display format" column if you want to change this default setting.
6. Now decide whether you want to monitor or modify the entered tags and, if applicable, enter the desired values for modifying.

Opening a watch table

Requirement

A watch table has been created.

Procedure

To open a watch table, follow these steps:

1. Open the "Watch and force tables" folder below the desired CPU.
2. Double-click on the required watch table in the folder.

Result

The selected watch table opens.

Copying and pasting a watch table

Requirement

A watch table has been created.

Procedure

To copy a watch table, follow these steps:

1. Right-click the watch table that you want to copy.
2. In the context menu, select "Copy".
3. In the project tree, open the folder structure for the CPU in which you want to paste the copied watch table.
4. Right-click on the "Watch and force tables" folder.
5. In the context menu, select "Paste".
6. Alternatively, you can select the entire contents of the watch table and Drag & Drop it onto another watch table.

Result

A copy of the selected watch table is placed in the "Watch and force tables" folder of the relevant CPU.

Saving a watch table

Prerequisite

A watch table has been created.

Procedure

To save a watch table, follow these steps:

1. In the project tree select the watch table you want to save.
2. If you wish to change the preset name of the table, select the "Rename" command in the context menu and enter a new name for the table.
3. In the "Project" menu, select "Save". Note that this save operation will save the entire project.

Result

The contents of the watch table and the project are saved.

Note

You can reuse saved watch tables to monitor and modify tags when retesting your program.

11.4.3.6 Entering tags in the watch table

Basic information on entering tags in the watch table

Recommended procedure

- Select the tags whose values you want to monitor or modify, and enter them in the watch table.
- When entering tags into the watch table, please note that these tags must be previously defined in the PLC tag table.
- When entering tags, work from the outside to the inside. This means that you start by entering the tags for the inputs in the watch table. Then, you enter the tags that are affected by the inputs or that affect the outputs. Finally, you enter the tags for the outputs.

Example of filling out a watch table

- Enter the absolute address to be monitored or modified in the "Address" column.
- Enter the symbolic name for the tag in the "Name" column.

- Select the display format you require from the drop-down list in the "Display format" column, if you do not want to use the default setting.
- Now decide whether you want to monitor or modify the entered tags. Enter the desired values for modifying as well as a comment in the corresponding columns of the watch table.

Create comment row

If required, you can create a comment row by entering the string "//" in the "Name" column.

Syntax check

When you enter the tags in the watch table, the syntax of each cell is checked when you exit the cell. Incorrect entries are marked in red.

Note

When you place the mouse pointer in a cell marked in red, brief information is displayed with additional notes on the error.

See also

Permitted operands for the watch table (Page 1861)

Permissible modify values for the watch table (Page 1862)

Permitted operands for the watch table

Permissible operands for the watch table

The following table shows the operands that are permitted for the watch table:

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	BOOL	I1.0, Q1.7, M10.1 I0.0:P; Q0.0:P
Input/output/bit memory	BYTE	IB1/QB10/MB100 IB1:P; QB1:P
Input/output/bit memory	WORD	IW1; QW10; MW100 IW2:P; QW3:P
Input/output/bit memory	DWORD	ID4; QD10; MD100 ID2:P; QD1:P
Timers	TIMER	T1
Counters	COUNTER	C1
Data block	BOOL	DB1.DBX1.0
Data block	BYTE	DB1.DBB1

Permitted operand	Example of data type	Example (International mnemonics)
Data block	WORD	DB1.DBW1
Data block	DWORD	DB1.DBD1

Note

Please observe the following notes to work with the watch table.

- You cannot enter "DB0..." because it is used by the system!
- Peripheral outputs can be modified but not monitored.
- Peripheral inputs can be monitored but not modified.

Notice

Danger of a time-out while monitoring peripheral inputs and controlling peripheral outputs

Note that the monitoring of peripheral inputs and the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

See also

Basic information on entering tags in the watch table (Page 1860)

Overview of the valid data types (Page 1908)

Permissible modify values for the watch table

Entry of modify values in the watch table

The following table shows the operands that are permitted for the entry of modify values in the watch table:

Table 11-1 Bit operands

Possible bit operands	Example for permitted modify values
I1.0	True
M1.7	False
Q1.0	0
Q1.1:P	1
DB1.DBX1.1	2#0
M1.6	2#1

Table 11-2 Byte operands

Possible byte operands	Example for permitted modify values
IB1	2#00110011
MB12	B#16#1F
QB10	1F
QB11:P	'a'
DB1.DBB1	10

Table 11-3 Word operands

Possible word operands	Example for permitted modify values
IW1	2#0011001100110011
MW12	W#16#ABCD
MW14	ABCD
QW10	B#(12, 34)
QW12:P	12345
DB1.DBW1	'ab'
MW16	S5T#9s_340ms
MW18	C#123
MW9	D#2006-12-31

Table 11-4 Double word operands

Possible double word operands	Example for permitted modify values
ID1	2#00110011001100110011001100110011
QD10	Dw#16#abcdef10
QD12:P	ABCDEF10
DB1.DBD2	b#(12,34,56,78)
MD8	L#-12
MD12	L#12
MD16	123456789
MD20	123456789
MD24	T#12s345ms
MD28	Tod#1:2:34.567
MD32	P#e0.0

Table 11-5 Timers

Possible operands of the "Timer" type	Permitted control values	Explanation
T1	0 ms	Time value in milliseconds (ms)
T12	20 ms	Time value in milliseconds (ms)

Possible operands of the "Timer" type	Permitted control values	Explanation
T14	12345 ms	Time value in milliseconds (ms)
T16	S5t#12s340ms	Time value 12s 340 ms

Table 11-6 Counters

Possible operands of the "Counter" type	Permitted control values
C1	0
C14	20
C16	C#123

Notes on timers and counters

- Timers

  Note

  Modifying a timer influences only the value, not the status. Timer T1 can be modified to the value "0", but the result of logic operation for A T1 is not changed.

  The time sequences "s5t" and "s5time" can be written in both lower-case and upper-case characters.

- Counter

  Note

  Modifying a counter influences only the value, not the status. Counter C1 can be changed to the value "0", but the result of the logic operation for A C1 is not changed.

Overview of the display formats

Display formats in the watch table

The display format you select specifies the representation of a tag value.

When entering the address a display format is automatically preset. If you want to change this, you can select a display format from the drop-down list in the "Display formats" column. The drop-down list only offers the display formats which are valid for this data type. The display format that appears first in the list is the pre-selected format.

Example

The following table shows the 32-bit data types permitted for all CPU families in the watch table and their possible display formats:

Data type	Possible display formats
BOOL	Bool, Hex, BCD, Octal, Bin, Dec, Dec+/-
BYTE	Hex, BCD, Octal, Bin, Dec, Dec+/-, Character
WORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Unicode_Character, SIMATIC_Timer, Date, Counter
DWORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Unicode_Character, Floating-point number, Time of day, Timer, Pointer
SINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
INT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter, Date
DINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
USINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
UINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter, Date
UDINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
REAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character,Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Pointer
DATE	Date, Dec, Hex, BCD, Bin
TIME_OF_DAY	Time of day, Dec, Hex, BCD, Bin
TIME	Timer, Hex, BCD, Bin
DATE_AND_TIME	Date and time,
TIMER	SIMATIC_Timer, Hex, BCD, Bin
CHAR	Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
WCHAR	Unicode_Character, Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
STRING	Character string
WSTRING	Unicode_character string
POINTER	Pointer, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Block number
COUNTER	Counter, Hex, BCD, Bin
S5TIME	SIMATIC_Timer, Hex, BCD, Bin

For the S7-1200 CPU family, all 32-bit data types are permitted (see table above), as well as the 64-bit data type LREAL with the following possible display formats:

Data type	Possible display formats
LREAL	In a project created with TIA Portal < V12: Floating-point number Note: The display of LREAL is limited to 13 digits plus exponent.
LREAL	In a project created with TIA Portal >= V12: Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Date and time Note: The display of LREAL is limited to 13 digits plus exponent.

For the S7-1500 CPU family, in addition to 32-bit data types, the 64-bit data types listed in the table are also permitted with the following possible display formats:

Data type	Possible display formats
LWORD	Hex, Octal, BCD, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Date and time
LINT	Dec+/-, Dec, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer, Date and time
ULINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer, Date and time
LREAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Date and time
LTIME	Timer, Dec+/-, Dec, Hex
LTOD	Time of day, Dec, Hex, BCD, Bin
LDT	Date and time, Dec, Hex

For more information, refer to the description of the valid data types (Page 1908).

Note

Rounding of floating-point numbers

In the watch table, floating-point numbers are stored as binary numbers in IEEE format. Because not every floating point number (real, longreal) that can be displayed on the user interface can be mapped to the IEEE format, there is a possibility that floating-point numbers will be rounded. If a rounded floating-point number in the watch table is copied and, in turn, inserted in another input field, the rounding may cause a slight difference.

Note

Only symbolic addressing is possible

In the watch table, LongDataTypes such as LWORD or LREAL can only be addressed symbolically.

Selecting the display format for tags

Procedure

To select the display format of the tags, follow these steps:

1. Enter the desired address in the watch table.
2. Click the desired cell in the "Display format" column, and open the drop-down list. The permissible display formats are shown in the drop-down list.
3. Select the desired display format from the drop-down list.

Note

If the selected display format cannot be applied, then the last selected display format will be displayed automatically.

See also

Overview of the valid data types (Page 1908)

Creating and editing comment lines

Basic principles of comment lines

In addition to the row related comments in the comment column, you can now also create complete comment lines to enhance the structure of the watch table.

The contents of the comment line are stored in the "Languages & Resources" folder in the "Project texts" tab and can be compiled in other project languages.

Creating comment lines

To create comment lines, follow these steps:

1. Open the watch table and enter the required addresses.
2. To create a comment line, enter the character string "//" in the "Name" column. No comment lines can be created in the other columns.

3. Enter the required comment in the comment line. The entered comment is shown in green.
4. To show all comments you entered, double-click "Project texts" in the project tree under "Languages & Resources".
5. If you are working in multi-lingual projects and want your comment to be translated into other languages, you can set the project languages required in addition to the editing language in the project tree under "Languages & Resources > Project languages".

Deleting comment lines

To delete comment lines, follow these steps:

1. Open a watch table containing comment lines.
2. Delete the entire comment including the introductory string "//", if you no longer required this.
3. Alternatively, delete only the introductory string "//". In this case the existing comment is retained and is displayed in the "Comment" column in the watch table.

Note

Deleting comment lines

When you delete comment lines the project languages and any existing translations for these comments are also deleted.

11.4.3.7 Monitoring tags in the watch table

Introduction to monitoring tags in the watch table

Introduction

The watch table allows you to monitor the tags of the configured input and output modules in the CPU, depending on the monitoring and modify mode (Page 1869) selected. To monitor tags, an online connection to the CPU must exist.

Notice

Danger of a time-out while monitoring peripheral inputs

Note that the monitoring of peripheral inputs can result in a time-out.

The CPU assumes the "STOP" mode.

Options for monitoring tags

The following options are available for monitoring tags:

- Monitor now
  This command starts the monitoring of the visible tags in the active watch table immediately and once only.
- Monitor all
  This command starts the monitoring of all visible tags in the active watch table, depending on the selected watch mode:
  - In basic mode, the monitoring mode is set to "permanent" by default.
  - In expanded mode, you can specify defined trigger points for the monitoring of tags.

  Note

  If the monitoring mode is changed while in expanded mode and then a switch is made to basic mode, the monitoring mode set before will also be applied in basic mode.

CPU-specific limitations when monitoring tags

The following CPU-specific differences exist:

- CPU S7-300/400:
  CPUs from this family can only monitor the first 30 characters of a string.
- CPU S7-1200/1500:
  CPUs from this family can monitor a string up to the total size of 254 characters.

Setting the monitoring and modify mode

Introduction

By selecting the monitoring and modify mode, you specify the trigger point and the duration of the tag monitoring in the watch table and the force table.

Possible monitoring and modify modes (duration of monitoring or modifying)

The following monitoring and modifying modes are available:

Trigger	Execution	CPU status	Duration
Permanent	Permanent During monitoring: The inputs are monitored at the end of the cycle and the outputs at the start of the cycle. During modify: The inputs are modified at the start of the cycle and the outputs at the end of the cycle.	RUN	Will be executed until the user stops the action or the online connection to the CPU is interrupted.
Permanently, at start of scan cycle	Permanently, at start of scan cycle	RUN	Will be executed until the user stops the action or the online connection to the CPU is interrupted.
Permanently, at end of scan cycle	Permanently, at end of scan cycle	RUN	Will be executed until the user stops the action or the online connection to the CPU is interrupted.
Permanently, at transition to STOP	Permanently, at transition from RUN to STOP	RUN > STOP	Will be executed until the user stops the action or the online connection to the CPU is interrupted.
Once only, at start of scan cycle	Once only, at start of scan cycle	RUN	Ends automatically after being executed once.
Once only, at end of scan cycle	Once only, at end of scan cycle	RUN	Ends automatically after being executed once.
Once only, at transition to STOP	Once only, at transition from RUN to STOP	STOP > RUN	Ends automatically after being executed once.

Special features when using "Permanent" mode

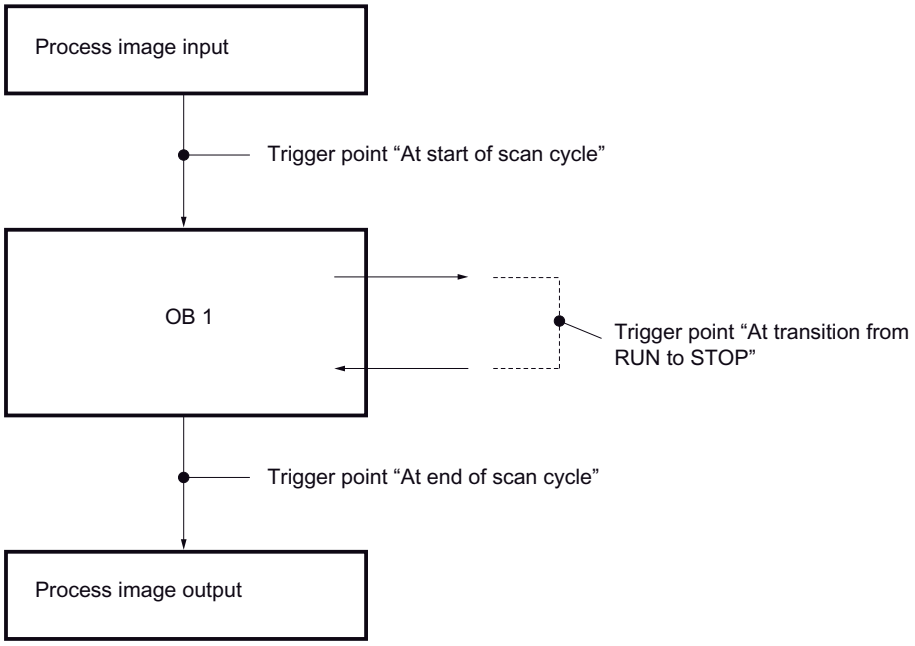
The "Permanent" mode is executed differently for the monitoring and modifying of tags.

- Monitoring: The inputs are monitored at the end of the cycle and the outputs at the start of the cycle.
- Modifying: The inputs are modified at the start of the cycle and the outputs at the end of the cycle.

Selecting the trigger point

The trigger points "Beginning of scan cycle", "End of scan cycle", and "Switch to stop" specify the time at which the tags are to be read from the CPU or updated in the CPU.

The following diagram shows the position of these trigger points:



Position of the trigger points

From the position of the trigger points, it follows that:

- Modifying of inputs is only appropriate at the beginning of the scan cycle (corresponding to the beginning of the user program OB 1), because otherwise the process image input is updated again after modifying and is thus overwritten.
- Modifying of outputs is only appropriate at the end of the scan cycle (corresponding to the end of the user program OB 1), because otherwise the process image output can be overwritten by the user program.
- The modified value is displayed in the "Monitor value" column, provided that monitoring is active and the modified value is not overwritten by the user program.

Monitoring tags

When tags are being modified, the following applies to the trigger points:

- If you have specified the modify mode as "once only", you will receive an alarm if the selected tags cannot be modified.
- In "permanent" modify mode, you do not receive an alarm.

Note regarding the "Modify now" command

You can modify the values of selected tags immediately using the "Online > Modify >Modify now" command. This command is executed once only and as quickly as possible without reference to a defined position (trigger point) in the user program. This function is used mainly for modifying when the CPU is in STOP mode.

"Monitor all" command for tags

Introduction

The "Monitor all" command allows you to start monitoring the visible tags in the active watch table. The default setting for the monitoring mode in basic mode of the watch table is "permanent". In expanded mode, you can specify defined trigger points for the monitoring of tags. In this case, the tags are monitored with reference to the specified trigger points.

Notice

Danger of a time-out while monitoring peripheral inputs

Note that the monitoring of peripheral inputs can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To execute the "Monitor all" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the watch table.
2. Switch to expanded mode by clicking the icon "Show/hide advanced setting columns" in the toolbar.
3. If you want to change the default monitoring mode for a tag, click the appropriate cell in the "Monitor with trigger" column and select the desired monitoring mode from the drop-down list.
4. Click the "Monitor all" icon in the toolbar.

Result

The tags of the active watch table are monitored using the monitoring mode selected.

See also

Icons in the watch table (Page 1857)

Entering tags in the watch table (Page 1860)

Basic mode and expanded mode in the watch table (Page 1856)

"Monitor now" command for tags

Introduction

The "Monitor now" command starts the monitoring of tags immediately without reference to defined trigger points. The tag values are read out once only and displayed in the watch table.

Notice

Danger of a time-out while monitoring peripheral inputs

Note that the monitoring of peripheral inputs can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To execute the "Monitor now" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the watch table.
2. Click the "Monitor now" icon in the toolbar.

Result

The tags of the active watch table are monitored immediately and once only.

See also

Icons in the watch table (Page 1857)

Entering tags in the watch table (Page 1860)

Basic mode and expanded mode in the watch table (Page 1856)

11.4.3.8 Modifying tags in the watch table

Introduction to modifying tags

Introduction

The watch table allows you to modify the tags of the configured input and output modules in the CPU, depending on the monitoring and modify mode (Page 1869) selected.

To monitor the tags, an online connection to the CPU must exist.

Danger

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

Notice

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

Notice

Danger due to modifying an identical operand in parallel with different modify values in more than one watch table

When working with more than one watch table, avoid modifying identical operands permanently multiple times with different modify values.

If an identical operand is modified permanently with different modify values at the same time in different watch tables, all watch tables will display the last modified value, because the modify value assigned last will be used in this case.

Options for modifying tags

The following options are available for modifying tags:

- Modify to "0"
  This command modifies the selected address to the modify value "0".
- Modify to "1"
  This command modifies the selected address to the modify value "1".
- Modify once only and immediately
  This command modifies all selected addresses in the active watch table "once only and immediately".

- Modify with trigger
  This command modifies all selected addresses in the active watch table using the monitoring and modify mode (Page 1869) selected.
  The "Modify with trigger" function is only available in expanded mode. You will not receive an alarm indicating whether or not the selected addresses were actually modified with the specified value. You should use the "Modify once only and immediately" function if you require such a confirmation.
- Enable peripheral outputs
  This command disables the command output disable.
  This function can only be executed in expanded mode, when the CPU is in STOP and the option Force (Page 1898) of tags is not enabled. If desired, deactivate this function in the force table.

Note

When modifying, note the following:

Modifying of tags **cannot** be undone.

Modify tags to "0"

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

Danger

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

Notice

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags to "0", follow these steps:

1. Enter the desired address in the watch table.
2. Select the "Online > Modify > Modify to 0" command in order to modify the selected address with the specified value.

Result

The selected address is modified to "0".

Note

When modifying, note the following:

Modifying can **not** be undone!

Modify tags to "1"

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

Danger

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

Notice

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags to "1", follow these steps:

1. Enter the desired address in the watch table.
2. Select the "Online > Modify > Modify to 1" command in order to modify the selected address with the specified value.

Result

The selected address is modified to "1".

Note

When modifying, note the following:

Modifying can **not** be undone!

"Modify now" command for tags

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them immediately. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

Danger

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

Notice

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags immediately, follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Select the "Online > Modify > Modify once and now" command in order to immediately modify the selected address once only with the specified value.

Result

The selected addresses are modified immediately and once only.

Note

When modifying, note the following:

Modifying can **not** be undone!

"Modify with trigger" command for tags

Introduction

You can assign values to addresses dependent on the defined monitoring and modify mode and modify them. The modify command is executed as specified in the monitoring and modify mode, with reference to the defined trigger position in the user program.

Danger

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

Notice

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.
- The watch table has to be in expanded mode.

Procedure

To modify tags "with trigger", follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Switch to expanded mode using the icon "Show/hide advanced settings columns" in the toolbar or the "Online > Expanded mode" command.
   The "Monitor with trigger" and "Modify with trigger" columns are displayed.
4. In the "Modify with trigger" column, select the desired modify mode from the drop-down list box. The following options are available:
   - Permanent
   - Permanently, at start of scan cycle
   - Once only, at start of scan cycle
   - Permanently, at end of scan cycle
   - Once only, at end of scan cycle
   - Permanently, at transition to STOP
   - Once only, at transition to STOP
5. Start modifying using the "Online > Modify > Modify with trigger" command.
6. Confirm the prompt with "Yes" if you want to start modifying with trigger.

Result

The selected tags are modified using the selected monitoring and modify mode. The yellow triangle is no longer displayed.

Note

When modifying, note the following:

Modifying can **not** be undone!

Enable peripheral outputs

Introduction

The "Enable peripheral outputs" function deactivates the command output disable of the peripheral outputs. You can then modify the peripheral outputs when the CPU is in STOP mode. This function is available in the watch table in "Expanded mode" only.

Danger

Danger when enabling the peripheral outputs:

Attention, the enabling of the peripheral outputs can cause serious personal injury and material damage!
Make certain that dangerous conditions cannot occur before you execute the "Enable peripheral outputs" function.

Prerequisites

- A watch table has been created.
- An online connection to the CPU exists.
- The CPU is in STOP mode before you can enable the peripheral outputs.
- The watch table has to be in expanded mode.
- The option Force (Page 1898) of tags must not be enabled.

Note

"Enable peripheral outputs" function

- This function is possible only in STOP mode. The function is exited by an operating state change of the CPU and by the termination of the online connection.
- While the function is enabled, forcing is not possible.

Procedure

To enable the peripheral outputs in STOP mode, follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Switch to expanded mode using the icon "Show/hide advanced settings columns" in the toolbar or the "Online > Expanded mode" command.
   The "Monitor with trigger" and "Modify with trigger" columns are displayed.
4. Change the relevant CPU to STOP using the operator panel.

5. Right-click to open the shortcut menu and select "Enable peripheral outputs".
6. Confirm the prompt with "Yes" if you want to unlock the command output disable for the peripheral outputs.
7. Modify the peripheral outputs using the "Online > Modify > Modify now" command.

Result

The peripheral outputs are modified with the selected modify values. The yellow triangle is no longer displayed.

Enabling the peripheral outputs

The "Enable peripheral outputs" function remains active until:

- The "Enable peripheral outputs" command is deactivated again via the shortcut menu or via the "Online > Modify > Enable peripheral outputs" command.
- The CPU is no longer in STOP mode.
- The online connection is terminated.

Note

When modifying, note the following:

Modifying can **not** be undone!

11.4.4 Testing with the force table

11.4.4.1 Introduction for testing with the force table

Overview

You can use the force table to assign permanent values to individual tags

of the user program. This action is referred to as "forcing".

The following functions are available in the force table:

- **Monitoring tags**
  This allows the current values of the individual tags of a user program or a CPU to be displayed on the programming device or PC. Tags can be monitored with or without a trigger condition.
- **Forcing tags**
  This function lets you assign a fixed value to individual I/O tags of the user program.

Monitoring and forcing tags

The monitoring and forcing of tags is always dependent on the operand scope of the CPU used.

The following tags can be monitored:

- Inputs, outputs, and bit memories
- Contents of data blocks
- Peripheral inputs

The following tags can be forced:

- Peripheral inputs
- Peripheral outputs

Example

Independent of the CPU used, only I/O can be forced, such as: "Tag_1":P or "QW0:P" or "IW0:P". Note that "Tag_1":P must not be the symbolic name of a bit memory.

Possible applications

One advantage of the force table is that you can simulate different test environments and overwrite tags in the CPU with a permanent value. This enables you to intervene in the ongoing process for regulating purposes.

See also

Layout of the force table (Page 1883)

Basic mode and expanded mode in the force table (Page 1884)

Icons in the force table (Page 1885)

Open and edit force table (Page 1886)

11.4.4.2 Safety precautions when forcing tags

Safety precautions when forcing tags

Because the forcing function allows you to intervene permanently in the process, observance of the following notices is essential:

Danger

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Caution

Prevent personal injury and material damage!

- Before you start the "Force" function, you should ensure that no one else is currently executing this function on the same CPU.
- Forcing can only be stopped by clicking the "Stop forcing" icon or using the "Online > Force > Stop forcing" command. Closing the active force table does **not** stop the forcing!
- Forcing can **not** be undone!
- Review the differences between " modifying tags" (Page 1873) and "forcing tags" (Page 1898).
- If a CPU does not support the "Force" function, the relevant icons cannot be selected.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is **not** possible on this CPU. If desired, deactivate this function in the watch table.

11.4.4.3 Layout of the force table

Introduction

In the force table, enter the CPU-wide tags that you have defined and selected and which are to be forced in the allocated CPU. Only peripheral inputs and peripheral outputs can be forced.

For each CPU created in the project, a force table will automatically be created in the "Watch and force tables" folder. Only one force table can be allocated to a CPU. This force table displays all the addresses forced in the allocated CPU.

Layout of the force table

The columns displayed in the force table depend on the mode you are working in: basic mode or expanded mode.

In expanded mode the "Monitor with trigger" column is also displayed.

Meaning of the columns

The following table shows the meaning of the individual columns in basic mode and expanded mode:

Mode	Column	Meaning
Basic mode	i	Identification column
	Name	Name of the inserted tag
	Address	Address of the inserted tag
	Display format	Selected display format
	Monitor value	Values of the tags, dependent on the selected display format.
	Force value	Value with which the tag is forced.
	F ("Force")	Select the tag to be forced by activating the corresponding check box.
	Comment	Comment for documentation of the tags
The following additional column is shown in expanded mode:	Monitor with trigger	Display of selected monitoring mode

See also

Icons in the force table (Page 1885)

Basic mode and expanded mode in the force table (Page 1884)

11.4.4.4 Basic mode and expanded mode in the force table

Difference between basic mode and expanded mode in the force table

In expanded mode the "Monitor with trigger" column is also displayed in the force table.

You will find a detailed list of the columns under Layout of the force table (Page 1883).

Switching between basic mode and expanded mode

You have the following options of toggling between the basic and expanded mode:

- Click the icon "Show/hide advanced setting columns". Click this icon again to return to the basic mode.
  Or:
- In the "Online" menu, select the "Expanded mode" check box. Deactivate this check box to return to the basic mode.

Functionality in expanded mode

The following functionality is only possible in expanded mode:

- Monitor with trigger
- Monitor peripheral inputs

11.4.4.5 Icons in the force table

Meaning of the icons

The following table shows the meaning of the icons in the force table:

Icon	Meaning
	Identifies a table inside the project tree as a force table.
	Identification column
	Inserts a row before the selected row.
	Inserts a row after the selected row.
	Displays all columns of expanded mode. If you click this icon again, the columns of the expanded mode will be hidden.
	Updates all operands and values currently being forced on the CPU in the open force table.
	Starts forcing for all addresses of the selected tags. If forcing is already running, the previous action is replaced without interruption.
	Stops forcing of addresses in the force table.
	Starts monitoring of the visible tags in the force table. The default setting for monitoring in basic mode is "permanent". In expanded mode an additional column is shown and you can set certain trigger points for monitoring tags.
	Starts monitoring of the visible tags in the force table. This command is executed immediately and the tags are monitored once.
	Displays the check box for the selection of tags to be forced.
	Indicates that an address cannot be fully forced. Example: It is possible to force the address QW0:P, but it is not possible to force the address QD0:P because this address area is potentially not available on the CPU.
	Indicates that an address cannot be monitored.
	Indicates that an address is being forced.
	Indicates that an address is being partly forced.
	Indicates that the associated peripheral address is being forced.
	Indicates that a syntax error occurred.
	Indicates that the address is selected but has not been forced yet.

Icon	Meaning
	Indicates that the selected tag was monitored for the value "1".
	Indicates that the selected tag was monitored for the value "0".
	Indicates that the address is being used more than once.
	Indicates that a substitute value is being used. Substitute values are values that are output to the process in case of signal output module faults or are used instead of a process value in the user program in case of signal input module faults. The substitute values can be assigned by the user (e.g., retain old value).

See also

Layout of the force table (Page 1883)

11.4.4.6 Open and edit force table

Display force table

Introduction

You cannot create a new force table; one force table already exists for each CPU. It is permanently allocated to this CPU and cannot be copied or duplicated.

Requirements

A project with an allocated CPU has to be open.

Displaying a force table

The force table is always displayed below a CPU in the "Watch and force tables" folder.

Open force table

Requirements

A project with an allocated CPU must be created.

Procedure

Proceed as follows to open a force table:

1. Open the "Watch and force tables" folder below the desired CPU.
2. Double-click the "Force table" in this folder.

Result

The selected force table opens.

Save force table

Requirements

A project with an allocated CPU has been created.

Procedure

Proceed as follows to save a force table:

1. Enter the desired changes in the force table.
2. Select the "Save" command in the "Project" menu or click the "Save project" icon in the toolbar. Note that this save operation will save the entire project.

Result

The contents of the force table and the associated project are saved.

Note

You cannot rename a force table.

11.4.4.7 Entering tags in the force table

Basic principles for entering tags in the force table

Recommended procedure

Select the tags whose values you want to monitor or force, and enter them in the force table.

When entering tags in the force table, please note that these tags must be previously defined in the PLC tag table.

Example of filling out a force table

- You can enter the absolute address that is to be forced or monitored in the "Address" column or you can enter the symbolic name of the tag in the "Name" column.
- Select the display format you require from the drop-down list in the "Display format" column, if you do not want to use the default setting.

- Now you have to decide whether you want to monitor or force the entered tags. Enter the required force value and a comment in the appropriate columns of the force table.
- Note that only peripheral inputs and peripheral outputs can be forced and review the Safety precautions when forcing tags (Page 1899).

Create comment line

If required, you can create a comment row by entering the string "//" in the "Name" column.

Syntax check

When you enter tags in the force table, the syntax of each cell will be checked when you exit the cell. Incorrect entries are marked in red.

Note

When you place the mouse pointer in a cell marked in red, brief information is displayed with additional notes on the error.

Permitted operands for the force table

Permitted operands for the force table

The following table shows the operands that are permitted for forcing in the force table:

Permitted operand	Example of data type	Example (International mnemonics)
Peripheral input/peripheral output	BOOL	I0.0:P; Q0.0:P
Peripheral input/peripheral output	BYTE	IB1:P; QB1:P
Peripheral input/peripheral output	WORD	IW2:P; QW3:P
Peripheral input/peripheral output	DWORD	ID2:P; QD1:P

The following table shows the operands that are permitted for monitoring in the force table:

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	BOOL	I1.0, Q1.7, M10.1 I0.0:P
Input/output/bit memory	BYTE	IB1/QB10/MB100 IB1:P
Input/output/bit memory	WORD	IW1; QW10; MW100 IW2:P

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	DWORD	ID4; QD10; MD100 ID2:P
Timers	TIMER	T1
Counters	COUNTER	C1
Data block	BOOL	DB1.DBX1.0
Data block	BYTE	DB1.DBB1
Data block	WORD	DB1.DBW1
Data block	DWORD	DB1.DBD1

Note

You cannot enter "DB0..." because it is used by the system!

Permitted force values for the force table

Entering force values in the force table

The following table shows the operands that are permitted for entering force values in the force table:

Table 11-7 Bit operands

Possible bit operands	Example for permitted force values
I1.0:P	True
I1.1:P	False
Q1.0P	0
Q1.1:P	1
I2.0:P	2#0
I2.1:P	2#1

Table 11-8 Byte operands

Possible byte operands	Example for permitted force values
IB1:P	2#00110011
IB2:P	B#16#1F
QB14:P	1F
QB10:P	'a'
IB3:P	10

Table 11-9 Word operands

Possible word operands	Example for permitted force values
IW0:P	2#0011001100110011
IW2:P	W#16#ABCD
QW10:P	ABCD
QW12:P	B#(12, 34)
IW4:P	'ab'
IW6:P	12345
IW8:P	S5T#9S_340ms
IW10:P	C#123
IW12:P	D#2006-12-31

Table 11-10 Double word operands

Possible double word operands	Example for permitted force values
ID0:P	2#00110011001100110011001100110011
ID4:P	1.2
QD10:P	1.234.e4
QD14:P	Dw#16#abcdef10
ID8:P	16#ABCDEF10
ID12:P	b#(12,34,56,78)
ID16:P	L#-12
ID20:P	L#12
ID24:P	123456789
ID28:P	123456789
ID32:P	T#12s345ms
ID36:P	Tod#14:20:40.645
ID40:P	P#e0.0

Overview of the display formats

Display formats in the force table

The display format you select specifies the representation of a tag value.

When entering the address a display format is automatically preset. If you want to change this, you can select a display format from the drop-down list in the "Display formats" column. The drop-down list only offers the display formats which are valid for this data type. The display format that appears first in the list is the pre-selected format.

Example

The following table shows the 32-bit data types permitted for all CPU families in the force table and their possible display formats:

Data type	Possible display formats
BOOL	Bool, Hex, BCD, Octal, Bin, Dec, Dec+/-
BYTE	Hex, BCD, Octal, Bin, Dec, Dec+/-, Character
WORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, SIMATIC_Timer, Date, Unicode_Character, Counter
DWORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Floating-point number, Time of day, Timer, Pointer, Unicode_Character
SINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
INT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter, Date
DINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
USINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
UINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter, Date
UDINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
REAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character,Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Pointer
DATE	Date, Dec, Hex, BCD, Bin
TIME_OF_DAY	Time of day, Dec, Hex, BCD, Bin
TIME	Timer, Hex, BCD, Bin
DATE_AND_TIME	Date and time,
TIMER	SIMATIC_Timer, Hex, BCD, Bin
CHAR	Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
WCHAR	Unicode_Character, Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
STRING	Character string
WSTRING	Unicode_character string
POINTER	Pointer, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec +/-,Dec_Sequence, Floating-point number, Time of day, Timer, Block number
COUNTER	Counter, Hex, BCD, Bin
S5TIME	SIMATIC_Timer, Hex, BCD, Bin

For the S7-1200 CPU family, all 32-bit data types are permitted (see table above), as well as the 64-bit data type LREAL with the following possible display formats:

Data type	Possible display formats
LREAL	In a project created with TIA Portal < V12: Floating-point number Note: The display of LREAL is limited to 13 digits plus exponent.
LREAL	In a project created with TIA Portal >= V12: Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Date and time Note: The display of LREAL is limited to 13 digits plus exponent.

For the S7-1500 CPU family, in addition to 32-bit data types, the 64-bit data types listed in the table are also permitted with the following possible display formats:

Data type	Possible display formats
LWORD	Hex, Octal, BCD, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Date and time
LINT	Dec+/-, Dec, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer, Date and time
ULINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer, Date and time
LREAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Date and time
LTIME	Timer, Dec+/-, Dec, Hex
LTOD	Dec, Hex, BCD, Bin, Time of day
LDT	Dec, Hex, Date and time

For more information, refer to the description of the valid data types (Page 1908).

Note

Rounding of floating-point numbers

In the force table, floating-point numbers are stored as binary numbers in IEEE format. Because not every floating point number (real, longreal) that can be displayed on the user interface can be mapped to the IEEE format, there is a possibility that floating-point numbers will be rounded. If a rounded floating-point number in the force table is copied and, in turn, inserted in another input field, the rounding may cause a slight difference.

Note

Only symbolic addressing is possible

In the force table, LongDataTypes such as LWORD or LREAL can only be addressed symbolically.

Selecting the display format for tags

Procedure

To select the display format of the tags, follow these steps:

1. Enter the desired address in the force table.
2. Click the desired cell in the "Display format" column, and open the drop-down list. The permitted display formats are shown in the drop-down list.
3. Select the desired display format from the drop-down list.

Note

If the selected display format cannot be applied, then the last selected display format will be displayed automatically.

Creating and editing comment lines

Basic principles of comment lines

In addition to the row-related comments in the comment column, you can now also create complete comment lines to enhance the structure of the force table.

The contents of the comment line are stored in the "Languages & Resources" folder in the "Project texts" tab and can be compiled in other project languages.

Creating comment lines

To create comment lines, follow these steps:

1. Open the force table and enter the required addresses.
2. To create a comment line, enter the character string "//" in the "Name" column. No comment lines can be created in the other columns.

3. Enter the required comment in the comment line. The entered comment is shown in green.
4. To show all comments you entered, double-click "Project texts" in the project tree under "Languages & Resources".
5. If you are working in multi-lingual projects and want your comment to be translated into other languages, you can set the project languages required in addition to the editing language in the project tree under "Languages & Resources > Project languages".

Deleting comment lines

To delete comment lines, follow these steps:

1. Open a force table containing comment lines.
2. Delete the entire comment including the introductory string "//", if you no longer require this.
3. Alternatively, delete only the introductory string "//". In this case the existing comment is retained and is displayed in the "Comment" column in the force table.

Note

Deleting comment lines

When you delete comment lines the project languages and any existing translations for these comments are also deleted.

11.4.4.8 Monitoring tags in the force table

Introduction to monitoring tags in the force table

Introduction

Use the force table to monitor the tags of the configured input and output modules in the CPU, dependent on the monitoring mode (Page 1895) you have selected. An online connection to the CPU must exist to monitor tags.

Options for monitoring tags

The following options are available for monitoring tags:

- Monitor all
  This command starts the monitoring of all visible tags in the active force table, dependent on the selected monitoring mode:
  - In basic mode, the monitoring mode is set to "permanent" by default.
  - In expanded mode, you can specify defined trigger points for the monitoring of tags.

  Note

  If the monitoring mode is changed while in expanded mode and then a switch is made to basic mode, the monitoring mode set before will also be applied in basic mode.

- Monitor now
  This command starts the monitoring of the visible tags in the active force table immediately and once only.

CPU-specific limitations when monitoring tags

The following CPU-specific differences exist:

- CPU S7-300/400:
  CPUs from this family can only monitor the first 30 characters of a string.
- CPU S7-1200:
  CPUs from this family can monitor a string up to the total size of 254 characters.

Setting the monitoring mode in the force table

Introduction

By selecting the monitoring mode, you specify the trigger point and the duration of tag monitoring in the force table.

Possible monitoring mode (duration of monitoring)

The following selection options are available:

- Permanent
- Once only, at start of scan cycle
- Once only, at end of scan cycle
- Permanently, at start of scan cycle
- Permanently, at end of scan cycle
- Once only, at transition to STOP
- Permanently, at transition to STOP

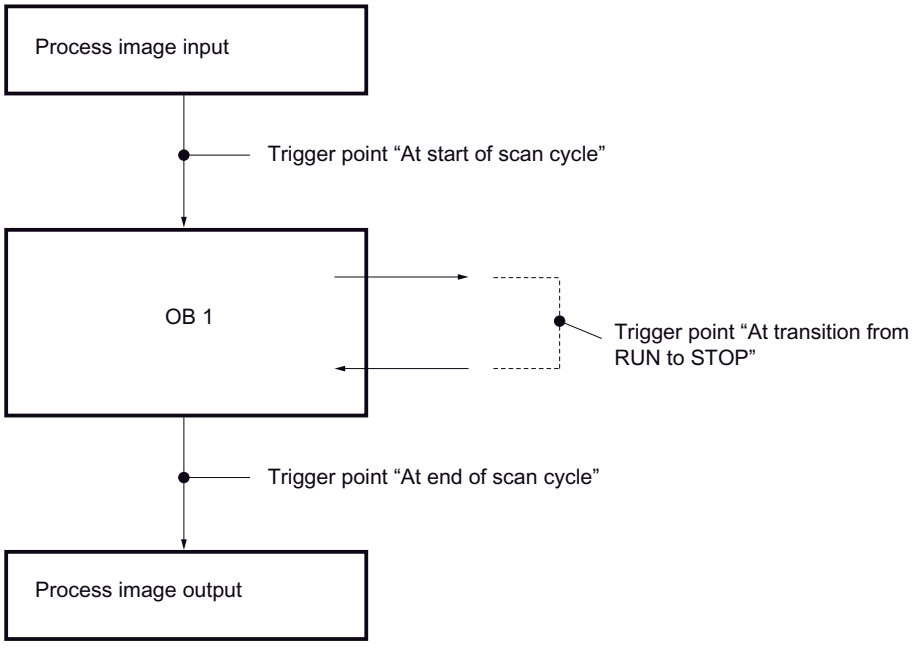
Special features when using "Permanent" mode

The "Permanent" mode is executed as follows for the monitoring of tags: The inputs are monitored at the end of the cycle and the outputs at the start of the cycle.

Selecting the trigger point

The trigger points "Beginning of scan cycle", "End of scan cycle", and "Switch to stop" specify the time at which the tags are to be read from the CPU or updated in the CPU.

The following diagram shows the position of these trigger points:



"Monitor all" command for tags

Introduction

Use the "Monitor all" command to start monitoring the visible tags in the active force table. In basic mode of the force table, the default setting for the monitoring mode is "permanent". In expanded mode, you can specify defined trigger points for the monitoring of tags. In this case, the tags are monitored with reference to the specified trigger points.

Requirements

An online connection to the CPU exists.

Procedure

To execute the "Monitor all" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the force table.
2. Switch to expanded mode by clicking the icon "Show/hide advanced setting columns" in the toolbar.
3. If you want to change the default monitoring mode for a tag, click the appropriate cell in the "Monitor with trigger" column and select the desired monitoring mode from the drop-down list.
4. Click the "Monitor all" icon in the toolbar.

Result

The tags of the active force table will be monitored using the set monitoring mode.

"Monitor now" command for tags

Introduction

The "Monitor now" command starts the monitoring of tags immediately without reference to defined trigger points. The tag values are read out only once and displayed in the force table.

Requirements

An online connection to the CPU exists.

Procedure

To execute the "Monitor now" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the force table.
2. Click the "Monitor now" icon in the toolbar.

Result

The tags of the active force table are monitored immediately and once only.

11.4.4.9 Forcing tags in the force table

Introduction to forcing tags

Introduction

You can use the force table to assign permanent values to individual tags of the user program. This action is referred to as forcing. Only peripheral inputs and peripheral outputs can be forced.

To use the forcing function, you must have an online connection to the CPU and the utilized CPU must support this functionality.

If you open a force table in the "Watch and force tables" folder below a CPU on which a force job is already running, you will first be prompted to update the display of the forced operand. Forcing or stopping forcing in the open force table is only possible following this command.

Possible applications

By permanently assigning defined values to tags, you can specify defined default settings for your user program and, thus, test the programmed functions. Forcing is possible in basic mode and in expanded mode (Page 1884).

Caution when forcing tags

Before forcing, make sure that you review the safety precautions for this procedure (Page 1899).

Danger

Prevent personal injury and material damage!

Remember that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Options for forcing tags

The following options are available for forcing tags:

- Force to "0"
  This command forces the selected address in the CPU to the force value "0".
- Force to "1"
  This command forces the selected address in the CPU to the force value "1".

- Force all
  This command starts the forcing of enabled addresses in the active force table or replaces an existing force job without interruption.
- Stop forcing
  This command stops the forcing of all addresses in the active force table.

Constraints when forcing tags

Note the following constraints when forcing:

- Forcing is always dependent on the operand scope of the CPU used.
- In principle, only peripheral inputs and peripheral outputs can be forced.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is not possible. If desired, deactivate this function in the watch table.

Unique aspects when forcing tags

Note that forcing of tags will overwrite values in the CPU and will continue even after the online connection to the CPU is terminated.

- **Stop forcing**
  Terminating the online connection is not sufficient to stop the forcing operation! To stop forcing, you must select the "Online > Force > Stop forcing" command. Only then will the tags that are visible in the active force table no longer be forced.
- **Stop forcing of individual tags**
  The "Online > Force > Stop forcing" command always applies to all tags displayed in the force table. To stop forcing individual tags, you must clear the check mark for forcing of these tags in the force table and restart forcing using the "Online > Force > Force all" command.

Safety precautions when forcing tags

Safety precautions when forcing tags

Because the forcing function allows you to intervene permanently in the process, observance of the following notices is essential:

Danger

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Caution

Prevent personal injury and material damage!

- Before you start the "Force" function, you should ensure that no one else is currently executing this function on the same CPU.
- Forcing can only be stopped by clicking the "Stop forcing" icon or using the "Online > Force > Stop forcing" command. Closing the active force table does **not** stop the forcing!
- Forcing can **not** be undone!
- Review the differences between " modifying tags" (Page 1873) and "forcing tags" (Page 1898).
- If a CPU does not support the "Force" function, the relevant icons cannot be selected.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is **not** possible on this CPU. If desired, deactivate this function in the watch table.

Updating forced operands

Introduction

If a force job is already running on a CPU, after opening the force table, you first need to make sure that the operands and values currently being forced on the CPU are displayed in the force table.

The command "Online" > "Force" > "Update forced operands" updates all operands and values currently being forced on the CPU in the open force table.

"Force" or "Stop forcing" in the open force table is only possible following this command.

Caution when forcing tags

Before forcing, make sure you are familiar with the safety precautions when forcing tags (Page 1899).

Danger

Prevent personal injury and material damage!

Remember that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Requirement

- An online connection to the CPU is possible.
- A force job is currently running on the CPU being used.

Procedure

To update the forced operands and values, follow these steps:

1. Open a force table.
2. Establish an online connection to the CPU.
3. Confirm the "Update forced operands" dialog that follows with "Yes".

Result

All forced operands in the open force table are updated with the relevant values. A red "F" is displayed in the first column indicating which operands are being forced.

This enables the "Force all" and "Stop forcing" buttons and you can select these functions.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing function!
- To stop forcing, the forced address must be visible in the active force table.

Force tags to "0"

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 1899).

Danger

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags to "0", follow these steps:

1. Open the force table.
2. Enter the desired address in the force table.
3. Select the "Online > Force> Force to 0" command in order to force the selected address with the specified value.
4. Confirm the next dialog with "Yes".

Result

The selected address is forced to "0". The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected values is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Force tags to "1"

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 1899).

Danger

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Endanger the life or health of personnel
- Cause damage to machinery or the entire plant.

Requirements

- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags to "1", follow these steps:

1. Open the force table.
2. Enter the desired address in the force table.
3. Select the "Online > Force> Force to 1" command in order to force the selected address with the specified value.
4. Confirm the next dialog with "Yes".

Result

The selected address is forced to "1". The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected values is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

"Force all" command for tags

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

If forcing is already active, this forcing operation is replaced without interruption by the "Online > Force > Force all" command. Any forced addresses that are not selected will no longer be forced.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 1899).

Danger

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags with the "Online > Force > Force all" command, follow these steps:

1. Open the force table.
2. Enter the desired addresses and force values in the force table.
3. Select the addresses to be forced by selecting the check boxes for forcing in the column after the "Force value".
   A yellow triangle appears behind the selected check box, indicating that the address is selected for forcing but is not being forced at the moment.
4. Select the "Online > Force> Force all" command in order to force the selected addresses with the specified values.
5. Confirm the next dialog with "Yes".

Result

The selected addresses are forced to the specified values. The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected addresses is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

11.4.4.10 Stop forcing tags

Stop forcing all tags

Introduction

Note the following before you stop forcing tags:

- Stopping forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 1899).

Danger

Prevent personal injury and material damage!

Note that an incorrect action when stopping the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- Tags are forced in a force table.
- An online connection to the CPU exists.

- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is not enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

Proceed as follows to stop **forcing all tags** :

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command in order to stop forcing the displayed addresses.
3. Confirm the "Stop forcing" dialog with "Yes".

Result

The forcing of all tags is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is flagged for forcing but is not being forced at the moment.

Stop forcing individual tags

Introduction

Note the following before you stop forcing tags:

- Stopping forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 1883).

Danger

Prevent personal injury and material damage!

Note that an incorrect action when stopping the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- Tags are forced in a force table.
- An online connection to the CPU exists.

- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is not enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

Proceed as follows to stop **forcing individual tags** :

1. Open the force table.
2. Deactivate the check boxes for the addresses that are no longer to be forced.
3. Reselect the "Online > Force" command.

Result

Forcing of the disabled addresses will be stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is flagged for forcing but is not being forced at the moment.

11.5 Data types

11.5.1 Overview of the valid data types

Validity of data type groups

The data type groups define the properties of the data, for example, the representation of the contents and the valid memory areas. In the user program, you can use predefined data type or also data types that you have defined.

The following tables show the availability of predefined data types in the various S7-CPUs:

Table 11-11 Binary numbers

Binary numbers	S7-300/400	S7-1200	S7-1500
BOOL (Page 1912)	X	X	X
Bit strings			
BYTE (Page 1913)	X	X	X
WORD (Page 1914)	X	X	X
DWORD (Page 1915)	X	X	X
LWORD (Page 1915)	-	-	X

Table 11-12 Integers

Integers	S7-300/400	S7-1200	S7-1500
SINT (Page 1917)	-	X	X
INT (Page 1918)	X	X	X
DINT (Page 1920)	X	X	X
USINT (Page 1918)	-	X	X
UINT (Page 1919)	-	X	X
UDINT (Page 1921)	-	X	X
LINT (Page 1922)	-	-	X
ULINT (Page 1923)	-	-	X

Table 11-13 Floating-point numbers

Floating-point numbers	S7-300/400	S7-1200	S7-1500
REAL (Page 1925)	X	X	X
LREAL (Page 1926)	-	X	X

Table 11-14 Timers

Timers	S7-300/400	S7-1200	S7-1500
S5TIME (Page 1929)	X	-	X
TIME (Page 1930)	X	X	X
LTIME (Page 1931)	-	-	X

Table 11-15 Date and time

Date and time	S7-300/400	S7-1200	S7-1500
DATE (Page 1931)	X	X	X
TIME_OF_DAY (TOD) (Page 1932)	X	X	X
LTOD (LTIME_OF_DAY) (Page 1932)	-	-	X
DT (DATE_AND_TIME) (Page 1933)	X	-	X
LDT (Page 1934)	-	-	X
DTL (Page 1935)	-	X	X

Table 11-16 Character

Character	S7-300/400	S7-1200	S7-1500
CHAR (Page 1936)	X	X	X
WCHAR (Page 1937)	-	X	X
STRING (Page 1937)	X	X	X
WSTRING (Page 1939)	-	X	X

Table 11-17 Array

Array	S7-300/400	S7-1200	S7-1500
ARRAY [….] OF <type> (Page 1941)	X	X	X

Table 11-18 Structures

Structures	S7-300/400	S7-1200	S7-1500
STRUCT (Page 1945)	X	X	X

Table 11-19 Pointer

Pointer	S7-300/400	S7-1200	S7-1500
POINTER (Page 1946)	X	-	X
ANY (Page 1948)	X	-	X
VARIANT (Page 1951)	-	X	X

Table 11-20 Parameter types

Parameter types	S7-300/400	S7-1200	S7-1500
TIMER (Page 1953)	X	-	X
COUNTER (Page 1953)	X	-	X
BLOCK_FC (Page 1953)	X	-	X
BLOCK_FB (Page 1953)	X	-	X
BLOCK_DB (Page 1953)	X	-	-
BLOCK_SDB (Page 1953)	X	-	-
BLOCK_SFB (Page 1953)	X	-	-
BLOCK_SFC (Page 1953)	X	-	-
BLOCK_OB (Page 1953)	X	X	X
VOID (Page 1953)	X	X	X

Table 11-21 PLC data types

PLC data types	S7-300/400	S7-1200	S7-1500
PLC data type (Page 1954)	X	X	X

Table 11-22 System data types

System data types	S7-300/400	S7-1200	S7-1500
IEC_TIMER (Page 1955)	X[1)]	X	X
IEC_LTIMER (Page 1955)	-	-	X
IEC_SCOUNTER (Page 1955)	-	X	X
IEC_USCOUNTER (Page 1955)	-	X	X

System data types	S7-300/400	S7-1200	S7-1500
IEC_COUNTER (Page 1955)	X[2)]	X	X
IEC_UCOUNTER (Page 1955)	-	X	X
IEC_DCOUNTER (Page 1955)	-	X	X
IEC_UDCOUNTER (Page 1955)	-	X	X
IEC_LCOUNTER (Page 1955)	-	-	X
IEC_ULCOUNTER (Page 1955)	-	-	X
ERROR_STRUCT (Page 1955)	-	X	X
NREF (Page 1955)	-	X	X
CREF (Page 1955)	-	X	X
FBTREF (Page 1955)	-	-	-
VREF (Page 1955)	-	-	-
STARTINFO (Page 1955)	X	-	X
SSL_HEADER (Page 1955)	X	-	X
CONDITIONS (Page 1955)	-	X	X
TADDR_Param (Page 1955)	-	X	X
TCON_Param (Page 1955)	-	X	X
HSC_Period (Page 1955)	-	X	-
[1)] For S7-300/400 CPUs, the data type is represented by TP, TON and TOF.			
[2)] For S7-300/400 CPUs, the data type is represented by CTU, CTD and CTUD.			

Table 11-23 Hardware data types

Hardware data types	S7-300/400	S7-1200	S7-1500
REMOTE (Page 1957)	-	X	X
GEOADDR (Page 1957)	-	-	X
HW_ANY (Page 1957)	-	X	X
HW_DEVICE (Page 1957)	-	X	X
HW_DPMASTER (Page 1957)	-	-	X
HW_DPSLAVE (Page 1957)	-	X	X
HW_IO (Page 1957)	-	X	X
HW_IOSYSTEM (Page 1957)	-	X	X
HW_SUBMODULE (Page 1957)	-	X	X
HW_MODULE (Page 1957)	-	-	X
HW_INTERFACE (Page 1957)	-	X	X
HW_IEPORT (Page 1957)	-	X	X
HW_HSC (Page 1957)	-	X	X
HW_PWM (Page 1957)	-	X	X
HW_PTO (Page 1957)	-	X	X
AOM_AID (Page 1957)	-	X	X
AOM_IDENT (Page 1957)	-	X	X
EVENT_ANY (Page 1957)	-	X	X
EVENT_ATT (Page 1957)	-	X	X

Hardware data types	S7-300/400	S7-1200	S7-1500
EVENT_HWINT (Page 1957)	-	X	X
OB_ANY (Page 1957)	-	X	X
OB_DELAY (Page 1957)	-	X	X
OB_TOD (Page 1957)	-	X	X
OB_CYCLIC (Page 1957)	-	X	X
OB_ATT (Page 1957)	-	X	X
OB_PCYCLE (Page 1957)	-	X	X
OB_HWINT (Page 1957)	-	X	X
OB_DIAG (Page 1957)	-	X	X
OB_TIMEERROR (Page 1957)	-	X	X
OB_STARTUP (Page 1957)	-	X	X
PORT (Page 1957)	-	X	X
RTM (Page 1957)	-	X	X
PIP (Page 1957)	-	-	X
CONN_ANY (Page 1957)	-	X	X
CONN_PRG (Page 1957)	-	X	X
CONN_OUC (Page 1957)	-	X	X
CONN_R_ID (Page 1957)	-	-	X
DB_ANY (Page 1957)	-	X	X
DB_WWW (Page 1957)	-	X	X
DB_DYN (Page 1957)	-	X	X

Note

Depending on the CPU version, the actually valid data types can deviate slightly from the table.

11.5.2 Binary numbers

11.5.2.1 BOOL (bit)

Description

An operand of data type BOOL represents a bit value and contains one of the following values:

- TRUE
- FALSE

The following table shows the properties of data type BOOL:

Length (bits)	Format	Range of values	Examples of value input
1	Boolean	FALSE or TRUE BOOL#0 or BOOL#1 BOOL#FALSE or BOOL#TRUE	TRUE BOOL#1 BOOL#TRUE
	Unsigned integers	0 or 1	1
	Binary numbers	2#0 or 2#1	2#0
	Octal numbers	8#0 or 8#1	8#1
	Hexadecimal numbers	16#0 or 16#1	16#1

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", the bit has a length of 1 byte.

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.2.2 Bit strings

BYTE (byte)

Description

An operand of data type BYTE is a bit string of 8 bits.

The following table shows the properties of data type BYTE:

Length (bits)	Format	Value range	Examples of value input
8	Unsigned integers[1)]	-128 to +127 or 0 to +255	15, BYTE#15, B#15
	Binary numbers	2#0 to 2#11111111	2#00001111, BYTE#2#00001111, B#2#00001111
	Octal numbers	8#0 to 8#377	8#17, BYTE#8#17, B#8#17
	Hexadecimal numbers	B#16#0 to B#16#FF, 16#0 to 16#FF	16#0F, BYTE#16#0F, B#16#0F
[1)] The value range depends on the relevant interpretation or conversion.			

Note

The BYTE data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the SINT and USINT data types.

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

WORD

Description

An operand of data type WORD is a bit string of 16 bits.

The following table shows the properties of data type WORD:

Length (bits)	Format	Value range	Examples of value input
16	Unsigned integers	-32768 to 65535	61680, WORD#61680, W#61680
	Binary numbers	2#0 to 2#1111111111111111	2#1111000011110000, WORD#2#1111000011110000, W#2#1111000011110000
	Octal numbers	8#0 to 8#177777	8#170360, WORD#8#170360, W#8#170360
	Hexadecimal numbers	W#16#0 to W#16#FFFF, 16#0 to 16#FFFF	16#F0F0, WORD#16#F0F0, W#16#F0F0
	BCD	C#0 to C#999	C#55
	Decimal sequence	B#(0, 0) to B#(255, 255)	B#(127, 200)

Note

The WORD data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the INT and UINT data types.

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

DWORD

Description

An operand of data type DWORD is a bit string of 32 bits.

The following table shows the properties of data type DWORD:

Length (bits)	Format	Value range	Examples of value input
32	Unsigned integers	-2147483648 to 4294967295	15793935, DWORD#15793935, DW#15793935
	Binary numbers	2#0 to 2#11111111111111111111111111111111	2#00000000111100001111111100001111, DWORD#2#00000000111100001111111100001111, DW#2#00000000111100001111111100001111
	Octal numbers	8#0 to 8#37777777777	8#74177417, DWORD#8#74177417, DW#8#74177417
	Hexadecimal numbers	DW#16#00000000 to DW#16#FFFFFFFF, 16#00000000 to 16#FFFFFFFF	16#00F0FF0F, DWORD#16#00F0FF0F, DW#16#00F0FF0F
	Decimal sequence	B#(0, 0, 0, 0) to B#(255, 255, 255, 255)	B#(127, 200, 127, 200)

Note

The DWORD data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the DINT and UDINT data types.

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

LWORD

Description

An operand of data type LWORD is a bit string of 64 bits.

The following table shows the properties of data type LWORD:

Length (bits)	Format	Value range	Examples of value input
64	Unsigned integers	-9223372036854775808 to 18446744073709551615	26123590360715, LWORD#26123590360715, LW#26123590360715
	Binary numbers	2#0 to 2#1111111111111111111111111111111111111111111111111111111111111111	2#0000000000000000000010111110000100101111010100101101111010001011, LWORD#2#0000000000000000000101111100001001011110101001011101000101, LW#2#0000000000000000001011111000010010111101010010111010001011
	Octal numbers	8#0 to 8#1777777777777777777777	8#13724557213, LWORD#8#13724557213, LW#8#13724557213
	Hexadecimal numbers	LW#16#00000000 to LW#16#FFFFFFFFFFFFFFFF, 16#00000000 to 16#FFFFFFFFFFFFFFFF	16#000000005F52DE8B, LWORD#16#000000005F52DE8B, LW#16#000000005F52DE8B
	Decimal sequence	B#(0, 0, 0, 0, 0, 0, 0, 0) to B#(255, 255, 255, 255, 255, 255, 255, 255)	B#(127, 200, 127, 200, 127, 200, 127, 200)

Note

The LWORD data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the LINT and ULINT data types.

See also

Overview of the valid data types (Page 1908)

Implicit conversion (Page 1961)

Explicit conversion (Page 2010)

Overview of data type conversion (Page 1959)

Data type conversion for S7-1200: (Page 2091)

11.5.3 Integers

11.5.3.1 SINT (8-bit integers)

Description

An operand of data type SINT (Short INT) has a length of 8 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 6 represent the number value. The signal state of bit 7 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

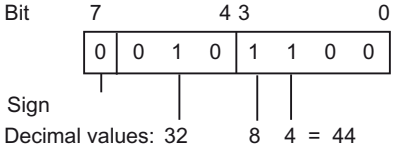
An operand of data type SINT occupies one BYTE in the memory.

The following table shows the properties of data type SINT:

Length (bits)	Format	Range of values	Examples of value input
8	Signed integers	-128 to 127	+44, SINT#+44 The value range is a maximum of SINT#255 when using the type SINT#. This value is interpreted as a whole number with -1.
	Binary numbers (only positive)	2#0 to 2#01111111	2#00101100, SINT#2#00101100
	Octal numbers (only positive)	8#0 to 8#177	8#54, SINT#8#54
	Hexadecimal numbers (only positive)	16#0 to 16#7F	16#2C, SINT#16#2C The value range is a maximum of SINT#16#FF when using the type SINT#. This value is interpreted as a whole number with -1.

Example

The following figure shows the integer +44 as a binary number:



See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.3.2 USINT (8-bit integers)

Description

An operand of data type USINT (Unsigned Short INT) has a length of 8 bits and contains unsigned numerical values:

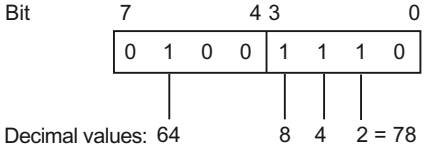
An operand of data type USINT occupies one BYTE in the memory.

The following table shows the properties of data type USINT:

Length (bits)	Format	Range of values	Examples of value input
8	Unsigned integers	0 to 255	78, USINT#78
	Binary numbers	2#0 to 2#11111111	2#01001110, USINT#2#01001110
	Octal numbers	8#0 to 8#377	8#116, USINT#8#116
	Hexadecimal numbers	16#0 to 16#FF	16#4E, USINT#16#4E

Example

The following figure shows the integer 78 as a binary number:



See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.3.3 INT (16-bit integers)

Description

An operand of data type INT has a length of 16 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 14 represent the number value. The signal state of bit 15 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

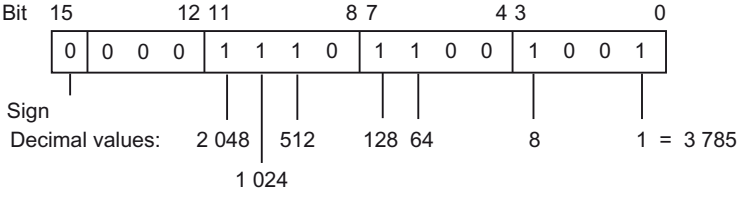
An operand of data type INT occupies two BYTE in the memory.

The following table shows the properties of data type INT:

Length (bits)	Format	Range of values	Examples of value input
16	Signed integers	-32768 to 32767	+3785, INT#+3785
	Binary numbers (only positive)	2#0 to 2#0111111111111111	2#0000111011001001, INT#2#0000111011001001
	Octal numbers	8#0 to 8#77777	8#7311, INT#8#7311
	Hexadecimal numbers (only positive)	16#0 to 16#7FFF	16#0EC9, INT#16#0EC9

Example

The following figure shows the integer +3785 as a binary number:



See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.3.4 UINT (16-bit integers)

Description

An operand of data type UINT (Unsigned INT) has a length of 16 bits and contains unsigned numerical values.

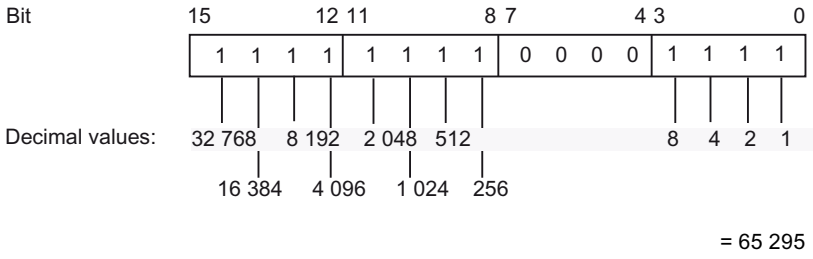
An operand of data type UINT occupies two BYTE in the memory.

The following table shows the properties of data type UINT:

Length (bits)	Format	Range of values	Examples of value input
16	Unsigned integers	0 to 65535	65295, UINT#65295
	Binary numbers	2#0 to 2#1111111111111111	2#1111111100001111, UINT#2#1111111100001111
	Octal numbers	8#0 to 8#177777	8#177417, UINT#8#177417
	Hexadecimal numbers	16#0 to 16#FFFF	16#FF0F, UINT#16#FF0F

Example

The following figure shows the integer 65295 as a binary number:



See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.3.5 DINT (32-bit integers)

Description

An operand of data type DINT (Double INT) has a length of 32 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 30 represent the number value. The signal state of bit 31 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

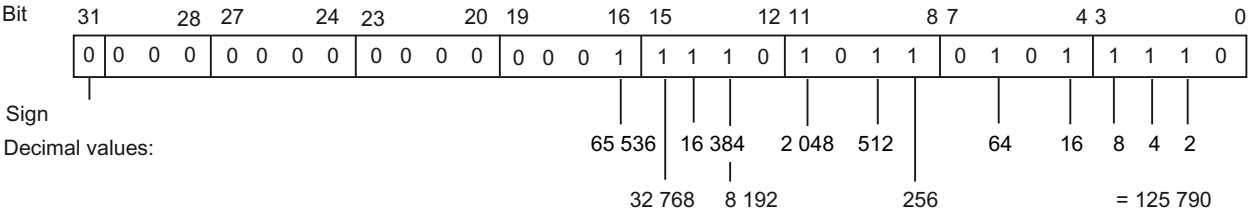
An operand of data type DINT occupies four BYTE in the memory.

The following table shows the properties of data type DINT:

Length (bits)	Format	Range of values	Examples of value input
32	Signed integers	-2147483648 to +2147483647	125790, DINT#125790, L#275
	Binary numbers (only positive)	2#0 to 2#01111111111111111111111111111111	2#00000000000000011110101101011110, DINT#2#00000000000000011110101101011110
	Octal numbers (only positive)	8#0 to 8#17777777777	8#365536, DINT#8#365536
	Hexadecimal numbers	16#00000000 to 16#7FFFFFFF	16#0001EB5E, DINT#16#0001EB5E

Example

The following figure shows the integer +125790 as a binary number:



See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.3.6 UDINT (32-bit integers)

Description

An operand of data type UDINT (Unsigned Double INT) has a length of 32 bits and contains unsigned numerical values.

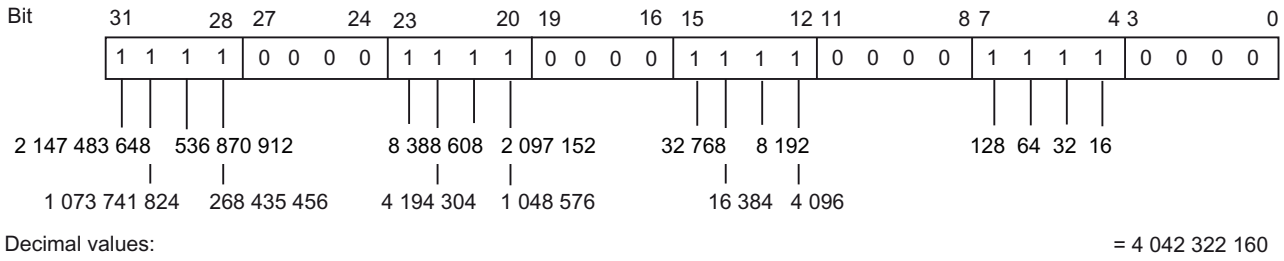
An operand of data type UDINT occupies four BYTE in the memory.

The following table shows the properties of data type UDINT:

Length (bits)	Format	Range of values	Examples of value input
32	Unsigned integers	0 to 4294967295	4042322160, UDINT#4042322160
	Binary numbers	2#0 to 2#11111111111111111111111111111111	2#11110000111100001111000011110000, UDINT#2#11110000111100001111000011110000
	Octal numbers	8#0 to 8#37777777777	8#36074170360, UDINT#8#36074170360
	Hexadecimal numbers	16#00000000 to 16#FFFFFFFF	16#F0F0F0F0, UDINT#16#F0F0F0F0

Example

The following figure shows the integer 4042322160 as a binary number:



See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.3.7 LINT (64-bit integers)

Description

An operand of data type LINT (Long INT) has a length of 64 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 62 represent the number value. The signal state of bit 63 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

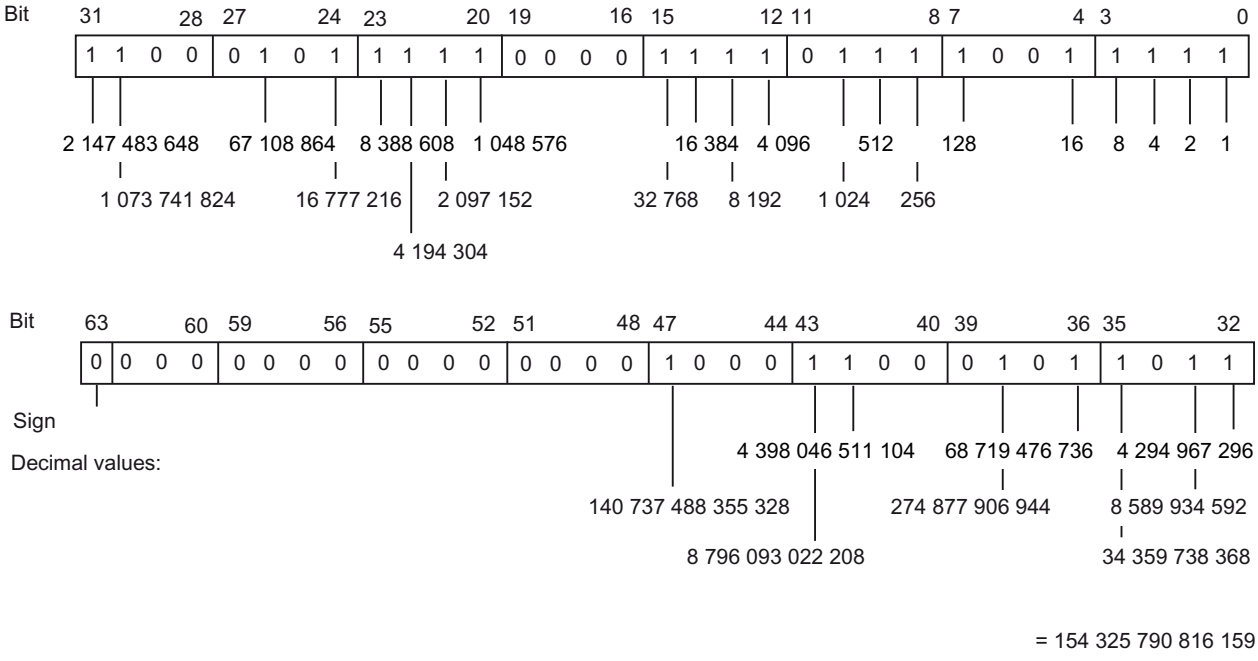
An operand of data type LINT occupies eight BYTE in the memory.

The following table shows the properties of data type LINT:

Length (bits)	Format	Value range	Examples of value input
64	Signed integers	-9223372036854775808 to +9223372036854775807	+154325790816159, LINT# +154325790816159
	Binary numbers (only positive)	2#0 to 2#0111111111111111111111111111111111111111111111111111111111111111	2#0000000000000000100011000101101111000101111100001111011110011111, LINT#2#0000000000000000100011000100101101111000101111100001111011110011111
	Octal numbers	8#0 to 8#7777777777777777777	8#4305570574173637, LINT#8#4305570574173637
	Hexadecimal numbers (only positive)	16#0 to 16#7FFFFFFFFFFFFFFF	16#00008C5BC5F0F79F, LINT#16#00008C5BC5F0F79F

Example

The following figure shows the integer +154325790816159 as a binary number:



See also

Overview of the valid data types (Page 1908)

Overview of data type conversion (Page 1959)

Implicit conversion (Page 1961)

Explicit conversion (Page 2010)

Data type conversion for S7-1200: (Page 2091)

11.5.3.8 ULINT (64-bit integers)

Description

An operand of data type ULINT (Unsigned Long INT) has a length of 64 bits and contains unsigned numerical values.

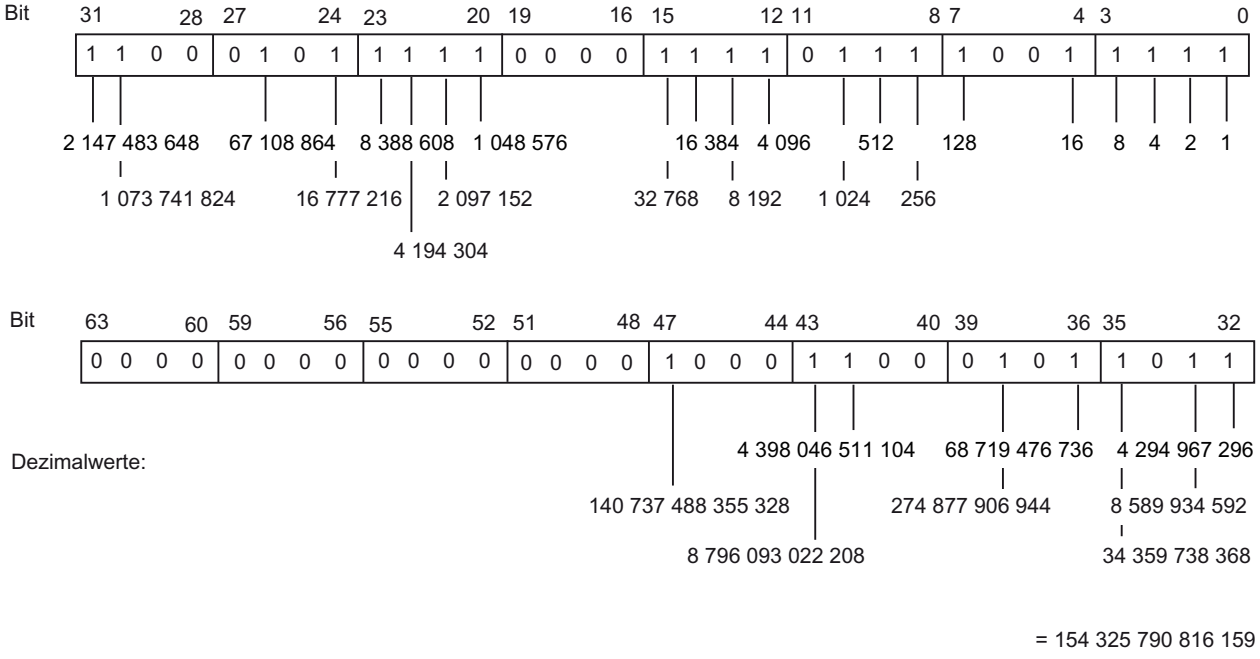
An operand of data type ULINT occupies eight BYTE in the memory.

The following table shows the properties of data type ULINT:

Length (bits)	Format	Value range	Examples of value input
64	Unsigned integers	0 to 18446744073709551615	154325790816159, ULINT#154325790816159
	Binary numbers	2#0 to 2#1111111111111111111111111111111111111111111111111111111111111111	2#0000000000000000100011000101101110001011111000011110111100111111, ULINT#2#0000000000000000100011000101101111000101111100001111011110011111
	Octal numbers	8#0 to 8#1777777777777777777777	8#4305570574173637, ULINT#8#4305570574173637
	Hexadecimal numbers	16#0 to 16#FFFFFFFFFFFFFFFF	16#00008C5BC5F0F79F, ULINT#16#00008C5BC5F0F79F

Example

The following figure shows the integer 154325790816159 as a binary number:



See also

Overview of the valid data types (Page 1908)

Overview of data type conversion (Page 1959)

Implicit conversion (Page 1961)

Explicit conversion (Page 2010)

Data type conversion for S7-1200: (Page 2091)

11.5.4 Floating-point numbers

11.5.4.1 REAL

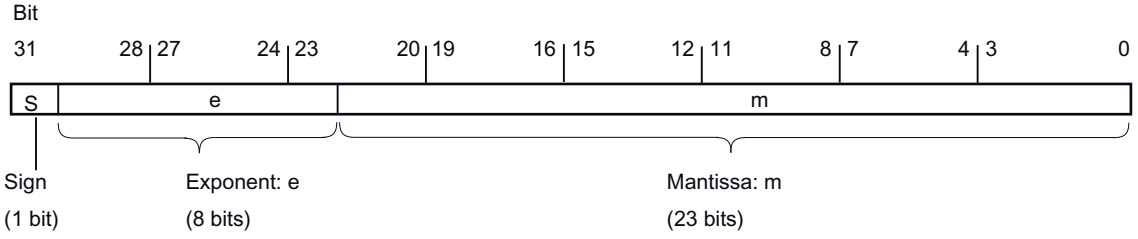
Description

Operands of the data type REAL have a length of 32 bits and are used to represent floating-point numbers. An operand of the REAL data type consists of the following three components:

- Sign: The sign is determined by the signal state of bit 31. The bit 31 assume the value "0" (positive) or "1" (negative).
- 8-bit exponents to basis 2: The exponent is increased by a constant (base, +127), so that it has a value range of 0 to 255.
- 23-bit mantissa: Only the fraction part of the mantissa is shown. The integer part of the mantissa is always 1 with normalized floating-point numbers and is not stored.

The REAL data type is processed with a precision of 6 digits.

The following figure shows the structure of the REAL data type:



Note

With floating-point numbers, only the precision defined by the IEEE754 standard is stored. Additionally specified decimals are rounded off according to IEEE754.

The number of decimal places may decrease for frequently nested arithmetic calculations.

If more decimal places are specified than can be stored by the data type, the number is rounded to the value corresponding to the precision allowed by this value range.

The following table shows the properties of data type REAL:

Length (bits)	Format	Value range	Examples of value input
32	Floating-point numbers according to IEEE754	-3.402823e+38 to -1.175495e-38 ±0,0 +1.175495e-38 to +3.402823e+38	1.0e-5; REAL#1.0e-5
	Floating-point numbers		1.0; REAL#1.0

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

Calculating with floating-point numbers (REAL and LREAL) in SCL (Page 290)

11.5.4.2 LREAL

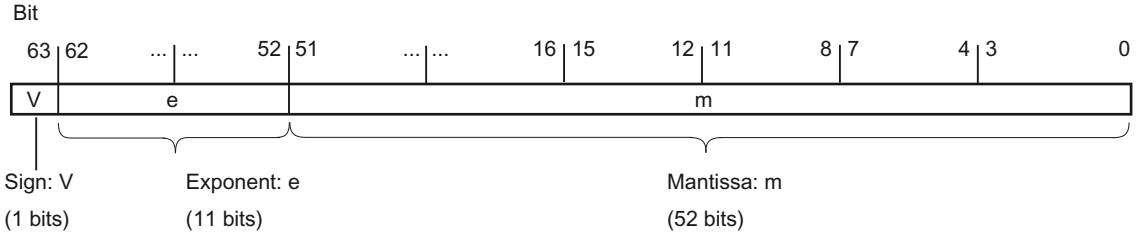
Description

Operands of the data type LREAL have a length of 64 bits and are used to represent floating-point numbers. An operand of the LREAL data type consists of the following three components:

- Sign: The sign is determined by the signal state of bit 63. The bit 63 assumes the value "0" (positive) or "1" (negative).
- 11-bit exponents to base 2: The exponent is increased by a constant (base, +1023), so that it has a value range of 0 to 2047.
- 52-bit mantissa: Only the fraction part of the mantissa is shown. The integer part of the mantissa is always 1 with normalized floating-point numbers and is not stored.

The LREAL data type is processed with a precision of 15 digits.

The following figure shows the structure of the LREAL data type:



The following table shows the properties of data type LREAL:

Length (bits)	Format	Value range	Examples of value input
64	Floating-point numbers according to IEEE754	-1.7976931348623158e+308 to -2.2250738585072014e-308 ±0,0 +2.2250738585072014e-308 to +1.7976931348623158e+308	1.0e-5; LREAL#1.0e-5
	Floating-point numbers		1.0; LREAL#1.0

Note

With floating-point numbers, only the precision defined by the IEEE754 standard is stored. Additionally specified decimals are rounded off according to IEEE754.

The number of decimal places may decrease for frequently nested arithmetic calculations.

If more decimal places are specified than can be stored by the data type, the number is rounded to the corresponding value of the precision allowed by this value range .

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

Calculating with floating-point numbers (REAL and LREAL) in SCL (Page 290)

11.5.4.3 Invalid floating-point numbers

Description

We distinguish between four number ranges for data types REAL and LREAL:

- normalized numbers stored with full accuracy
- denormalized numbers not stored with full accuracy
- Infinite numbers: +Inf/-Inf (infinity)
- Invalid numbers: NaN (Not a Number)

Note

Floating-point numbers are stored as specified by the IEEE754 standard. Results of conversion or arithmetic functions with a denormalized, infinite or NaN (Not a Number) floating point depend on the CPU.

If you are not working with normalized floating-point numbers in mathematical functions, the result will show significant differences depending on the series of the CPU which you are using.

A CPU cannot calculate with denormalized floating-point numbers, with the exception of older CPU versions of the S7-300 and S7-400 series. The bit pattern of a denormalized number is interpreted as a zero. If the result of calculation falls into this range, it is continued with zero; the status bits OV and OS are set (number range undershoot).

Even though the values of invalid floating-point numbers can only be displayed with limited accuracy for mathematical functions, numbers with an exponent of -39 (e.g. 2.4408e-039) can

be monitored in the TIA Portal and therefore do not necessarily represent a faulty result. This means that floating-point values may be located outside the range of valid numerical values.

Note

The following applies to CPUs of the series S7-1200 V1, V2 and V3:

The comparison operation "Equal" uses the bit pattern of the invalid floating-point number. If two "NaN numbers" with the same bit pattern are compared, the output of the "Equal" comparison operation returns the result TRUE.

Note

The following applies to CPUs of the S7-1200 V4 and S7-1500 series:

If two invalid numbers (NaN) are compared with each other, the result is always FALSE, regardless of the bit pattern of the invalid number or the relation (>, >, ...).

Note

Comparison of denormalized floating-point numbers

For the comparison operation "Equal" with two denormalized floating-point numbers, the output for CPUs of the S7-300/400 series is set to the signal state "0" and for CPUs of the S7-1200/1500 series to the signal state "1".

If the input variables of a mathematical function represent an invalid floating-point number, an invalid floating-point number will also be output as result.

You have the following options to evaluate possible errors caused by invalid floating-point numbers:

- In LAD/FBD and SCL, you can query the enable output ENO for FALSE
- In STL, you can evaluate the status bit OV

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

Calculating with floating-point numbers (REAL and LREAL) in SCL (Page 290)

11.5.5 Timers

11.5.5.1 S5TIME (duration)

Format

Data type S5TIME stores the duration in BCD format. The duration is the product from a time in the range 0 to 999 and a time basis. The time basis indicates the interval at which a timer decrements the time value by one unit until it reaches "0". The resolution of the times can be controlled via the time basis.

The following table shows the range of values for data type S5TIME:

Length (bits)	Format	Range of values	Examples of value input
16	S7 time in increments of 10 ms (default value)	S5T#0MS to S5T#2H_46M_30S_0MS	S5T#10s, S5TIME#10s
	Hexadecimal numbers	16#0 to 16#3999	16#2

The following table shows the time base coding for S5TIME:

Time basis	Binary code for time basis
10 ms	00
100 ms	01
1 s	10
10 s	11

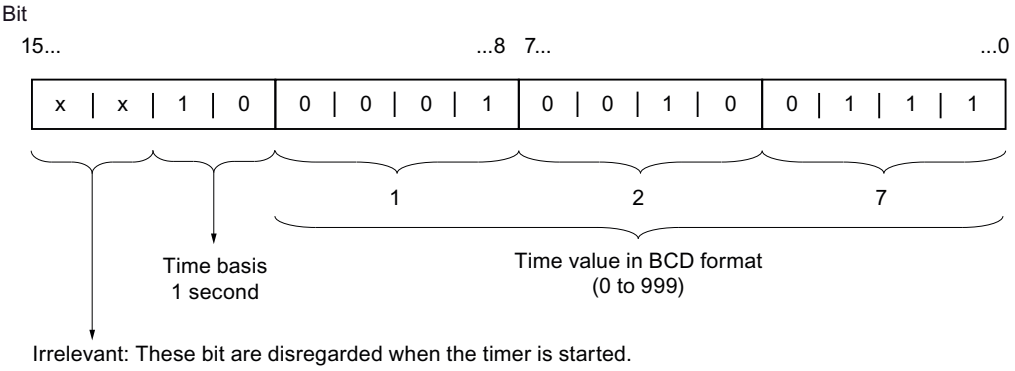
Always observe range limits and the resolution of time values when using data type S5TIME with timers. The table below specifies the range associated with each resolution:

Resolution	Range
0.01 s	10 ms to 9 s 990 ms
0.1 s	100 ms to 1 min 39 s 900 ms
1 s	1 s to 16 min 39 s
10 s	10 s to 2 h 46 min 30 s

Values that exceed 2h46m30s are not accepted.

Example

The following figure shows the content of the time operand for a time value of 127 and a time base of 1 s:



See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.5.2 TIME (IEC time)

Description

The contents of an operand of the data type TIME is interpreted as milliseconds. The representation contains information for days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms).

The following table shows the properties of data type TIME:

Length (bits)	Format	Value range	Examples of value input
32	Signed duration	T#-24d20h31m23s648ms to T#+24d20h31m23s647ms	T#10d20h30m20s630ms, TIME#10d20h30m20s630ms
	Hexadecimal numbers	16#00000000 to 16#7FFFFFFF	16#0001EB5E

It is not necessary to specify all time units. T#5h10s is a valid entry, for example. If only one unit is specified, the absolute value of days, hours, and minutes must not exceed the high or low limits. When more than one time unit is specified, the value must not exceed 24 days, 23 hours, 59 minutes, 59 seconds or 999 milliseconds.

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.5.3 LTIME (IEC time)

Description

The contents of an operand of data type LTIME is interpreted as nanoseconds. The representation contains information for days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms), microseconds (us) and nanoseconds (ns).

The following table shows the properties of data type LTIME:

Length (bits)	Format	Range of values	Examples of value input
64	Signed duration	LT#-106751d23h47m16s854ms775us808ns to LT#+106751d23h47m16s854ms775us807ns	LT#11350d20h25m14s830ms652us315ns, LTIME#11350d20h25m14s830ms652us315ns
	Hexadecimal numbers	16#0 to 16#8000000000000000	16#2

It is not necessary to specify all time units. LT#5h10s is therefore a valid entry, for example. If only one unit is specified, the absolute value of days, hours, and minutes must not exceed the high or low limits. When more than one time unit is specified, the value must not exceed 106751 days, 23 hours, 59 minutes, 59 seconds, 999 milliseconds, 999 microseconds or 999 nanoseconds.

See also

Overview of the valid data types (Page 1908)

Overview of data type conversion (Page 1959)

Implicit conversion (Page 1961)

Explicit conversion (Page 2010)

Data type conversion for S7-1200: (Page 2091)

11.5.6 Date and time

11.5.6.1 DATE

Format

The DATE data type saves the date as an unsigned integer. The representation contains the year, the month, and the day.

The contents of an operand of DATE data type correspond in hexadecimal format to the number of days since 01-01-1990 (16#0000).

The following table shows the properties of data type DATE:

Length (bytes)	Format	Range of values	Examples of value input
2	IEC date (Year-Month-Day)	D#1990-01-01 to D#2168-12-31	D#2009-12-31, DATE#2009-12-31
	Hexadecimal numbers	16#0000 to 16#FF62	16#00F2

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.6.2 TIME_OF_DAY (TOD)

Format

Data type TOD (TIME_OF_DAY) occupies a double word and stores the number of milliseconds since the beginning of the day (0:00 h) as unsigned integer.

The following table shows the properties of data type TOD:

Length (bytes)	Format	Value range	Examples of value input
4	Time-of-day (hours:minutes:seconds.milliseconds)	TOD#00:00:00.000 to TOD#23:59:59.999	TOD#10:20:30.400, TIME_OF_DAY#10:20:30.400

You always need to specify the hours, minutes and seconds. The specification of milliseconds is optional.

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.6.3 LTOD (LTIME_OF_DAY)

Format

Data type LTOD (LTIME_OF_DAY) occupies two double words and stores the number of nanoseconds since the beginning of the day (0:00 h) as unsigned integer.

The following table shows the properties of data type LTOD:

Length (bytes)	Format	Value range	Examples of value input
8	Time-of-day (hours:minutes:seconds.nanoseconds)	LTOD#00:00:00.000000000 to LTOD#23:59:59.999999999	LTOD#10:20:30.400_365_215, LTIME_OF_DAY#10:20:30.400_365_215

You always need to specify the hours, minutes and seconds. The specification of nanoseconds is optional.

See also

Overview of the valid data types (Page 1908)

Overview of data type conversion (Page 1959)

Implicit conversion (Page 1961)

Explicit conversion (Page 2010)

Data type conversion for S7-1200: (Page 2091)

11.5.6.4 DATE_AND_TIME (date and time of day)

Format

The DT (DATE_AND_TIME) data type saves the information on date and time of day in BCD format.

The following table shows the properties of data type DT:

Length (bytes)	Format	Range of values	Example of value input
8	Date and time (year-month-day-hour:minute:second:millisecond [3])	Min.: DT#1990-01-01-00:00:00.000 Max.: DT#2089-12-31-23:59:59.999	DT#2008-10-25-8:12:34.567, DATE_AND_TIME#2008-10-25-08:12:34.567

The following table shows the structure of the DT data type:

Byte	Contents	Range of values
0	Year	0 to 99 (Years 1990 to 2089) BCD#90 = 1990 ... BCD#0 = 2000 ... BCD#89 = 2089
1	Month	BCD#0 to BCD#12
2	Day	BCD#1 to BCD# 31
3	Hour	BCD#0 to BCD#23
4	Minute	BCD#0 to BCD#59
5	Second	BCD#0 to BCD#59
6	The two most significant digits of MSEC	BCD#0 to BCD#999
7 (4MSB) [1]	The least significant digit of MSEC	BCD#0 to BCD#9
7 (4LSB) [2]	Weekday	BCD#1 to BCD#7 BCD#1 = Sunday ... BCD#7 = Saturday
[1] MSB: Most significant bit [2] LSB: Least significant bit [3] Fixed point number		

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.6.5 LDT (DATE_AND_LTIME)

Format

Data type LDT (DATE_AND_LTIME) stores the date and time-of-day information in nanoseconds since 01/01/1970 0:0.

The following table shows the properties of data type LDT:

Length (bytes)	Format	Value range	Example of value input
8	Date and time (Year-Month-Day-Hour:Minute:Second.Nanoseconds)	Min.: LDT#1970-01-01-0:0:0.000000000 Max.: LDT#2263-04-11-23:47:16.854775808	LDT#2008-10-25-8:12:34.567
	Hexadecimal numbers	16#0 to 16#7FFF_FFFF_FFFF_FFFF	16#7FFF

See also

Overview of the valid data types (Page 1908)

Overview of data type conversion (Page 1959)

Implicit conversion (Page 1961)

Explicit conversion (Page 2010)

Data type conversion for S7-1200: (Page 2091)

11.5.6.6 DTL

Description

An operand of data type DTL has a length of 12 bytes and stores date and time information in a predefined structure.

The following table shows the properties of data type DTL:

Length (bytes)	Format	Value range	Example of value input
12	Date and time (Year-Month-Day-Hour:Minute:Second.Nanoseconds)	Min.: DTL#1970-01-01-00:00:00.0 Max.: DTL#2262-04-11-23:47:16.854775807	DTL#2008-12-16-20:30:20.250

The structure of data type DTL consists of several components each of which can contain a different data type and range of values. The data type of a specified value must match the data type of the corresponding components.

The following table shows the structure components of data type DTL and their properties:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2262
1			

Byte	Component	Data type	Value range
2	Month	USINT	1 to 12
3	Day	USINT	1 to 31
4	Weekday	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanosecond	UDINT	0 to 999999999
9			
10			
11			

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.7 Character strings

11.5.7.1 CHAR (character)

Description

An operand of data type CHAR has a length of 8 bits and occupies one BYTE in the memory.

The CHAR data type stores a single character in ASCII format. You can find information on the encoding of special characters under "See also > STRING".

The following table shows the value range of the CHAR data type:

Length (bits)	Format	Value range	Example of value inputs
8	ASCII characters	ASCII character set	'A', CHAR#'A'

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

STRING (Page 1937)

11.5.7.2 WCHAR (character)

Description

An operand of data type WCHAR (Wide Characters) has a length of 16 bits and occupies two BYTE in the memory.

The WCHAR data type saves a single character of an expanded character set which is stored in Unicode format. However, only a subset of the entire Unicode range is covered. When a control character is entered, it is represented with a dollar sign.

The following table shows the value range of the WCHAR data type:

Length (bits)	Format	Range of values	Example of value input
16	Unicode	$0000 - $D7FF	WCHAR#'a'

See also

Overview of the valid data types (Page 1908)

11.5.7.3 STRING

Description

An operand of the STRING data type saves several characters in a character string that can consist of up to 254 characters. In a character string, all characters of the ASCII code are permitted. The characters are specified in single quotation marks.

The following table shows the properties of a STRING tag:

Length (bytes)	Format	Value range	Example of value input
n + 2 *	ASCII character string incl. special characters	0 to 254 characters	'Name', STRING#'NAME'
* An operand of the STRING data type occupies two bytes more than the specified maximum length in the memory.			

A character string can also contain special characters. The escape character $ is used to identify control characters, dollar signs and single quotation marks.

The table below shows examples for the notation of special characters:

Character	Hex	Meaning	Example
$L or $l	0A	Line feed	'$LText', '$0AText'
$N	0A and 0D	Line break The line break occupies 2 characters in the character string.	'$NText', '$0A$0DText'
$P or $p	0C	Page feed	'$PText', '$0CText'
$R or $r	0D	Carriage return (CR)	'$RText','$0DText'
$T or $t	09	Tab	'$TText', '$09Text'

Character	Hex	Meaning	Example
$$	24	Dollar sign	'100$$t', '100$26'
$'	27	Single quotation mark	'$'Text$'','$27Text$27'

The maximum length of the character string can be specified during the declaration of an operand using square brackets after the keyword STRING (for example, STRING[4]). If the specification of the maximum length is omitted, the standard length of 254 characters is set for the respective operand.

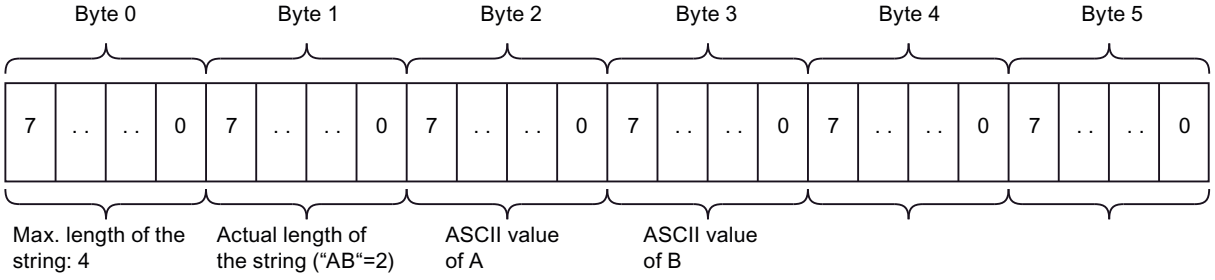
If the actual length of a specified character string is shorter than the declared maximum length, the characters are written to the character string left-justified and the remaining character spaces remain undefined. Only occupied character spaces are considered in the value processing.

Note

For S7-300/400 CPUs, please note: If a temporary tag of the STRING data type was defined, you must describe the BYTE "Max. length of string" with the defined length before you use the tags in the user program.

Example

The example below shows the byte sequence if the STRING[4] data type is specified with output value 'AB':



See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.7.4 WSTRING

Description

An operand of data type WSTRING (Wide String) stores several Unicode characters of data type WCHAR in one character string. If you do not specify a length, the character string has a preset length of 254 characters. In a character string, all characters of the Unicode format are permitted. This means you can also use Chinese characters in a character string.

When declaring an operand of data type WSTRING you can define its length using square brackets (for example WSTRING[10]). If you do not specify a length, the length of the WSTRING is set to 254 characters by default. You can declare a length of up to 16382 characters (WSTRING[16382]).

Note

Use in the watch table

If you wish to monitor or control a tag with data type WSTRING in the watch table, it cannot contain more than 254 characters.

The specification of the characters occurs in single quotes and always with the qualifier WSTRING#.

The table below shows the properties of a WSTRING tag:

Length (WORD)	Format	Range of values	Example of value input
n + 2 *	Unicode character string; n specifies the length of the character string.	Preset value: 0 to 254 characters Max. possible value: 0 to 16382	WSTRING#'Hello World'
* An operand of the WSTRING data type occupies two WORDs more in the memory than the specified maximum length.			

A character string can also contain special characters. The escape character $ is used to identify control characters, dollar signs and single quotation marks.

The table below shows examples for the notation of special characters:

Character	Hex	Meaning	Example
$L or $l	000A	Line feed	'$LText', '$000AText'
$N	000A and 000D	Line break The line break occupies 2 characters in the character string.	'$NText', '$000A $000DText'
$P or $p	000C	Page feed	'$PText', '$000CText'
$R or $r	000D	Carriage return (CR)	'$RText','$000DText'
$T or $t	0009	Tab	'$TText', '$0009Text'
$$	0024	Dollar sign	'100$$t', '100$0024t'
$'	0027	Single quotation mark	'$'Text$'','$0027Text $0027'

The maximum length of the character string can be specified during the declaration of an operand using square brackets after the keyword WSTRING (for example, WSTRING[4]). If the specification of the maximum length is omitted, the standard length of 254 characters is set for the respective operand.

If the actual length of a specified character string is shorter than the declared maximum length, the characters are written to the character string left-justified and the remaining character spaces remain undefined. Only occupied character spaces are considered in the value processing.

Note

Conversion of WSTRING tags

Implicit conversion of the WSTRING data type is not possible. Explicit conversion from the WSTRING data type to STRING is generally possible. However, normally only the conversion of characters in the code range from 0 - 127 will work in all Windows code pages. For all characters outside this range, the code page character and the Unicode character must be in exactly the same position for the conversion to work without errors.

Access to block parameters of data type WSTRING

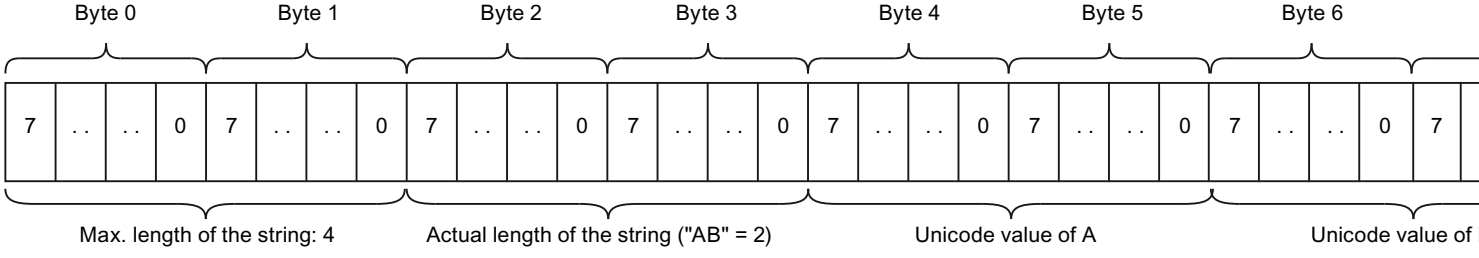
Operands of the data type WSTRING can be transferred as parameters up to the maximum length for blocks with "optimized" access.

For function blocks (FB) with "standard" access, operands of the data type WSTRING can be declared as parameters in all sections of the block interface except in the section "InOut". For a function (FC) with "standard" access, only operands of the data type STRING can be transferred as parameters.

The function value of an FC in the "Return" section and expressions in the SCL programming language are another exception to this rule. In these cases, the WSTRING tag must not be longer than 1022 characters. If you need a WSTRING tag with more than 1022 characters, declare a tag of the data type "WSTRING" with the required character length in the section "Temp" and assign the function value to the tag.

Example

The example below shows the byte sequence if the WSTRING[4] data type is specified with output value 'AB':



See also

Overview of the valid data types (Page 1908)

11.5.8 Array

11.5.8.1 Format of array (16-bit limits)

Description

The Array data type represents a data structure that consists of a fixed number of components of the same data type. All data types except Array are permitted.

A tag with the Array data type always starts at a WORD limit.

The array components are addressed by means of an index. In the array declaration, the index limits are defined in square brackets after the keyword Array. The low limit must be smaller than or equal to the high limit. An array may contain up to six dimensions, the limits of which can be specified separated by a comma.

The following table shows the property of the Array data type:

Length	Format	Index limits	Data type
Number of components * length of the data type	Array [low limit...high limit] of <data type>	[-32768..32767] of <data type>	Bit strings, integers, floating-point numbers, timers, character strings, structures

Example

The following example shows how operands of data type Array can be declared:

Name	Declaration	Comment
Measured value	Array[1..20] of REAL	One-dimensional array with 20 components
Time-of-day	Array[-5..5] of INT	One-dimensional array with 11 components
Character	Array[1..2, 3..4] of CHAR	Two-dimensional array with 4 components

Maximum array limits

The maximum Array limits depend on the following factors:

- Data type of the Array elements
- Maximum storage capacity of the CPU (you can find more information in the relevant device manual)

See also

Overview of the valid data types (Page 1908)

11.5.8.2 Format of array (32-bit limits)

Description

The Array data type represents a data structure that consists of a fixed number of components of the same data type. All data types except Array are permitted.

The array components are addressed by means of an index. In the array declaration, the index limits are defined in square brackets after the keyword Array. The low limit must be smaller than or equal to the high limit. An array may contain up to six dimensions, the limits of which can be specified separated by a comma.

Note

Depending on the CPU, the storage capacity of a data block is limited and the number of components of the Array is therefore also limited. However, you may initialize the addressing of the array components at any position within index limits.

The following table shows the property of the Array data type:

Length	Format	Index limits	Data type
Number of components * length of the data type	Array [low limit...high limit] of <data type>	[-2147483648..2147483647] of <data type>	Bit strings, integers, floating-point numbers, timers, character strings, structures

Note

The length of the array depends on whether the block was created with the "Standard" or "with optimized access" block property.

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", a bit requires 1 byte of memory. This is also true when you use an ARRAY of <data type>. An ARRAY [0..1] of BOOL, for example, thus requires 2 bytes in an optimized block.

Example

The following example shows how operands of data type Array can be declared:

Name	Declaration	Comment
Measured value	Array[1..20] of REAL	One-dimensional array with 20 components
Time-of-day	Array[-5..5] of INT	One-dimensional array with 11 components
Character	Array[1..2, 3..4] of CHAR	Two-dimensional array with 4 components

Maximum array limits

The maximum array limits depend on the following factors:

- Data type of the array elements
- Memory reserve (only in blocks with optimized access)
  You can find additional information on this topic in the section "Loading block changes without reinitialization".
- Maximum size of a data block for a CPU (you can find more information in the respective device manual)
- The entire length of the array is available within a data block. Within a program block (OB, FB or FC), the possible length is reduced by the memory capacity required by the program code.

Example based on a CPU of the S7-1200 series

The following table shows the maximum number of elements within a block with the "with optimized access" block property:

Data type width (bits)	Maximum number of elements	Note
1	524272	= 65534*8
8	65534	Refer to the respective device manual of the CPU for the value.
16	32767	= 65534/2 (integer division, remainder 0)
32	16383	= 65534/4 (integer division, remainder 2)
64	8191	= 65534/8 (integer division, remainder 6)

Due to various technical/internal constraints, the actual usable memory area may be approximately 70 - 100 bytes less. The memory area may be further restricted due to a default setting, for example, by the "Load without reinitialization" block property.

See also

Overview of the valid data types (Page 1908)

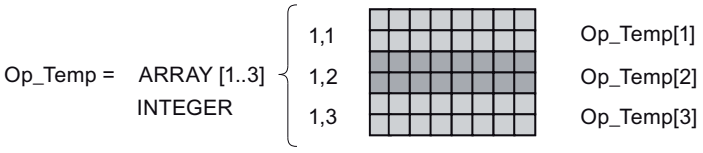
11.5.8.3 Example of a one-dimensional array

Declaration

The following table shows the declaration of a one-dimensional Array tag:

Name	Data type	Comment
Op_Temp	Array[1..3] of INT	One-dimensional array tag with 3 components.

The following figure shows the structure of the declared array tag:



Access to ARRAY components

The individual array components are accessed via an index.. The index of the first ARRAY component is [1], of the second [2], and of the third [3]. To access the value of the second ARRAY component, you need to declare "Op_Temp[2]" in the program.

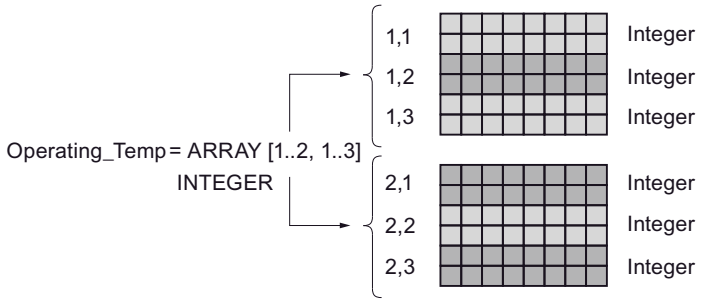
11.5.8.4 Example of a multi-dimensional array

Declaration

The following table shows the declaration of a two-dimensional Array tag:

Name	Data type	Value	Comment
Betr_Temp	Array[1..2, 1..3] of INT	1,1,4(0)	Two-dimensional array tag with 6 components. The first two components are assigned the value "1". The remaining four components are assigned the value "0".

The following figure shows the structure of the declared array tag:



Access to the array components

The values of the individual array components are accessed via an index. The index of the first array component is, for example, [1,1] and the index of the fourth array component is [2,1]. For example, you need to declare "Betr_Temp[2,1]" in the program to enable access to the value of the fourth array component.

Additional access option

You can also declare the "Betr_Temp" TAG as a six-dimensional array. The following table shows an example of the declaration of a six-dimensional Array tag:

Name	Data type	Value	Comment
Betr_Temp	Array[1..3, 1..2, 1..3, 1..4, 1..3, 1..4] of INT	-	Six-dimensional array tag

The index of the first array component is in this case [1,1,1,1,1,1] and the index of the last component is [3,2,3,4,3,4]. Intermediate values are accessed by entering the corresponding value for each dimension.

11.5.9 Structures

11.5.9.1 STRUCT

Description

Data type STRUCT represents a data structure that consists of a fixed number of components of various data types. Components of STRUCT or ARRAY data type can also be nested in a structure. The nesting depth is hereby limited to eight levels. Structures can be used to group data according to the process control system and to transfer parameters as one data unit.

You can create up to 65534 structures for a CPU of the S7-1200 or S7-1500 series. Each of these structures can include up to 252 components. In addition, you can create up to 65534 function blocks, 65535 functions and 65535 organization blocks with up to 252 components each.

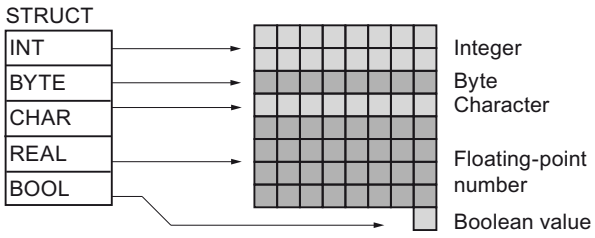
A component of the ARRAY data type always starts at a WORD limit.

The following table shows the properties of data type STRUCT:

Length	Format	Range of values	Example of value input
A STRUCT variable starts with one byte with even address and occupies the memory up to the next word limit.	STRUCT	The value ranges of the used data types apply.	The value input rules of the used data types apply.

Example

The following figure shows an example of the structure of a STRUCT variable:



See also

Overview of the valid data types (Page 1908)

11.5.10 Pointer

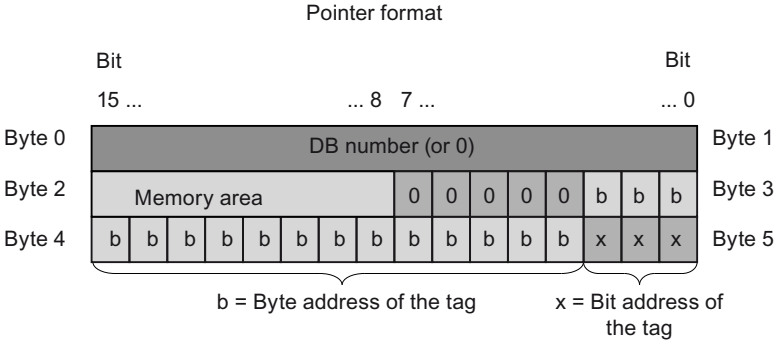
11.5.10.1 POINTER

Description

A parameter of the type POINTER is a pointer that can point to a specific tag. It occupies 6 bytes (48 bits) in memory and may contain the following tag information:

- DB number, or 0 if the data is not stored in a DB
- Memory area in the CPU
- Tag address

The following figure shows the structure of parameter type POINTER:



Types of pointer

Depending on the information, you can use parameter type POINTER to declare the following four types of pointer:

- Area-internal pointer:
  An area-internal pointer contains information on the address of a tag.
- Cross-area pointer:
  A cross-area pointer contains information on the memory area and the address of a tag.
- DB pointer:
  You can use a DB pointer to point to a data block tag. A DB pointer also contains a data block number in addition to the memory area and the address of a tag.
- Zero pointer:
  Use the zero pointer to indicate a missing value. A missing value may indicate that no value exists, or that the value is not yet known. A zero value represents the absence of a value, but is also a value.

The following table shows the formats for the declaration of various pointer types:

P#ByteRepresentation	Format	Example of value input	Description
Symbolic	P#Byte.Bit	"MyTag"	Area-internal pointer
	P#OperandAreaByte.Bit	"MyTag"	Cross-area pointer
	P#Data_block.Data_operand	"MyDB"."MyTag"	DB pointer
	P#Zero value	-	Zero pointer
Absolute	P#Byte.Bit	P#20.0	Area-internal pointer
	P#OperandAreaByte.Bit	P#M20.0	Cross-area pointer
	P#Data_block.Data_operand	P#DB10.DBX20.0	DB pointer
	P#Zero value	P#0.0, ZERO	Zero pointer

You can enter a parameter of the type POINTER without prefix (P#). The entry is then automatically converted to the POINTER format.

Note

If you use the prefix P#, you can only point to memory areas with "standard" access mode.

Memory areas

The following table shows the hexadecimal codes of the memory areas for parameter type POINTER:

Hexadecimal code	Memory area	Description
B#16#80[1)]	P	Peripherals on a CPU S7-300/400
16#1	P	Peripheral inputs on a CPU S7-1500

Hexadecimal code	Memory area	Description
16#2	P	Peripheral outputs on a CPU S7-1500
B#16#81	I	Memory area of inputs
B#16#82	Q	Memory area of outputs
B#16#83	M	Memory area of bit memory
B#16#84	DBX	Data block
B#16#85	DIX	Instance data block
B#16#86	L	Local data
B#16#87	V	Previous local data
[1] These data types can only be used for the POINTER pointer on a CPU S7-300/400.		

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.10.2 ANY

Description

An ANY type parameter points to the start of a data area and specifies its length. An ANY pointer occupies 10 bytes of memory and may contain the following information:

- Data type:
  Data type of the elements of the data area
- Repetition factor:
  Number of elements of the data area
- DB number:
  Data block that contains the declaration of data area elements.
- Memory area:
  Memory area of the CPU that stores the data area elements.
- Start address of the data in the format "byte.bit":
  Data area start identified by the ANY pointer.
- Zero pointer:
  Use the zero pointer to indicate a missing value. A missing value may indicate that no value exists, or that the value is not yet known. A zero value represents the absence of a value, but is also a value.

In the programming languages SCL and STL, memory of any kind can be transferred upon a block call if an ANY pointer has been programmed at a block parameter.

The ANY pointer cannot, however, store any information on the structure of the memory. For example, the ANY pointer does not save the information that it points to a tag of the PLC data type. The ANY pointer sees this as an ARRAY of BYTE.

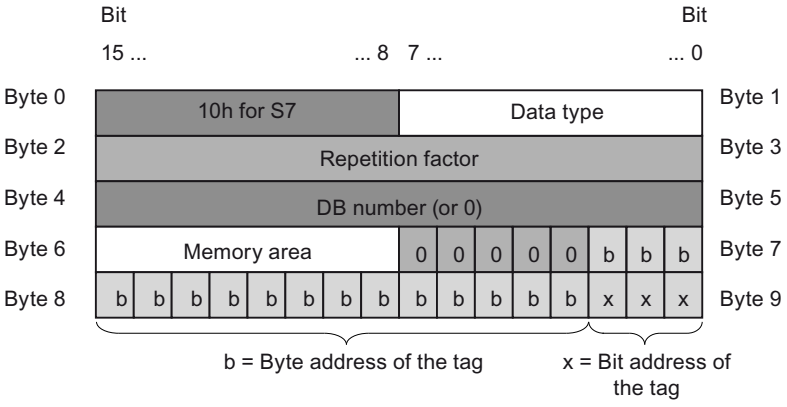
Parameters of the ANY data type can be passed to system function blocks (SFBs) or system functions (SFCs).

Note

Memory area

For an S7-1500 CPU, the ANY pointer can also only point to memory areas with "Standard" access mode.

The following figure shows the structure of the ANY pointer:



An ANY pointer cannot identify structures. It can only be assigned to local tags.

The following table shows the formats for the declaration of an ANY pointer:

Representation	Format	Example of value input	Description
Symbolic	P#DataBlock.MemoryArea DataAddress Type Number	"MyDB".StructTag.InitialComponents	Area with 10 words in global DB11 starting with DBB20.0
	P#MemoryArea DataAddress Type Number	"MyMarkerTag"	Area with 4 bytes starting with MB 20.0
		"MyTag"	Input I1.0
	P#Zero value	-	Zero value
Absolute	P#DataBlock.MemoryArea DataAddress Type Number	P#DB11.DBX20.0 INT 10	Area with 10 words in global DB11 starting with DBB20.0
	P#MemoryArea DataAddress Type Number	P#M20.0 BYTE 10	Area with 10 bytes starting with MB 20.0
		P#I1.0 BOOL 8	Range with 8 bits from input I1.0 (the range length specified must be divisible by 8.)
	P#Zero value	P#P0.0 VOID 0, NULL [1]	Zero value
[1] In the programming languages LAD and FBD, only NULL is a value entry for the zero value.			

Coding of data types

The following table lists the coding of data types for the ANY pointer:

Hexadecimal code	Data type	Description
B#16#00	NIL	Null pointer
B#16#01[1)]	BOOL	Bits
B#16#02	BYTE	bytes, 8 bits
B#16#03	CHAR	8-bit characters
B#16#04	WORD	16-bit words
B#16#05	INT	16-bit integers
B#16#06	DWORD	32-bit words
B#16#07	DINT	32-bit integers
B#16#08	REAL	32-bit floating-point numbers
B#16#0B	TIME	Time duration
B#16#0C	S5TIME	Time duration
B#16#09	DATE	Date
B#16#0A	TOD	Date and time
B#16#0E	DT	Date and time
B#16#13	STRING	Character string
B#16#17[1)]	BLOCK_FB	Function block
B#16#18[1)]	BLOCK_FC	Function
B#16#19[1)]	BLOCK_DB	Data block
B#16#1A[1)]	BLOCK_SDB	System data block
B#16#1C[1)]	COUNTER	Counter
B#16#1D[1)]	TIMER	Timer
[1)] These data types can only be used for the ANY pointer on a CPU S7-300/400.		

Coding of the memory area

The following table lists the coding of the memory areas for the ANY pointer:

Hexadecimal code	Area	Description
B#16#80[1)]	P	I/O
B#16#81	I	Memory area of inputs
B#16#82	Q	Memory area of outputs
B#16#83	M	Memory area of bit memory
B#16#84	DBX	Data block
B#16#85	DIX	Instance data block
B#16#86	L	Local data
B#16#87	V	Previous local data
[1)] These memory areas can only be used for the ANY pointer on an S7-300/400 CPU.		

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

11.5.10.3 VARIANT

Description

A parameter of the VARIANT type is a pointer that can point to tags of different data types other than an instance. The VARIANT pointer can be an object of an elementary data type, such as INT or REAL. It can also be a STRING, DTL, ARRAY of STRUCT, UDT, or ARRAY of UDT. The VARIANT pointer can recognize structures and point to individual structure components. An operand of data type VARIANT occupies no space in the instance data block or work memory. However, it will occupy memory space on the CPU.

A tag of the VARIANT type is not an object but rather a reference to another object. Individual elements of the VARIANT type can only be declared on formal parameters within the block interface of a function in the VAR_IN, VAR_IN_OUT and VAR_TEMP sections. For this reason, it cannot be declared in a data block or in the static section of the block interface of a function block, for example, because its size is unknown. The size of the referenced objects can change.

You can use VARIANT to generate generic function blocks or functions. When a block is called, you can connect the parameters of the block to tags of any data type. When a block is called, the type information of the tag is transferred in addition to a pointer to the tag. The code of the block can then be executed according to its type in line with the tag transferred during runtime.

If, for example, a block parameter of a function has the VARIANT data type, then a tag of the integer data type can be transferred at one point in the program, and a tag of the PLC data type can be transferred at another point in the program. With the help of the VARIANT instructions, the function is then in a position to react to the situation without errors.

Note

You can only point to a complete data block if it was originally derived from a user-defined data type (UDT).

The following table shows the properties of the VARIANT pointer:

Length (bytes)	Repre-sentation	Format	Example of value input
0	Symbolic	Operand	"TagResult"
		NameDataBlock.NameOper-and.Component	"Data_TIA_Portal".StructVariable.First-Component
	Absolute	Operand	%MW10
		DataBlockNumber.Operand Type Length (valid only for blocks with standard access)	P#DB10.DBX10.0 INT 12
		P#Zero value	P#0.0 VOID 0, ZERO

Note

If you use the prefix P#, you can only point to memory areas with "standard" access mode.

Coding of data types

If you use absolute addressing with P#, the following data types are permitted:

Hexadecimal code	Data type	Description
B#16#00	NIL	Null pointer
B#16#01	BOOL	Bits
B#16#02	BYTE	bytes, 8 bits
B#16#03	CHAR	8-bit characters
B#16#04	WORD	16-bit words
B#16#05	INT	16-bit integers
B#16#06	DWORD	32-bit words
B#16#07	DINT	32-bit integers
B#16#08	REAL	32-bit floating-point numbers
B#16#0B	TIME	Time duration
B#16#0C	S5TIME	Time duration
B#16#09	DATE	Date
B#16#0A	TOD	Date and time
B#16#0E	DT	Date and time
B#16#13	STRING	Character string
B#16#17	BLOCK_FB	Function block
B#16#18	BLOCK_FC	Function
B#16#19	BLOCK_DB	Data block
B#16#1A	BLOCK_SDB	System data block
B#16#1C	COUNTER	Counter
B#16#1D	TIMER	Timer

Example

The following example shows how VARIANT works using the "MOVE: Move value" instruction of STL:

STL	Explanation
CALL MOVE	// The "Move value" instruction is called.
VARIANT	// Data type of the instruction
IN := "Data_TIA_Portal".StructVariable.FirstCompo-nent	// The contents of the "FirstComponent" operand are moved from the "Data_TIA_Portal" DB.
OUT := "MotorDB".StructResult.TagResult	// And transferred to the "TagResult" operand from the "MotorDB" DB.

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

VARIANT instructions (Page 263)

11.5.11 Parameter types

11.5.11.1 Parameter types

Description

The parameter types are data types for formal parameters that are transferred to called blocks. A parameter type can also be a PLC data type.

The following table shows the available parameter data types and their purpose:

Parameter type	Length (bits)	Description
TIMER	16	Is used to specify a timer that is used in the called code block. If you supply a formal parameter of the TIMER parameter type, the associated actual parameter must be a timer. Example: T1
COUNTER	16	Is used to specify a counter that is used in the called code block. If you supply a formal parameter of the COUNTER parameter type, the associated actual parameter must be a counter. Example: C10
BLOCK_FC	16	Is used to specify a block that is used as input in the called code block. The declaration of the parameter determines the block type (for example FB, FC, DB) that is to be used. If you supply a formal parameter of the BLOCK parameter type, specify a block address as the actual parameter. Example: DB3
BLOCK_FB	16	
BLOCK_DB	16	
BLOCK_SDB	16	
BLOCK_SFB	16	
BLOCK_SFC	16	
BLOCK_OB	16	
VOID	-	The VOID parameter type does not save any values. This parameter type is used if the return values of an output are not required. The VOID parameter type can be specified at the STATUS output, for example, if no error information is required.

See also

Overview of the valid data types (Page 1908)

11.5.12 PLC data types

11.5.12.1 PLC data types

Description

PLC data types are data structures that you define and that can be used multiple times within the program. The structure of a PLC is made up of several components, each of which can contain different data types. You define the type of components during the declaration of the PLC data type.

You can create up to 65534 PLC data types for a CPU of the S7-1200 or S7-1500 series. Each of these PLC data types can include up to 252 components.

PLC data types can be used for the following applications:

- PLC data types can be used as data types for variables in the variable declaration of logic blocks or in data blocks.
- PLC data types can be used as templates for the creation of global data blocks with identical data structures.
- PLC data types can be used in S7-1200 and S7-1500 as a template for the creation of structured PLC tags.

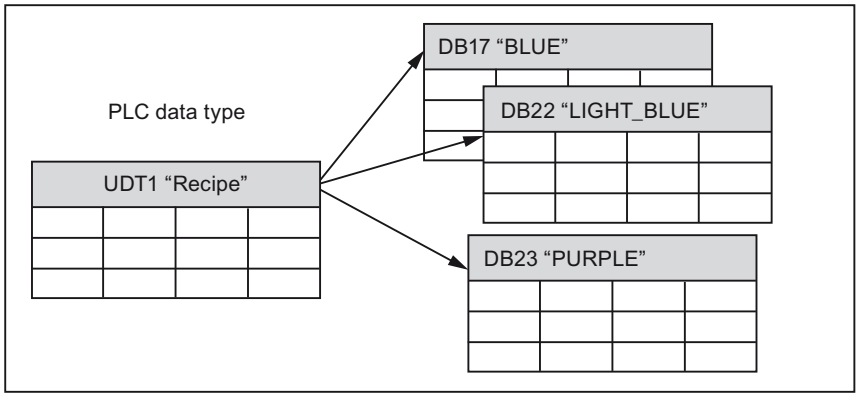
11.5.12.2 Example of a PLC data type

Example

You can declare PLC data types as the type when creating data blocks. Based on this type you can create a number of data blocks, all of which have the same data structure. These data blocks can be adapted by entering different actual values for the corresponding task.

For instance, create a PLC data type for a recipe for blending paints. You can then assign this data type to several data blocks, each of which contains other quantity information.

The following figure shows this application:



11.5.13 System data types

11.5.13.1 System data types

Description

The system data types (SDT) are made available by the system and have a predefined structure. The structure of a system data type consists of a fixed number of components that can have various data types. It is not possible to change the structure of a system data type.

The system data types can only be used for specific instructions. The following table shows the available system data types and their purpose:

System data type	Length (bytes)	Description
IEC_TIMER	16	Structure of a timer whose timer values are of TIME data type. This data type is used for the "TP", "TOF", "TON", "TONR", "RT" and "PT" instructions, for example.
IEC_LTIMER	32	Structure of a timer whose timer values are of LTIME data type. This data type is used for the "TP", "TOF", "TON", "TONR", "RT" and "PT" instructions, for example.
IEC_SCOUNTER	3	Structure of a counter whose count values are of SINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_USCOUNTER	3	Structure of a counter whose count values are of USINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_COUNTER	6	Structure of a counter whose count values are of INT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_UCOUNTER	6	Structure of a counter whose count values are of UINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_DCOUNTER	12	Structure of a counter whose count values are of DINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_UDCOUNTER	12	Structure of a counter whose count values are of UDINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_LCOUNTER	24	Structure of a counter with count values of data type UDINT. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.

System data type	Length (bytes)	Description
IEC_ULCOUNTER	24	Structure of a counter with count values of data type UINT. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
ERROR_STRUCT	28	Structure of an error information to a programming or I/O access error. This data type is used, for example, for the "GET_ERROR" instruction.
CREF	8	Components of the ERROR_STRUCT data type, in which information about the address of a block is saved.
NREF	8	Components of the ERROR_STRUCT data type, in which information about the address of an operand is saved.
VREF	12	Is used for storage of a VARIANT pointer. This data type is, for example, used for instructions from S7-1200 Motion Control.
STARTINFO	12	Specifies the data structure in which the start information is saved. This data type is used, for example, for the "RD_SINFO" instruction.
SSL_HEADER	4	Specifies the data structure in which information about the data records are saved during the reading of the system status lists. This data type is used, for example, for the "RDSYSST" instruction.
CONDITIONS	52	User-defined data structure defining the conditions for start and end of a data reception. This data type is used, for example, for the "RCV_CFG" instruction.
TADDR_Param	8	Specifies the structure of a data block which stores descriptions of connections for Open User Communication via UDP. This data type is used for the "TUSEND" and "TURSV" instructions, for example.
TCON_Param	64	Specifies the structure of a data block which stores descriptions of connections for Open User Communication via Industrial Ethernet (PROFINET). This data type is used for the "TSEND" and "TRSV" instructions, for example.
HSC_Period	12	Specifies the structure of the data block for the time period measurement with the extended high-speed counter. This data type is used, for example, for the "CTRL_HSC_EXT" instruction.

See also

Overview of the valid data types (Page 1908)

11.5.14 Hardware data types

11.5.14.1 Hardware data types

Description

The hardware data types are made available by the CPU. The number of available hardware data types depends on the CPU.

Constants of a specific hardware data type are stored based on the modules set in the hardware configuration. When an instruction for controlling or activating a configured module is inserted in the user program, the available constants can be used for the parameters.

The following table shows the available hardware data types and their purpose:

Data type	Basic data type	Description
REMOTE	ANY	Serves to specify the address of a remote CPU. This data type is used, for example, for the "PUT" and "GET" instructions.
GEOADDR	HW_IOSYSTEM	Geographical address information
HW_ANY	WORD	Identification of any hardware component, e.g. a module.
HW_DEVICE	HW_ANY	Identification of a DP slave/PROFINET IO device
HW_DPMASTER	HW_INTERFACE	Identification of a DP master
HW_DPSLAVE	HW_DEVICE	Identification of a DP slave
HW_IO	HW_ANY	Identification number of the CPU or the interface The number is automatically allocated and is stored in the properties of the CPU or of the interface in the hardware configuration.
HW_IOSYSTEM	HW_ANY	Identification of a PN/IO system or DP master system
HW_SUBMODULE	HW_IO	Identification of a central hardware component
HW_MODULE	HW_IO	Identification of a module
HW_INTERFACE	HW_SUBMODULE	Identification of an interface component
HW_IEPORT	HW_SUBMODULE	Identification of a port (PN/IO)
HW_HSC	HW_SUBMODULE	Identification of a high-speed counter This data type is used, for example, for the "CTRL_HSC" and "CTRL_HSC_EXT" instructions.
HW_PWM	HW_SUBMODULE	Identification of a pulse width modulation This data type is used, for example, for the "CTRL_PWM" instruction.
HW_PTO	HW_SUBMODULE	Identification of a pulse encoder This data type is used for Motion Control.
AOM_AID	DWORD	Is used only in connection with a system function block.
AOM_IDENT	DWORD	Identification of an object in the runtime system of the AS
EVENT_ANY	AOM_IDENT	Used to identify any event

Data type	Basic data type	Description
EVENT_ATT	EVENT_ANY	Is used to specify an event that can be assigned dynamically to an OB This data type is used, for example, for the "ATTACH" and "DETACH" instructions.
EVENT_HWINT	EVENT_ATT	Is used to specify a hardware interrupt event
OB_ANY	INT	Serves to specify any organization block.
OB_DELAY	OB_ANY	Used to specify an organization block that is called when a time-delay interrupt occurs. This data type is used, for example, for the "SRT_DINT" and "CAN_DINT" instructions.
OB_TOD	OB_ANY	Specifies the number of a time-of-day interrupt OB. This data type is used, for example, for the "SET_TINT", "CAN_TINT", "ACT_TINT" and "QRY_TINT" instructions.
OB_CYCLIC	OB_ANY	Is used to specify an organization block that is called when a watchdog interrupt occurs.
OB_ATT	OB_ANY	Is used to specify an organization block that can be assigned dynamically to an event. This data type is used, for example, for the "ATTACH" and "DETACH" instructions.
OB_PCYCLE	OB_ANY	Is used to specify an organization block that can be assigned to an event of the "Cyclic program" event class.
OB_HWINT	OB_ATT	Is used to specify an organization block that is called when a hardware interrupt occurs.
OB_DIAG	OB_ANY	Is used to specify an organization block that is called when a diagnostic interrupt occurs.
OB_TIMEERROR	OB_ANY	Is used to specify an organization block that is called when a time error occurs.
OB_STARTUP	OB_ANY	Is used to specify an organization block that is called when a startup event occurs.
PORT	HW_SUBMODULE	Serves to specify a communication port. This data type is used for point-to-point communication.
RTM	UINT	Serves to specify the number of an operating hours counter. This data type is used, for example, for the "RTM" instruction.
PIP	UINT	Is used to create and connect a "Synchronous Cycle" OB. This data type is used for the SFCs 26, 27, 126 and 127.
CONN_ANY	WORD	Serves to specify any connection.
CONN_PRG	CONN_ANY	Serves to specify a connection for open communication over UDP.
CONN_OUC	CONN_ANY	Used to specify a connection for open communication over Industrial Ethernet (PROFINET).
CONN_R_ID	DWORD	Data type for the R_ID parameter on the S7 communication blocks.

Data type	Basic data type	Description
DB_ANY	UINT	Identification (number) of any DB The data type "DB_ANY" has the length 0 in the section "Temp".
DB_WWW	DB_ANY	Number of a DB generated via the Web application (for example, "WWW" instruction) The data type "DB_WWW" has the length 0 in the section "Temp".
DB_DYN	DB_ANY	Number of a DB generated by the user program

See also

Overview of the valid data types (Page 1908)

11.5.15 Data type conversion for S7-1500:

11.5.15.1 Overview of data type conversion

Introduction

If you combine several operands in an instruction, you must make sure that the data types are compatible. This also applies when you assign or supply block parameters. If the operands are not of the same data type, a conversion has to be carried out.

There are two options for conversion:

- Implicit conversion
  Implicit conversion is supported by the programming languages LAD, FBD, SCL and GRAPH. Implicit conversion is not possible in the STL programming language.
- Explicit conversion
  You use an explicit conversion instruction before the actual instruction is executed.

Note

Converting bit strings to SCL

All bit strings (BYTE, WORD, DWORD and LWORD) are handled like the corresponding unsigned integers (USINT, UINT, UDINT and ULINT) in expressions. Therefore, implicit conversion from DWORD to REAL is carried out like a conversion from UDINT to REAL, for example.

Implicit conversion

Implicit conversion is executed automatically if the data types of the operands are compatible. This compatibility test can be carried out according to strict or less strict criteria:

- With IEC check (default setting)
  If IEC check is set, the following rules are applied:
  - Implicit conversion of BOOL into other data types is not possible.
  - Only the data types REAL, BYTE, WORD, DWORD, DINT, INT, SINT, UDINT, UINT, USINT, TIME, LDT, DTL, DT, TOD, WCHAR and CHAR can be converted implicitly.
  - The bit length of the source data type may not exceed the bit length of the destination data type. A WORD data type operand, for example, cannot be specified at a parameter at which the BYTE data type is expected.
- Without IEC check
  If IEC check is not set, the following rules are applied:
  - Implicit conversion of BOOL into other data types is not possible.
  - Only the data types REAL, LREAL, BYTE, WORD, DWORD, LWORD, SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, TIME, LTIME, S5TIME, LDT, DTL, TOD, LTOD, DATE, STRING, WSTRING, WCHAR and CHAR can be converted implicitly.
  - The bit length of the source data type may not exceed the bit length of the destination data type. A DWORD data type operand, for example, cannot be specified at a parameter at which the WORD data type is expected.
  - The bit length of an operand entered at in-out parameters (InOut) must be the same as the programmed bit length for the parameter in question.

  Note

  Implicit conversion without IEC check

  The programming editor uses a gray rectangle to mark operands that are implicitly converted. The dark gray rectangle signals that an implicit conversion is possible without any accuracy loss, for example, if you convert the data type INT to DINT. A light gray rectangle signals that implicit conversion is possible, but errors could occur during runtime. If, for example, you are converting the data type LINT to DINT and an overflow occurs, the enable output ENO is set to "0".

For more information about the setting of the IEC check and the implicit conversion, refer to "See also".

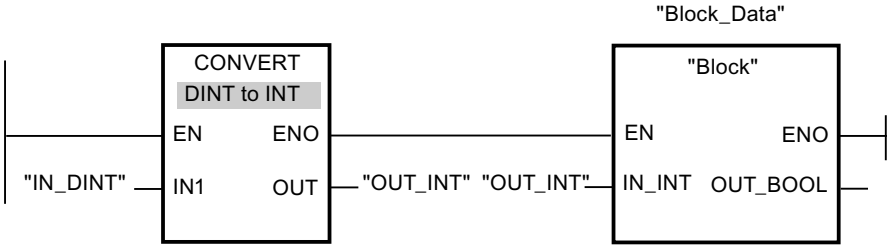
Explicit conversion

If the operands are not compatible and implicit conversion is therefore not possible, you can perform an explicit conversion. You can do this by using the conversion instructions in the "Instructions" Task Card or you can insert the conversion into the program manually. You can find the format for explicit conversion functions under "See also".

Any overflow will be displayed on the ENO enable output. An overflow occurs, for example, if the value of the source data type is greater than the value of the destination data type.

You can find additional information on explicit conversion under "See also".

The following figure shows an example in which explicit data type conversion must be carried out:



The "Block" function block expects a tag of the INT data type at the "IN_INT" input parameter. The value of the "IN_DINT" tag has therefore to the converted first from DINT to INT. Conversion is performed if the value of the "IN_DINT" tag is within the admissible value range for the INT data type. Otherwise an overflow is reported. However, a conversion takes place even in the event of an overflow, but the values are truncated and the ENO enable output is set to "0".

See also

Implicit conversion (Page 1961)

Explicit conversion (Page 2010)

11.5.15.2 Implicit conversion

Setting and disabling the IEC check

The data types of the operands used are checked for compatibility. This compatibility test can be carried out according to criteria that are more or less strict. If "IEC check" is activated, stricter criteria are applied.

You can set the IEC check centrally for all new blocks of the project or for individual blocks.

Setting IEC check for new blocks

To set the IEC check for all new blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. Select or clear the "IEC check" check box in the "Default settings for new blocks" group.
   The IEC check is enabled or disabled for all new blocks in the program.

Setting IEC check for a block

To set the IEC check for a block, proceed as follows:

1. Open the block.
2. Open the "Properties" tab in the inspector window.
3. Select the "Attributes" group in the area navigation.
4. Select or clear the "IEC check" check box.
   The IEC check is enabled or disabled for this block. The setting is stored together with the project.

Binary numbers

Implicit conversion of BOOL

Implicit conversion options

The BOOL data type cannot be implicitly converted.

See also

BOOL (bit) (Page 1912)

Overview of data type conversion (Page 1959)

Setting and disabling the IEC check (Page 1961)

Explicit conversion of BOOL (Page 2010)

Bit strings

Implicit conversion of BYTE

Implicit conversion options

The following table shows the options for the implicit conversion of BYTE data type:

Source	Destination	With IEC check	Without IEC check	Explanation
BYTE	BOOL	-	-	No implicit conversion
	WORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	DWORD	X	X	
	LWORD	X	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

BYTE (byte) (Page 1913)

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)
Explicit conversion of BYTE (Page 2012)

Implicit conversion of WORD

Implicit conversion options

The following table shows the options for the implicit conversion of WORD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
WORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The low byte is transferred to the destination data type; the high byte is ignored.
	DWORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LWORD	X	X	
	SINT	-	X	The low byte is transferred to the destination data type; the high byte is ignored.
	USINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LDT	-	-	No implicit conversion
	DTL	-	-	
	DT	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	TOD	-	-	No implicit conversion
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

WORD (Page 1914)
Setting and disabling the IEC check (Page 1961)
Overview of data type conversion (Page 1959)
Explicit conversion of WORD (Page 2015)

Implicit conversion of DWORD

Implicit conversion options

The following table shows the options for the implicit conversion of DWORD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The right byte is transferred to the destination data type; the left byte is ignored.
	WORD	-	X	
	LWORD	X	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	-	-	No implicit conversion
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

DWORD (Page 1915)

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)
Explicit conversion of DWORD (Page 2018)

Implicit conversion of LWORD

Implicit conversion options

The following table shows the options for the implicit conversion of LWORD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The right byte is transferred to the destination data type; the left byte is ignored.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	-	No implicit conversion
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	S5TIME	-	-	No implicit conversion
	LDT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

LWORD (Page 1915)

Explicit conversion of LWORD (Page 2022)

Integers

Implicit conversion of SINT

Implicit conversion options

The following table shows the options for the implicit conversion of SINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
SINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	USINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value transfer from SINT #-1 -> INT #-1, not filled with "0".)
	INT	X	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	LINT	X	X	
	ULINT	-	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

SINT (8-bit integers) (Page 1917)

Explicit conversion of SINT (Page 2026)

Implicit conversion of USINT

Implicit conversion options

The following table shows the options for the implicit conversion of USINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
USINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from USINT #10 -> DINT #10 or USINT #128 -> SINT #-128)
	INT	X	X	
	UINT	X	X	
	DINT	X	X	
	UDINT	X	X	
	LINT	X	X	
	ULINT	X	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

USINT (8-bit integers) (Page 1918)

Explicit conversion of USINT (Page 2029)

Implicit conversion of INT

Implicit conversion options

The following table shows the options for the implicit conversion of INT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
INT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from INT #-1 -> SINT #-1, or INT #-32 767 -> UINT #32 769)
	USINT	-	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	LINT	X	X	
	ULINT	-	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	TOD	-	-	No implicit conversion
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

INT (16-bit integers) (Page 1918)

Explicit conversion of INT (Page 2032)

Implicit conversion of UINT

Implicit conversion options

The following table shows the options for the implicit conversion of UINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
UINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from UINT #100 -> DINT #100 or UINT #60 000 -> INT #-5536)
	USINT	-	X	
	INT	-	X	
	DINT	X	X	
	UDINT	X	X	
	LINT	X	X	
	ULINT	X	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	TOD	-	-	No implicit conversion
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

UINT (16-bit integers) (Page 1919)

Explicit conversion of UINT (Page 2036)

Implicit conversion of DINT

Implicit conversion options

The following table shows the options for the implicit conversion of DINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	UDINT	-	X	
	LINT	X	X	
	ULINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

DINT (32-bit integers) (Page 1920)

Explicit conversion of DINT (Page 2039)

Implicit conversion of UDINT

Implicit conversion options

The following table shows the options for the implicit conversion of UDINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
UDINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	LINT	X	X	
	ULINT	X	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

UDINT (32-bit integers) (Page 1921)

Explicit conversion of UDINT (Page 2043)

Implicit conversion of LINT

Implicit conversion options

The following table shows the options for the implicit conversion of LINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> SINT #-1 or LINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	ULINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 9 007 199 254 740 992)
	TIME	-	-	No implicit conversion
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (duration in nanoseconds)
	S5TIME	-	-	No implicit conversion
	LDT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

LINT (64-bit integers) (Page 1922)

Explicit conversion of LINT (Page 2046)

Implicit conversion of ULINT

Implicit conversion options

The following table shows the options for the implicit conversion of ULINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
ULINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from ULINT #-1 -> SINT #-1 or ULINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 9 007 199 254 740 992)
	TIME	-	-	No implicit conversion
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (duration in nanoseconds)
	S5TIME	-	-	No implicit conversion
	LDT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

ULINT (64-bit integers) (Page 1923)

Explicit conversion of ULINT (Page 2049)

Floating-point numbers

Implicit conversion of REAL

Implicit conversion options

The following table shows the options for the implicit conversion of REAL data type:

Source	Destination	With IEC check	Without IEC check	Explanation
REAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	LWORD	-	-	No implicit conversion
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the destination data type. (For example, rounding off and value conversion of REAL #2.5 -> INT #2, or negative numbers REAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined REAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	LREAL	X	X	The value is transferred to the destination data type.
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

REAL (Page 1925)

Explicit conversion of REAL (Page 2052)

Implicit conversion of LREAL

Implicit conversion options

The following table shows the options for the implicit conversion of LREAL data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LREAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the destination data type. (For example, rounding off and value conversion of LREAL #2.5 -> INT #2, or negative numbers LREAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined LREAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	X	The value is transferred to the destination data type.
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

LREAL (Page 1926)

Explicit conversion of LREAL (Page 2055)

Timers

Implicit conversion of S5TIME

Implicit conversion options

The following table shows the options for the implicit conversion of data type S5TIME:

Source	Destination	With IEC check	Without IEC check	Explanation
S5TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	DWORD	-	-	No implicit conversion
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	

x: Conversion possible

-: Conversion not possible

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

S5TIME (duration) (Page 1929)

Explicit conversion of S5TIME (Page 2058)

Implicit conversion of TIME

Implicit conversion options

The following table shows the options for the implicit conversion of TIME data type:

Source	Destination	With IEC check	Without IEC check	Explanation
TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	LWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	UDINT	-	X	
	LINT	-	-	No implicit conversion
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	S5TIME	-	-	
	LTIME	X	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds. (1 ms = 1 000 000 ns)
	LDT	-	-	No implicit conversion
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	If the source value is between 0 s and 84599.999 s, the bit pattern of the source value is transferred unchanged to the destination data type. (Display in nanoseconds) The target value remains unchanged otherwise. The result of the conversion shows the time that has passed since midnight.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

TIME (IEC time) (Page 1930)

Explicit conversion of TIME (Page 2060)

Implicit conversion of LTIME

Implicit conversion options

The following table shows the options for the implicit conversion of LTIME data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LTIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds.
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds.
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	S5TIME	-	-	
	TIME	-	X	If the source value is outside the value range of the destination data type, the target value remains unchanged. (0.123456789 s becomes 0.123 s)
	LDT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds since 1/1/1970.
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds since 1/1/1970.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

LTIME (IEC time) (Page 1931)

Explicit conversion of LTIME (Page 2063)

Date and time-of-day

Implicit conversion of DT

Implicit conversion options

The following table shows the options for the implicit conversion of DT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	X	X	The source value is transferred unchanged with the same value to the destination data type. (12/24/2012 14:30 remains 12/24/2012 14:30)
	DTL	X	X	
	DATE	-	-	No implicit conversion
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

DATE_AND_TIME (date and time of day) (Page 1933)

Explicit conversion of DT (Page 2066)

Implicit conversion of LDT

Implicit conversion options

The following table shows the options for the implicit conversion of LDT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LDT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	S5TIME	-	-	No implicit conversion
	DT	-	X	The bit pattern of the source value is converted with a loss in accuracy to the destination data type. (12/24/2012 12:34:56.123456789 becomes 12/24/2012 12:34:56.123)
	DTL	X	X	The source value is transferred unchanged with the same value to the destination data type. (12/24/2012 14:30 remains 12/24/2012 14:30)
	DATE	-	-	No implicit conversion
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

Explicit conversion of LDT (Page 2069)

LDT (DATE_AND_LTIME) (Page 1934)

Implicit conversion of DTL

Implicit conversion options

The following table shows the options for the implicit conversion of DTL data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DTL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	X	X	The source value is transferred unchanged with the same value to the destination data type. (12/24/2012 14:30 remains 12/24/2012 14:30)
	DT	-	X	The bit pattern of the source value is converted with a loss in accuracy to the destination data type. (12/24/2012 12:34:56.123456789 becomes 12/24/2012 12:34:56.123)
	DATE	-	-	No implicit conversion
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

DTL (Page 1935)

Explicit conversion of DTL (Page 2072)

Implicit conversion of DATE

Implicit conversion options

The following table shows the options for the implicit conversion of data type DATE:

Source	Destination	With IEC check	Without IEC check	Explanation
DATE	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion is equal to the number of days since 01/01/1990.
	DWORD	-	-	No implicit conversion
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion is equal to the number of days since 01/01/1990.
	UINT	-	X	
	DINT	-	-	No implicit conversion
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DT	-	-	
	DTL	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	

x: Conversion possible
-: Conversion not possible

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

DATE (Page 1931)

Explicit conversion of DATE (Page 2075)

Implicit conversion of TOD

Implicit conversion options

The following table shows the options for the implicit conversion of TOD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
TOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	LWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	UDINT	-	X	
	LINT	-	-	No implicit conversion
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DT	-	-	
	DTL	-	-	
	DATE	-	-	
	LTOD	X	X	The source value is transferred unchanged with the same value to the destination data type. (12:34:56.123 becomes 12:34:56.123000000)
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	

x: Conversion possible

-: Conversion not possible

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

TIME_OF_DAY (TOD) (Page 1932)

Explicit conversion of TOD (Page 2077)

Implicit conversion of LTOD

Implicit conversion options

The following table shows the options for the implicit conversion of LTOD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LTOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of nanoseconds since the start of day (0:00 hrs).
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of nanoseconds since the start of day (0:00 hrs).
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of nanoseconds since the start of day (0:00 hrs).
	S5TIME	-	-	No implicit conversion
	LDT	-	-	
	DT	-	-	
	DTL	-	-	
	DATE	-	-	
	TOD	-	X	The source value is transferred unchanged with the same value rounded off to the destination data type. (12:34:56.123456789 becomes 12:34:56.123)
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	

x: Conversion possible

-: Conversion not possible

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

Explicit conversion of LTOD (Page 2079)

LTOD (LTIME_OF_DAY) (Page 1932)

String

Implicit conversion of CHAR

Implicit conversion options

The following table shows the options for the implicit conversion of CHAR data type:

Source	Destination	With IEC check	Without IEC check	Explanation
CHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	WCHAR	-	-	
	STRING	X	X	The STRING is shortened to length 1 and includes the character.
	WSTRING	-	-	No implicit conversion
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

CHAR (character) (Page 1936)

Explicit conversion of CHAR (Page 2082)

Implicit conversion of WCHAR

Implicit conversion options

The following table shows the options for the implicit conversion of WCHAR data type:

Source	Destination	With IEC check	Without IEC check	Explanation
WCHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	CHAR	-	-	
	STRING	-	-	
	WSTRING	X	X	The WSTRING is shortened to length 1 and includes the character.
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

WCHAR (character) (Page 1937)

Explicit conversion of WCHAR (Page 2084)

Implicit conversion of STRING

Implicit conversion options

The following table shows the options for the implicit conversion of STRING data type:

Source	Destination	With IEC check	Without IEC check	Explanation
STRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	CHAR	-	X	The first character of the STRING is returned if the STRING includes one or more characters. Otherwise, the character is output with coding $00.
	WCHAR	-	-	No implicit conversion
	WSTRING	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

STRING (Page 1937)

Explicit conversion of STRING (Page 2086)

Implicit conversion of WSTRING

Implicit conversion options

The following table shows the options for the implicit conversion of WSTRING data type:

Source	Destination	With IEC check	Without IEC check	Explanation
WSTRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	CHAR	-	-	
	WCHAR	-	X	The first character of the WSTRING is returned if the WSTRING includes one or more characters. Otherwise, the character is output with coding $0000.
	STRING	-	-	No implicit conversion

x: Conversion possible

-: Conversion not possible

See also

Setting and disabling the IEC check (Page 1961)

Overview of data type conversion (Page 1959)

WSTRING (Page 1939)

Explicit conversion of WSTRING (Page 2089)

11.5.15.3 Explicit conversion

Binary numbers

Explicit conversion of BOOL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BOOL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
BOOL	BYTE	X	Only the LSB (Least Significant Bit) is set in the destination data type. The enable output ENO is always "1".	BOOL_TO_BYTE
	WORD	X		BOOL_TO_WORD
	DWORD	X		BOOL_TO_DWORD
	LWORD	X		BOOL_TO_LWORD
	SINT	X		BOOL_TO_SINT
	USINT	X		BOOL_TO_USINT
	INT	X		BOOL_TO_INT
	UINT	X		BOOL_TO_UINT
	DINT	X		BOOL_TO_DINT
	UDINT	X		BOOL_TO_UDINT
	LINT	X		BOOL_TO_LINT
	ULINT	X		BOOL_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	LTIME	-		-
	S5TIME	-		-
	LDT	-		-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

Overview of data type conversion (Page 1959)

Implicit conversion of BOOL (Page 1962)

BOOL (bit) (Page 1912)

Bit strings

Explicit conversion of BYTE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BYTE data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
BYTE[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	BYTE_TO_BOOL
	WORD[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_WORD
	DWORD[1)]	X		BYTE_TO_DWORD
	LWORD[1)]	X		BYTE_TO_LWORD
	SINT	X		BYTE_TO_SINT
	USINT	X		BYTE_TO_USINT
	INT	X		BYTE_TO_INT
	UINT	X		BYTE_TO_UINT
	DINT	X		BYTE_TO_DINT
	UDINT	X		BYTE_TO_UDINT
	LINT	X		BYTE_TO_LINT
	ULINT	X		BYTE_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_TIME
	LTIME	X		BYTE_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_TOD
	LTOD	X		BYTE_TO_LTOD
	DATE	X		BYTE_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_CHAR
	WCHAR	X		BYTE_TO_WCHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.				

See also

Overview of data type conversion (Page 1959)

Implicit conversion of BYTE (Page 1963)

BYTE (byte) (Page 1913)

Explicit conversion of WORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WORD[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	WORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_BYTE
	DWORD[1)]	X		WORD_TO_DWORD
	LWORD[1)]	X		WORD_TO_LWORD

	SINT	X	ENO = TRUE
			#sint1 := WORD_TO_SINT(16#FFFF); // -1 to
			#sint1 := WORD_TO_SINT(16#FF80); // -128
			#sint1 := WORD_TO_SINT(16#0); // 0 to
			#sint1 := WORD_TO_SINT(16#007F); // 127
			
			ENO = FALSE
			#sint1 := WORD_TO_SINT(16#FF7F); // -129 to
			#sint1 := WORD_TO_SINT(16#8000); // -32768
			#sint1 := WORD_TO_SINT(16#0080); // 128 to
			#sint1 := WORD_TO_SINT(16#7FFF); // 32767
				WORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_USINT
	INT	X		WORD_TO_INT
	UINT	X		WORD_TO_UINT
	DINT	X		WORD_TO_DINT
	UDINT	X		WORD_TO_UDINT
	LINT	X		WORD_TO_LINT
	ULINT	X		WORD_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_TIME
	LTIME	X		WORD_TO_LTIME
	S5TIME	X		WORD_TO_S5TIME
	LDT	X		WORD_TO_LDT
	DT	-	No explicit conversion	-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_TOD
	LTOD	X		WORD_TO_LTOD
	DATE	X		WORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_CHAR
	WCHAR	X		WORD_TO_WCHAR
WORD_ BCD16	INT	X	The value to be converted has data type WORD and is accepted as a BCD-coded value between -999 and +999. The result is available after conversion as an integer (in binary notation) of the type INT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	WORD_BCD16_TO_INT
BCD16	INT	X		BCD16_TO_INT

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.

See also

Implicit conversion of WORD (Page 1964)

Overview of data type conversion (Page 1959)

WORD (Page 1914)

Explicit conversion of DWORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DWORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DWORD[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	DWORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_BYTE
	WORD[1)]	X		DWORD_TO_WORD
	LWORD[1)]	X		DWORD_TO_LWORD

	SINT	X	ENO = TRUE
			#sint1 := DWORD_TO_SINT(16#FFFF_FFFF); // -1 to
			#sint1 := DWORD_TO_SINT(16#FFFF_FF80); // -128
			#sint1 := DWORD_TO_SINT(16#0); // 0 to
			#sint1 := DWORD_TO_SINT(16#0000_007F); // 127
			
			ENO = FALSE
			#sint1 := DWORD_TO_SINT(16#FFFF_FF7F); // -129
			#sint1 := DWORD_TO_SINT(16#8000_0000); // -2147483648
			#sint1 := DWORD_TO_SINT(16#0000_0080); // 128 to
			#sint1 := DWORD_TO_SINT(16#7FFF_FFFF); // 2147483647
				DWORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_USINT

	INT	X	ENO = TRUE
			#int1 := DWORD_TO_INT(16#FFFF_FFFF); // -1 to
			#int1 := DWORD_TO_INT(16#FFFF_8000); // -32768
			#int1 := DWORD_TO_INT(16#0); // 0 to
			#int1 := DWORD_TO_INT(16#0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := DWORD_TO_INT(16#FFFF_7FFF); // -32769
			#int1 := DWORD_TO_INT(16#8000_0000); //
				DWORD_TO_INT

Source	Destination	Conversion	Explanation	Mnemonics of the instruction

			-2147483648
			#int1 := DWORD_TO_INT(16#8000); // 32768 to
			#int1 := DWORD_TO_INT(16#7FFF_FFFF); // 2147483647	

	UINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If no errors occur during the conversion, the signal state of ENO = 1; if an error occurs during processing, the signal state of ENO = 0.	DWORD_TO_UINT
	DINT	X		DWORD_TO_DINT
	UDINT	X		DWORD_TO_UDINT
	LINT	X		DWORD_TO_LINT
	ULINT	X		DWORD_TO_ULINT
	REAL	X		DWORD_TO_REAL
	LREAL	-	No explicit conversion	-
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_TIME
	LTIME	X		DWORD_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_TOD
	LTOD	X		DWORD_TO_LTOD
	DATE	X		DWORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_CHAR
	WCHAR	X		DWORD_TO_WCHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DWORD_BCD32	DINT	X	The value to be converted has data type DWORD and is accepted as a BCD-coded value between -9999999 and +9999999. The result is available after conversion as an integer (in binary notation) of the type DINT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	DWORD_BCD32_TO_DINT
BCD32	DINT	X		BCD32_TO_DINT

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.

See also

Implicit conversion of DWORD (Page 1966)

Overview of data type conversion (Page 1959)

DWORD (Page 1915)

Explicit conversion of LWORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LWORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LWORD[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	LWORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_BYTE
	WORD[1)]	X		LWORD_TO_WORD
	DWORD[1)]	X		LWORD_TO_DWORD

	SINT	X	ENO = TRUE
			#sint1 := LWORD_TO_SINT(16#FFFF_FFFF_FFFF_FFFF); // -1 to
			#sint1 := LWORD_TO_SINT(16#FFFF_FFFF_FFFF_FF80); // -128
			#sint1 := LWORD_TO_SINT(16#0); // 0 to
			#sint1 := LWORD_TO_SINT(16#0000_0000_0000_007F); // 127
			
			ENO = FALSE
			#sint1 := LWORD_TO_SINT(16#FFFF_FFFF_FFFF_FF7F); // -129
			#sint1 := LWORD_TO_SINT(16#8000_0000_0000_0000); // -9223372036854775808
			#sint1 := LWORD_TO_SINT(16#0000_0000_0000_0080); // 128
			#sint1 := LWORD_TO_SINT(16#7FFF_FFFF_FFFF_FFFF); // 9223372036854775807
				LWORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_USINT

	INT	X	ENO = TRUE
			#int1 := LWORD_TO_INT(16#FFFF_FFFF_FFFF_FFFF); // -1 to
			#int1 := LWORD_TO_INT(16#FFFF_FFFF_FFFF_8000); // -32768
			#int1 := LWORD_TO_INT(16#0); // 0 to
			#int1 := LWORD_TO_INT(16#0000_0000_0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := LWORD_TO_INT(16#FFFF_FFFF_FFFF_7FFF); // -32769
			#int1 := LWORD_TO_INT(16#8000_0000_0000_0000); // -2147483648
				LWORD_TO_INT

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			#int1 := LWORD_TO_INT(16#0000_0000_0000_8000); // 32768 to #int1 := LWORD_TO_INT(16#7FFF_FFFF_FFFF_FFFF); // 2147483647	
	UINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_UINT

	DINT	X	ENO = TRUE
			#dint1 := LWORD_TO_DINT(16#FFFF_FFFF_FFFF_FFFF); // -1 to
			#dint1 := LWORD_TO_DINT(16#FFFF_FFFF_8000_0000); // -2147483648
			#dint1 := LWORD_TO_DINT(16#0); // 0 to
			#dint1 := LWORD_TO_DINT(16#0000_0000_7FFF_FFFF); // 2147483647
			
			ENO = FALSE
			#dint1 := LWORD_TO_DINT(16#FFFF_FFFF_7FFF_FFFF); // -2147483649 to
			#dint1 := LWORD_TO_DINT(16#8000_0000_0000_0000); // -9223372036854775808
			#dint1 := LWORD_TO_DINT(16#0000_0000_8000_0000); // 2147483648 to
			#dint1 := LWORD_TO_DINT(16#7FFF_FFFF_FFFF_FFFF); // 9223372036854775807
				LWORD_TO_DINT

	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_UDINT
	LINT	X		LWORD_TO_LINT
	ULINT	X		LWORD_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If no errors occur during the conversion, the signal state of ENO = 1; if an error occurs during processing, the signal state of ENO = 0.	LWORD_TO_LREAL
	TIME	X		LWORD_TO_TIME
	LTIME	X		LWORD_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_TOD
	LTOD	X		LWORD_TO_LTOD
	DATE	X		LWORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_CHAR
	WCHAR	X		LWORD_TO_WCHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.				

See also

Implicit conversion of LWORD (Page 1967)

Overview of data type conversion (Page 1959)

LWORD (Page 1915)

Integers

Explicit conversion of SINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the SINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
SINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	SINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	SINT_TO_BYTE
	WORD[1)]	X		SINT_TO_WORD
	DWORD[1)]	X		SINT_TO_DWORD
	LWORD[1)]	X		SINT_TO_LWORD
	USINT	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	SINT_TO_USINT
	INT	X		SINT_TO_INT
	UINT	X		SINT_TO_UINT
	DINT	X		SINT_TO_DINT
	UDINT	X		SINT_TO_UDINT
	LINT	X		SINT_TO_LINT
	ULINT	X		SINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	SINT_TO_REAL, NORM_X
	LREAL	X		SINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	SINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	SINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	SINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value	SINT_TO_TOD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			"-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0)	SINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	SINT_TO_DATE
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	SINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		SINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	SINT_TO_CHAR
	WCHAR[1)]	X		SINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of SINT (Page 1969)

Overview of data type conversion (Page 1959)

SINT (8-bit integers) (Page 1917)

Explicit conversion of USINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the USINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
USINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	USINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	USINT_TO_BYTE
	WORD[1)]	X		USINT_TO_WORD
	DWORD[1)]	X		USINT_TO_DWORD
	LWORD[1)]	X		USINT_TO_LWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign is changed during the conversion, the enable output ENO is set to "0".	USINT_TO_SINT
	INT	X	The bit pattern of the source value is converted and transferred to the destination data type.	USINT_TO_INT
	UINT	X		USINT_TO_UINT
	DINT	X		USINT_TO_DINT
	UDINT	X		USINT_TO_UDINT
	LINT	X		USINT_TO_LINT
	ULINT	X		USINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	USINT_TO_REAL, NORM_X
	LREAL	X		USINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	USINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	USINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	USINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in milliseconds since 0:0)	USINT_TO_TOD
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in nanoseconds since 0:0)	USINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in days since 1990-1-1)	USINT_TO_DATE
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	USINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		USINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type.	USINT_TO_CHAR
	WCHAR[1)]	X		USINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of USINT (Page 1970)

Overview of data type conversion (Page 1959)

USINT (8-bit integers) (Page 1918)

Explicit conversion of INT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the INT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
INT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	INT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	INT_TO_BYTE
	WORD[1)]	X		INT_TO_WORD
	DWORD[1)]	X		INT_TO_DWORD
	LWORD[1)]	X		INT_TO_LWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	INT_TO_SINT
	USINT	X		INT_TO_USINT
	UINT	X		INT_TO_UINT
	DINT	X		INT_TO_DINT
	UDINT	X		INT_TO_UDINT
	LINT	X		INT_TO_LINT
	ULINT	X		INT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	INT_TO_REAL, NORM_X
	LREAL	X		INT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	INT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	INT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	INT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO	INT_TO_TOD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			is set to "0". (interpretation in milliseconds since 0:0; check for 24h limit)	
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0; check for 24h limit)	INT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1; check for negative value)	INT_TO_DATE
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	INT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		INT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	INT_TO_CHAR
	WCHAR[1)]	X		INT_TO_WCHAR
	BCD16	X	The value to be converted has type INT and is accepted as an integer with a value between -999 and +999. The result is available after conversion as a BCD-coded number of the type WORD. A real conversion takes place. If the value is outside the target area, a synchronous error is not triggered, but rather only the status bit OV is set.	INT_TO_BCD16
	BCD16_WORD	X		INT_TO_BCD16_WORD

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of INT (Page 1972)

Overview of data type conversion (Page 1959)

INT (16-bit integers) (Page 1918)

Explicit conversion of UINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the UINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
UINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	UINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If bits are lost in the process, the enable output ENO is set to "0".	UINT_TO_BYTE
	WORD[1)]	X		UINT_TO_WORD
	DWORD[1)]	X		UINT_TO_DWORD
	LWORD[1)]	X		UINT_TO_LWORD
	SINT	X		UINT_TO_SINT
	USINT	X		UINT_TO_USINT
	INT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UINT_TO_INT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	UINT_TO_DINT
	UDINT	X		UINT_TO_UDINT
	LINT	X		UINT_TO_LINT
	ULINT	X		UINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	UINT_TO_REAL, NORM_X
	LREAL	X		UINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	UINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	UINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	UINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in milliseconds since 0:0; check for 24h limit)	UINT_TO_TOD
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in nanoseconds since 0:0; check for 24h limit)	UINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in days since 1990-1-1; check for negative value)	UINT_TO_DATE, T_CONV
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		UINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the destination data type. The enable output ENO is set to "0" in the event of overflow.	UINT_TO_CHAR
	WCHAR[1)]	X		UINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of UINT (Page 1973)

Overview of data type conversion (Page 1959)

UINT (16-bit integers) (Page 1919)

Explicit conversion of DINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	DINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	DINT_TO_BYTE
	WORD[1)]	X		DINT_TO_WORD
	DWORD[1)]	X		DINT_TO_DWORD
	LWORD[1)]	X		DINT_TO_LWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	DINT_TO_SINT
	USINT	X		DINT_TO_USINT
	INT	X		DINT_TO_INT
	UINT	X		DINT_TO_UINT
	UDINT	X		DINT_TO_UDINT
	LINT	X		DINT_TO_LINT
	ULINT	X		DINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	DINT_TO_REAL, NORM_X
	LREAL	X		DINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	DINT_TO_TIME, T_CONV
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	DINT_TO_LTIME, T_CONV
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	DINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1"	DINT_TO_TOD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			(16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0)	DINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	DINT_TO_DATE
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	DINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		DINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	DINT_TO_CHAR
	WCHAR[1)]	X		DINT_TO_WCHAR
	BCD32	X	The value to be converted has type DINT and is accepted as an integer with a value between –999999 and +9999999. The result is available after conversion as a BCD-coded number of the type DWORD. The enable output is set to "0" in the event of overflow. A real conversion takes place. If the value is outside the target area, a synchronous error is not triggered, but rather only the status bit OV is set.	DINT_TO_BCD32
	BCD32_DWORD	X		DINT_TO_BCD32_DWORD

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of DINT (Page 1975)

Overview of data type conversion (Page 1959)

DINT (32-bit integers) (Page 1920)

Explicit conversion of UDINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the UDINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
UDINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	UDINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If bits are lost in the process, the enable output ENO is set to "0".	UDINT_TO_BYTE
	WORD[1)]	X		UDINT_TO_WORD
	DWORD[1)]	X		UDINT_TO_DWORD
	LWORD[1)]	X		UDINT_TO_LWORD
	SINT	X		UDINT_TO_SINT
	USINT	X		UDINT_TO_USINT
	INT	X		UDINT_TO_INT
	UINT	X		UDINT_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UDINT_TO_DINT
	LINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	UDINT_TO_LINT
	ULINT	X		UDINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	UDINT_TO_REAL, NORM_X
	LREAL	X		UDINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	UDINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	UDINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	UDINT_TO_LDT
	DT	-	No explicit conversion	-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0; check for 24h limit)	UDINT_TO_TOD, T_CONV
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0; check for 24h limit)	UDNT_TO_LTOD, T_CONV
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1; check for negative value)	UDINT_TO_DATE
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UDINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		UDINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the destination data type. The enable output ENO is set to "0" in the event of overflow.	UDINT_TO_CHAR
	WCHAR[1)]	X		UDINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of UDINT (Page 1977)

Overview of data type conversion (Page 1959)

UDINT (32-bit integers) (Page 1921)

Explicit conversion of LINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	LINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	LINT_TO_BYTE
	WORD[1)]	X		LINT_TO_WORD
	DWORD[1)]	X		LINT_TO_DWORD
	LWORD[1)]	X		LINT_TO_LWORD
	SINT	X		LINT_TO_SINT
	USINT	X		LINT_TO_USINT
	INT	X		LINT_TO_INT
	UINT	X		LINT_TO_UINT
	DINT	X		LINT_TO_DINT
	UDINT	X		LINT_TO_UDINT
	ULINT	X		LINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	LINT_TO_REAL, NORM_X
	LREAL	X		LINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	LINT_TO_TIME, T_CONV
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	LINT_TO_LTIME, T_CONV
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	LINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative	LINT_TO_TOD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0)	LINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	LINT_TO_DATE
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	LINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		LINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	LINT_TO_CHAR
	WCHAR[1)]	X		LINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of LINT (Page 1979)

Overview of data type conversion (Page 1959)

LINT (64-bit integers) (Page 1922)

Explicit conversion of ULINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the ULINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
ULINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	ULINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	ULINT_TO_BYTE
	WORD[1)]	X		ULINT_TO_WORD
	DWORD[1)]	X		ULINT_TO_DWORD
	LWORD[1)]	X		ULINT_TO_LWORD
	SINT	X		ULINT_TO_SINT
	USINT	X		ULINT_TO_USINT
	INT	X		ULINT_TO_INT
	UINT	X		ULINT_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign bit is overwritten during the conversion, the enable output ENO is set to "0".	ULINT_TO_DINT
	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	ULINT_TO_UDINT
	LINT	X		ULINT_TO_LINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	ULINT_TO_REAL, NORM_X
	LREAL	X		ULINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	ULINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	ULINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	ULINT_TO_LDT
	DT	-	No explicit conversion	-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in milliseconds since 0:0)	ULINT_TO_TOD, T_CONV
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in nanoseconds since 0:0)	ULINT_TO_LTOD, T_CONV
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in days since 1990-1-1)	ULINT_TO_DATE
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	ULINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		ULINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type.	ULINT_TO_CHAR
	WCHAR[1)]	X		ULINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of ULINT (Page 1981)

Overview of data type conversion (Page 1959)

ULINT (64-bit integers) (Page 1923)

Floating-point numbers

Explicit conversion of REAL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the REAL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
REAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	REAL_TO_DWORD
	LWORD	-	No explicit conversion	-
	SINT	X	The value is converted into the destination data type. The result of conversion depends on the instruction employed. The enable output ENO is set to "0" if the admissible range of values of the destination data type is exceeded during conversion or the value to be converted has an invalid floating-point number.	REAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		REAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		REAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		REAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		REAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		REAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LINT	X		REAL_TO_LINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	ULINT	X		REAL_TO_ULINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LREAL	X	The value is converted into the destination data type. The result of the conversion depends on the instruction used, e.g. TRUNC(2.5) = 2.0; CEIL(2.5) = 3.0	REAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	LTIME	-		-
	S5TIME	-		-
	LDT	-		-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	X	The value is converted into a character string. The enable output ENO is set to "0" if the character string length is exceeded or the value to be converted has an invalid floating-point number. The min. length of the character string is 14 characters.	REAL_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		REAL_TO_WSTRING
	CHAR	-	No explicit conversion	-
	WCHAR	-		

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of REAL (Page 1983)

Overview of data type conversion (Page 1959)

REAL (Page 1925)

Explicit conversion of LREAL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LREAL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LREAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LREAL_TO_LWORD
	SINT	X	The value is converted into the destination data type. The result of conversion depends on the instruction employed. The enable output ENO is set to "0" if the admissible range of values of the destination data type is exceeded during conversion or the value to be converted has an invalid floating-point number.	LREAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		LREAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		LREAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		LREAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		LREAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		LREAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LINT	X		LREAL_TO_LINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	ULINT	X		LREAL_TO_ULINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	REAL	X	The value is converted into the destination data type. If the permitted value range of the destination data type is violated during the conversion or if the value to be converted is an invalid floating-point number, the enable output ENO is set to "0". A loss in accuracy is tolerated.	LREAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	LTIME	-		-
	S5TIME	-		-
	LDT	-		-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	X	The value is converted into a character string. The enable output ENO is set to "0" if the character string length is exceeded or the value to be converted has an invalid floating-point number. The min.	LREAL_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		LREAL_TO_WSTRING

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			length of the character string is 21 characters.	
	CHAR	-	No explicit conversion	-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of LREAL (Page 1985)

Overview of data type conversion (Page 1959)

LREAL (Page 1926)

Timers

Explicit conversion of S5TIME

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the S5TIME data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
S5TIME	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	S5TIME_TO_WORD
	DWORD	-	No explicit conversion	-
	LWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	LINT	-		-
	ULINT	-		-
	REAL	-		-
	LREAL	-		-
	TIME	X	A conversion of the source value into the destination data type takes place here (for example, s5t#10ms becomes T#10ms)	S5TIME_TO_TIME
	LTIME	X	A conversion of the source value into the destination data type takes place here (for example, s5t#10ms becomes LTIME#10ms)	S5TIME_TO_LTIME
	LDT	-	No explicit conversion	-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of S5TIME (Page 1986)

Overview of data type conversion (Page 1959)

S5TIME (duration) (Page 1929)

Explicit conversion of TIME

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TIME data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
TIME	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TIME_TO_BYTE
	WORD[1)]	X		TIME_TO_WORD
	DWORD[1)]	X		TIME_TO_DWORD
	LWORD[1)]	X		TIME_TO_LWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the destination data type.	TIME_TO_SINT
	USINT	X		TIME_TO_USINT
	INT	X		TIME_TO_INT
	UINT	X		TIME_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The result of conversion shows the duration in milliseconds.	TIME_TO_DINT, T_CONV
	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the destination data type. A change in sign has the result that the enable output ENO is set to "0".	TIME_TO_UDINT
	LINT	X		TIME_TO_LINT
	ULINT	X		TIME_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	X	A conversion of the source value into the destination data type takes place here (for example, T#10ms becomes s5t#10ms)	TIME_TO_TIME
	LTIME	X	A conversion of the source value into the destination data type takes place here (for example, T#10ms becomes LTIME#10ms)	TIME_TO_LTIME
	LDT	-	No explicit conversion	-
	DT	-		-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the source value is outside the value range of TOD, the destination data type remains unchanged.	TIME_TO_TOD
	LTOD	-	No explicit conversion	-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of TIME (Page 1988)

Overview of data type conversion (Page 1959)

TIME (IEC time) (Page 1930)

Explicit conversion of LTIME

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LTIME data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LTIME	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LTIME_TO_BYTE
	WORD[1)]	X		LTIME_TO_WORD
	DWORD[1)]	X		LTIME_TO_DWORD
	LWORD[1)]	X		LTIME_TO_LWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as nanoseconds to the destination data type. A change in sign has the result that the enable output ENO is set to "0".	LTIME_TO_SINT
	USINT	X		LTIME_TO_USINT
	INT	X		LTIME_TO_INT
	UINT	X		LTIME_TO_UINT
	DINT	X		LTIME_TO_DINT
	UDINT	X		LTIME_TO_UDINT
	LINT	X		LTIME_TO_LINT, T_CONV
	ULINT	X		LTIME_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	X	A conversion of the source value into the destination data type takes place here (for example, LTIME#10ms becomes s5t#10ms)	LTIME_TO_S5TIME
	TIME	X	A conversion of the source value into the destination data type takes place here (for example, LTIMET#10ms becomes T#10ms)	LTIME_TO_TIME
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, LTIMET#10ms becomes LDT#1970-1-1-0:0:0.010000000)	LTIME_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	LTOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the source value is outside the value range of LTOD, the destination data type remains unchanged.	LTIME_TO_LTOD
	DATE	-	No explicit conversion	-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of LTIME (Page 1990)

Overview of data type conversion (Page 1959)

LTIME (IEC time) (Page 1931)

Date and time-of-day

Explicit conversion of DT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DT	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	LINT	-		-
	ULINT	-		-
	REAL	-		-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes LDT#1990-1-2-0:0:1.0)	DT_TO_LDT
	DTL	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes DTL#1990-1-2-0:0:1.0)	DT_TO_DTL
	TOD	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes TOD#1990-1-2-0:0:1.0)	DT_TO_TOD
	LTOD	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes LTOD#1990-1-2-0:0:1.0)	DT_TO_LTOD
	DATE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DT_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of DT (Page 1992)

Overview of data type conversion (Page 1959)

DATE_AND_TIME (date and time of day) (Page 1933)

Explicit conversion of LDT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LDT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LDT	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LDT_TO_BYTE
	WORD[1)]	X		LDT_TO_WORD
	DWORD[1)]	X		LDT_TO_DWORD
	LWORD[1)]	X		LDT_TO_LWORD
	SINT	X		LDT_TO_SINT
	USINT	X		LDT_TO_USINT
	INT	X		LDT_TO_INT
	UINT	X		LDT_TO_UINT
	DINT	X		LDT_TO_DINT
	UDINT	X		LDT_TO_UDINT
	LINT	X		LDT_TO_LINT
	ULINT	X		LDT_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LDT_TO_LTIME
	DT	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes DT#1990-1-2-0:0:1.0)	LDT_TO_DT
	DTL	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes DTL#1990-1-2-0:0:1.0)	LDT_TO_DTL
	TOD	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes TOD#1990-1-2-0:0:1.0)	LDT_TO_TOD
	LTOD	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes LTOD#1990-1-2-0:0:1.0)	LDT_TO_LTOD
	DATE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LDT_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of LDT (Page 1994)

Overview of data type conversion (Page 1959)

LDT (DATE_AND_LTIME) (Page 1934)

Explicit conversion of DTL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DTL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DTL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	LINT	-		-
	ULINT	-		-
	REAL	-		-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, DTL#1990-1-2-0:0:1.0 becomes LDT#1990-1-2-0:0:1.0)	DTL_TO_LDT
	DT	X	A conversion of the source value into the destination data type takes place here (for example, DTL#1990-1-2-0:0:1.0 becomes DT#1990-1-2-0:0:1.0)	DTL_TO_DT
	TOD	X	During the conversion, the time of day is extracted from the DTL format and written to the destination data type.	DTL_TO_TOD, T_CONV
	LTOD	X		DTL_TO_LTOD
	DATE	X	During the conversion, the date is extracted from the DTL format and written to the destination data type. The enable output ENO is set to "0" in the event of overflow.	DTL_TO_DATE, T_CONV
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of DTL (Page 1996)

Overview of data type conversion (Page 1959)

DTL (Page 1935)

Explicit conversion of DATE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DATE data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DATE	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DATE_TO_BYTE
	WORD[1)]	X		DATE_TO_WORD
	DWORD[1)]	X		DATE_TO_DWORD
	LWORD[1)]	X		DATE_TO_LWORD
	SINT	X	The number of days since 1/1/1990 is returned as result.	DATE_TO_SINT
	USINT	X		DATE_TO_USINT
	INT	X		DATE_TO_INT
	UINT	X		DATE_TO_UINT
	DINT	X		DATE_TO_DINT
	UDINT	X		DATE_TO_UDINT
	LINT	X		DATE_TO_LINT
	ULINT	X		DATE_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	DT	X	The conversion sets the date in the destination type.	DATE_TO_DT
	LDT	X		DATE_TO_LDT
	DTL	X		DATE_TO_DTL
	TOD	-	No explicit conversion	-
	LTOD	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of DATE (Page 1998)

Overview of data type conversion (Page 1959)

DATE (Page 1931)

Explicit conversion of TOD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TOD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
TOD	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TOD_TO_BYTE
	WORD[1)]	X		TOD_TO_WORD
	DWORD[1)]	X		TOD_TO_DWORD
	LWORD[1)]	X		TOD_TO_LWORD
	SINT	X	The number of milliseconds since midnight is returned as result.	TOD_TO_SINT
	USINT	X		TOD_TO_USINT
	INT	X		TOD_TO_INT
	UINT	X		TOD_TO_UINT
	DINT	X		TOD_TO_DINT
	UDINT	X	The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).	TOD_TO_UDINT, T_CONV
	LINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TOD_TO_LINT
	ULINT	X		TOD_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	X	The duration since midnight is returned as result.	TOD_TO_TIME
	LTIME	-	No explicit conversion	-
	DT	X	The conversion sets the time in the destination data type.	TOD_TO_DT
	LDT	X		TOD_TO_LDT
	DTL	X		TOD_TO_DTL
	DATE	-	No explicit conversion	-
	LTOD	X	A conversion of the source value into the destination data type takes place here (for example, TOD#1:0:0.0 becomes LTOD#1:0:0.0)	TOD_TO_LTOD
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of TOD (Page 2000)

Overview of data type conversion (Page 1959)

TIME_OF_DAY (TOD) (Page 1932)

Explicit conversion of LTOD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LTOD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LTOD	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LTOD_TO_BYTE
	WORD[1)]	X		LTOD_TO_WORD
	DWORD[1)]	X		LTOD_TO_DWORD
	LWORD[1)]	X		LTOD_TO_LWORD
	SINT	X	The number of nanoseconds since midnight is returned as result.	LTOD_TO_SINT
	USINT	X		LTOD_TO_USINT
	INT	X		LTOD_TO_INT
	UINT	X		LTOD_TO_UINT
	DINT	X		LTOD_TO_DINT
	UDINT	X		LTOD_TO_UDINT
	LINT	X		LTOD_TO_LINT
	ULINT	X		LTOD_TO_ULINT, T_CONV
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LTOD_TO_LTIME
	DT	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes DT#1:0:0.0)	LTOD_TO_DT
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes LDT#1:0:0.0)	LTOD_TO_LDT
	DTL	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes DTL#1:0:0.0)	LTOD_TO_DTL
	DATE	-	No explicit conversion	-
	TOD	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes TOD#1:0:0.0)	LTOD_TO_TOD
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of LTOD (Page 2002)

Overview of data type conversion (Page 1959)

LTOD (LTIME_OF_DAY) (Page 1932)

String

Explicit conversion of CHAR

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the CHAR data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
CHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	CHAR_TO_BYTE
	WORD[1)]	X		CHAR_TO_WORD
	DWORD[1)]	X		CHAR_TO_DWORD
	LWORD[1)]	X		CHAR_TO_LWORD
	SINT	X		CHAR_TO_SINT
	USINT	X		CHAR_TO_USINT
	INT	X		CHAR_TO_INT
	UINT	X		CHAR_TO_UINT
	DINT	X		CHAR_TO_DINT
	UDINT	X		CHAR_TO_UDINT
	LINT	X		CHAR_TO_LINT
	ULINT	X		CHAR_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	X	The value is converted into the first character in the character string (STRING). The length "1" is set after conversion if the character string length is not defined. If the character string length is defined, it remains unchanged following conversion.	CHAR_TO_STRING, S_CONV
	WSTRING	-	No explicit conversion	-
	WCHAR	X		CHAR_TO_WCHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data type WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of CHAR (Page 2004)

Overview of data type conversion (Page 1959)

CHAR (character) (Page 1936)

Explicit conversion of WCHAR

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WCHAR data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WCHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WCHAR_TO_BYTE
	WORD[1)]	X		WCHAR_TO_WORD
	DWORD[1)]	X		WCHAR_TO_DWORD
	LWORD[1)]	X		WCHAR_TO_LWORD
	SINT	X		WCHAR_TO_SINT
	USINT	X		WCHAR_TO_USINT
	INT	X		WCHAR_TO_INT
	UINT	X		WCHAR_TO_UINT
	DINT	X		WCHAR_TO_DINT
	UDINT	X		WCHAR_TO_UDINT
	LINT	X		WCHAR_TO_LINT
	ULINT	X		WCHAR_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	X	The value is converted into the first character in the character string (WSTRING). The length "1" is set after conversion if the character string length is not defined. If the character string length is defined, it remains unchanged following conversion.	WCHAR_TO_WSTRING
	CHAR	X		WCHAR_TO_CHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data type CHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Overview of data type conversion (Page 1959)

WCHAR (character) (Page 1937)

Implicit conversion of WCHAR (Page 2005)

Explicit conversion of STRING

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the STRING data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
STRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are admissible for conversion: • Digit • Sign • Dot The first character in the character string may be a sign (+, -) or a digit. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator to the left of the decimal point is permitted but is ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	STRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		STRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		STRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		STRING_TO_UINT, S_CONV, STRG_VAL
	DINT	X		STRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		STRING_TO_UDINT, S_CONV, STRG_VAL
	LINT	X		STRING_TO_LINT, S_CONV, STRG_VAL
	ULINT	X		STRING_TO_ULINT, S_CONV, STRG_VAL
	REAL	X		STRING_TO_REAL, S_CONV, STRG_VAL
	LREAL	X		STRING_TO_LREAL, S_CONV, STRG_VAL
	S5TIME	-	No explicit conversion	-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	CHAR	X	The first character in the character string (STRING) is transferred to the destination data type. The value "0" is written to the destination data type if the character string is empty.	STRING_TO_CHAR, S_CONV
	WCHAR	-	No explicit conversion	-
	WSTRING	X		STRING_TO_WSTRING
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of STRING (Page 2007)

Overview of data type conversion (Page 1959)

STRING (Page 1937)

Explicit conversion of WSTRING

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WSTRING data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WSTRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (WSTRING) and stops at the end of the string or at the first inadmissible character. The following characters are admissible for conversion: • Digit • Sign • Dot The first character in the character string may be a sign (+, -) or a digit. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator to the left of the decimal point is permitted but is ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	WSTRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		WSTRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		WSTRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		WSTRING_TO_UINT, S_CONV, STRG_VAL
	DINT	X		WSTRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		WSTRING_TO_UDINT, S_CONV, STRG_VAL
	LINT	X		WSTRING_TO_LINT, S_CONV, STRG_VAL
	ULINT	X		WSTRING_TO_ULINT, S_CONV, STRG_VAL
	REAL	X		WSTRING_TO_REAL, S_CONV, STRG_VAL
	LREAL	X		WSTRING_TO_LREAL, S_CONV, STRG_VAL
	S5TIME	-	No explicit conversion	-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	CHAR	-		-
	WCHAR	X	The first character in the character string (WSTRING) is transferred to the destination data type. The value "0" is written to the destination data type if the character string is empty.	WSTRING_TO_WCHAR
	STRING	X		WSTRING_TO_STRING
x: Conversion possible - : Conversion not possible				

See also

Overview of data type conversion (Page 1959)

WSTRING (Page 1939)

Implicit conversion of WSTRING (Page 2008)

11.5.16 Data type conversion for S7-1200:

11.5.16.1 Overview of data type conversion

Introduction

If you link several operands in an instruction, you must make sure that the data types are compatible. This applies also for assignments or for supplying block parameters. If the operands are not the same data type, a conversion has to be carried out.

There are two options for the conversion:

- Implicit conversion
  The conversion take place automatically when the instruction is executed.
- Explicit conversion
  You use an explicit conversion instruction before the actual instruction is executed.

Note

The data type conversion options described always refer to the latest version of the CPU (V. 4) Conversions marked as possible may not be available in CPU versions 1 - 3.

Note

Converting bit strings in SCL

All bit strings (BYTE, WORD, and DWORD) are handled like the corresponding unsigned integers (USINT, UINT, and UDINT) in expressions. Therefore, implicit conversion from DWORD to REAL is carried out like a conversion from UDINT to REAL, for example.

Implicit conversion

An implicit conversion is executed automatically if the data types of the operands are compatible. This compatibility test can be carried out according to criteria that are more or less strict:

- With IEC check (default)
  If IEC check is set, the following rules are applied:
  - Implicit conversion of BOOL to other data types is not possible.
  - Only the REAL, BYTE, WORD, DINT, INT, SINT, UDINT, UINT, USINT, TIME, DT, STRING, CHAR and WCHAR data types can be converted implicitly.
  - The bit length of the source data type must not exceed the bit length of the target data type. An operand of data type WORD, for example, cannot be declared at a parameter at which data type BYTE is expected.
- Without IEC check
  If IEC check is not set, the following rules are applied:
  - Implicit conversion of BOOL to other data types is not possible.
  - Only the REAL, LREAL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, TIME, DTL, TOD, DATE, STRING, CHAR and WCHAR data types can be converted implicitly.
  - The bit length of the source data type must not exceed the bit length of the target data type. An operand of data type DWORD, for example, cannot be declared at a parameter at which data type WORD is expected.
  - The bit length of an operand entered in-out parameters InOut) must be the same as the programmed bit length for the parameter in question.

Note

Implicit conversion without IEC check

The programming editor uses a gray rectangle to mark operands that are implicitly converted. The dark gray rectangle signals that an implicit conversion is possible without any accuracy loss, for example, if you convert the data type SINT to INT. A light gray rectangle signals that implicit conversion is possible, but errors could occur during runtime. If, for example, you are converting the data type DINT to INT and an overflow occurs, the enable output ENO is set to "0".

For more information about the setting of the IEC check and the implicit conversion, refer to "See also".

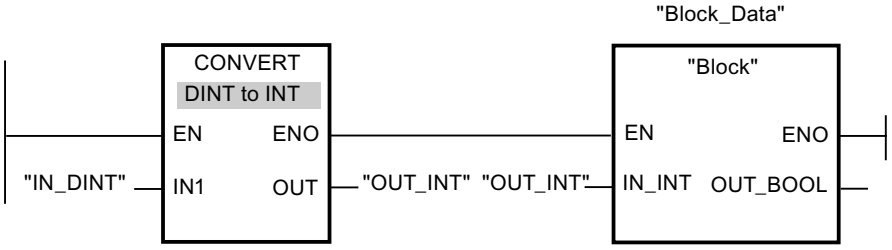
Explicit conversion

If the operands are not compatible and an implicit conversion is therefore not possible, you can use an explicit conversion instruction. You can find the conversion instructions in the "Instructions" task card.

A possible overflow is displayed at the ENO enable output. An overflow is created, for example, if the value of the source data type is greater than the value of the target data type.

For more information about explicit conversion, refer to "See also".

The following figure shows an example in which an explicit data type conversion must be carried out:



The "Block" function block expects a tag of the INT data type at the "IN_INT" input parameter. Therefore, the value of the "IN_DINT" tag must first be converted from DINT to INT. If the value of the "IN_DINT" tag is within the permitted value range of the INT data type, the conversion takes place. Otherwise, an overflow is signaled. A conversion still takes place even in case of an overflow, but the values are cut off and the enable output ENO is set to "0".

See also

Implicit conversion (Page 2093)

Explicit conversion (Page 2113)

Setting and canceling the IEC check (Page 2093)

11.5.16.2 Implicit conversion

Setting and canceling the IEC check

The data types of the operands used are check for compatibility. This compatibility test can be carried out according to criteria that are more or less strict. If "IEC check" is activated, stricter criteria are applied.

You can set the IEC check centrally for all new blocks of the project or for individual blocks.

Setting IEC check for new blocks

To set the IEC check for all new blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. Select or clear the "IEC check" check box in the "Default settings for new blocks" group.
   The IEC check is enabled or disabled for all new blocks in the program.

Setting IEC check for a block

To set the IEC check for a block, proceed as follows:

1. Open the block.
2. Open the "Properties" tab in the Inspector window.
3. Select the "Attributes" group in the area navigation.
4. Select or clear the "IEC check" check box.
   The IEC check is enabled or disabled for this block. The setting is stored together with the project.

Binary numbers

Implicit conversion of BOOL

Options for implicit conversion

The implicit conversion of the BOOL data type is not possible.

See also

BOOL (bit) (Page 1912)

Bit strings

Implicit conversion of BYTE

Options for implicit conversion

The following table shows the options for implicit conversion of the BYTE data type:

Source	Target	With IEC check	Without IEC check	Explanation
BYTE	BOOL	-	-	No implicit conversion
	WORD	x	x	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DWORD	x	x	
	SINT	-	x	
	USINT	-	x	
	INT	-	x	
	UINT	-	x	
	DINT	-	x	
	UDINT	-	x	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	x	The bit pattern of the source value is transferred unchanged to the target data type.
	WCHAR	-	x	
x: Conversion possible -: Conversion not possible				

See also

BYTE (byte) (Page 1913)
Setting and canceling the IEC check (Page 2093)
Overview of data type conversion (Page 2091)
Explicit conversion of BYTE (Page 2114)

Implicit conversion of WORD

Options for implicit conversion

The following table shows the options for implicit conversion of the WORD data type:

Source	Target	With IEC check	Without IEC check	Description
WORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The least significant byte is transferred to the target data type, while the most significant byte is ignored.
	DWORD	X	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	SINT	-	X	The least significant byte is transferred to the target data type, while the most significant byte is ignored.
	USINT	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged to the target data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

WORD (Page 1914)

Setting and canceling the IEC check (Page 2093)

Overview of data type conversion (Page 2091)

Explicit conversion of WORD (Page 2116)

Implicit conversion of DWORD

Options for implicit conversion

The following table shows the options for implicit conversion of the DWORD data type:

Source	Target	With IEC check	Without IEC check	Explanation
DWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The right bytes are transferred to the target data type; the left bytes are ignored.
	WORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	UDINT	-	X	
	REAL	-	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	-	-	No implicit conversion
	TIME	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DTL	-	-	No implicit conversion
	TOD	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

DWORD (Page 1915)

Setting and canceling the IEC check (Page 2093)

Overview of data type conversion (Page 2091)

Explicit conversion of DWORD (Page 2119)

Integers

Implicit conversion of SINT

Options for implicit conversion

The following table shows the options for implicit conversion of the SINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
SINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	USINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value transfer from SINT #-1 -> INT #-1, not filled with "0".)
	INT	X	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

SINT (8-bit integers) (Page 1917)

Setting and canceling the IEC check (Page 2093)

Overview of data type conversion (Page 2091)

Explicit conversion of SINT (Page 2123)

Implicit conversion of USINT

Options for implicit conversion

The following table shows the options for implicit conversion of the USINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
USINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from USINT #10 -> DINT #10 or USINT #128 -> SINT #-128)
	INT	X	X	
	UINT	X	X	
	DINT	X	X	
	UDINT	X	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "1", for example, is converted to the value "1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

USINT (8-bit integers) (Page 1918)

Setting and canceling the IEC check (Page 2093)

Overview of data type conversion (Page 2091)

Explicit conversion of USINT (Page 2126)

Implicit conversion of INT

Options for implicit conversion

The following table shows the options for the implicit conversion of the INT data type:

Source	Target	With IEC check	Without IEC check	Explanation
INT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from INT #-1 -> SINT #-1, or INT #-32 767 -> UINT #32 769)
	USINT	-	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

INT (16-bit integers) (Page 1918)
Setting and canceling the IEC check (Page 2093)
Overview of data type conversion (Page 2091)
Explicit conversion of INT (Page 2129)

Implicit conversion of UINT

Options for implicit conversion

The following table shows the options for implicit conversion of the UINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
UINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from UINT #100 -> DINT #100 or UINT #60 000 -> INT #-5536)
	USINT	-	X	
	INT	-	X	
	DINT	X	X	
	UDINT	X	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "1", for example, is converted to the value "1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

UINT (16-bit integers) (Page 1919)

Setting and canceling the IEC check (Page 2093)

Overview of data type conversion (Page 2091)

Explicit conversion of UINT (Page 2132)

Implicit conversion of DINT

Options for implicit conversion

The following table shows the options for implicit conversion of the DINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
DINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	UDINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DTL	-	-	No implicit conversion
	TOD	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

DINT (32-bit integers) (Page 1920)

Setting and canceling the IEC check (Page 2093)

Overview of data type conversion (Page 2091)

Explicit conversion of DINT (Page 2135)

Implicit conversion of UDINT

Options for implicit conversion

The following table shows the options for implicit conversion of the UDINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
UDINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the format of the target data type. (The value "1", for example, is converted to the value "1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DTL	-	-	No implicit conversion
	TOD	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	

x: Conversion possible
-: Conversion not possible

See also

UDINT (32-bit integers) (Page 1921)

Setting and canceling the IEC check (Page 2093)

Overview of data type conversion (Page 2091)

Explicit conversion of UDINT (Page 2138)

Floating-point numbers

Implicit conversion of REAL

Options for implicit conversion

The following table shows the options for implicit conversion of the REAL data type:

Source	Target	With IEC check	Without IEC check	Explanation
REAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the target data type. (for example, rounding off and value conversion of REAL #2.5 -> INT #2 or negative number REAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined REAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LREAL	X	X	The value is transferred to the target data type.
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

REAL (Page 1925)

Setting and canceling the IEC check (Page 2093)

Overview of data type conversion (Page 2091)

Explicit conversion of REAL (Page 2141)

Implicit conversion of LREAL

Options for implicit conversion

The following table shows the options for implicit conversion of the LREAL data type:

Source	Target	With IEC check	Without IEC check	Explanation
LREAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the target data type. (for example, rounding off and value conversion of REAL #2.5 -> INT #2 or negative number REAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined REAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Explicit conversion of LREAL (Page 2144)

Timers

Implicit conversion of TIME

Options for implicit conversion

The following table shows the options for implicit conversion of the TIME data type:

Source	Target	With IEC check	Without IEC check	Explanation
TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion shows the duration in milliseconds.
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion shows the duration in milliseconds.
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	DTL	-	-	
	TOD	-	X	The bit pattern of a source value that is less than 24 h (86 400 00 ms) is transferred without changes to the target data type. No further changes are made to the target value. The result of the conversion shows the time that has passed since midnight.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	

x: Conversion possible
-: Conversion not possible

See also

TIME (IEC time) (Page 1930)

Setting and canceling the IEC check (Page 2093)

Overview of data type conversion (Page 2091)

Explicit conversion of TIME (Page 2147)

Date and time

Implicit conversion of DATE

Options for implicit conversion

The following table shows the options for implicit conversion of the DATE data type:

Source	Target	With IEC check	Without IEC check	Explanation
DATE	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	DWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	INT	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	UINT	-	X	
	DINT	-	-	No implicit conversion
	UDINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	DTL	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	TOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	

x: Conversion possible
-: Conversion not possible

See also

DATE (Page 1931)
Setting and canceling the IEC check (Page 2093)
Overview of data type conversion (Page 2091)
Explicit conversion of DATE (Page 2149)

Implicit conversion of TOD

Options for implicit conversion

The following table shows the options for implicit conversion of the TOD data type:

Source	Target	With IEC check	Without IEC check	Explanation
TOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	DTL	-	-	No implicit conversion
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME_OF_DAY (TOD) (Page 1932)

Setting and canceling the IEC check (Page 2093)

Overview of data type conversion (Page 2091)

Explicit conversion of TOD (Page 2150)

Implicit conversion of DTL

Options for implicit conversion

The DTL data type cannot be implicitly converted.

See also

Explicit conversion of DTL (Page 2151)

Character strings

Implicit conversion of CHAR

Options for implicit conversion

The following table shows the options for implicit conversion of the CHAR data type:

Source	Target	With IEC check	Without IEC check	Explanation
CHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	WCHAR	-	-	
	STRING	X	X	The STRING is shortened to length 1 and includes the character.
	WSTRING	-	-	No implicit conversion
x: Conversion possible -: Conversion not possible				

See also

CHAR (character) (Page 1936)
Setting and canceling the IEC check (Page 2093)
Overview of data type conversion (Page 2091)
Explicit conversion of CHAR (Page 2152)

Implicit conversion of WCHAR

Options for implicit conversion

The following table shows the options for the implicit conversion of the WCHAR data type:

Source	Target	With IEC check	Without IEC check	Explanation
WCHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	CHAR	-	-	
	STRING	-	-	
	WSTRING	X	X	The WSTRING is shortened to length 1 and includes the character.
x: Conversion possible -: Conversion not possible				

Implicit conversion of STRING

Options for implicit conversion

The following table shows the options for implicit conversion of the STRING data type:

Source	Target	With IEC check	Without IEC check	Explanation
STRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	DATE	-	-	
	TOD	-	-	
	CHAR	-	X	The first character of the STRING is returned if the STRING includes one or more characters. Otherwise, the character is output with coding $00.
	WCHAR	-	-	No implicit conversion
	WSTRING	-	-	
x: Conversion possible -: Conversion not possible				

See also

Explicit conversion of STRING (Page 2154)

Implicit conversion of WSTRING

Options for implicit conversion

The following table shows the options for the implicit conversion of the WSTRING data type:

Source	Target	With IEC check	Without IEC check	Explanation
WSTRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	DATE	-	-	
	TOD	-	-	
	CHAR	-	-	
	WCHAR	-	X	The first character of the WSTRING is returned if the WSTRING includes one or more characters. Otherwise, the character is output with coding $0000.
	STRING	-	-	No implicit conversion
x: Conversion possible -: Conversion not possible				

11.5.16.3 Explicit conversion

Binary numbers

Explicit conversion of BOOL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BOOL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
BOOL	BYTE	X	Only the LSB (Least Significant Bit) is set in the target data type. The enable output ENO is always "1".	BOOL_TO_BYTE
	WORD	X		BOOL_TO_WORD
	DWORD	X		BOOL_TO_DWORD
	SINT	X		BOOL_TO_SINT
	USINT	X		BOOL_TO_USINT
	INT	X		BOOL_TO_INT
	UINT	X		BOOL_TO_UINT
	DINT	X		BOOL_TO_DINT
	UDINT	X		BOOL_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

BOOL (bit) (Page 1912)

Implicit conversion of BYTE (Page 2095)

Overview of data type conversion (Page 2091)

Bit strings

Explicit conversion of BYTE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BYTE data type:

Source	Target	Conversion	Explanation	Mnemonics of Instruction
BYTE[1)]	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bits are not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	BYTE_TO_BOOL
	WORD[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	BYTE_TO_WORD
	DWORD[1)]	X		BYTE_TO_DWORD
	SINT	X		BYTE_TO_SINT
	USINT	X		BYTE_TO_USINT
	INT	X		BYTE_TO_INT
	UINT	X		BYTE_TO_UINT
	DINT	X		BYTE_TO_DINT
	UDINT	X		BYTE_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	BYTE_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	BYTE_TO_TOD
	DATE	X		BYTE_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	BYTE_TO_CHAR
	WCHAR	X		BYTE_TO_WCHAR

Source	Target	Conversion	Explanation	Mnemonics of Instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD) are interpreted as an unsigned integer with the same bit length.. Data type BYTE is interpreted as USINT, WORD as UINT and DWORD as UDINT.				

See also

BYTE (byte) (Page 1913)

Implicit conversion of BYTE (Page 2095)

Overview of data type conversion (Page 2091)

Explicit conversion of WORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WORD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
WORD[1)]	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	WORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_BYTE
	DWORD[1)]	X		WORD_TO_DWORD

	SINT	X	ENO = TRUE
			#sint1 := WORD_TO_SINT(16#FFFF); // -1 to
			#sint1 := WORD_TO_SINT(16#FF80); // -128
			#sint1 := WORD_TO_SINT(16#0); // 0 to
			#sint1 := WORD_TO_SINT(16#007F); // 127
			
			ENO = FALSE
			#sint1 := WORD_TO_SINT(16#FF7F); // -129 to
			#sint1 := WORD_TO_SINT(16#8000); // -32768
			#sint1 := WORD_TO_SINT(16#0080); // 128 to
			#sint1 := WORD_TO_SINT(16#7FFF); // 32767
				WORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_USINT
	INT	X		WORD_TO_INT
	UINT	X		WORD_TO_UINT
	DINT	X		WORD_TO_DINT
	UDINT	X		WORD_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_TOD
	DATE	X		WORD_TO_DATE

Source	Target	Conversion	Explanation	Mnemonics of the instruction
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_CHAR
	WCHAR	X		WORD_TO_WCHAR
WORD_BCD16	INT	X	The value to be converted has data type WORD and is accepted as a BCD-coded value between -999 and +999. The result is available after conversion as an integer (in binary notation) of the type INT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	WORD_BCD16_TO_INT
BCD16	INT	X		BCD16_TO_INT

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD) are interpreted as an unsigned integer with the same bit length.. Data type BYTE is interpreted as USINT, WORD as UINT and DWORD as UDINT.

See also

WORD (Page 1914)

Implicit conversion of WORD (Page 2096)

Overview of data type conversion (Page 2091)

Explicit conversion of DWORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DWORD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DWORD[1)]	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	DWORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_BYTE
	WORD[1)]	X		DWORD_TO_WORD

	SINT	X	ENO = TRUE
			#sint1 := DWORD_TO_SINT(16#FFFF_FFFF); // -1 bis
			#sint1 := DWORD_TO_SINT(16#FFFF_FF80); // -128
			#sint1 := DWORD_TO_SINT(16#0); // 0 bis
			#sint1 := DWORD_TO_SINT(16#0000_007F); // 127
			
			ENO = FALSE
			#sint1 := DWORD_TO_SINT(16#FFFF_FF7F); // -129
			#sint1 := DWORD_TO_SINT(16#8000_0000); // -2147483648
			#sint1 := DWORD_TO_SINT(16#0000_0080); // 128 bis
			#sint1 := DWORD_TO_SINT(16#7FFF_FFFF); // 2147483647
				DWORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_USINT

	INT	X	ENO = TRUE
			#int1 := DWORD_TO_INT(16#FFFF_FFFF); // -1 bis
			#int1 := DWORD_TO_INT(16#FFFF_8000); //
				DWORD_TO_INT

Source	Target	Conversion	Explanation	Mnemonics of the instruction

			-32768
			#int1 := DWORD_TO_INT(16#0); // 0
			bis
			#int1 :=
			DWORD_TO_INT(16#0000_7FFF); //
			32767
			
			ENO = FALSE
			#int1 :=
			DWORD_TO_INT(16#FFFF_7FFF); //
			-32769
			#int1 :=
			DWORD_TO_INT(16#8000_0000); //
			-2147483648
			#int1 :=
			DWORD_TO_INT(16#8000); // 32768
			bis
			#int1 :=
			DWORD_TO_INT(16#7FFF_FFFF); //
			2147483647	

	UINT	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_UINT
	DINT	X		DWORD_TO_DINT
	UDINT	X		DWORD_TO_UDINT
	REAL	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If no errors occur during the conversion, the signal state of ENO = 1; if an error occurs during processing, the signal state of ENO = 0.	DWORD_TO_REAL
	LREAL	-	No explicit conversion	-
	TIME	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_TOD
	DATE	X		DWORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_CHAR
	WCHAR	X		DWORD_TO_WCHAR

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DWORD_BCD32	DINT	X	The value to be converted has data type DWORD and is accepted as a BCD-coded value between -9999999 and +9999999. The result is available after conversion as an integer (in binary notation) of the type DINT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	DWORD_BCD32_TO_DINT
BCD32	DINT	X		BCD32_TO_DINT
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD) are interpreted as an unsigned integer with the same bit length.. Data type BYTE is interpreted as USINT, WORD as UINT and DWORD as UDINT.				

See also

DWORD (Page 1915)

Implicit conversion of DWORD (Page 2097)

Overview of data type conversion (Page 2091)

Integers

Explicit conversion of SINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the SINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
SINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	SINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	SINT_TO_BYTE
	WORD[1)]	X		SINT_TO_WORD
	DWORD[1)]	X		SINT_TO_DWORD
	USINT	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type.	SINT_TO_USINT
	INT	X		SINT_TO_INT
	UINT	X		SINT_TO_UINT
	DINT	X		SINT_TO_DINT
	UDINT	X		SINT_TO_UDINT
	REAL	X	The value is converted into the format of the target data type (the value "-1" will be converted into the value "-1.0" with the "Convert value" (CONVERT) instruction.	SINT_TO_REAL, NORM_X
	LREAL	X		SINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	SINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0)	SINT_TO_TOD
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1)	SINT_TO_DATE
	STRING	X	The value is converted to a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	SINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		SINT_TO_WSTRING

Source	Target	Conversion	Explanation	Mnemonics of the instruction
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type.	SINT_TO_CHAR
	WCHAR[1)]	X		SINT_TO_WCHAR
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.				

See also

SINT (8-bit integers) (Page 1917)

Implicit conversion of SINT (Page 2098)

Overview of data type conversion (Page 2091)

Explicit conversion of USINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the USINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
USINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	USINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	USINT_TO_BYTE
	WORD[1)]	X		USINT_TO_WORD
	DWORD[1)]	X		USINT_TO_DWORD
	SINT	X	The bit pattern of the source value is transferred unchanged to the target data type. If the sign is changed during the conversion, the enable output ENO is set to "0".	USINT_TO_SINT
	INT	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	USINT_TO_INT
	UINT	X		USINT_TO_UINT
	DINT	X		USINT_TO_DINT
	UDINT	X		USINT_TO_UDINT
	REAL	X	The value is converted into the format of the target data type (the value "1" will be converted into the value "1.0" with the "Convert value" (CONVERT) instruction.	USINT_TO_REAL, NORM_X
	LREAL	X		USINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	USINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	USINT_TO_TOD
	DATE	X		USINT_TO_DATE
	STRING	X	The value is converted to a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	USINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		USINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the target data type.	USINT_TO_CHAR
	WCHAR[1)]	X		USINT_TO_WCHAR

Source	Target	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width (the non-existing sign is replaced with zeros) and then the bits are copied. The source type determines the interpretation.				

See also

USINT (8-bit integers) (Page 1918)

Implicit conversion of USINT (Page 2099)

Overview of data type conversion (Page 2091)

Explicit conversion of INT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the INT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
INT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	INT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	INT_TO_BYTE
	WORD[1)]	X		INT_TO_WORD
	DWORD[1)]	X		INT_TO_DWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	INT_TO_SINT
	USINT	X		INT_TO_USINT
	UINT	X		INT_TO_UINT
	DINT	X		INT_TO_DINT
	UDINT	X		INT_TO_UDINT
	REAL	X	The value is converted to the format of the target data type (the value "-1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "-1.0").	INT_TO_REAL, NORM_X
	LREAL	X		INT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	INT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0; check for 24h limit)	INT_TO_TOD
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0". (interpretation	INT_TO_DATE

Source	Target	Conversion	Explanation	Mnemonics of the instruction
			in days since 1990-1-1; check for negative value)	
	STRING	X	The value is converted to a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	INT_TO_STRING, S_CONV, VAL_STRG)
	WSTRING	X		INT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	INT_TO_CHAR
	WCHAR[1)]	X		INT_TO_WCHAR
	BCD16	X	The value to be converted has type INT and is accepted as an integer with a value between -999 and +999. The result is available after conversion as a BCD-coded number of the type WORD. A real conversion takes place. If the value is outside the target area, a synchronous error is not triggered, but rather only the status bit OV is set.	INT_TO_BCD16
	BCD16_WORD	X		INT_TO_BCD16_WORD
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.				

See also

INT (16-bit integers) (Page 1918)

Implicit conversion of INT (Page 2100)

Overview of data type conversion (Page 2091)

Explicit conversion of UINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the UINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
UINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	UINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If bits are lost in the process, the enable output ENO is set to "0".	UINT_TO_BYTE
	WORD[1)]	X		UINT_TO_WORD
	DWORD[1)]	X		UINT_TO_DWORD
	SINT	X		UINT_TO_SINT
	USINT	X		UINT_TO_USINT
	INT	X	The bit pattern of the source value is transferred unchanged to the target data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UINT_TO_INT
	DINT	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	UINT_TO_DINT
	UDINT	X		UINT_TO_UDINT
	REAL	X	The value is converted to the format of the target data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "1.0").	UINT_TO_REAL, NORM_X
	LREAL	X		UINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	UINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0; check for 24h limit)	UINT_TO_TOD
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output	UINT_TO_DATE, T_CONV

Source	Target	Conversion	Explanation	Mnemonics of the instruction
			ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1; check for negative value)	
	STRING	X	The value is converted to a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		UINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type. The enable output ENO is set to "0" in the event of overflow.	UINT_TO_CHAR
	WCHAR[1)]	X		UINT_TO_WCHAR
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data type CHAR are first extended to the necessary width (the non-existing sign is replaced with zeros) and then the bits are copied. The source type determines the interpretation.				

See also

UINT (16-bit integers) (Page 1919)

Implicit conversion of UINT (Page 2101)

Overview of data type conversion (Page 2091)

Explicit conversion of DINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	DINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	DINT_TO_BYTE
	WORD[1)]	X		DINT_TO_WORD
	DWORD[1)]	X		DINT_TO_DWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	DINT_TO_SINT
	USINT	X		DINT_TO_USINT
	INT	X		DINT_TO_INT
	UINT	X		DINT_TO_UINT
	UDINT	X		DINT_TO_UDINT
	REAL	X	The value is converted to the format of the target data type (the value "-1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "-1.0").	DINT_TO_REAL, NORM_X
	LREAL	X		DINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	DINT_TO_TIME, T_CONV
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	DINT_TO_TOD
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target	DINT_TO_DATE

Source	Target	Conversion	Explanation	Mnemonics of the instruction
			data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	
	STRING	X	The value is converted to a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	DINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		DINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	DINT_TO_CHAR
	WCHAR[1)]	X		DINT_TO_WCHAR
	BCD32	X	The value to be converted has type DINT and is accepted as an integer with a value between –999999 and +9999999. The result is available after conversion as a BCD-coded number of the type DWORD. The enable output is set to "0" in the event of overflow. A real conversion takes place. If the value is outside the target area, a synchronous error is not triggered, but rather only the status bit OV is set.	DINT_TO_BCD32
	BCD32_DWORD	X		DINT_TO_BCD32_DWORD

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

DINT (32-bit integers) (Page 1920)

Implicit conversion of DINT (Page 2102)

Overview of data type conversion (Page 2091)

Explicit conversion of UDINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the UDINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
UDINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	UDINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type. If bits are lost in the process, the enable output ENO is set to "0".	UDINT_TO_BYTE
	WORD[1)]	X		UDINT_TO_WORD
	DWORD[1)]	X		UDINT_TO_DWORD
	SINT	X		UDINT_TO_SINT
	USINT	X		UDINT_TO_USINT
	INT	X		UDINT_TO_INT
	UINT	X		UDINT_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged to the target data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UDINT_TO_DINT
	REAL	X	The value is converted to the format of the target data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "1.0").	UDINT_TO_REAL, NORM_X
	LREAL	X		UDINT_TO_LREAL, NORM_X
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the target data type.	UDINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0; check for 24h limit)	UDINT_TO_TOD, T_CONV
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output	UDINT_TO_DATE

Source	Target	Conversion	Explanation	Mnemonics of the instruction
			ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1; check for negative value)	
	STRING	X	The value is converted to a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UDINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		UDINT_TO_WCHAR
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type. The enable output ENO is set to "0" in the event of overflow.	UDINT_TO_CHAR
	WCHAR[1)]	X		UDINT_TO_WCHAR
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data type CHAR are first extended to the necessary width (the non-existent sign is replaced with zeros) and then the bits are copied. The source type determines the interpretation.				

See also

UDINT (32-bit integers) (Page 1921)

Implicit conversion of UDINT (Page 2103)

Overview of data type conversion (Page 2091)

Floating-point numbers

Explicit conversion of REAL

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the REAL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
REAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	X	The bit pattern of the source value is transferred unchanged to the target data type.	REAL_TO_DWORD
	SINT	X	The value is converted to the target data type. The result of the conversion depends on the instruction used. Enable output ENO is set to "0" if the valid range of values of the target data type is exceeded during conversion, or if the value to be converted is an invalid floating-point number.	REAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		REAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		REAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		REAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		REAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		REAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LREAL	X	The value is converted to the target data type. The result of the conversion depends on the instruction used, e.g. TRUNC(2.5) = 2.0; CEIL(2.5) = 3.0	REAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	X	The value is converted to a character string. Enable output ENO is set to "0" if the character string length is exceeded, or if the value to be converted is an invalid floating-point number. The string has a minimum length of 14 characters.	REAL_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		REAL_TO_WSTRING
	CHAR	-	No explicit conversion	-
	WCHAR	-		-

Source	Target	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible -: Conversion not possible				

See also

REAL (Page 1925)

Implicit conversion of REAL (Page 2104)

Overview of data type conversion (Page 2091)

Explicit conversion of LREAL

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the LREAL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
LREAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	SINT	X	The value is converted to the target data type. The result of the conversion depends on the instruction used. Enable output ENO is set to "0" if the valid range of values of the target data type is exceeded during conversion, or if the value to be converted is an invalid floating-point number.	LREAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		LREAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		LREAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		LREAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		LREAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		LREAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	REAL	X	The value is converted to the target data type. Enable output ENO is set to "0" if the valid range of values of the target data type is exceeded during conversion, or if the value to be converted is an invalid floating-point number. A loss in accuracy is tolerated.	LREAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	X	The value is converted to a character string. Enable output ENO is set to "0" if the character string length is exceeded, or if the value to be converted is an invalid floating-point number. The string has a minimum length of 21 characters.	LREAL_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		LREAL_TO_WSTRING
	CHAR	-	No explicit conversion	-
	WCHAR	-		-
x: Conversion possible -: Conversion not possible				

See also

LREAL (Page 1926)

Overview of data type conversion (Page 2091)

Timers

Explicit conversion of TIME

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the TIME data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
TIME	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type.	TIME_TO_BYTE
	WORD[1)]	X		TIME_TO_WORD
	DWORD[1)]	X		TIME_TO_DWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the target data type.	TIME_TO_SINT
	USINT	X		TIME_TO_USINT
	INT	X		TIME_TO_INT
	UINT	X		TIME_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion shows the duration in milliseconds.	TIME_TO_DINT, T_CONV
	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the target data type. Enable output ENO is set to "0" if the sign changes.	TIME_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If the source value exceeds the range of values of TOD, the target data type remains unchanged.	TIME_TO_TOD
	DATE	-	No explicit conversion	-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

Source	Target	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible -: Conversion not possible [1] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.				

See also

TIME (IEC time) (Page 1930)

Implicit conversion of TIME (Page 2106)

Overview of data type conversion (Page 2091)

Clock and calendar

Explicit conversion of DATE

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DATE data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DATE	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DATE_TO_BYTE
	WORD[1)]	X		DATE_TO_WORD
	DWORD[1)]	X		DATE_TO_DWORD
	SINT	X	The number of days since 1/1/1990 is returned as result.	DATE_TO_SINT
	USINT	X		DATE_TO_USINT
	INT	X		DATE_TO_INT
	UINT	X		DATE_TO_UINT
	DINT	X		DATE_TO_DINT
	UDINT	X		DATE_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	DTL	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DATE_TO_DTL
	TOD	-	No explicit conversion	-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.

See also

DATE (Page 1931)

Implicit conversion of DATE (Page 2107)

Overview of data type conversion (Page 2091)

Explicit conversion of TOD

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the TOD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
TOD	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type.	TOD_TO_BYTE
	WORD[1)]	X		TOD_TO_WORD
	DWORD[1)]	X		TOD_TO_DWORD
	SINT	X	The number of milliseconds since midnight is returned as result.	TOD_TO_SINT
	USINT	X		TOD_TO_USINT
	INT	X		TOD_TO_INT
	UINT	X		TOD_TO_UINT
	DINT	X		TOD_TO_DINT
	UDINT	X	The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).	TOD_TO_UDINT, T_CONV
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The duration since midnight is returned as result.	TOD_TO_TIME
	DTL	X	The date is set to 1.1.1970 as a result.	TOD_TO_DTL
	DATE	-	No explicit conversion	-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

-: Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

TIME_OF_DAY (TOD) (Page 1932)

Implicit conversion of TOD (Page 2108)

Overview of data type conversion (Page 2091)

Explicit conversion of DTL

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DTL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DTL	BYTE	-	No explicit conversion	-
	WORD	-		-
	DWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	REAL	-		-
	LREAL	-		-
	TIME	-		-
	TOD	X	During the conversion, the time-of-day is extracted from the DTL format and written to the target data type.	DTL_TO_TOD, T_CONV
	DATE	X	During the conversion, the date is extracted from the DTL format and written to the target data type. The enable output ENO is set to "0" in the event of overflow.	DTL_TO_DATE, T_CONV
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible -: Conversion not possible				

See also

DTL (Page 1935)

Overview of data type conversion (Page 2091)

Character strings

Explicit conversion of CHAR

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the CHAR data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
CHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	CHAR_TO_BYTE
	WORD[1)]	X		CHAR_TO_WORD
	DWORD[1)]	X		CHAR_TO_DWORD
	SINT	X		CHAR_TO_SINT
	USINT	X		CHAR_TO_USINT
	INT	X		CHAR_TO_INT
	UINT	X		CHAR_TO_UINT
	DINT	X		CHAR_TO_DINT
	UDINT	X		CHAR_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	X	The value is converted to the first character in the character string (STRING). If the length of the character string is not defined, the length "1" is set after the conversion. If the length of the character string is defined, it remains unchanged after the conversion.	CHAR_TO_STRING
	WSTRING	-	No explicit conversion	-
	WCHAR	X		CHAR_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.

See also

CHAR (character) (Page 1936)

Implicit conversion of CHAR (Page 2109)

Overview of data type conversion (Page 2091)

Explicit conversion of WCHAR

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the WCHAR data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
WCHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WCHAR_TO_BYTE
	WORD[1)]	X		WCHAR_TO_WORD
	DWORD[1)]	X		WCHAR_TO_DWORD
	SINT	X		WCHAR_TO_SINT
	USINT	X		WCHAR_TO_USINT
	INT	X		WCHAR_TO_INT
	UINT	X		WCHAR_TO_UINT
	DINT	X		WCHAR_TO_DINT
	UDINT	X		WCHAR_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	X	The value is converted to the first character in the character string (WSTRING). If the length of the character string is not defined, the length "1" is set after conversion. If the length of the character string is defined, it remains unchanged after conversion.	WCHAR_TO_WSTRING
	CHAR	X		WCHAR_TO_CHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.

Explicit conversion of STRING

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the STRING data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
STRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are permitted for conversion: • Digit • Sign • Dot The first character of the string may be a sign (+, -) or a number. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousands separator is permitted to the left of the decimal point but will be ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	STRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		STRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		STRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		STRING_TO_UINT, S_CONV, STRG_VAL
	DINT	X		STRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		STRING_TO_UDINT, S_CONV, STRG_VAL
	REAL	X		STRING_TO_REAL, S_CONV, STRG_VAL
	LREAL	X		STRING_TO_LREAL, S_CONV, STRG_VAL
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	CHAR	X	The first character in the character string (STRING) is transferred to the destination data type. If the string is empty, then the value "0" will be written in the destination data type.	STRING_TO_CHAR, S_CONV
	WCHAR	-	No explicit conversion	-
	WSTRING	X		STRING_TO_WSTRING
x: Conversion possible - : Conversion not possible				

See also

STRING (Page 1937)

Overview of data type conversion (Page 2091)

Explicit conversion of WSTRING

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the WSTRING data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
WSTRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are permitted for conversion: • Digit • Sign • Dot The first character of the string may be a sign (+, -) or a number. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousands separator is permitted to the left of the decimal point but will be ignored. If the layout of the character string is invalid for the conversion, or an overflow occurs, the enable output ENO will be set to "0".	WSTRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		WSTRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		WSTRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		WSTRING_TO_UINT, S_CONV, STRG_VAL
	DINT	X		WSTRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		WSTRING_TO_UDINT, S_CONV, STRG_VAL
	REAL	X		WSTRING_TO_REAL, S_CONV, STRG_VAL
	LREAL	X		WSTRING_TO_LREAL, S_CONV, STRG_VAL
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	CHAR	-		-
	WCHAR	X	The first character in the character string (WSTRING) is transferred to the target data type. If the string is empty, the value "0" will be written in the target data type.	WSTRING_TO_WCHAR
	STRING	X		WSTRING_TO_STRING
x: Conversion possible - : Conversion not possible				

11.5.17 Data type conversion for S7-300, S7-400

11.5.17.1 Overview of data type conversion

Introduction

If you combine several operands in an instruction, you must make sure that the data types are compatible. This also applies when you assign or supply block parameters. If the operands are not of the same data type, a conversion has to be carried out.

There are two options for conversion:

- Implicit conversion
  Implicit conversion is supported by the programming languages LAD, FBD, SCL and GRAPH. Implicit conversion is not possible in the STL programming language.
- Explicit conversion

Note

Converting bit strings to SCL

All bit strings (BYTE, WORD, DWORD and LWORD) are handled like the corresponding unsigned integers (USINT, UINT, UDINT and ULINT) in expressions. Therefore, implicit conversion from DWORD to REAL is carried out like a conversion from UDINT to REAL, for example.

Implicit conversion

Implicit conversion is executed automatically if the data types of the operands are compatible. This compatibility test can be carried out according to strict or less strict criteria:

- With IEC check
  The following rules apply in the LAD, FBD and GRAPH programming languages when the IEC check has been set:
  - The only data types for which implicit conversion is possible are BYTE and WORD.
  - The bit length of the source data type may not exceed the bit length of the target data type. A WORD data type operand, for example, cannot be specified at a parameter at which the BYTE data type is expected.

  The following rules apply in the SCL programming language when the IEC check has been set:
  - Implicit conversion of bit strings into other data types is not possible. A WORD data type operand, for example, cannot be specified at a parameter at which the INT data type is expected.
  - The bit length of the source data type may not exceed the bit length of the target data type. A WORD data type operand, for example, cannot be specified at a parameter at which the BYTE data type is expected.
- Without IEC check (default setting)
  The following rules apply in the LAD, FBD and GRAPH programming languages when the IEC check has not been set:
  - Implicit conversion of data types BYTE, WORD, DWORD, INT, DINT, TIME, S5TIME, TOD, DATE and CHAR is possible.
  - The bit length of the source data type may not exceed the bit length of the target data type. A DWORD data type operand, for example, cannot be specified at a parameter at which the WORD data type is expected.

  The following rules apply in the SCL programming language when the IEC check has not been set:
  - Bit strings can be implicitly converted into other data types. A WORD data type operand can, for example, be given at a parameter at which the WORD data type is expected.
  - Bit strings cannot be implicitly converted into floating-point numbers. A WORD data type operand, for example, cannot be specified at a parameter at which the REAL data type is expected.
  - Bit strings can only be implicitly converted into the data types TIME, TOD, DATE and CHAR if these have the same bit length. A DWORD data type operand, for example, cannot be specified at a parameter at which the DATE data type is expected.
  - The bit length of the source data type may not exceed the bit length of the target data type. A DINT data type operand, for example, cannot be specified at a parameter at which the INT data type is expected.
  - The bit length of an operand entered at in-out parameters must be the same as the programmed bit length for the parameter in question.

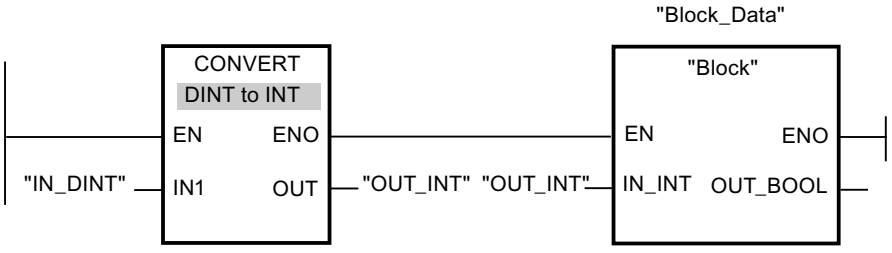
Explicit conversion

If the operands are not compatible and implicit conversion is therefore not possible, you can use an explicit conversion instruction. You can find the conversion instructions in the "Instructions" task card.

Any overflow will be displayed on the ENO enable output. An overflow occurs, for example, if the value of the source data type is greater than the value of the target data type.

You can find additional information on explicit conversion under "See also".

The following figure shows an example in which explicit data type conversion must be carried out:



The "Block" function block expects a tag of the INT data type at the "IN_INT" input parameter. The value of the "IN_DINT" tag has therefore to the converted first from DINT to INT. Conversion is performed if the value of the "IN_DINT" tag is within the admissible value range for the INT data type. Otherwise an overflow is reported. However, a conversion takes place even in the event of an overflow, but the values are truncated and the ENO enable output is set to "0".

See also

Setting and disabling the IEC check (Page 2159)

Implicit conversion (Page 2159)

Explicit conversion (Page 2171)

11.5.17.2 Implicit conversion

Setting and disabling the IEC check

The data types of the operands used are checked for compatibility. This compatibility test can be carried out according to criteria that are more or less strict. If "IEC check" is activated, stricter criteria are applied.

You can set the IEC check centrally for all new blocks of the project or for individual blocks.

Setting IEC check for new blocks

To set the IEC check for all new blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. Select or clear the "IEC check" check box in the "Default settings for new blocks" group.
   The IEC check is enabled or disabled for all new blocks in the program.

Setting IEC check for a block

To set the IEC check for a block, proceed as follows:

1. Open the block.
2. Open the "Properties" tab in the inspector window.
3. Select the "Attributes" group in the area navigation.
4. Select or clear the "IEC check" check box.
   The IEC check is enabled or disabled for this block. The setting is stored together with the project.

See also

Overview of data type conversion (Page 2157)

Binary numbers

Implicit conversion of BOOL

Implicit conversion options

The BOOL data type cannot be implicitly converted.

See also

BOOL (bit) (Page 1912)

Bit strings

Implicit conversion of BYTE

Implicit conversion options

The following table shows the options for the implicit conversion of BYTE data type:

Source	Target	With IEC check	Without IEC check	Explanation
BYTE	BOOL	-	-	No implicit conversion
	WORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	DWORD	X	X	
	INT	-	-	No implicit conversion
	DINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	REAL	-	-	No implicit conversion
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged to the target data type.
x: Conversion possible -: Conversion not possible				

See also

BYTE (byte) (Page 1913)

Overview of data type conversion (Page 2157)

Explicit conversion of BYTE (Page 2172)

Setting and disabling the IEC check (Page 2159)

Implicit conversion of WORD

Implicit conversion options

The following table shows the options for the implicit conversion of WORD data type:

Source	Target	With IEC check	Without IEC check	Explanation
WORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	Only the low byte is transferred to the destination data type.
	DWORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	INT	-	X	
	DINT	-	X	
	REAL	-	-	No implicit conversion
	TIME	-	-	
	S5TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	DT	-	-	No implicit conversion
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	STRING	-	-	No implicit conversion
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

WORD (Page 1914)

Overview of data type conversion (Page 2157)

Explicit conversion of WORD (Page 2174)

Setting and disabling the IEC check (Page 2159)

Implicit conversion of DWORD

Implicit conversion options

The following table shows the options for the implicit conversion of DWORD data type:

Source	Target	With IEC check	Without IEC check	Explanation
DWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	WORD	-	X	
	INT	-	-	No implicit conversion
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	REAL	-	-	No implicit conversion
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	S5TIME	-	-	No implicit conversion
	DT	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DWORD (Page 1915)

Overview of data type conversion (Page 2157)

Explicit conversion of DWORD (Page 2176)

Setting and disabling the IEC check (Page 2159)

Integers

Implicit conversion of INT

Options for implicit conversion

The following table shows the options for the implicit conversion of INT data type:

Source	Target	With IEC check	Without IEC check	Explanation
INT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	DWORD	-	-	No implicit conversion
	DINT [1)]	X	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	REAL	X	X	
	TIME	-	-	No implicit conversion
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	

x: Conversion possible
-: Conversion not possible
[1)]: Only possible in SCL

See also

INT (16-bit integers) (Page 1918)
Overview of data type conversion (Page 2157)
Explicit conversion of INT (Page 2179)
Setting and disabling the IEC check (Page 2159)

Implicit conversion of DINT

Implicit conversion options

The following table shows the options for the implicit conversion of DINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
DINT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	INT	-	-	No implicit conversion
	REAL	-	-	
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	S5TIME	-	-	No implicit conversion
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DINT (32-bit integers) (Page 1920)

Overview of data type conversion (Page 2157)

Explicit conversion of STRING (Page 2192)

Setting and disabling the IEC check (Page 2159)

Floating-point numbers

Implicit conversion of REAL

Implicit conversion options

The REAL data type cannot be implicitly converted.

See also

REAL (Page 1925)

Overview of data type conversion (Page 2157)

Explicit conversion of CHAR (Page 2191)

Setting and disabling the IEC check (Page 2159)

Timers

Implicit conversion of TIME

Implicit conversion options

The following table shows the options for the implicit conversion of TIME data type:

Source	Target	With IEC check	Without IEC check	Explanation
TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	INT	-	-	No implicit conversion
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	REAL	-	-	No implicit conversion
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME (IEC time) (Page 1930)

Overview of data type conversion (Page 2157)

Explicit conversion of TIME (Page 2186)

Setting and disabling the IEC check (Page 2159)

Implicit conversion of S5TIME

Implicit conversion options

The following table shows the options for the implicit conversion of S5TIME data type:

Source	Target	With IEC check	Without IEC check	Explanation
S5TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	DWORD	-	-	No implicit conversion
	INT	-	-	
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

S5TIME (duration) (Page 1929)

Overview of data type conversion (Page 2157)

Explicit conversion of S5TIME (Page 2187)

Setting and disabling the IEC check (Page 2159)

Date and time

Implicit conversion of DATE

Implicit conversion options

The following table shows the options for the implicit conversion of DATE data type:

Source	Target	With IEC check	Without IEC check	Explanation
DATE	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of days since 01/01/1990.
	DWORD	-	-	No implicit conversion
	INT	-	-	
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DATE (Page 1931)
Overview of data type conversion (Page 2157)
Setting and disabling the IEC check (Page 2159)

Implicit conversion of TOD

Implicit conversion options

The following table shows the options for the implicit conversion of TOD data type:

Source	Target	With IEC check	Without IEC check	Explanation
TOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	INT	-	-	No implicit conversion
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME_OF_DAY (TOD) (Page 1932)

Overview of data type conversion (Page 2157)

Setting and disabling the IEC check (Page 2159)

Implicit conversion of DT

Implicit conversion options

The DT data type cannot be implicitly converted.

See also

DATE_AND_TIME (date and time of day) (Page 1933)

Overview of data type conversion (Page 2157)

Explicit conversion of DT (Page 2190)

Setting and disabling the IEC check (Page 2159)

Character strings

Implicit conversion of CHAR

Implicit conversion options

The following table shows the options for the implicit conversion of CHAR data type:

Source	Target	With IEC check	Without IEC check	Explanation
CHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	WORD	-	-	No implicit conversion
	DWORD	-	-	
	INT	-	-	
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	X	X	The bit pattern of the source value is transferred unchanged to the destination data type.
x: Conversion possible -: Conversion not possible				

See also

CHAR (character) (Page 1936)

Overview of data type conversion (Page 2157)

Explicit conversion of CHAR (Page 2191)

Setting and disabling the IEC check (Page 2159)

Implicit conversion of STRING

Implicit conversion options

The STRING data type cannot be implicitly converted.

See also

STRING (Page 1937)

Overview of data type conversion (Page 2157)

Explicit conversion of STRING (Page 2192)

Setting and disabling the IEC check (Page 2159)

11.5.17.3 Explicit conversion

Binary numbers

Explicit conversion of BOOL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BOOL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
BOOL	BYTE	X	Only the LSB (Least Significant Bit) is set in the destination data type. The enable output ENO is always "1".	CONVERT
	WORD	X		
	DWORD	X		
	INT	X		BOOL_TO_INT
	DINT	X		BOOL_TO_DINT
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion (Page 2159)

Overview of data type conversion (Page 2157)

BOOL (bit) (Page 1912)

Bit strings

Explicit conversion of BYTE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BYTE data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
BYTE	BOOL	X	The following possibilities can occur: • If the source is "0", the target data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the target data type is set according to LSB and the enable output ENO is "0".	CONVERT
	WORD	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. The remaining bits are set to "0".	CONVERT
	DWORD	X		
	INT	X		
	DINT	X		
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. The remaining bits are set to "0".	CONVERT
x: Conversion possible - : Conversion not possible				

See also

BYTE (byte) (Page 1913)

Overview of data type conversion (Page 2157)

Implicit conversion of BYTE (Page 2161)

Explicit conversion of WORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WORD	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	CONVERT
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the permitted value range of the destination data type is violated, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	
	DWORD	X		
	INT	X		
	DINT	X		
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	MOVE
	DT	-	No explicit conversion	-
	TOD	-		
	DATE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	MOVE
	STRING	-	No explicit conversion	-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_CHAR
	BLOCK_DB	X	The bit pattern of WORD is interpreted as a data block number.	WORD_TO_BLOCK_DB

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WORD_BCD[1)]	INT	X	If the permitted value range of the destination data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	WORD_BCD_TO_INT
BCD[1)]	INT	X		BCD_TO_INT

x: Conversion possible

- : Conversion not possible

[1)] The value to be converted has data type WORD and is accepted as a BCD-coded value between -999 and +999. The result is available after conversion as an integer (in binary notation) of the type INT.

See also

WORD (Page 1914)

Overview of data type conversion (Page 2157)

Implicit conversion of WORD (Page 2162)

Explicit conversion of DWORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DWORD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DWORD	BOOL	X	The following possibilities can occur: • If the source is "0", the target data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and the enable output ENO is "0".	CONVERT
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	
	WORD	X		

	INT	X	ENO = TRUE
			#int1 := DWORD_TO_INT(16#FFFF_FFFF); // -1 bis
			#int1 := DWORD_TO_INT(16#FFFF_8000); // -32768
			#int1 := DWORD_TO_INT(16#0); // 0 bis
			#int1 := DWORD_TO_INT(16#0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := DWORD_TO_INT(16#FFFF_7FFF); // -32769
			#int1 := DWORD_TO_INT(16#8000_0000); // -2147483648
			#int1 := DWORD_TO_INT(16#8000); // 32768 bis
			#int1 := DWORD_TO_INT(16#7FFF_FFFF); // 2147483647	

	DINT	X	The bit pattern of the source value is transferred unchanged	MOVE
	REAL	X		
	TIME	X		

Source	Target	Conversion	Explanation	Mnemonics of the instruction
			right-justified to the target data type.	
	S5TIME	-	No explicit conversion	-
	DT	-		
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	MOVE
	DATE	-	No explicit conversion	-
	STRING	-		
	CHAR	-		
DWORD_BCD[1)]	DINT	X	If the permitted value range of the target data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	DWORD_BCD_TO_DINT
BCD[1)]	DINT	X		BCD_TO_DINT

x: Conversion possible

- : Conversion not possible

[1)] The value to be converted has data type DWORD and is accepted as a BCD-coded value between -9999999 and +9999999. The result is available after conversion as an integer (in binary notation) of the type DINT.

See also

DWORD (Page 1915)

Overview of data type conversion (Page 2157)

Implicit conversion of DWORD (Page 2163)

Integers

Explicit conversion of INT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the INT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
INT	BOOL	X	The value is first converted to WORD and then to BOOL. The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	INT_TO_BOOL
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The enable output ENO is set to "0" if a negative value is converted to an unsigned destination data type or if overflow occurs.	CONVERT
	WORD	X		MOVE
	DWORD	X		
	DINT	X		CONVERT, ITD
	REAL	X	The value is converted into the format of the destination data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) into the value "1.0").	CONVERT, SCALE, ITR
	TIME	-	No explicit conversion	-
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	S_CONV, CONVERT
	CHAR	X	The bit pattern of the source value is transferred unchanged to the destination data type. With the conversion of negative values or with an overflow, the enable output ENO will be set to "0".	CONVERT
	BCD[1)]	X	If the permitted value range of the destination data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	INT_TO_BCD
	BCD_WORD [1)]	X		INT_TO_BCD_WORD

x: Conversion possible

- : Conversion not possible

[1)] The value to be converted has type INT and is accepted as an integer with a value between -999 and +999. The result is available after conversion as a BCD-coded number of the type WORD.

See also

INT (16-bit integers) (Page 1918)

Overview of data type conversion (Page 2157)

Implicit conversion of INT (Page 2164)

Explicit conversion of DINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DINT	BOOL	X	The value is first converted to DWORD and then to BOOL. The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	DINT_TO_BOOL
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The enable output ENO is set to "0" if a negative value is converted to an unsigned destination data type or if overflow occurs.	CONVERT
	WORD	X		
	DWORD	X		MOVE
	INT	X		CONVERT
	REAL	X	The value is converted into the format of the destination data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) into the value "1.0").	CONVERT, DTR, SCALE
	TIME	X	The bit pattern of the source value is transferred unchanged to the destination data type.	T_CONV, MOVE
	S5TIME	-	No explicit conversion	-
	DT	-		
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	CONVERT
	DATE	X		
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	S_CONV, CONVERT
	CHAR	X	The bit pattern of the source value is transferred unchanged to the destination data type. With the conversion of negative values or with an overflow, the enable output ENO will be set to "0".	DINT_TO_CHAR
	BCD[1)]	X	If the permitted value range of the destination data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	DINT_TO_BCD
	BCD_DWORD[1)]	X		DINT_TO_BCD_DWORD

Source	Target	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible				

See also

DINT (32-bit integers) (Page 1920)

Overview of data type conversion (Page 2157)

Implicit conversion (Page 2159)

Floating-point numbers

Explicit conversion of REAL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the REAL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
REAL	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	X	The bit pattern of the source value is transferred unchanged to the destination data type.	CONVERT, MOVE
	INT	X	The value is converted into the destination data type. The result of conversion depends on the instruction employed. The enable output ENO is set to "0" if the admissible range of values of the destination data type is exceeded during conversion or the value to be converted has an invalid floating-point number.	CONVERT, ROUND, RND, CEIL, RND+, FLOOR, RND-, TRUNC, UNSCALE
	DINT	X		
	TIME	-	No explicit conversion	-
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	X	The value is converted into a character string. The enable output ENO is set to "0" if the character string length is exceeded or the value to be converted has an invalid floating-point number.	S_CONV, CONVERT
	CHAR	-	No explicit conversion	-

x: Conversion possible

- : Conversion not possible

See also

REAL (Page 1925)

Overview of data type conversion (Page 2157)

Implicit conversion (Page 2159)

Timer operations

Explicit conversion of TIME

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TIME data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
TIME	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.	CONVERT
	INT	-	No explicit conversion	-
	DINT	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.	T_CONV, CONVERT
	REAL	-	No explicit conversion	-
	S5TIME	X	The value is converted to the destination data type format. The enable output ENO is set to "0" in the event of overflow.	T_CONV, CONVERT
	DT	-	No explicit conversion	-
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

TIME (IEC time) (Page 1930)

Overview of data type conversion (Page 2157)

Implicit conversion of TIME (Page 2166)

Explicit conversion of S5TIME

Options for explicit conversion in LAD, FBD, STL and GRAPH

The following table shows the options and instructions for the explicit conversion of the S5TIME data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
S5TIME	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	X	The value is converted to the destination data type format.	S5TIME_TO_WORD
	DWORD	-	No explicit conversion	-
	INT	-		
	DINT	-		
	REAL	-		
	TIME	X	The value is converted to the destination data type format.	T_CONV, CONVERT
	DT	-	No explicit conversion	-
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

S5TIME (duration) (Page 1929)

Overview of data type conversion (Page 2157)

Implicit conversion of S5TIME (Page 2167)

Date and time

Explicit conversion of DATE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DATE data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DATE	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	X	The value is converted to the destination data type format.	DATE_TO_WORD
	DWORD	-	No explicit conversion	-
	INT	X	The value is converted to the destination data type format.	DATE_TO_INT
	DINT	X		DATE_TO_DINT
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

DATE (Page 1931)

Overview of data type conversion (Page 2157)

Implicit conversion of DATE (Page 2168)

Explicit conversion of TOD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TOD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
TOD	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	X	The value is converted to the destination data type format.	TOD_TO_DWORD
	INT	-	No explicit conversion	-
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TOD_TO_DINT
	REAL	-	No explicit conversion	-
	TIME	X	The value is converted to the destination data type format.	TOD_TO_TIME
	S5TIME	-	No explicit conversion	-
	DT	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

TIME_OF_DAY (TOD) (Page 1932)

Overview of data type conversion (Page 2157)

Implicit conversion of TOD (Page 2169)

Explicit conversion of DT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DT	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	-		
	INT	-		
	DINT	-		
	REAL	-		
	TIME	-		
	S5TIME	-		
	TOD	X	During conversion, times are extracted from the DTL format and transferred to the destination data type.	T_CONV, CONVERT
	DATE	X	During the conversion, the date information is extracted from the DTL format and transferred to the destination data type.	
	STRING	-	No explicit conversion	-
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

DATE_AND_TIME (date and time of day) (Page 1933)

Overview of data type conversion (Page 2157)

String

Explicit conversion of CHAR

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the CHAR data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
CHAR	BOOL	-	No explicit conversion	-
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	CONVERT
	WORD	X		
	DWORD	X		
	INT	X		
	DINT	X		
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	X	The value is converted into the first character in the character string (STRING). The length "1" is set after conversion if the character string length is not defined. If the character string length is defined, it remains unchanged following conversion.	S_CONV, CONVERT
x: Conversion possible - : Conversion not possible				

See also

CHAR (character) (Page 1936)

Overview of data type conversion (Page 2157)

Implicit conversion of CHAR (Page 2170)

Explicit conversion of STRING

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the STRING data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
STRING	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	-		
	INT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are admissible for conversion: • Digit • Sign • Dot The first character in the character string may be a sign (+, -) or a digit. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator to the left of the decimal point is permitted but is ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	S_CONV, CONVERT
	DINT	X		
	REAL	X		
	TIME	-	No explicit conversion	-
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	CHAR	X	The first character in the character string (STRING) is transferred to the target data type. The value "0" is written to the target data type if the character string is empty.	S_CONV, CONVERT
x: Conversion possible - : Conversion not possible				

See also

STRING (Page 1937)

Overview of data type conversion (Page 2157)

Implicit conversion (Page 2159)

Additional conversion functions

Additional explicit conversion functions

Additional options for explicit conversion in SCL

The following table shows the additional options and operations for explicit conversion in SCL:

Source	Target	Explanation	Mnemonics of the instruction
WORD	BLOCK_DB	The bit pattern of WORD is interpreted as a data block number.	WORD_TO_BLOCK_DB
BLOCK_DB	WORD	The data block number is interpreted as a bit pattern of WORD.	BLOCK_DB_TO_WORD

11.6 Instructions

11.6.1 General parameters of the instructions

11.6.1.1 Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions

Asynchronous instructions

For instructions that work asynchronously, the function is executed with several calls.

Identification of the job

If you use asynchronous instructions to trigger a process interrupt, output control commands to DP slaves, start a data transfer, or abort a non-configured connection with one of the SFCs listed above and then call the same SFC again before the current job is completed, then the reaction of the SFC will depend on whether the second call involves the same job.

Parameter REQ

The input parameter REQ (request) is used solely to start the job:

- If you call the instruction for a job that is not currently active, the job is started with REQ = 1 (case 1).
- If a particular job has been started and not yet completed and you call the instruction again to perform the same job (for example, in a cyclic interrupt OB), then REQ is not evaluated by the instruction (case 2).

Parameter RET_VAL and BUSY

The output parameters RET_VAL and BUSY indicate the status of the job.

Pay attention to the note in section: Evaluating errors with output parameter RET_VAL (Page 2195)

- In case 1 (first call with REQ=1), the input parameter will be entered in RET_VALW#16#7001 if system resources are available and supply is correct. BUSY will be set.
  If the required system resources are currently being used or the input parameters have errors, the corresponding error code is entered in RET_VAL and BUSY has the value 0.
- In case 2 (interim call) W#16#7002 will be entered in RET_VAL (this corresponds to a warning: Job still being processed!), and BUSY will be set.
- The following applies to the last call for a job:
  - For instruction "DPNRM_DG (Page 3139)", the number of data in bytes will entered as integer in RET_VAL in case there are no errors in data transmission. BUSY has the value "0" in this case.
    If there is an error, then the error information will be entered in RET_VAL and you should not evaluate BUSY in this case.
  - For all other instructions, "0" will be entered in RET_VAL if the job was executed without errors and BUSY has the value "0" in this case. If there is an error, the error code is entered in RET_VAL and BUSY has the value "0" in this case.

Note

If the first and last call coincide, the reaction is the same for RET_VAL and BUSY as described for the last call.

Overview

The following table provides you with an overview of the relationships explained above. In particular, it shows the possible values of the output parameters if the execution of the job is not completed after an instruction call has been completed.

Note

Following every call, you must evaluate the relevant output parameters in your program.

Relationship between call, REQ, RET_VAL and BUSY during execution of a "running" job.

Number of the call	Type of call	REQ	RET_VAL	BUSY
1	First call	1	W#16#7001	1
			Error code	0
2 to (n - 1)	Intermediate call	irrelevant	W#16#7002	1
n	Last call	irrelevant	W#16#0000, if no errors have occurred.	0
			Error code if errors occurred	0

11.6.1.2 Evaluating errors with output parameter RET_VAL

Types of error information

An executed instruction indicates in the user program whether or not the CPU was able to execute the function of the instruction successfully.

You can obtain information about any errors that occurred in two ways:

- In the BR bit of the status word
- in the output parameter RET_VAL (return value).

Note

Before evaluating the output parameters specific to an instruction, you should always follow the steps below:

- First, evaluate the BR bit of the status word.
- Then check the output parameter RET_VAL.

If the BR bit indicates that an error has occurred or if RET_VAL contains a general error code, you should not evaluate the instruction-specific output parameters.

Error information in the return value

An instruction indicates that an error occurred during its execution by entering the value "0" in the binary result bit (BR) of the status word. Some instructions provide an additional error code at an output parameter known as the return value (RET_VAL). If a general error is entered in the output parameter RET_VAL (see below for explanation), this is only indicated by the value "0" in the BR bit of the status word.

The return value is of the data type integer (INT). The relationship of the return value to the value "0" indicates whether or not an error occurred during execution of the function.

CPU execution of the instruction	BR	Return value	Sign of the integer
With error(s)	0	less than "0"	negative (sign bit is "1")
Without error	1	greater than or equal to "0"	positive (sign bit is "0")

Reacting to error information

There are two types of error codes in RET_VAL:

- A general error code that all instructions can output and
- A specific error code that an instruction can output and which relates to its specific function.

You can write your program so that it reacts to the errors that occur during execution of an instruction. This way you prevent further errors occurring as a result of the first error.

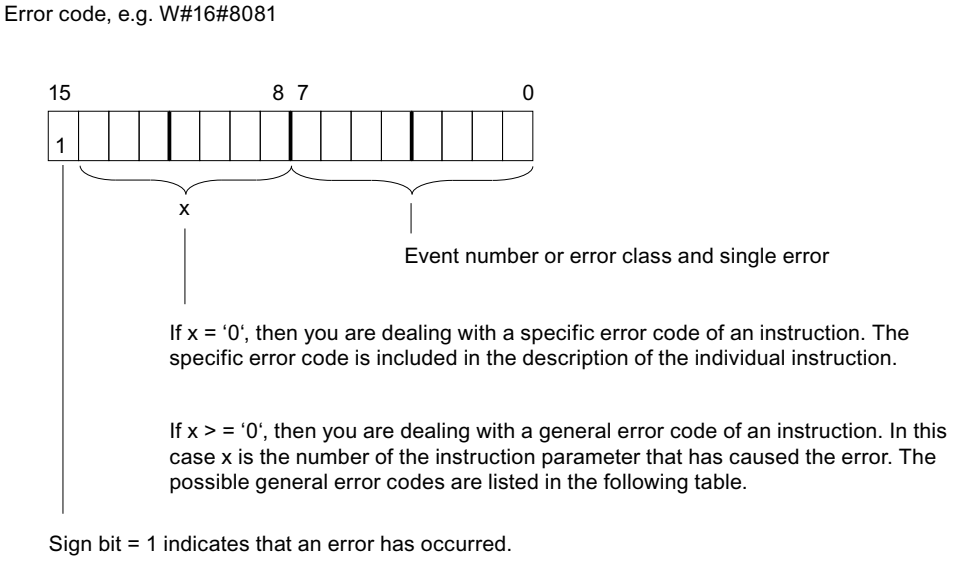
General and specific error information

The return value (RET_VAL) of an instruction provides one of the two following types of error codes:

- A general error code that relates to errors that can occur in any instruction.
- A specific error code that relates only to the particular instruction.

Even though the data type of the output parameter RET_VAL is an integer (INT), the error codes for the instruction are grouped according to hexadecimal values. If you want to examine a return value and compare the value with the error codes listed in this documentation, then display the error code in hexadecimal format.

The figure below shows the structure of a system function error code in hexadecimal format.

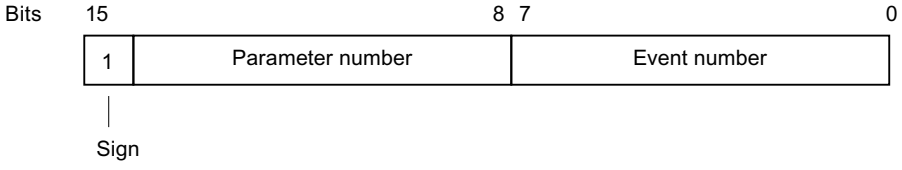


General error information

The general error code indicates errors that can occur in all instructions. A general error code consists of the following two numbers:

- A parameter number from 1 to 111, where 1 indicates the first parameter of the called instruction, 2 the second parameter, and so forth.
- An event number from 0 to 127. The event number indicates that a synchronous error occurred.

The following table lists the codes for general errors and an explanation of each error.



Note

If a general error code was entered in RET_VAL, then the following situations are possible:

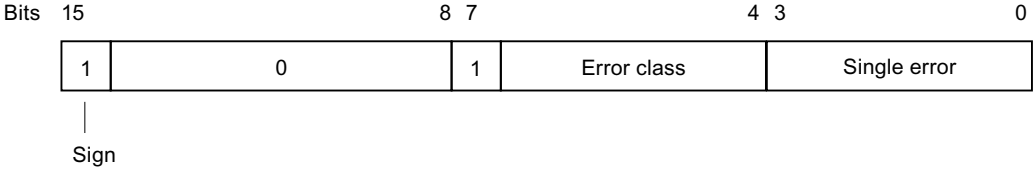
- The action associated with the instruction may have been started or already completed.
- A specific instruction error may have occurred when the action was performed. As a result of a general error that occurred later, the specific error could, however, no longer be indicated.

Specific error information

Some instructions have a return value that provides an error code specific for the instruction. A specific error code indicates errors that can occur only in specific instructions.

A specific error code consists of the following two numbers:

- An error class from 0 to 7.
- An error number from 0 to 15.



General error codes

The following table explains the general error codes of a return value. The error code is shown in hexadecimal format. The letter x in each code number is simply a place holder and represents the number of the system function parameter that caused the error.

General error codes

Error code (W#16#...)	Explanation
8x7F	Internal error This error code indicates an internal error at parameter x.
8x01	Illegal syntax ID at an VARIANT parameter
8x22	Range length error when reading a parameter.
8x23	Range length error when writing a parameter. This error code indicates that the parameter x is located either entirely or partly outside the range of an address, or that the length of a bit range is not a multiple of 8 with an VARIANT parameter.
8x24	Range error when reading a parameter.
8x25	Range error when writing a parameter. This error code indicates that the parameter x is located in a range that is illegal for the system function. Refer to the descriptions of the individual functions for information about the illegal ranges.

Error code (W#16#...)	Explanation
8x26	The parameter contains a timer cell number that is too high. This error code indicates that the timer cell specified in parameter x does not exist.
8x27	The parameter contains a counter cell number that is too high (counter number error). This error code indicates that the counter cell specified in parameter x does not exist.
8x28	Alignment error when reading a parameter.
8x29	Alignment error when writing a parameter. This error code indicates that the reference to parameter x is an operand with bit address that is not equal to 0.
8x30	The parameter is located in a read-only global DB.
8x31	The parameter is located in a read-only instance DB. This error code indicates that parameter x is located in a read-only data block. If the data block was opened by the system function itself, the system function always returns the value W#16#8x30.
8x32	The parameter contains a DB number that is too high (DB number error).
8x34	The parameter contains an FC number that is too high (FC number error).
8x35	The parameter contains an FB number that is too high (FB number error). This error code indicates that parameter x contains a block number higher than the highest permitted number.
8x3A	The parameter contains the number of a DB that is not loaded.
8x3C	The parameter contains the number of an FC that is not loaded.
8x3E	The parameter contains the number of an FB that is not loaded.
8x42	An access error occurred while the system was attempting to read a parameter from the peripheral input area.
8x43	An access error occurred while the system was attempting to write a parameter to the peripheral output area.
8x44	Error in the nth (n > 1) read access after an error occurred.
8x45	Error in the nth (n > 1) write access after an error occurred. This error code indicates that access to the required parameter is denied.

11.6.2 Basic instructions

11.6.2.1 LAD

Bit logic operations

---| |---: Normally open contact

Description

The activation of the normally open contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally open contact closes and the signal state at the output is set to the signal state of the input.

When the operand has signal state "0", the normally open contact is not activated and the signal state at the output of the instruction is reset to "0".

Two or more normally open contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally open contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

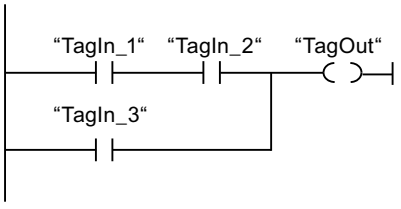
Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".

See also

Overview of the valid data types (Page 1908)

Example of controlling a conveyor belt (Page 3861)

Example of detecting the direction of a conveyor belt (Page 3863)

Example of detecting the fill level of a storage area (Page 3864)

Example of controlling room temperature (Page 3868)

---| / |---: Normally closed contact

Description

The activation of the normally closed contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally closed contact opens and the signal state at the output of the instruction is reset to "0".

When the operand has signal state "0", the normally closed contact is not enabled and the signal state of the input is transferred to the output.

Two or more normally closed contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally closed contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

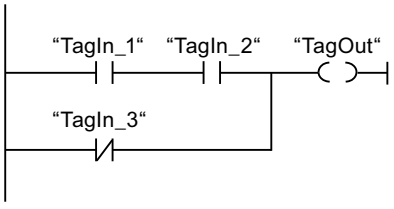
Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 1908)

Example of controlling a conveyor belt (Page 3861)

Example of detecting the direction of a conveyor belt (Page 3863)

Example of detecting the fill level of a storage area (Page 3864)

Example of controlling room temperature (Page 3868)

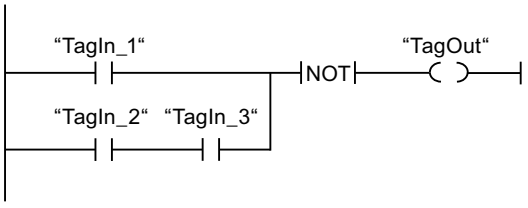
--|NOT|--: Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). If the signal state is "1" at the input of the instruction, the output of the instruction has signal state "0". If the signal state is "0" at the input of the instruction, the output has the signal state "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The signal state of the operands "TagIn_2" and "TagIn_3" is "1".

---( )---: Assignment

Description

You can use the "Assignment" instruction to set the bit of a specified operand. If the result of logic operation (RLO) at the input of the coil has signal state "1", the specified operand is set to signal state "1". If the signal state is "0" at the input of the coil, the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output.

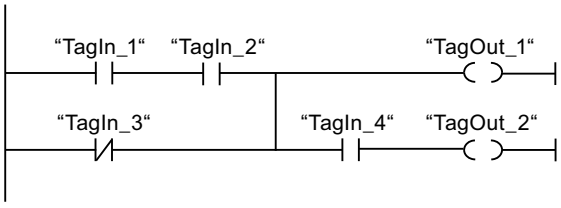
Parameter

The following table shows the parameters of the "Assignment" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



The "TagOut_1" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

The "TagOut_2" operand is set when one of the following conditions is fulfilled:

- Operands "TagIn_1", "TagIn_2", and "TagIn_4" have signal state "1".
- The signal state of the "TagIn_3" operand is "0" and the signal state of the "TagIn_4" operand is "1".

See also

Overview of the valid data types (Page 1908)

Example of detecting the fill level of a storage area (Page 3864)

Example of controlling room temperature (Page 3868)

--(/)--: Negate assignment

Description

The "Negate assignment" instruction inverts the result of logic operation (RLO) and assigns it to the specified operand. When the RLO at the input of the coil is "1", the operand is reset. When the RLO at the input of the coil is "0", the operand is set to signal state "1".

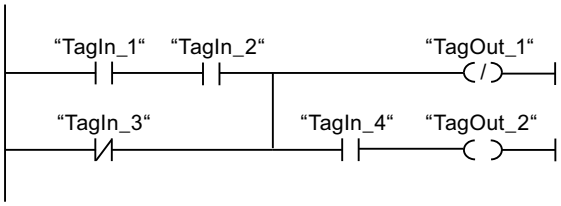
Parameter

The following table shows the parameters of the "Negate assignment" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



Operand "TagOut_1" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 1908)

---(R)---: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is reset to "0". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

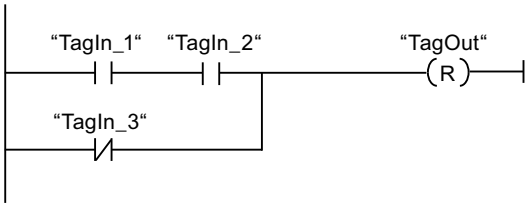
Parameter

The following table shows the parameters of the "Reset output" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Operand that is reset when RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 1908)

Example of controlling a conveyor belt (Page 3861)

Example of detecting the direction of a conveyor belt (Page 3863)

---(S)---: Set output

Description

You can use the "Set output" instruction to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is set to "1". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

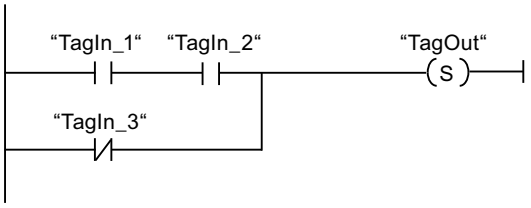
Parameter

The following table shows the parameters of the "Set output" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 1908)

Example of controlling a conveyor belt (Page 3861)

Example of detecting the direction of a conveyor belt (Page 3863)

SET_BF: Set bit field

Description

You use the instruction "Set bit field" to set multiple bits starting from a certain address.

You determine the number of bits to be set using the value of <Operand1>. The address of the first bit to be set is defined by <Operand2>. If the value of <Operand1> is greater than the number of bits in a selected byte, then the bits of the next byte will be set. The bits remain set until they are explicitly reset, for example, by another instruction.

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If the RLO at the input of the coil is "0", the instruction does not execute.

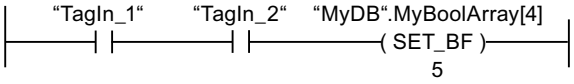
Parameter

The following table shows the parameters of the "Set bit field" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand2>	Output	BOOL	I, Q, M In the case of a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be set.
<Operand1>	Input	UINT	Constant	Number of bits to be set.

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are set starting at the address of the operand "MyDB".MyBoolArray[4]

See also

Overview of the valid data types (Page 1908)

RESET_BF: Reset bit field

Description

You use the "Reset bit field" instruction to reset several bits starting from a certain address.

You specify the number of bits to be reset using the value of <Operand1>. The address of the first bit to be reset is specified by <Operand2>. If the value of <Operand1> is greater than the number of bits in a selected byte, the bits of the next byte will be reset. The bits remain reset until they are explicitly set, for example, by another instruction.

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If the RLO at the input of the coil is "0", the instruction does not execute.

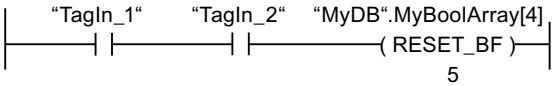
Parameter

The following table shows the parameters of the "Reset bit field" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand2>	Output	BOOL	I, Q, M In the case of a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be reset.
<Operand1>	Input	UINT	Constant	Number of bits to be reset.

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are reset starting at the address of the operand "MyDB".MyBoolArray[4]

See also

Overview of the valid data types (Page 1908)

SR: Set/reset flip-flop

Description

Use the instruction "Set/reset flip-flop" to set or reset the bit of the specified operand, depending on the signal state of the inputs S and R1. If the signal state is "1" at input S and "0" at input R1, the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

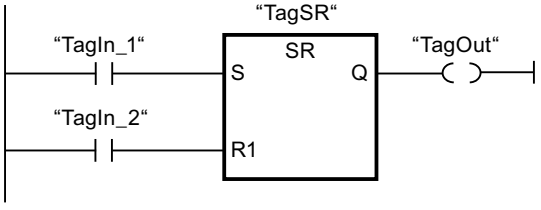
Parameter

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
S	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable setting
R1	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable reset-ting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is set or reset.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 1908)

RS: Reset/set flip-flop

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

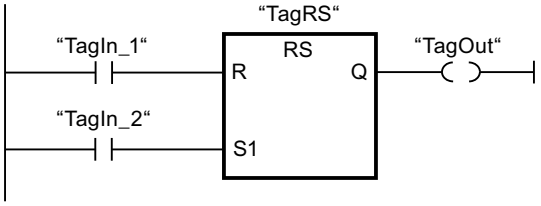
Parameter

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
R	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable reset-ting
S1	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable setting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is reset or set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 1908)

--|P|--: Scan operand for positive signal edge

Description

You can use the "Scan operand for positive signal edge" instruction to determine if there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan that is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a positive edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

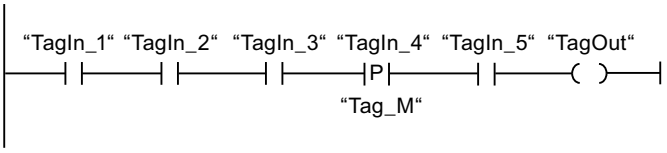
Parameter

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a rising edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

See also

Overview of the valid data types (Page 1908)

Example of detecting the direction of a conveyor belt (Page 3863)

--|N|--: Scan operand for negative signal edge

Description

You can use the "Scan operand for negative signal edge" instruction to determine if there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan that is saved in an edge memory bit <Operand2>. If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a negative signal edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

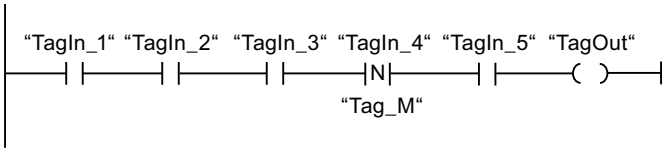
Parameter

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a negative signal edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

See also

Overview of the valid data types (Page 1908)

--(P)--: Set operand on positive signal edge

Description

You can use the "Set operand on positive signal edge" instruction to set a specified operand (<Operand1>) when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand2>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

When a positive signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be set (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

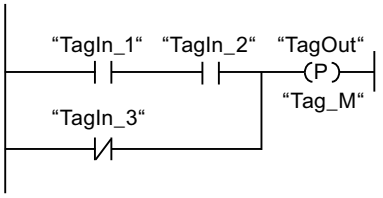
Parameter

The following table shows the parameters of the "Set operand on positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set by a positive edge.
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows how the instruction works:



Operand "TagOut" is set for one program cycle, when the signal state at the input of the coil switches from "0" to "1" (positive signal edge). In all other cases, the operand "TagOut" has the signal state "0".

See also

Overview of the valid data types (Page 1908)

--(N)--: Set operand on negative signal edge

Description

You can use the "Set operand on negative signal edge" instruction to set a specified operand (<Operand1>) when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand2>). If the instruction detects a change in the RLO from "1" to "0", there is a negative edge.

When a negative signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be set (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

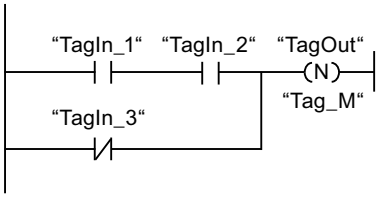
Parameter

The following table shows the parameters of the "Set operand on negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set by a negative edge.
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows how the instruction works:



Operand "TagOut" is set for one program cycle, when the signal state at the input of the coil switches from "1" to "0" (negative signal edge). In all other cases, the operand "TagOut" has the signal state "0".

See also

Overview of the valid data types (Page 1908)

P_TRIG: Scan RLO for positive signal edge

Description

Use the "Scan RLO for positive signal edge" instruction to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

If a positive edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

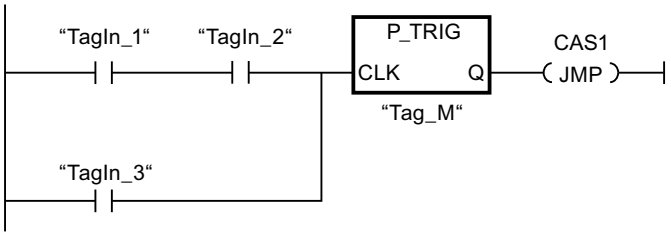
Parameter

The following table shows the parameters of the "Scan RLO for positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 1908)

N_TRIG: Scan RLO for negative signal edge

Description

Use the "Scan RLO for negative signal edge" instruction to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative edge.

If a negative signal edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

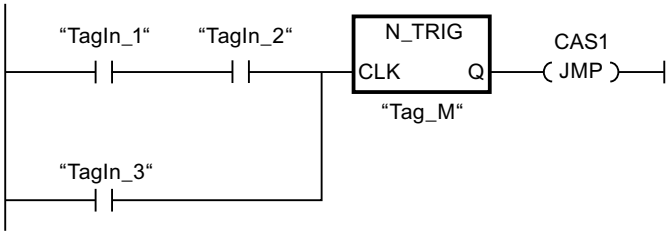
Parameter

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 1908)

R_TRIG: Detect positive signal edge

Description

With the "Detect positive signal edge" instruction, you can detect a state change from "0" to "1" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "0" to "1", a positive signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface.

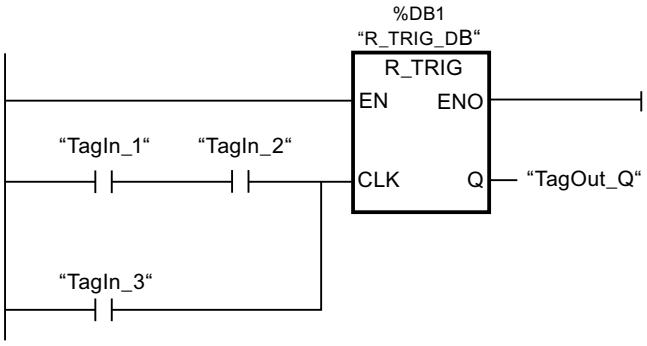
Parameter

The following table shows the parameters of the "Detect positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The previous state of the tag at the CLK input is stored in the "R_TRIG_DB" tag. If a change in the signal state from "0" to "1" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 1908)

F_TRIG: Detect negative signal edge

Description

With the "Detect negative signal edge" instruction, you can detect a state change from "1" to "0" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "1" to "0", a negative signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface.

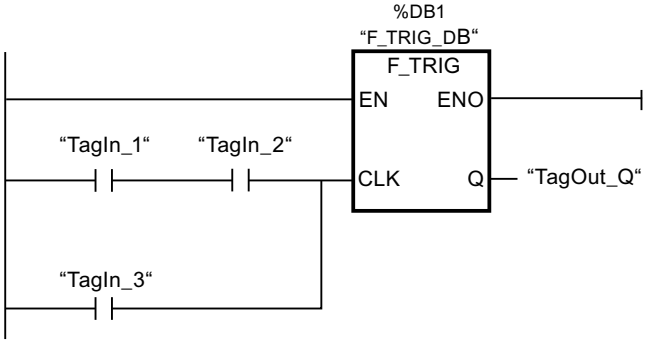
Parameter

The following table shows the parameters of the "Detect negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The previous state of the tag at the CLK input is stored in the "F_TRIG_DB" tag. If a change in the signal state from "1" to "0" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 1908)

Timers

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the output Q for a programmed duration. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. Output Q is set for the duration PT, regardless of the subsequent course of the input signal. Even if a new positive signal edge is detected, the signal state at the output Q is not affected as long as the PT time duration is running.

You can scan the current time value at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. When the duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TP_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

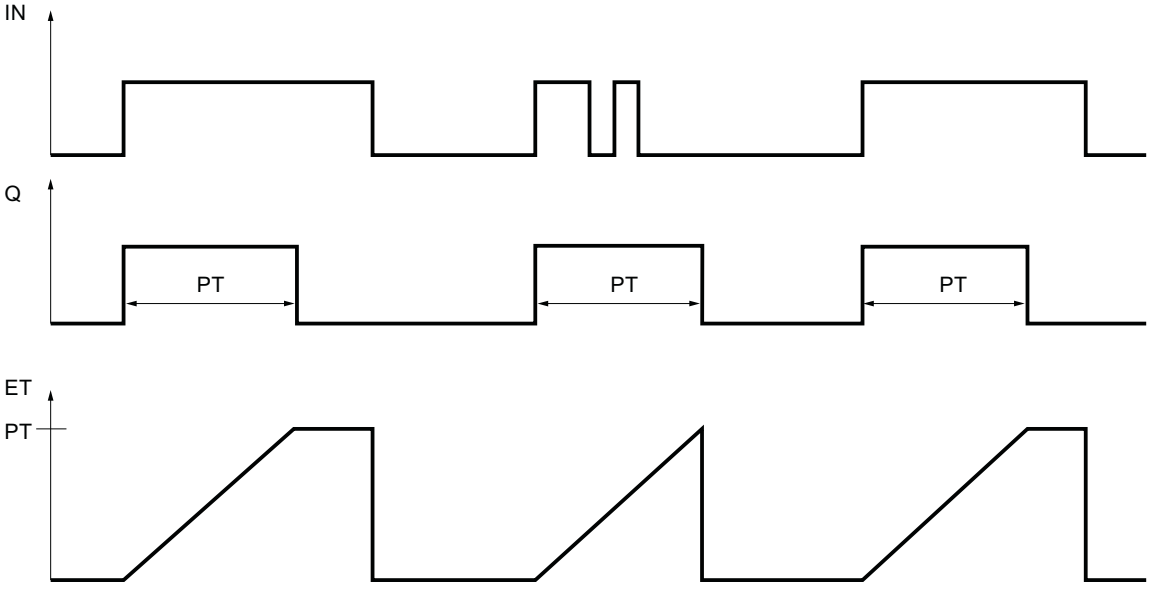
The following table shows the parameters of the "Generate pulse" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the pulse The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Pulse output
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:



See also

Overview of the valid data types (Page 1908)

Example of controlling room temperature (Page 3868)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the programmed duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the duration PT expires, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TON_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

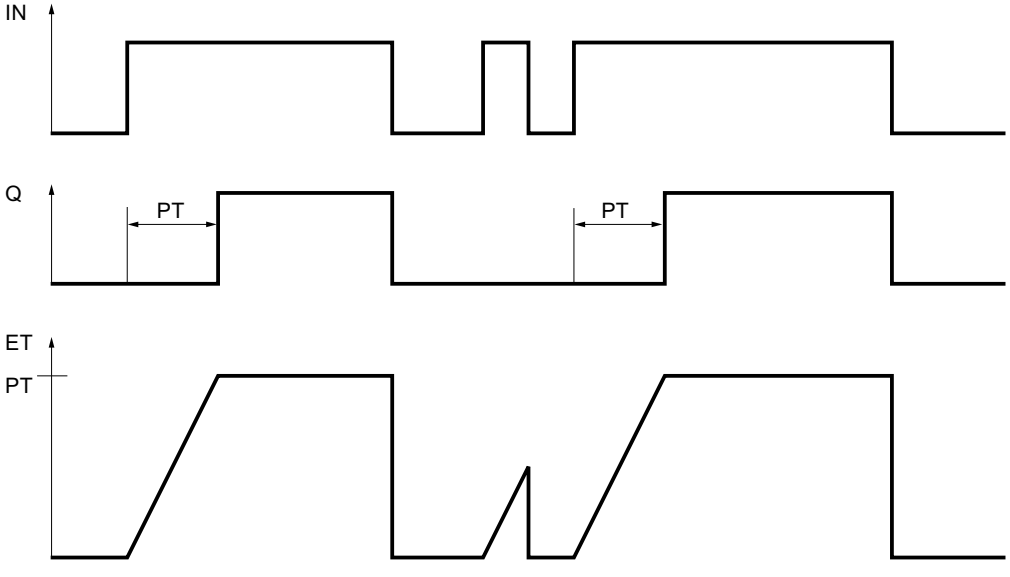
The following table shows the parameters of the "Generate on-delay" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:



See also

Overview of the valid data types (Page 1908)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed duration PT. The Q output is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). When the signal state at input IN changes back to "0", the programmed time PT starts. Output Q remains set as long as the duration PT is running. When duration PT expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q continues to be "1".

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. When the time duration PT expires, the ET output remains set to the current value until the IN input changes back to "1". If input IN switches to "1" before the duration PT has expired, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TOF_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it

in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

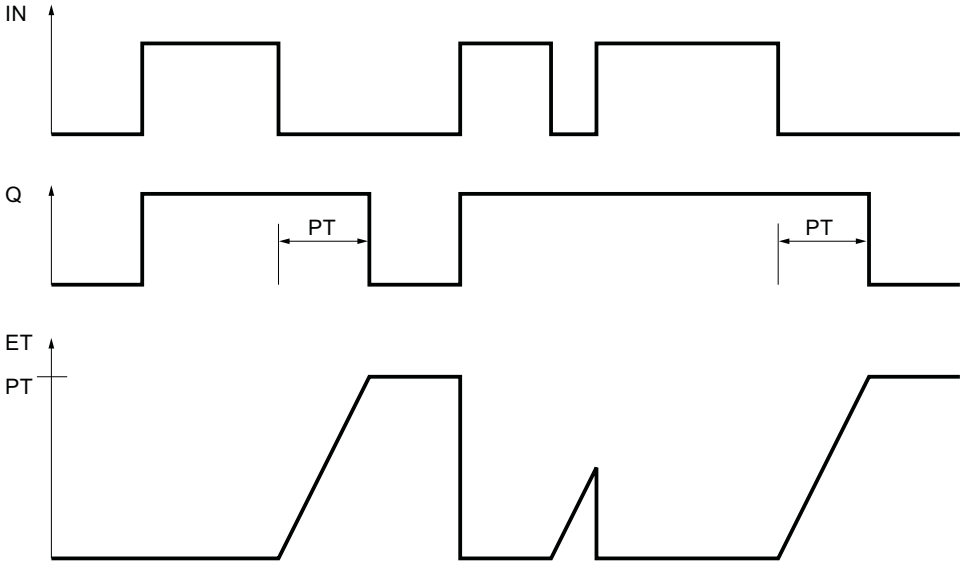
The following table shows the parameters of the "Generate off-delay" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is reset when the timer PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate off-delay" instruction:



See also

Overview of the valid data types (Page 1908)

TONR: Time accumulator

Description

The "Time accumulator" instruction is used to accumulate time values within a period set by the PT parameter. When the signal state at input IN changes from "0" to "1" (positive signal edge), the instruction is executed and the duration PT starts. While the duration PT is running, the time values are accumulated that are recorded when the IN input has signal state "1". The accumulated time is written to output ET and can be queried there. When the duration PT expires, the output Q has signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes from "1" to "0" (negative signal edge).

The R input resets the outputs ET and Q regardless of the signal state at the start input.

Each call of the "Time accumulator" instruction must be assigned to an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TONR_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

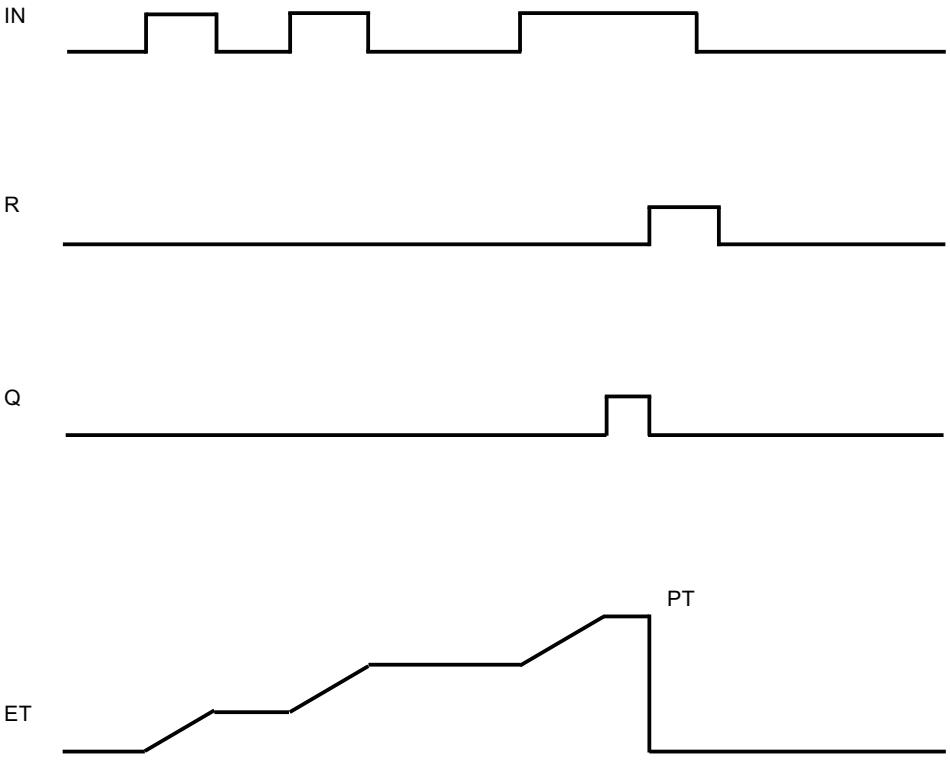
The following table shows the parameters of the "Time accumulator" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Start input
R	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, P, or constant	Reset input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P, or constant	Maximum duration of time recording The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Accumulated time

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:



See also

Overview of the valid data types (Page 1908)

---(TP)---: Start pulse timer

Description

Use the "Start pulse timer" instruction to start an IEC timer with a specified duration as pulse. The IEC timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC timer runs for the specified duration regardless of any subsequent changes in the RLO. The run of the IEC timer is also not affected by the detection of a new positive signal edge. As long as the IEC timer is running, the querying of the timer status for "1" returns the signal state "1". When the IEC timer has expired, the timer status returns the signal state "0".

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER or TP_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_LTIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the assigned timer is accessed.

The current timer status is stored in the Q structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0". The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Start pulse timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameter

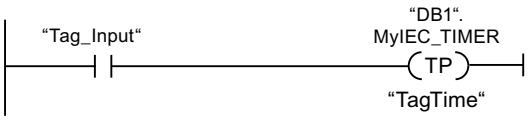
The following table shows the parameters of the "Start pulse timer" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TP_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME	D, L	IEC timer that is started.

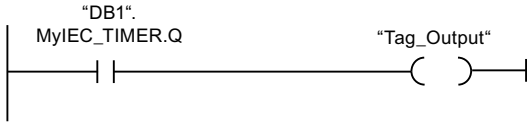
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start pulse timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The timer "DB1".MyIEC_TIMER is started for the time stored in the operand "TagTime".



As long as the timer "DB1". MyIEC_TIMER is running, the timer status ("DB1".MyIEC_TIMER.Q) has signal state "1" and the operand "Tag_Output" is set. When the IEC timer has expired, the signal state of the time status changes back to "0" and the "Tag_Output" operand is reset.

See also

Overview of the valid data types (Page 1908)

---(TON)---: Start on-delay timer

Description

Use the "Start on-delay timer" instruction to start an IEC timer with a specified duration as on-delay. The IEC timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC timer runs for the specified time. The output returns the signal state "1" if the RLO at the input of the instruction has the signal state "1". If the RLO changes to "0" before the time expires, the IEC timer is reset. In this case, querying the timer status for "1" returns signal state "0". The IEC timer restarts when the next positive signal edge is detected at the input of the instruction.

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start on-delay timer" stores its data in a structure of the data type IEC_TIMER or TON_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start on-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the assigned timer is accessed.

The current timer status is stored in the ET structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0". The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Start on-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameter

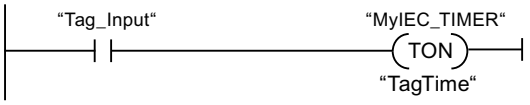
The following table shows the parameters of the "Start on-delay timer" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TON_TIME	IEC_TIMER, IEC_LTIMER, TON_TIME, TON_LTIME	D, L	IEC timer that is started.

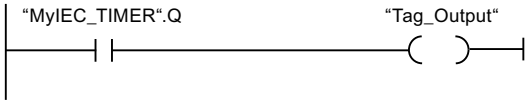
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start on-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The "MyIEC_TIMER timer is started for the time stored in the "TagTime" operand.



If the timer "MyIEC_TIMER" has expired and the operand "Tag_Input" has the signal state "1", querying the timer status ("MyIEC_TIMER).Q) returns signal state "1" and the "Tag_Output" operand is set. When the signal state of the operand "Tag_Input" changes to "0", the querying of the timer status returns the signal state "0" and the operand "Tag_Output" is reset.

See also

Overview of the valid data types (Page 1908)

---(TOF)---: Start off-delay timer

Description

Use the "Start off-delay timer" instruction to start an IEC timer with a specified duration as on-delay. The query of the timer status for "1" returns the signal state "0" if the result of logic operation (RLO) at the input of the instruction has the signal state "1". When the RLO changes from "1" to "0" (negative signal edge), the IEC timer starts with the specified time. The timer status remains at signal state "1" as long as the IEC timer is running. When the timer has run out and the RLO at the input of the instruction has the signal state "0", the timer status is set to the signal state "0". If the RLO changes to "1" before the time expires, the IEC timer is reset and the timer status remains at signal state "1".

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start off-delay timer" stores its data in a structure of the data type IEC_TIMER or TOF_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start off-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the assigned timer is accessed.

The current timer status is stored in the ET structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0". The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Start off-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

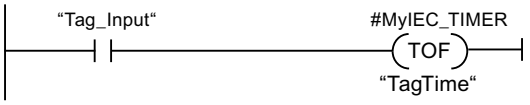
The following table shows the parameters of the "Start off-delay timer" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TOF_TIME	IEC_TIMER, IEC_LTIMER, TOF_TIME, TOF_LTIME	D, L	IEC timer that is started.

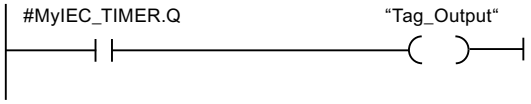
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start off-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "1" to "0". The timer #MyIEC_TIMER is started for the time stored in the operand "TagTime".



As long as timer #MyIEC_TIMER is running, the query of the time status (#MyIEC_TIMER.Q) returns the signal state "1" and operand "Tag_Output" is set. If the timer has expired and the operand "Tag_Input" has the signal state "0", the query of the timer status returns the signal state "0". If the signal state of the operand "Tag_Input" changes to "1" before timer #MyIEC_TIMER expires, the timer is reset. When the signal state of the operand "Tag_Input" is "1", the query of the timer status returns the signal state "1".

See also

Overview of the valid data types (Page 1908)

---(TONR)---: Time accumulator

Description

You can use the "Time accumulator" instruction to record how long the signal is at the input of instruction "1". The instruction is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The time is recorded as long at the RLO is "1". If the RLO changes to "0", the instruction is halted. If the RLO changes back to "1", the time recording is continued. The query of the timer status for "1" returns the signal state "1" if the recorded time exceeds the value of the specified duration and the RLO at the input of coil is "1".

The timer status and the currently expired timer can be reset to "0" using the "Reset timer" instruction.

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER or TONR_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the assigned timer is accessed.

The current timer status is stored in the ET structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0". The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the Time accumulator" instruction requires a preceding logic operation. It can be placed only at the end of the network.

Parameter

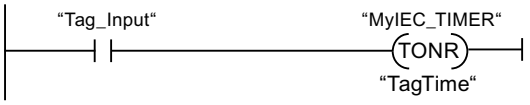
The following table shows the parameters of the "Time accumulator" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TONR_TIME	IEC_TIMER, IEC_LTIMER, TONR_TIME, TONR_LTIME	D, L	IEC timer that is started.

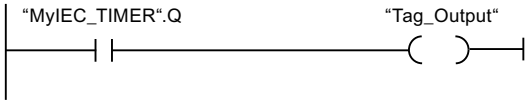
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Time accumulator" instruction executes on a positive signal edge in the RLO. The time is recorded as long as the operand "Tag_Input" has the signal state "1".



If the recorded time exceeds the value of the operand "TagTime", then the query of the timer status ("MyIEC_TIMER".Q) will return the signal state "1" and the operand "Tag_Output" will be set.

See also

Overview of the valid data types (Page 1908)

---(RT)---: Reset timer

Description

You can use the "Reset timer" instruction to reset an IEC timer to "0". The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If current is flowing to the coil (RLO is "1"), the structure components of the timer in the specified data block are reset to "0". If the RLO at the input of the instruction is "0", the timer remains unchanged.

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

You assign the "Reset timer" instruction an IEC timer that has been declared in the program.

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

Parameters

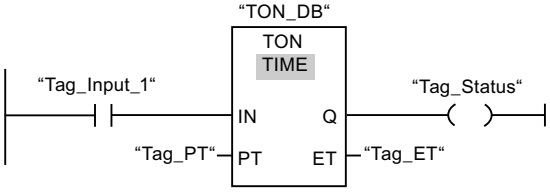
The following table shows the parameters of the "Reset timer" instruction":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer that is reset.

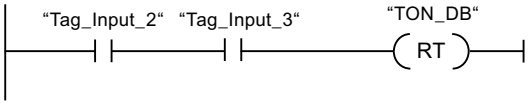
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The timer stored in the "TON_DB" instance data block starts running for the time duration specified by operand "Tag_PT".



If operands "Tag_Input_2" and "Tag_Input_3" have the signal state "1", the "Reset timer" instruction is executed and the timer stored in the "TON_DB" data block.

See also

Overview of the valid data types (Page 1908)

---(PT)---: Load time duration

Description

You can use the "Load time duration" instruction to set the time for an IEC timer. The instruction is executed in every cycle when the result of logic operation (RLO) at the input of the instruction has the signal state "1". The instruction writes the specified time to the structure of the specified IEC timer.

Note

If the specified IEC timer is running while the instruction executes, the instruction overwrites the current time of the specified IEC timer. This can change the timer status of the IEC timer.

You assign an IEC timer declared in the program to the "Load time duration" instruction.

The instruction data is updated only when the instruction is called and each time the assigned IEC timer is accessed. The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Parameter

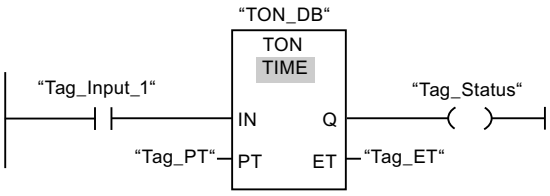
The following table shows the parameters of the "Load time duration" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer, the duration of which is set.

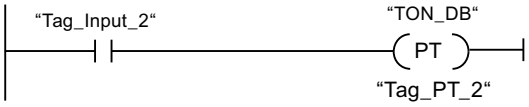
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".



The "Load time duration" instruction is executed when the operand "Tag_Input_2" has the signal state "1". The instruction writes the time duration "Tag_PT_2" in the instance data block "TON_DB" and at the same time overwrites the value of the operand "Tag_PT" within the data block. The signal state of the timer status may therefore change at the next query or when "MyTimer".Q or "MyTimer".ET are accessed.

See also

Overview of the valid data types (Page 1908)

Legacy

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as soon as the signal state at input S is "1". If the signal state at input S changes to "0" before the programmed duration expires, the timer is stopped. In this case, the signal state at output Q is "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign pulse timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

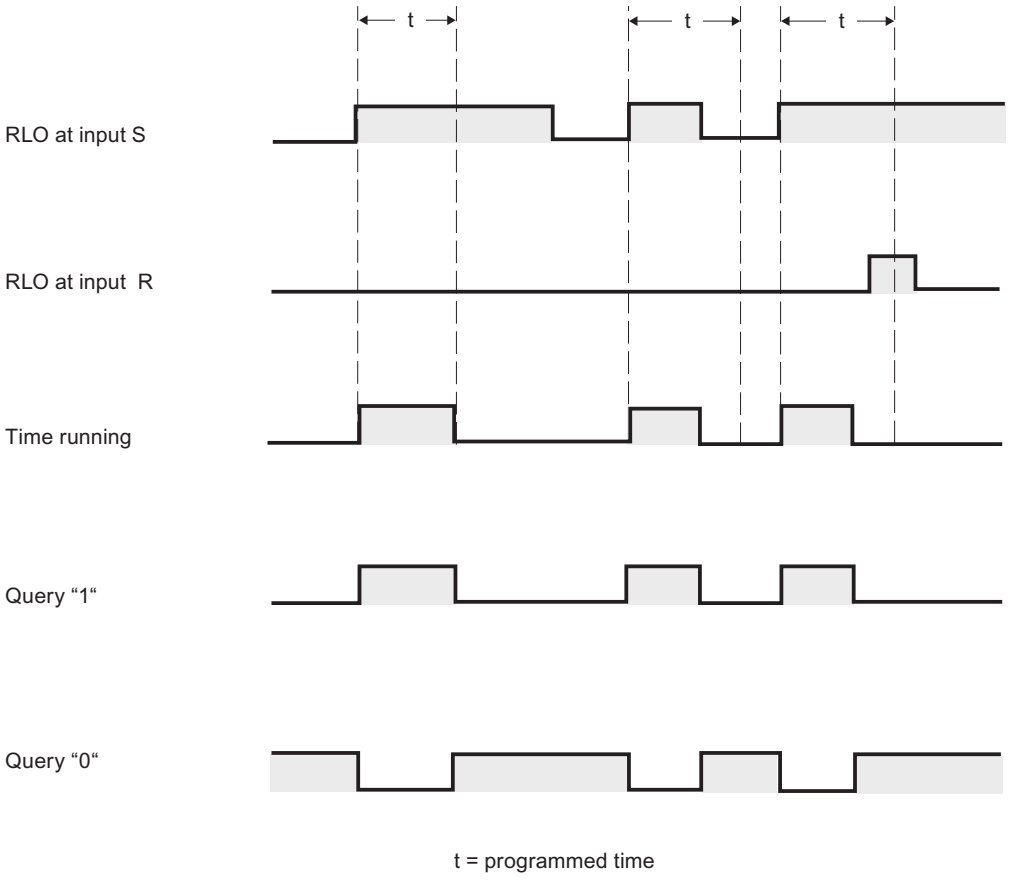
The following table shows the parameters of the instruction "Assign pulse timer parameters and start":

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

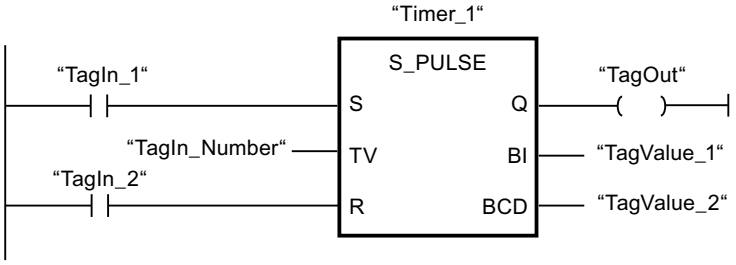
The following figure shows the pulse timing diagram of the instruction "Assign pulse timer parameters and start":



t = programmed time

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" as long as the operand "TagIn_1" has the signal state "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer "Timer_1" is stopped. The operand "TagOut" is reset to "0" in this case.

The operand "TagOut" has the signal state "1" as long as the timer is running and the operand "TagIn_1" has the signal state "1". When the timer expires or is reset, the operand "TagOut" is reset to "0".

See also

Overview of the valid data types (Page 1908)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". As long as the timer is running, the output Q has the signal state "1". When the timer expires, the output Q is reset to "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input TV.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign extended pulse timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

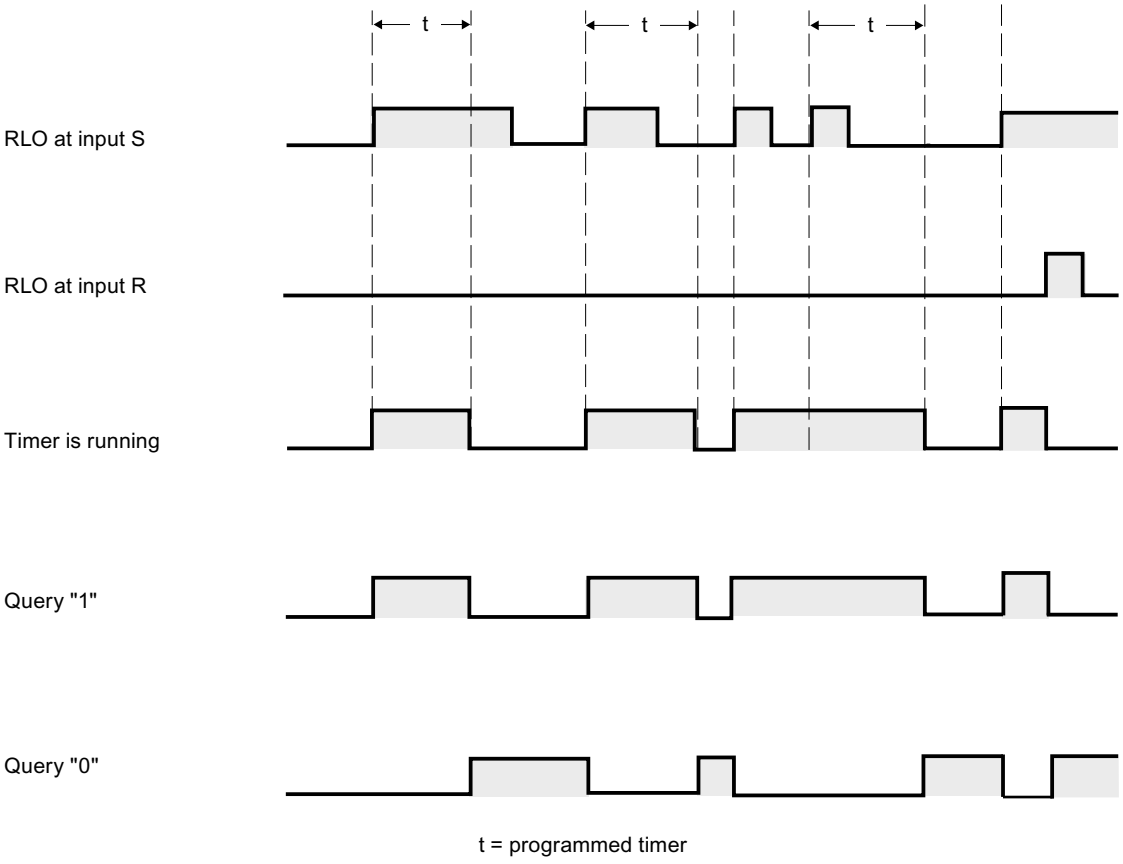
The following table shows the parameters of the instruction "Assign extended pulse timer parameters and start":

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

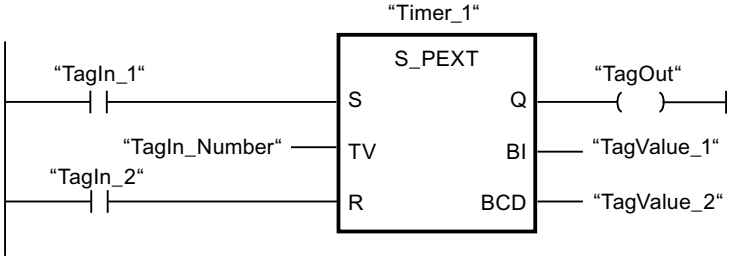
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative edge at input S. If the signal state of the operand "TagIn_1" changes from "0" to "1" before the timer expires, the timer is restarted.

The operand "TagOut" has the signal state "1" as long as the timer is running. When the timer expires or is reset, the operand "TagOut" is reset to "0".

See also

Overview of the valid data types (Page 1908)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as soon as the signal state at input S is "1". If the timer expires correctly and input S still has signal state "1" then output Q returns signal state "1". If the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the time is running and the signal state at input R changes from "0" to "1" then the current time value and the time base are also set to zero. In this case, the signal state at output Q is "0". The timer is reset if the signal state is "1" at the R input even if the timer is not running and the RLO at input S is "1".

Specify the timer of the instruction in the placeholder above the box. The timer must be declared with the data type TIMER.

The "Assign on-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

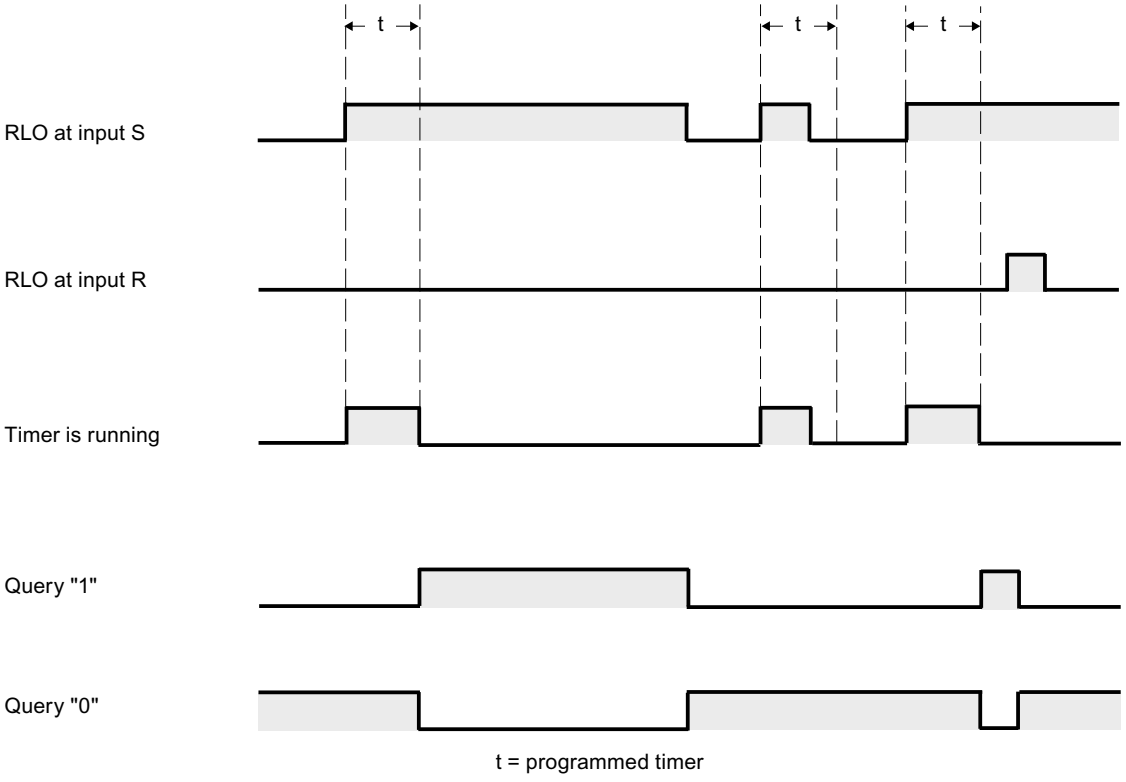
The following table shows the parameters of the "Assign on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

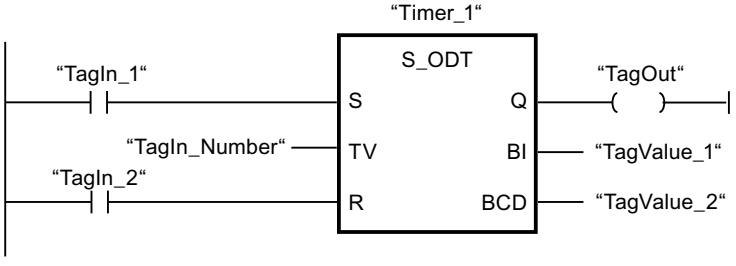
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". If the timer expires and the operand has the signal state "1", the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. The operand "TagOut" has the signal state "0" in this case.

See also

Overview of the valid data types (Page 1908)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". If the timer expires, the "Q" output returns signal state "1" regardless of the signal state at input "S". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input (TV).

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0" regardless of the signal state at start input S. In this case, the signal state at output Q is "0".

The "Assign retentive on-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

The following table shows the parameters of the "Assign retentive on-delay timer parameters and start" instruction:

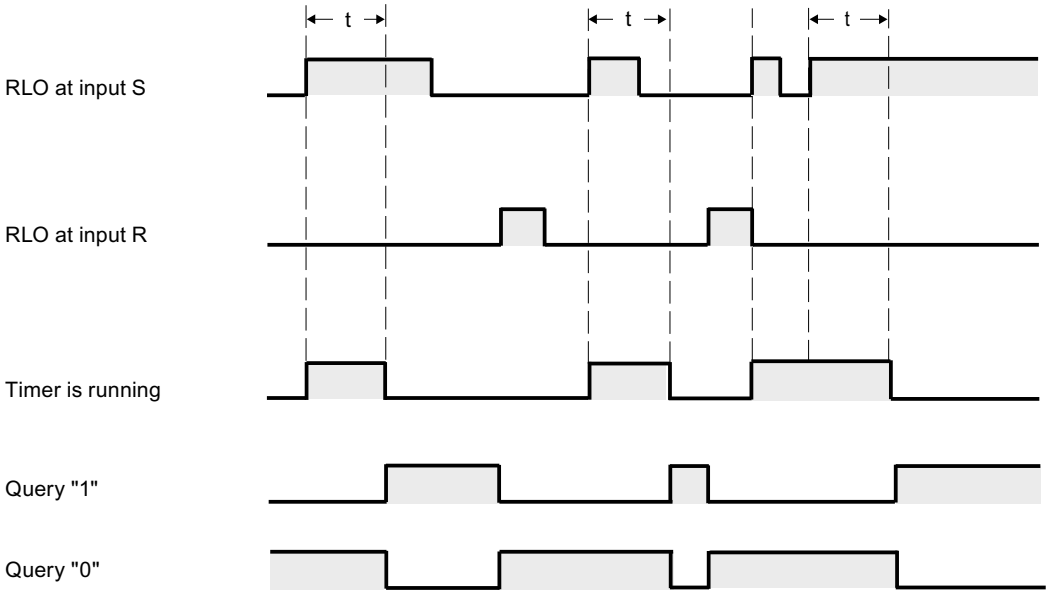
Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input

Parameter	Declaration	Data type	Memory area	Description
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

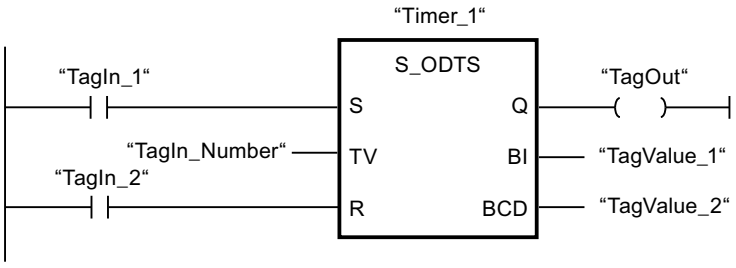
The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:



t = programmed timer

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number", even if the signal state of the operand "TagIn_1" changes to "0". When the timer expires, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is restarted.

See also

Overview of the valid data types (Page 1908)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV). As long as the timer is running or input S returns signal state "1", then output Q has signal state "1". When the timer expires and the signal state at input S is "0", output Q is reset to the signal state "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at input S.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0". In this case, the signal state at output Q is "0".

The "Assign off-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

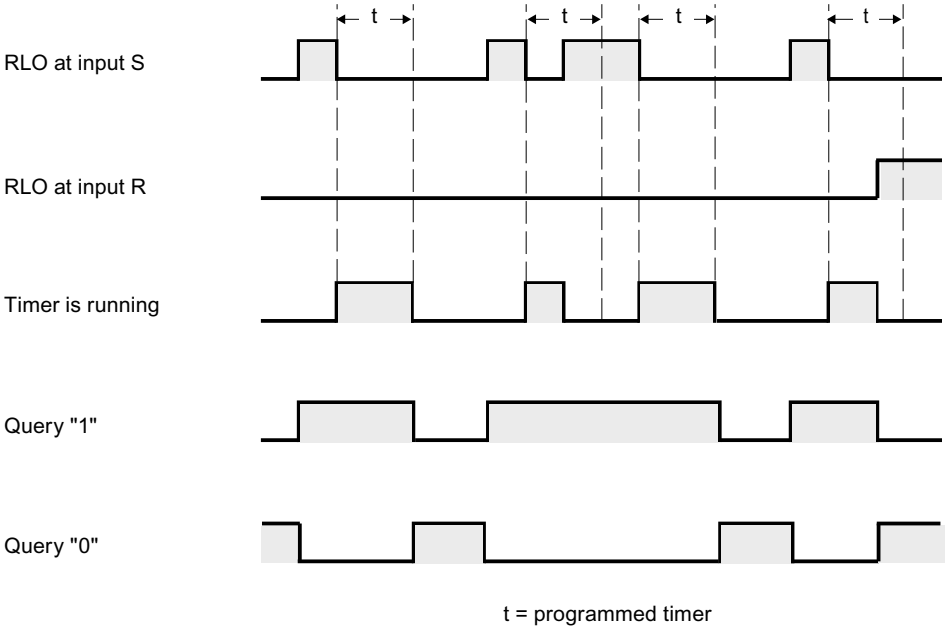
The following table shows the parameters of the "Assign off-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

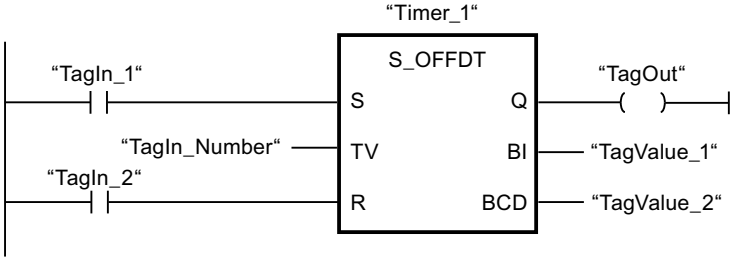
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "1" to "0". The timer expires with the time value of the operand "TagIn_Number". The operand "TagOut" is set to "1" when the timer is running and when the operand "TagIn_1" has the signal state "0". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is reset.

See also

Overview of the valid data types (Page 1908)

---(SP): Start pulse timer

Description

The "Start pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The time runs with the specified duration as long as the RLO has the signal state "1". As long as the timer is running, the querying of the timer status for "1" returns the signal state "1". If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer stops. In this case, the querying of the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

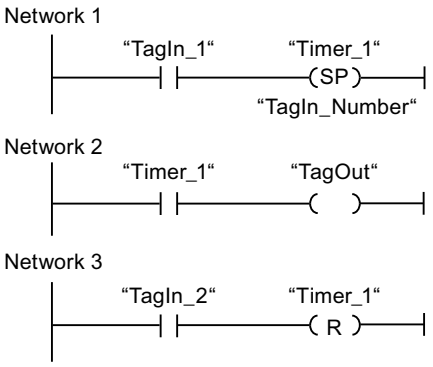
The following table shows the parameters of the "Start pulse timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" as long as the signal state of the operand "TagIn_1" is "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. As long as the timer is running, the operand "TagOut" has the signal state "1". A signal state change of the operand "TagIn_1" from "0" to "1" resets the timer, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 1908)

---(SE): Start extended pulse timer

Description

The "Start extended pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration, even if the RLO changes to the signal state "0". As long as the timer is running, the querying of the timer status for "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is restarted with the programmed duration. The querying of the timer status for "1" returns the signal state "0" when the timer expires.

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start extended pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

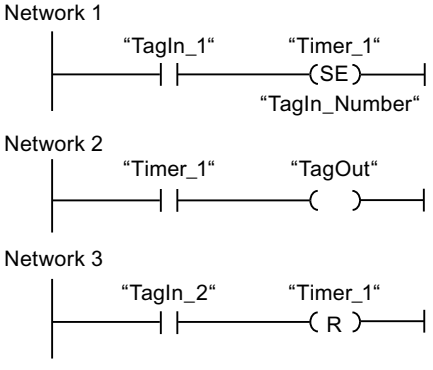
The following table shows the parameters of the "Start extended pulse timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative edge in the RLO. As long as the timer is running, the operand "TagOut" has the signal state "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" before the timer expires, the timer is restarted.

See also

Overview of the valid data types (Page 1908)

---(SD): Start on-delay timer

Description

The "Start on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer runs for the specified duration as long as the RLO is "1". If the timer expires and the RLO still has the signal state "1", the query of the timer status for "1" returns the signal state "1". If the RLO changes from "1" to "0" while the timer is running, the timer is stopped. In this case, the querying of the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start on-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

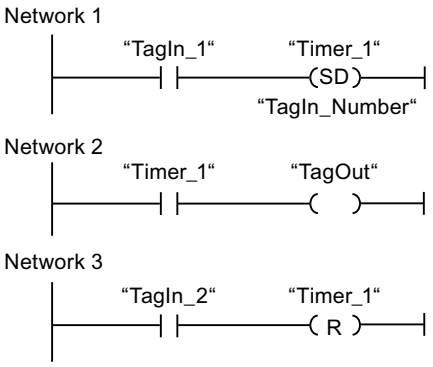
The following table shows the parameters of the "Start on-delay timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". If the timer expires and the RLO has the signal state "1", the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. If the signal state of the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 1908)

---(SS): Start retentive on-delay timer

Description

The "Start retentive on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration, even if the RLO changes to the signal state "0". When the timer expires, the querying of the timer status for "1" returns the signal state "1". After expiry of the timer, the timer can only be restarted if it is explicitly reset.

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start retentive on-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

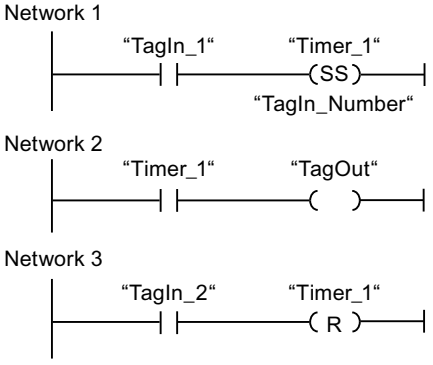
The following table shows the parameters of the "Start retentive on-delay timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". When the timer expires, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is restarted. If the signal state of the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 1908)

---(SF): Start off-delay timer

Description

The "Start off-delay timer" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration. As long as the timer is running, the querying of the timer status for "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is reset. The timer is always restarted when the RLO changes from "1" to "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start off-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

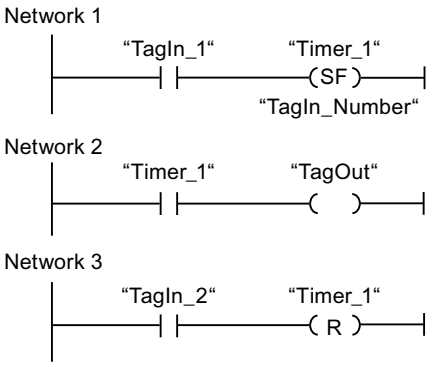
The following table shows the parameters of the "Start off-delay timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "1" to "0". The timer expires with the time value of the operand "TagIn_Number". As long as the timer is running, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" while the timer is running, the timer is restarted. If the signal state of the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 1908)

Counter operations

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the output CV. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can query the counter status in the Q output. The signal state at the Q output is determined by the PV parameter. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value at the CV output is reset to zero when the signal state at input R changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction "Count up":

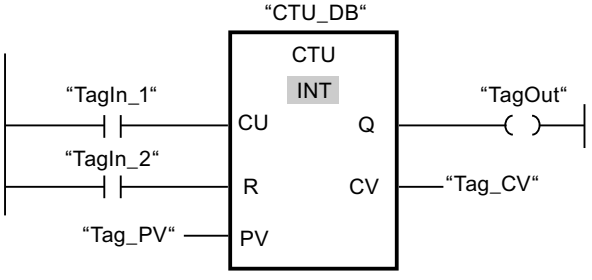
Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, D, L, P, or constant	I, Q, M, T, C, D, L, P or constant	Reset input
PV	Input	Integers	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CV" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 1908)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can query the counter status in the Q output. If the current counter value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value at the CV output is set to the value of the PV parameter when the signal state at the LD input changes to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction "Count down":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
LD	Input	BOOL	I, Q, M, D, L, P, or constant	I, Q, M, T, C, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Value to which the CV output is set with LD = 1.

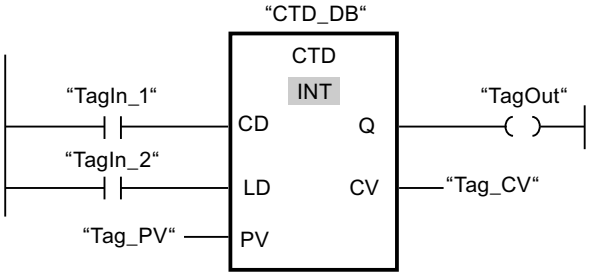
Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the instruction is executed and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the counter value is decremented until the low limit of the specified data type (INT = -32768) is reached.

The "TagOut" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 1908)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV output. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter. As long as the LD input has signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0".

You can query the status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction "Count up and down":

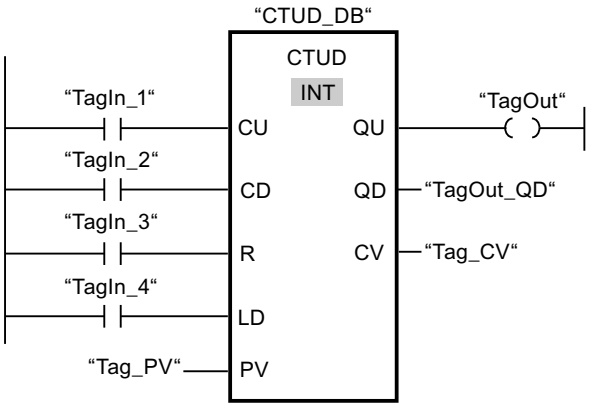
Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P, or constant	I, Q, M, T, C, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P, or constant	I, Q, M, T, C, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Value at which the output QU is set. / Value to which the CV output is set with LD = 1.
QU	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status of the down counter
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_1" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input, the current counter value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_2" input, the counter value is decremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the CU input, the counter value is incremented until it reaches the high limit of 32767. If input CD has a positive signal edge, the counter value is decremented until it reaches the low limit of INT = -32768.

The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut" output has signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut_QD" output has signal state "0".

See also

Overview of the valid data types (Page 1908)

Legacy

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CU is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameter

The following table shows the parameters of the "Assign parameters and count up" instruction:

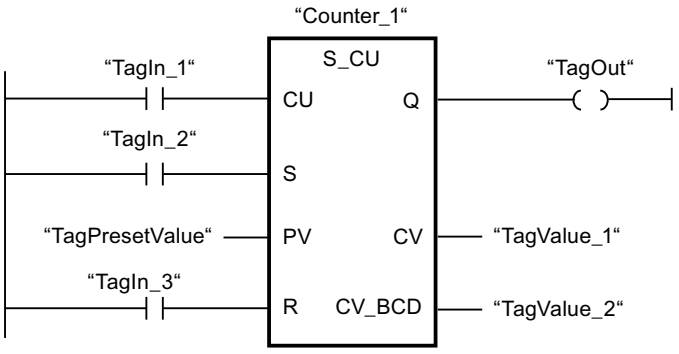
Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)

Parameter	Declaration	Data type	Memory area	Description
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. When the signal state at input "TagIn_2" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 1908)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value is decremented until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CD is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count down" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameter

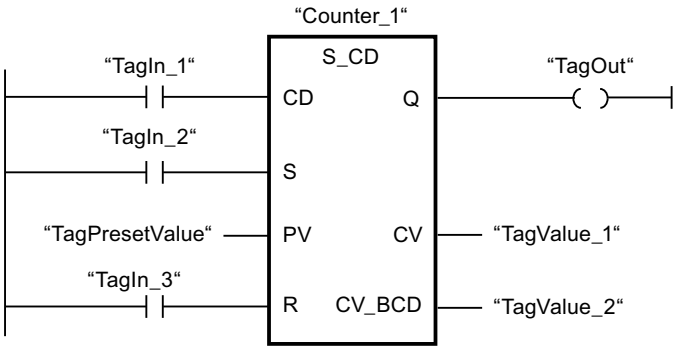
The following table shows the parameters of the "Assign parameters and count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decremented by one. When the signal state at input "TagIn_2" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 1908)

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment or decrement the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. If there is a positive signal edge at the CU and CD inputs in one program cycle, the counter value remains unchanged.

The counter value can be incremented until the high limit of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit "0" is reached, the counter value is not decremented any further.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO is "1" at the inputs CU and CD, the counter counts accordingly in the next scan cycle, even if no change in the signal edge was detected.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU, CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up / down" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

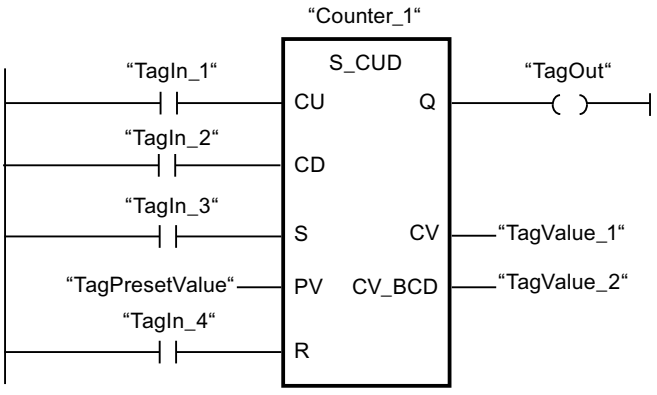
The following table shows the parameters of the "Assign parameters and count up / down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L, T, C or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
BI	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_1"" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Assign parameters and count up / down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input and the current counter value is less than "999", the counter value is incremented by one. When there is a positive signal edge at the "TagIn_2" input and the current counter value is greater than "0", the counter value is decremented by one.

When the signal state at input "TagIn_3" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_4" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 1908)

---(SC): Set counter value

Description

You can use the "Set counter value" instruction to set the value of a counter. The instruction is executed when the result of logic operation (RLO) at the input changes from "0" to "1". When the instruction is executed, the counter is set to the specified counter value.

The "Set counter value" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameter

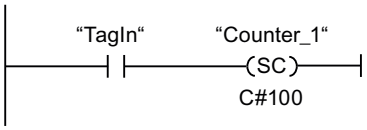
The following table lists the parameters of the "Set counter value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Count value>	Input	WORD	I, Q, M, D, L or constant	Value with which the counter is preset in the BCD format. (C#0 to C#999)
<Counter>	InOut/Input	COUNTER	C	Counter that is preset.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



the counter "Counter_1" starts with the value "100" when the signal state of the "TagIn" operand changes from "0" to "1".

See also

Overview of the valid data types (Page 1908)

---(CU): Count up

Description

With the "Count up" instruction you can increment the value of the specified counter by one if there is a positive signal edge in the result of logic operation (RLO). The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

The "Count up" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameter

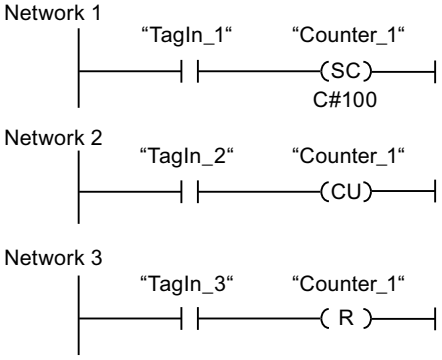
The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter whose value is incremented.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of the counter "Counter_1" is incremented by one when the signal state of the operand "TagIn_2" changes from "0" to "1".

When the operand "TagIn_3" has the signal state "1", the value of the counter "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 1908)

---(CD): Count down

Description

With the "Count down" instruction you can increment the value of the specified counter by one if there is a positive signal edge in the result of logic operation (RLO). The counter value can be decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer changed on a positive signal edge.

The "Count down" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameter

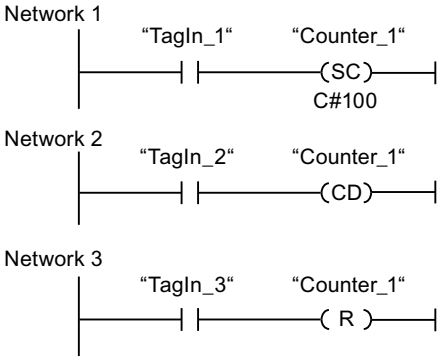
The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter whose value is decremented.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of the counter "Counter_1" is decremented by one when the signal state of the operand "TagIn_2" changes from "0" to "1".

When the operand "TagIn_3" has the signal state "1", the value of the counter "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 1908)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to determine if a first comparison value (<Operand1>) is equal to a second comparison value (<Operand2>).

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0

You can also use the "Equal" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width (length) of the operands must be the same. When floating-point numbers are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check.

Note

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, instead of the instruction "CMP ==: Equal", use the instruction "IN_RANGE: Value within range".

Note

Comparison of the data type PORT

To be able to compare the operands of the PORT data type with the "Equal" instruction, you need to select the WORD data type from the drop-down list of the instructions box.

Parameters

The following table lists the parameters of the "Equal" instruction:

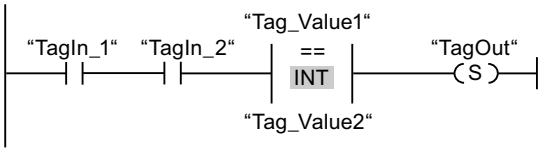
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" = "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to determine if a first comparison value (<Operand1>) is not equal to a second comparison value (<Operand2>).

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0

You can also use the "Not equal" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width (length) of the operands must be the same. If the floating-point numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

Note

Comparison of the data type PORT

To be able to compare the operands of the PORT data type with the "Not equal" instruction, you need to select the WORD data type from the drop-down list of the instructions box.

Parameters

The following table lists the parameters of the "Not equal" instruction:

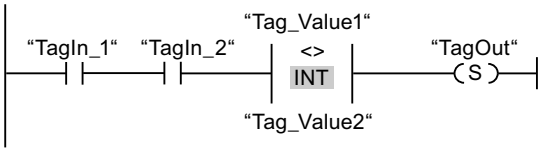
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" **<>** "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to determine if a first comparison value (<Operand1>) is greater than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'Hello World'	'Hello World'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'AAA'	'a'	0

You can also use the "Greater or equal" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is greater (more recent) than or equal to the point of time at <Operand2>.

Parameters

The following table lists the parameters of the "Greater or equal" instruction:

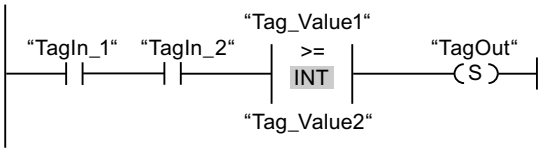
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" >= "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

Example of detecting the fill level of a storage area (Page 3864)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to determine if a first comparison value (<Operand1>) is less than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'Hello World'	'Hello World'	1
'HelloWorld'	'Hello World'	0
'BB'	'AA'	0
'AAA'	'AA'	0

You can also use the "Less or equal" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is less (less recent) than or equal to the point of time at <Operand2>.

Parameters

The following table lists the parameters of the "Less or equal" instruction:

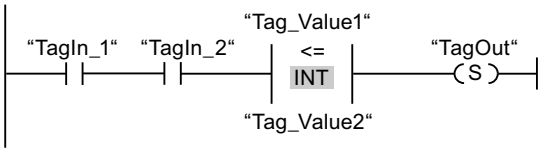
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <= "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

CMP >: Greater than

Description

You can use the "Greater than" instruction to determine if a first comparison value (<Operand1>) is greater than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'AA'	'aa'	0
'AAA'	'a'	0

You can also use the "Greater than" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is greater (more recent) than the point of time at <Operand2>.

Parameters

The following table lists the parameters of the "Greater than" instruction:

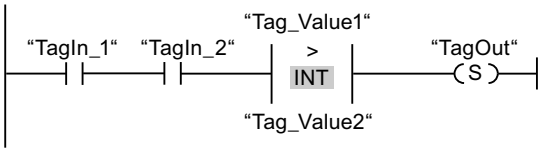
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" > "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

CMP <: Less than

Description

You can use the "Less than" instruction to determine if a first comparison value (<Operand1>) is less than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1

<Operand1>	<Operand2>	RLO of the instruction
'BB'	'AA'	0
'AAA'	'AA'	0

You can also use the "Less than" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is less (less recent) than the point of time at <Operand2>.

Parameters

The following table lists the parameters of the "Less than" instruction:

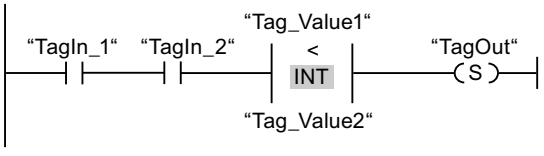
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" < "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

Example of detecting the fill level of a storage area (Page 3864)

IN_RANGE: Value within range

Description

You can use the "Value within range" instruction to determine if the value at the VAL input is within a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value within range" instruction compares the value at the VAL input with the values of the MIN and MAX inputs and sends the result to the box output. If the value at the VAL input fulfills the comparison MIN <= VAL or VAL <= MAX, the box output has the signal state "1". If the comparison is not fulfilled, the box output has the signal state "0".

If the box input has the signal state "0", the "Value within range" instruction is not executed.

The comparison function can only execute if the values to be compared are of the same data type and the box input is interconnected.

Parameters

The following table shows the parameters of the "Value within range" instruction:

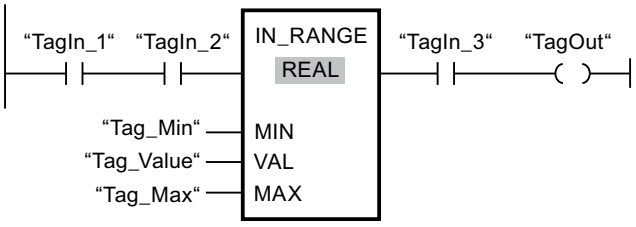
Parameter	Declaration	Data type	Memory area	Description
Box input	Input	BOOL	I, Q, M, D, L	Result of the previous logic operation
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The value of the operand "Tag_Value" is within the value range that is specified by the current values of the operands "Tag_Min" and "Tag_Max" (MIN <= VAL or VAL <= MAX).
- The operand "TagIn_3" has the signal state "1".

See also

Overview of the valid data types (Page 1908)

OUT_RANGE: Value outside range

Description

You can use the "Value outside range" instruction to determine if the value at the VAL input is outside a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value outside range" instruction compares the value at the VAL input with the values of the MIN and MAX inputs and sends the result to the box output. If the value at the VAL input fulfills the comparison MIN > VAL or VAL > MAX, the box output has the signal state "1". The box output also has the signal state "1" if a specified operand with the REAL data type shows an invalid value.

The box output returns the signal state "0", if the value at input VAL does not satisfy the MIN > VAL or VAL > MAX condition.

If the box input has the signal state "0", the "Value outside range" instruction is not executed.

The comparison function can only execute if the values to be compared are of the same data type and the box input is interconnected.

Parameters

The following table shows the parameters of the "Value outside range" instruction:

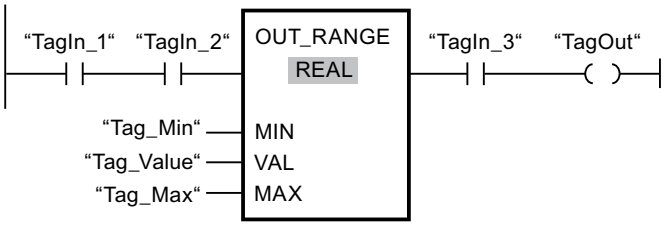
Parameter	Declaration	Data type	Memory area	Description
Box input	Input	BOOL	I, Q, M, D, L	Result of the previous logic operation
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The value of the operand "Tag_Value" is outside the value range that is specified by the values of the operands "Tag_Min" and "Tag_Max" (MIN > VAL or VAL > MAX).
- The operand "TagIn_3" has the signal state "1".

See also

Overview of the valid data types (Page 1908)

----I OK I----: Check validity

Description

You can use the "Check validity" instruction to check if the value of an operand (<operand>) is a valid floating-point number. The query is started in each program cycle when the signal state at the input of the instruction is "1".

The output of the instruction has signal state "1" when the value of the operand is a valid floating-point number at the time of the query and the input of the instruction has signal state "1". In all other cases, the signal state at the output of the "Check validity" instruction is "0".

You can use the "Check validity" instruction together with the EN mechanism. If you connect the instruction box to an EN enable input, the enable input is set only when the result of the validity query of the value is positive. You can use this function to ensure that an instruction is enabled only when the value of the specified operand is a valid floating-point number.

Parameters

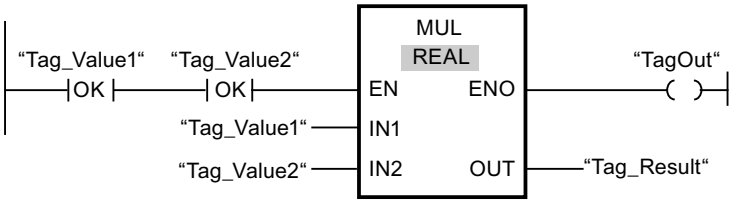
The following table shows the parameters of the "Check validity" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be queried.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the values of the operands "Tag_Value1" and "Tag_Value2" show valid floating-point numbers, the "Multiply" (MUL) instruction is activated and the ENO output is set. During the execution of the "Multiply" (MUL) instruction, the value of the operand "Tag_Value1" is multiplied with the value of the operand "Tag_Value2". The product of the multiplication is then stored in the operand "Tag_Result". If the instruction is executed without errors, the ENO and "TagOut" outputs are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

----| NOT_OK |----: Check invalidity

Description

You can use the "Check invalidity" instruction to check if the value of an operand (<operand>) is an invalid floating-point number. The query is started in each program cycle when the signal state at the input of the instruction is "1".

The output of the instruction has signal state "1" when the value of the operand is an invalid floating-point number at the time of the query and the input of the instruction has signal state "1". In all other cases, the signal state at the output of the "Check invalidity" instruction is "0".

Parameters

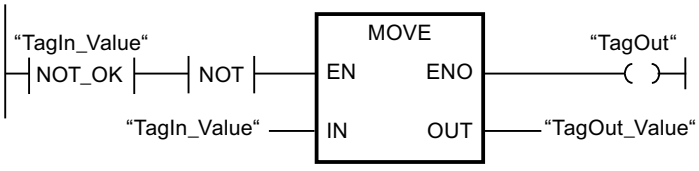
The following table shows the parameters of the "Check invalidity" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be queried.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the value of operand "TagIn_Value" is an invalid floating-point number, the "Move value" (MOVE) instruction will not be executed. The "TagOut" output is reset to signal state "0".

See also

Overview of the valid data types (Page 1908)

VARIANT

EQ_Type: Compare data type for EQUAL with the data type of a tag

Description

You can use the "Compare data type for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You are comparing the data type of the tag (<Operand1>), which you declared in the block interface, with the data type of a tag (<Operand2>) for "Equal to".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Parameters

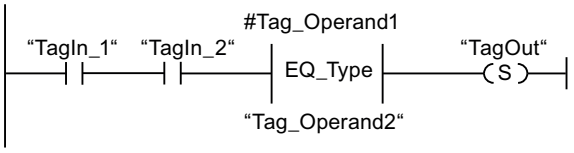
The following table lists the parameters of the "Compare data type for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	VARIANT	I, Q, M, L	First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L, P	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 1908)

NE_Type: Compare data type for UNEQUAL with the data type of a tag

Description

You can use the "Compare data type for UNEQUAL with the data type of a tag" instruction to query which data type a tag to which a VARIANT points does not have. You are comparing the data type of the tag (<Operand1>), which you declared in the block interface, with the data type of a tag (<Operand2>) for "Not equal to".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Parameters

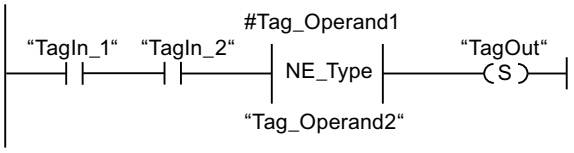
The following table lists the parameters of the "Compare data type for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	VARIANT	I, Q, M, L	First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L, P	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is not equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 1908)

EQ_ElemType: Compare data type of an ARRAY element for EQUAL with the data type of a tag

Description

You can use the "Compare data type of an ARRAY element for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You are comparing the data type of the tag (<Operand1>), which you declared in the block interface, with the data type of a tag (<Operand2>) for "Equal to".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag (<Operand1>) is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Parameters

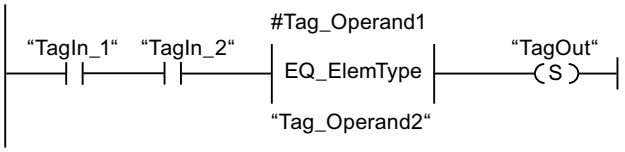
The following table lists the parameters of the "Compare data type of an ARRAY for EQUAL with the data type of a tag" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	VARIANT	I, Q, M, L	First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 1908)

NE_ElemType: Compare data type of an ARRAY element for UNEQUAL with the data type of a tag

Description

You can use the "Compare data type of an ARRAY element for UNEQUAL with the data type of a tag" instruction to query which data type a tag does not have to which a VARIANT points. You are comparing the data type of the tag (<Operand1>), which you declared in the block interface, with the data type of a tag (<Operand2>) for "Not equal to".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag (<Operand1>) is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Parameters

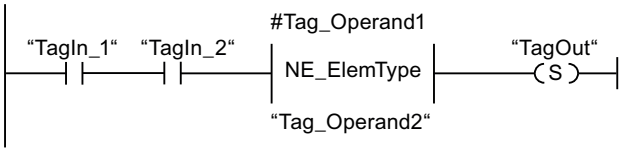
The following table lists the parameters of the "Compare data type of an ARRAY for UNEQUAL with the data type of a tag" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	VARIANT	I, Q, M, L	First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is not equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 1908)

IS_NULL: Check for EQUALS NULL pointer

Description

You can use the instruction "Check for EQUALS NULL pointer" to query whether the VARIANT points to a NULL pointer and therefore does not point to an object.

<Operand> must have the VARIANT data type.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "0" even if the ANY pointer is NULL.

Parameters

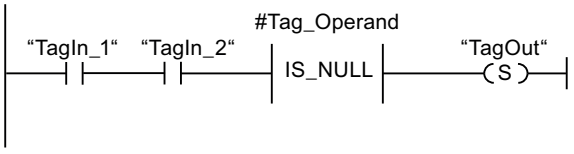
The following table shows the parameters of the "Check for EQUALS NULL pointer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	I, Q, M, L	Operand that is compared for EQUALS NULL

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand" operand does not point to an object.

See also

Overview of the valid data types (Page 1908)

NOT_NULL: Check for UNEQUALS NULL pointer

Description

You can use the instruction "Check for UNEQUALS NULL pointer" to query whether the VARIANT does not point to a NULL pointer and therefore points to an object.

<Operand> must have the VARIANT data type.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "1" even if the ANY pointer is NULL.

Parameters

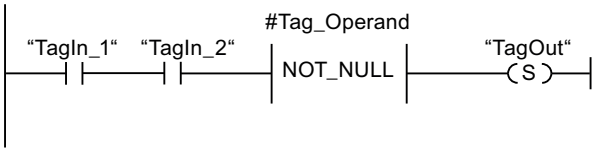
The following table shows the parameters of the "Check for UNEQUALS NULL pointer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	I, Q, M, L	Operand that is compared for UNEQUALS NULL

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand" operand points to an object.

See also

Overview of the valid data types (Page 1908)

IS_ARRAY: Check for ARRAY

Description

You can use the "Check for ARRAY" instruction to query whether the VARIANT points to a tag of the ARRAY data type.

<Operand> must have the VARIANT data type.

Parameters

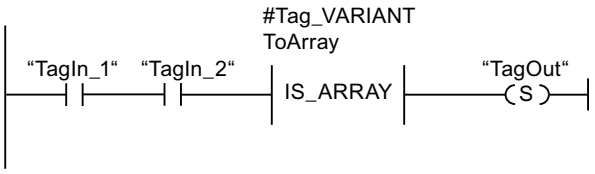
The following table shows the parameters of the "Check for ARRAY" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	I, Q, M, L	Operand that is queried for ARRAY

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_VARIANTToArray" operand has the ARRAY data type.

See also

Overview of the valid data types (Page 1908)

Math functions

CALCULATE: Calculate

Description

The "Calculate" instruction is used to define and execute an expression for the calculation of mathematical operations or complex logic operations depending on the selected data type.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. Depending on the data type selected, you can combine the functions of certain instructions to perform a complex calculation. The expression to be calculated is specified via a dialog you can open via the "Calculator" icon at the top of the instruction box. The expression can contain names of input parameters and the syntax of the instructions. Operand names and operand addresses cannot be specified.

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box.

The values of the inputs are used to execute the specified expression. Not all of the defined inputs have to be used in the expression. The result of the instruction is transferred to the output OUT.

Note

If one of the mathematical operations fails in the expression, no result is transferred to the output OUT and the enable output ENO returns the signal state "1".

If you use inputs in the expression that are not available in the box, they are inserted automatically. This requires that there are no gaps in the numbering of the inputs to be newly

defined in the expression. For example, you cannot use the IN4 input in the expression unless the IN3 input has been defined.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the "Calculate" instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.
- An error occurred during execution of one of the instructions in the expression.

The following table shows the instructions that can be executed together in the expression of the instruction "Calculate", depending on the selected data type:

Data type	Instruction	Syntax	Example
Bit strings	AND: AND logic operation	AND	IN1 AND IN2 OR IN3
	OR: OR logic operation	OR	
	XOR: EXCLUSIVE OR logic operation	XOR	
	INV: Create ones complement	NOT	
	SWAP: Swap [1)]	SWAP	
Integers	ADD: Add	+	(IN1 + IN2) * IN3; (ABS(IN2)) * (ABS(IN1))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	MOD: Return remainder of division	MOD	
	INV: Create ones complement	NOT	
	NEG: Create twos complement	-(in1)	
	ABS: Form absolute value	ABS( )	

Data type	Instruction	Syntax	Example
Floating-point numbers	ADD: Add	+	((SIN(IN2) * SIN(IN2) + (SIN(IN3) * SIN(IN3)) / IN3)); (SQR(SIN(IN2)) + (SQR(COS(IN3)) / IN2))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	EXPT: Exponentiate	**	
	ABS: Form absolute value	ABS( )	
	SQR: Form square	SQR( )	
	SQRT: Form square root	SQRT( )	
	LN: Form natural logarithm	LN( )	
	EXP: Form exponential value	EXP( )	
	FRAC: Return fraction	FRAC( )	
	SIN: Form sine value	SIN( )	
	COS: Form cosine value	COS( )	
	TAN: Form tangent value	TAN( )	
	ASIN: Form arcsine value	ASIN( )	
	ACOS: Form arcosine value	ACOS( )	
	ATAN: Form arctangent value	ATAN( )	
	NEG: Create twos complement	-(in1)	
	TRUNC: Truncate numerical value	TRUNC( )	
	ROUND: Round numerical value	ROUND( )	
	CEIL: Generate next higher integer from floating-point number	CEIL( )	
	FLOOR: Generate next lower integer from floating-point number	FLOOR( )	
[1] Not possible for data type BYTE.			

Parameters

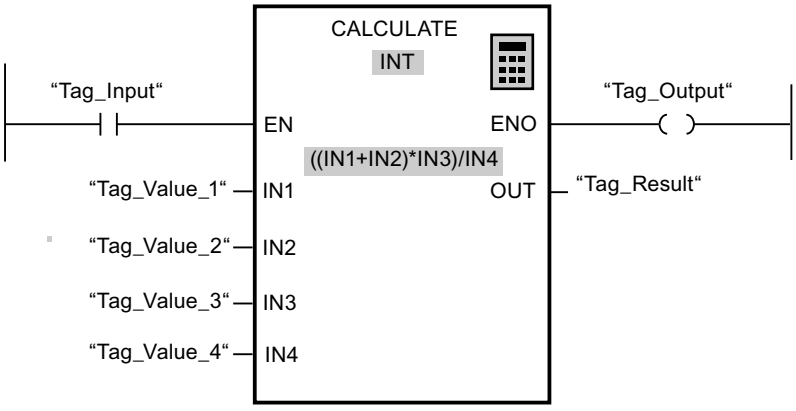
The following table shows the parameters of the "Calculate" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	First available input
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	Second available input
INn	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	Additionally inserted inputs
OUT	Output	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P	Output to which the end result is to be transferred.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value_1	4
IN2	Tag_Value_2	4
IN3	Tag_Value_3	3
IN4	Tag_Value_4	2
OUT	Tag_Result	12

If the "Tag_Input" input has the signal state"1", the "Calculate" instruction is executed. The value of operand "Tag_Value_1" is added to the value of operand "Tag_Value_2". The sum is multiplied with the value of operand "Tag_Value_3". The product is divided by the value of operand "Tag_Value_4". The quotient is transferred as end result to the operand "Tag_Result" at the OUT output of the instruction. If the instruction is executed without errors, the enable output ENO and operand "Tag_Output" are set to "1".

See also

Overview of the valid data types (Page 1908)

Example of calculating an equation (Page 3867)

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 and the value at input IN2 and query the sum at output OUT (OUT := IN1+IN2).

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. When

the instruction is executed, the values of all available input parameters are added. The sum is stored at the OUT output.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Add" instruction:

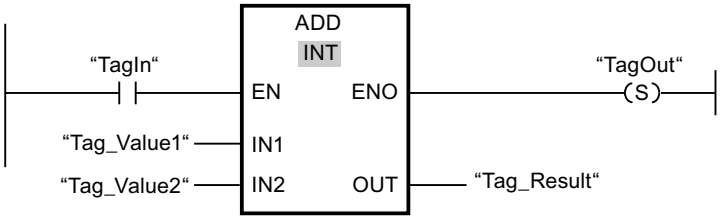
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second number to be added
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Optional input values that are added.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Sum

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Add" instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at output OUT (OUT := IN1-IN2).

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Subtract" instruction:

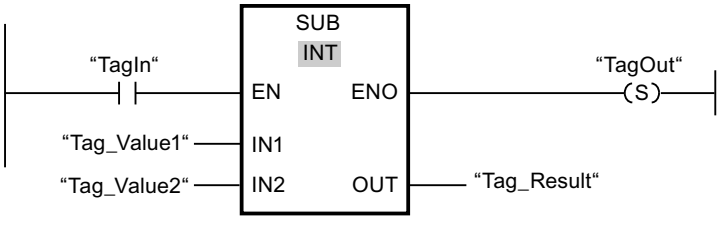
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Subtracting
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Difference

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Subtract" instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of

the subtraction is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at input IN1 with the value at input IN2 and query the product at output OUT (OUT := IN1*IN2).

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are multiplied. The product is stored at the OUT output.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Multiply" instruction:

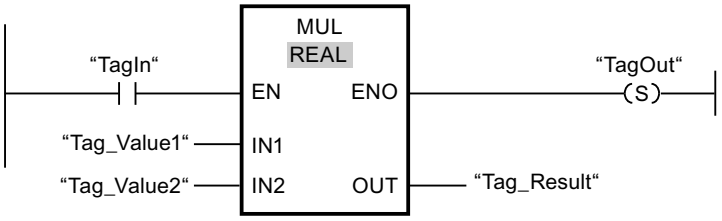
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Multiplier
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Number being multiplied
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Optional input values that can be multiplied.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Product

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The result of the multiplication is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT (OUT := IN1/IN2).

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Divide" instruction:

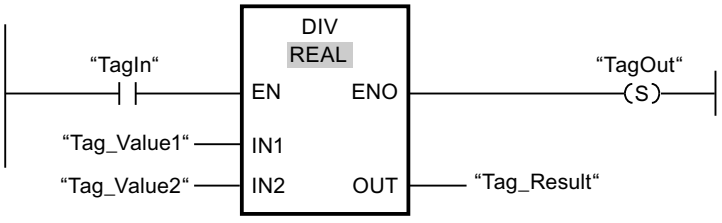
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Quotient value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Divide" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The division result is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at input IN1 by the value at input IN2 and query the remainder of division at output OUT.

Parameters

The following table shows the parameters of the "Return remainder of division" instruction:

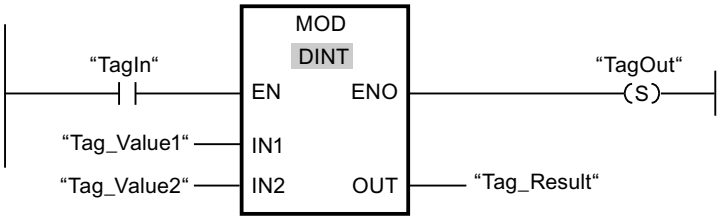
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers	I, Q, M, D, L, P, or constant	Dividend
IN2	Input	Integers	I, Q, M, D, L, P, or constant	Divisor
OUT	Output	Integers	I, Q, M, D, L, P	Remainder of division

You can select the data type of the instruction from the "<Auto ???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Return remainder of division" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder of division is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at the IN input and query the result at the OUT output. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

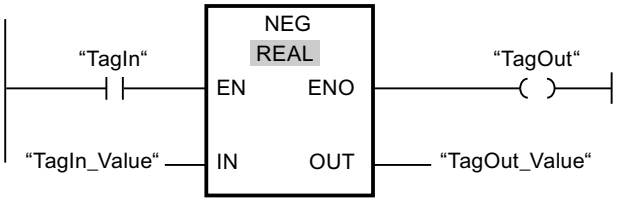
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Create twos complement" instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is provided at "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

INC: Increment

Description

You can use the "Increment" instruction to change the value of the operand at the IN/OUT parameter to the next higher value and query the result. The "Increment" instruction is only started when the signal state at the EN enable input is "1". If no overflow error occurs during the execution, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Increment" instruction:

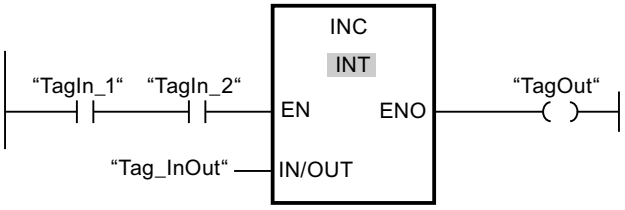
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	Value to be incremented.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of operand "Tag_InOut" is incremented by one and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

DEC: Decrement

Description

You can use the "Decrement" instruction to change the value of the operand at the IN/OUT parameter to the next lower value and query the result. The "Decrement" instruction is only started when the signal state at the EN enable input is "1". If the range of values of the selected data type is not exceeded during processing, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- A floating-point number has an invalid value.

Parameter

The following table shows the parameters of the "Decrement" instruction:

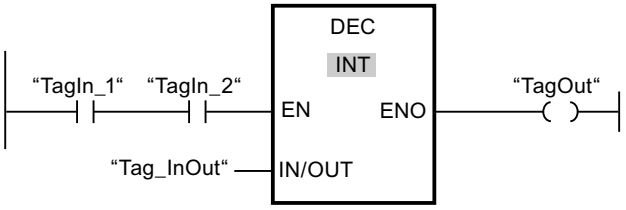
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	Value to be decremented.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of operand "Tag_InOut" is decremented by one and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ABS: Form absolute value

Description

You can use the "Form absolute value" instruction to calculate the absolute value of the value specified at input IN. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Form absolute value" instruction:

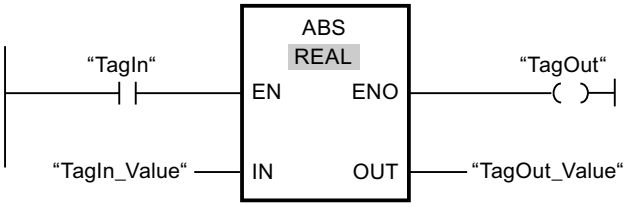
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

If operand "TagIn" has the signal state "1", the "Form absolute value" instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the available inputs and writes the lowest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

A minimum of two and a maximum of 100 inputs must be specified for the execution of the instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

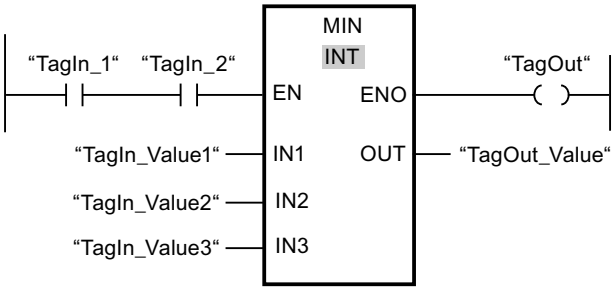
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second input value
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Additionally inserted inputs whose values are to be compared
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result
When the IEC check is not activated, you can also use tags of the data type TIME, LTIME, TOD, LTOD, DATE and LDT by selecting an equally long bit string or integer as data type of the instruction (e.g. instead of TIME => DINT, UDINT or DWORD = 32 bit)				

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the available inputs and writes the highest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

A minimum of two and a maximum of 100 inputs must be specified for the execution of the instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

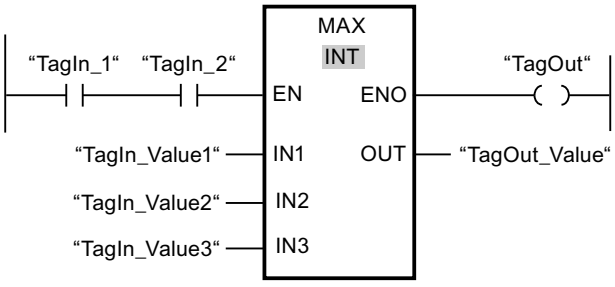
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second input value
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Additionally inserted inputs whose values are to be compared
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result
When the IEC check is not activated, you can also use tags of the data type TIME, LTIME, TOD, LTOD, DATE and LDT by selecting an equally long bit string or integer as data type of the instruction. (e.g. instead of TIME => DINT, UDINT or DWORD = 32 bit)				

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN, the output OUT is set to the value of the MN input. If the MX high limit is exceeded, the OUT output is set to the value of the MX input.

If the value at the MN input is greater than at the MX input, the result is undefined and the enable output ENO is "0".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN input is greater than the value of the MX input.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
MN	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Low limit
IN	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Input value
MX	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	High limit

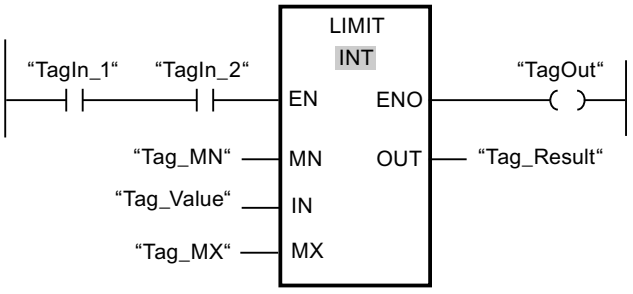
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
OUT	Output	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Set limit value" instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value of operand "Tag_Value" is less than the low limit, the value of operand "Tag_MN" is copied to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SQR: Form square

Description

The instruction "Form square" is used to square the value of a floating-point number at input IN and to write the result to output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form square":

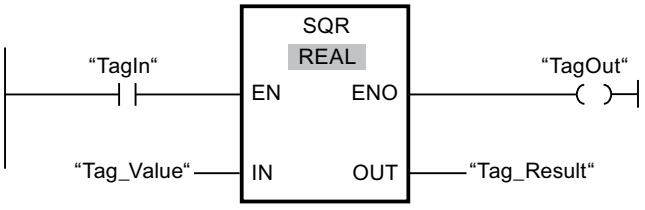
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If the operand "TagIn" has the signal state "1", the "Form square" instruction is executed. The instruction squares the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 1908)

SQRT: Form square root

Description

The instruction "Form square root" is used form the square root from the value of the floating-point number at input IN and to write the result to output OUT. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number. If the value at input IN is "0", the result is also "0".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form square root":

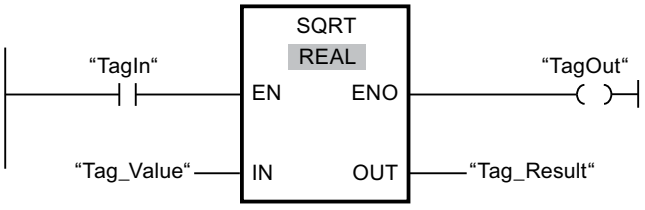
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square root of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If the operand "TagIn" has the signal state "1", the "Form square root" instruction is executed. The instruction calculates the square root of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 1908)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) of the value at input IN. The result is sent to output OUT and can be queried there. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form natural logarithm":

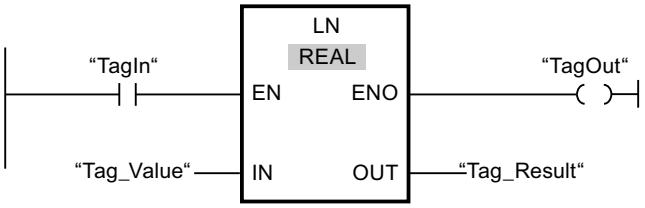
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Form natural logarithm" instruction is executed. The instruction forms the natural logarithm of the value at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 1908)

EXP: Form exponential value

Description

You can use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value specified at input IN. The result is provided at output OUT and can be queried there (OUT = e^{IN}).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form exponential value":

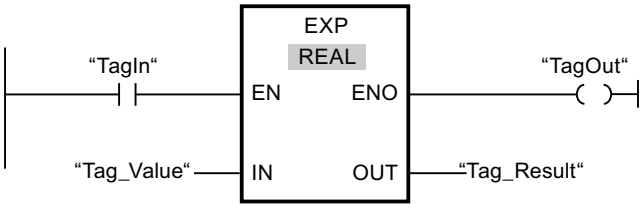
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Exponential value of input value IN

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Form exponential value" instruction is executed. The instruction forms the exponent from base e and the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 1908)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form sine value" instruction:

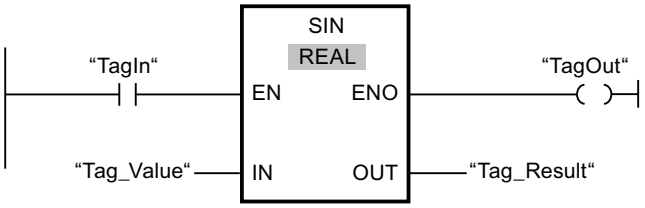
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Sine of the specified angle

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

If operand "TagIn" has the signal state "1", the "Form sine value" instruction is executed. The instruction calculates the sine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form cosine value" instruction:

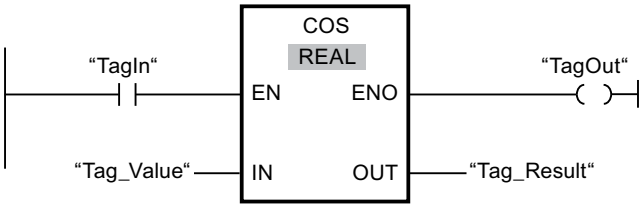
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Cosine of the specified angle

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

If operand "TagIn" has the signal state "1", the "Form cosine value" instruction is executed. The instruction calculates the cosine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the tangent of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form tangent value" instruction:

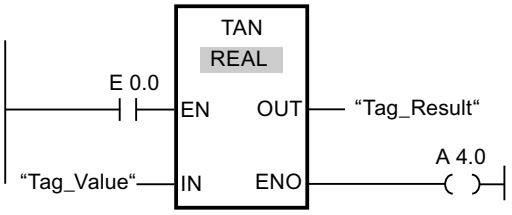
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Tangent of the specified angle

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the tangent of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from the arcsine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the "Form arcsine value" instruction:

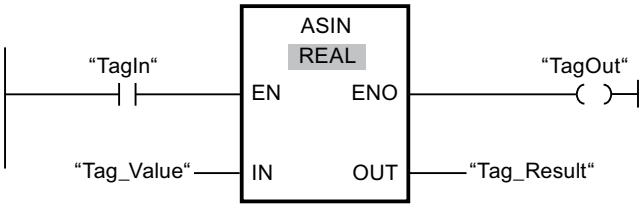
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

If operand "TagIn" has the signal state "1", the "Form arcsine value" instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from the cosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from 0 to +π.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the "Form arccosine value" instruction:

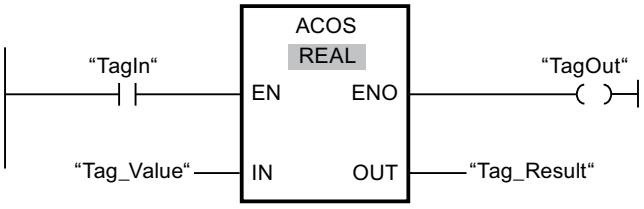
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

If operand "TagIn" has the signal state "1", the "Form arccosine value" instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from the tangent value specified at input IN, which corresponds to this value. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) at input IN. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form arctangent value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

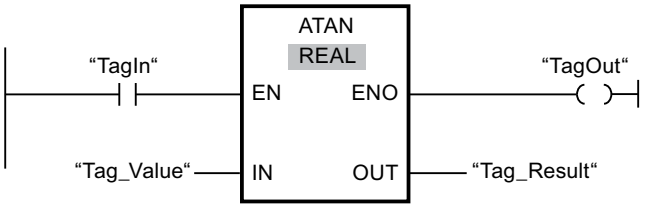
Parameters	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Tangent value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398 (π/4)

If the operand "TagIn" has the signal state "1", the "Form arctangent value" instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 1908)

Invalid floating-point numbers (Page 1927)

FRAC: Return fraction

Description

You can use the "Return fraction" instruction to determine the decimal places of the value at the IN input. The result of the query is stored at the OUT output and can be queried there. If, for example, the IN input has the value 123.4567, the OUT output returns the value 0.4567.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Return fraction" instruction:

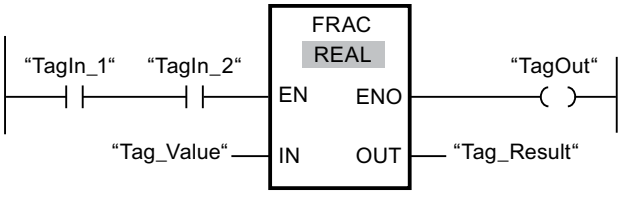
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Value, whose decimal places are to be determined.
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Decimal places of the value at the IN input

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	2.555
OUT	Tag_Result	0.555

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Return fraction" instruction is started. The decimal places from the value of operand "Tag_Value" are copied to operand "Tag_Result". If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

EXPT: Exponentiate

Description

You can use the "Exponentiate" instruction to raise the value at the IN1 input to a power specified with the value at the IN2 input. The result of the instruction is provided at the OUT output and can be queried there (OUT =$IN1^{IN2}$).

The IN1 input can only be assigned valid floating-point numbers. Integers can also be assigned to the IN2 input.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- Errors occur during execution of the instruction, for example, an overflow occurs.

Parameters

The following table shows the parameters of the "Exponentiate" instruction:

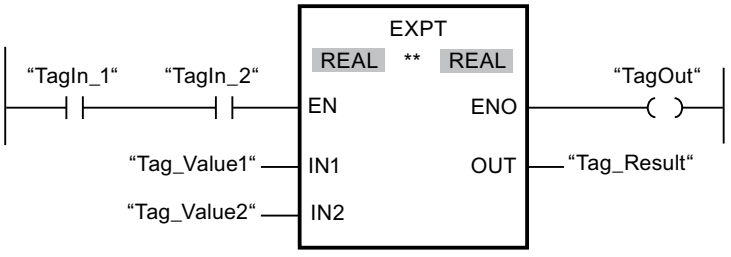
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Base value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Value with which the base value is exponentiated
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Exponentiate" instruction is started. The value of operand "Tag_Value1" is raised to a power specified with the value of operand "Tag_Value2". The result is stored in the "Tag_Result" output. If the instruction is

executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of ascending address.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data type at the IN parameter does not correspond to the specified data type at the OUT1 parameter.

The following table lists the possible transfers for the S7-1200 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE ,TOD, CHAR
WORD	WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
USINT	USINT, UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
INT	INT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UINT	UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UDINT	UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LREAL

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
TIME	TIME	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME
DATE	DATE	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, DATE
TOD	TOD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, CHAR, Character of a string[1)]
WCHAR	WCHAR	BYTE, WORD, DWORD, CHAR, WCHAR, character of a character string[1)]
Character of a string[1)]	Character of a string	CHAR, WCHAR, character of a string
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER

The following table lists the possible transfers for the S7-1500 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE ,TOD, LTOD, CHAR
WORD	WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, S5TIME, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
DWORD	DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
LWORD	LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LREAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
SINT	SINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
USINT	USINT, UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
INT	INT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UINT	UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD. LTOD
DINT	DINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UDINT	UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
LINT	LINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
ULINT	ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LWORD, LREAL
S5TIME	S5TIME	WORD, S5TIME
TIME	TIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME
LTIME	LTIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTIME
DATE	DATE	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, DATE
DT	DT	DT
LDT	LDT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LDT
TOD	TOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TOD
LTOD	LTOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, LWORD, CHAR, Character of a string[1)]
WCHAR	WCHAR	BYTE, WORD, DWORD, LWORD, CHAR, WCHAR, Character of a string[1)]
Character of a string[1)]	Character of a string	CHAR, WCHAR, character of a string

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
COUNTER	COUNTER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	TIMER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_LTIMER	IEC_LTIMER	IEC_LTIMER
IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNT-ER	IEC_LCOUNTER	IEC_LCOUNTER
IEC_UL-COUNTER	IEC_ULCOUNTER	IEC_ULCOUNTER

[1)] You can also use the "Move value" instruction to transfer individual characters of a string to operands of the CHAR or WCHAR data type. The number of the character to be transferred is specified in square brackets next to the operand name. "MyString[2]", for example, transfers the second character of the "MyString" string. The transfer of the operand of the data type CHAR or WCHAR to the individual character of a string is also possible. You can also replace a specific character of a string with the character of another string.

[2)] Transferring entire arrays (ARRAY) is possible only when the array components of the operands at input IN and at output OUT1 are of the same data type.

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at input IN is less than the bit length of the data type at output OUT1, the higher-order bits of the destination value will be overwritten with zeros.

In its initial state, the instruction box contains 1 output (OUT1). The number of outputs can be extended. The added outputs are numbered in ascending order on the box. During the execution of the instruction, the content of the operand at the input IN is transferred to all available outputs. The instruction box cannot be extended if structured data types (DTL, STRUCT, ARRAY) or characters of a string are transferred.

You can also use the "Move block" (MOVE_BLK) and "Move block uninterruptible" (UMOVE_BLK) instructions to move operands of the ARRAY data type. Operand of the data type STRING or WSTRING can be copied with the instruction "Move character string" (S_MOVE).

Parameter

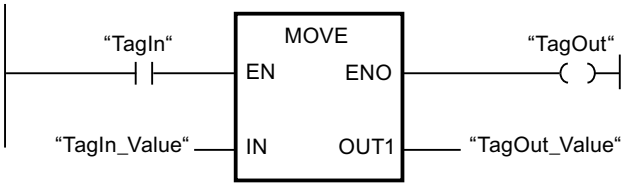
The following table lists the parameters of the "Move value" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, timers, date and time, CHAR, WCHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, timers, date and time, CHAR; WCHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data type, PLC data type (UDT)	I, Q, M, D, L or constant	Source value
OUT1	Output	Bit strings, integers, floating-point numbers, timers, date and time, CHAR, WCHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, timers, date and time, CHAR; WCHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data type, PLC data type (UDT)	I, Q, M, D, L	Operands to which in the source value is transferred.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If the operand "TagIn" has the signal state "1", the "Move value" instruction is executed. The instruction copies the contents of operand "TagIn_Value" to operand "TagOut_Value" and sets output "TagOut" to signal state "1".

See also

Overview of the valid data types (Page 1908)

MOVE_BLK: Move block (Page 2344)

UMOVE_BLK: Move block uninterruptible (Page 2350)

S_MOVE: Move character string (Page 3001)

Deserialize: Deserialize

Description

You can use the "Deserialize" instruction to convert the sequential representation of a PLC data type (UDT) back to a PLC data type and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type is located must have the ARRAY of BYTE data type and be declared with standard access. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion.

The instruction enables you to convert multiple sequential representations of converted PLC data types back to their original state step-by-step.

If you only want to convert back a single sequential representation of a PLC data type (UDT), you can also use the instruction "TRCV: Receive data via communication connection".

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", the bit has a length of 1 byte.

Parameters

The following table shows the parameters of the "Deserialize" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_ARRAY	Input	VARIANT	I, Q, M, L	Global data block in which the generated data stream is stored
DEST_VARIABLE	InOut	VARIANT	I, Q, M, L	Tag in which the returned converted PLC data type (UDT) is stored

Parameters	Declaration	Data type	Memory area	Description
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

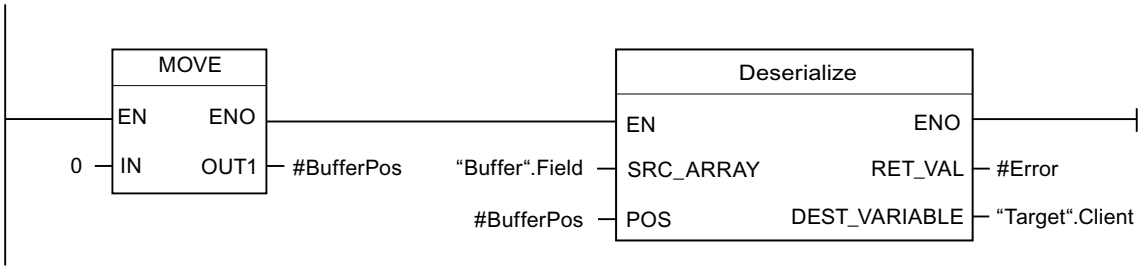
The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The data block at the DEST_VARIABLE parameter is not a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains no value.
8151	Code generation error at the SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8250	The VARIANT data type at the DEST_VARIABLE parameter contains no value.
8251	Code generation error at the DEST_VARIABLE parameter
8254	Invalid data type at the DEST_VARIABLE parameter
8382	The value at parameter POS is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

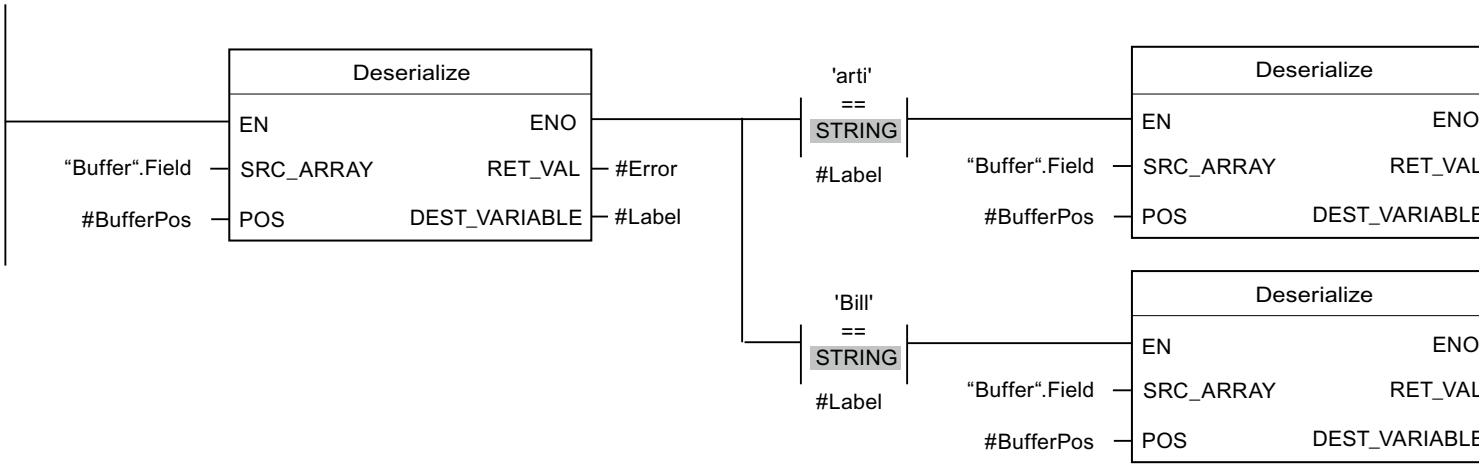
The following example shows how the instruction works:

Network 1:



The "Move value" instruction moves the value "0" to the "#BufferPos" operand. The "Deserialize" instruction deserializes the sequential representation of the customer data from the "Buffer" data block and writes it to the "Target" data block. The number of bytes used by the converted customer data is stored in the "#BufferPos" operand.

Network 2:



The "Deserialize" instruction deserializes the sequential representation of the separator sheet, which was saved with the sequential representation after the customer data, from the "Buffer" data block and writes the characters to the "#Label" operand. The characters are compared using comparison instructions "arti" and "Bill". If the comparison for "arti" = TRUE, these are article data that have been deserialized and written to the "Target" data block. If the comparison for "Bill" = TRUE, these are billing data that have been deserialized and written to the "Target" data block.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table shows the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 1908)

PLC data types (Page 1954)

Serialize: Serialize

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT) to a sequential representation without losing parts of their structure.

You can use the instruction to cache multiple structured data from your program in a buffer, which is preferably in a global data block, and send it to another CPU. The memory area in which the converted PLC data types are stored must have the ARRAY of BYTE data type and be declared with standard access. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion.

The operand at the POS parameter contains information about the number of bytes used by the converted PLC data types.

If you want to send a single PLC data type (UDT), you can directly call the instruction "TSEND: Send data via communication connection".

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", the bit has a length of 1 byte.

Parameters

The following table shows the parameters of the instruction "Serialize":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_VARIABLE	Input	VARIANT	I, Q, M, L	PLC data type (UDT) that is converted to sequential representation.

Parameters	Declaration	Data type	Memory area	Description
DEST_ARRAY	InOut	VARIANT	I, Q, M, L	Data block in which the generated data stream is stored.
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

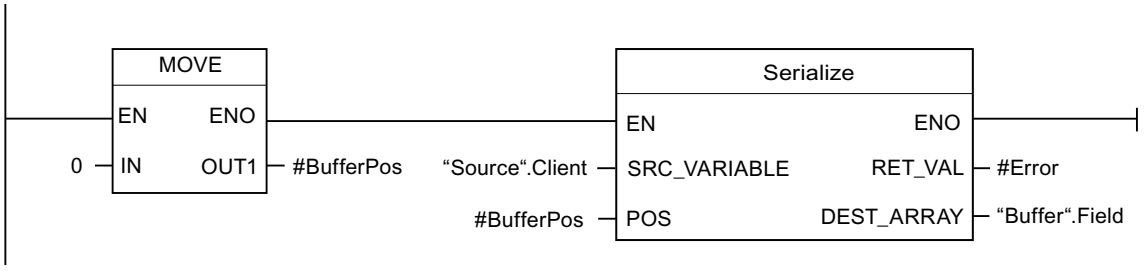
The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains no value.
8152	Code generation error at the SRC_VARIABLE parameter
8236	The data block at the DEST_ARRAY parameter is not a block with standard access.
8250	The VARIANT data type at the DEST_ARRAY parameter contains no value.
8252	Code generation error at the DEST_ARRAY parameter
8253	There is not enough free memory available at the DEST_ARRAY parameter.
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at parameter POS is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

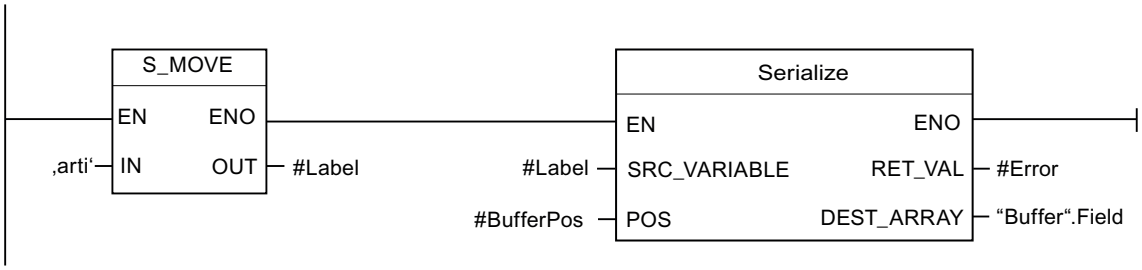
The following example shows how the instruction works:

Network 1:



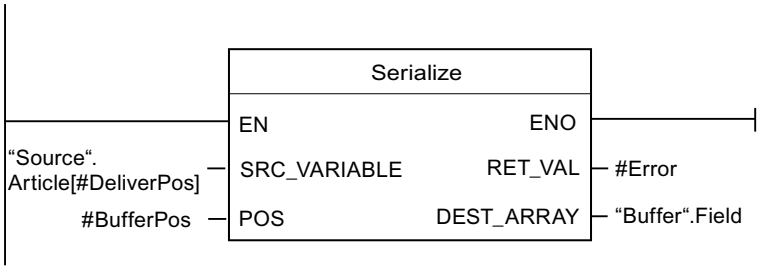
The "Move value" instruction moves the value "0" to the "#BufferPos" operand. The "Serialize" instruction serializes the customer data from the "Source" data block and writes it in sequential representation to the "Buffer" data block. The number of bytes used by the sequential representation is stored in the "#BufferPos" operand.

Network 2:



A kind of separator sheet is inserted now to make it easier to deserialize the sequential representation later. The "Move string" instruction moves the "arti" characters to the "#Label" operand. The "Serialize" instruction writes these characters after the customer data to the "Buffer" data block. The number of bytes that the characters need is added to the number already stored in the "#BufferPos" operand.

Network 3:



The "Serialize" instruction serializes the data of a specific article, which is calculated in runtime, from the "Source" data block and writes it in sequential representation to the "Buffer" data block after the "arti" characters.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table shows the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT

Name of the PLC data types	Name	Data type
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 1908)

PLC data types (Page 1954)

MOVE_BLK: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be moved to the destination area is specified at input COUNT. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source area and the destination area have the same data type.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Parameters

The following table lists the parameters of the "Move block" instruction:

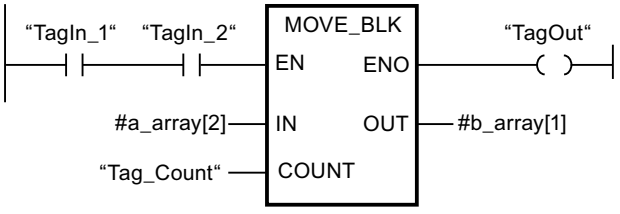
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P, or constant	Number of elements to be moved from the source area to the destination area.
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Move block" instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MOVE_BLK_VARIANT: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). You can copy a complete ARRAY or elements of an ARRAY to another ARRAY of the same data type. The size (number of elements) of source and destination ARRAY may be different. You can copy multiple or single elements within an ARRAY.

When you use the instruction at the time to the block is created, the ARRAY does not yet have to be known, as the source and the designation are transferred per VARIANT.

Counting at the parameters SRC_INDEX and DEST_INDEX always begins with the low limit "0", regardless of the later declaration of the ARRAY.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is copied than is made available.

Note

VARIANT in connection with BOOL data type

If you want to interconnect a parameter of data type VARIANT (source or destination area) with a tag of data type BOOL or an ARRAY of BOOL, you have the following options:

1. You can address it symbolically
   Example: Parameter SRC: "Data_block".myArray
2. You can address it with help of ANY pointer. However, you must note that the specified length of the area must be dividable by 8, or the instruction will otherwise not be executed.
   Example: Parameter SRC: P#DB123.DBX456.0 BOOL 1000

Parameters

The following table lists the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
SRC	Input	VARIANT (which points to an ARRAY or an individual ARRAY element), ARRAY of <Data_type>	I, Q, M, L	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L or constant	Number of elements which are copied Assign the value "1" to the parameter COUNT, if no ARRAY is specified at parameter SRC or at Parameter DEST.
SRC_INDEX	Input	DINT	I, Q, M, D, L or constant	• The SRC_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter SRC, the integer at parameter SRC_INDEX specifies the first element within the source area from which it is to be copied. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter SRC or only one single element of an ARRAY is specified, then assign the value "0" at parameter SRC_INDEX.
DEST_INDEX	Input	DINT	I, Q, M, D, L or constant	• The DEST_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter DEST, the integer at parameter DEST_INDEX specifies the first element within the destination area to it is to be copied. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter DEST, then assign the value "0" at parameter DEST_INDEX.
DEST	Output [1)]	VARIANT	I, Q, M, L	Destination area into which the contents of the source block are copied.

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
1) The DEST parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

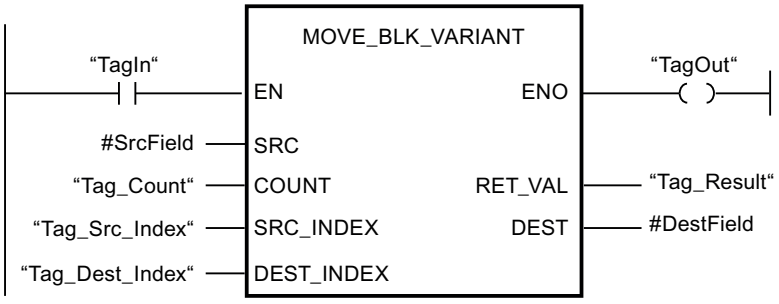
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at the SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value
8382	The value at the SRC_INDEX parameter is outside the limits of the VARIANT.
8383	The value at parameter SRC_INDEX is outside the high limit of the ARRAY.
8482	The value at the DEST_INDEX parameter is outside the limits of the VARIANT.
8483	The value at parameter DEST_INDEX is outside the high limit of the ARRAY.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at the DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Declaration in the block interface	Operand	Value
SRC	Input	#SrcField	The local operand #SrcField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[0..10] of "MOVE_UDT"
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	The local operand #DestField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[10..20] of "MOVE_UDT"

If the operand "TagIn" has the signal state "1", the "Move block" instruction is executed. Two elements are copied from the source area, beginning from the fourth element of the ARRAY of UDT, to the destination area. The copies are inserted in the ARRAY of UDT starting from the fourth element. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

VariantGet: Read out VARIANT tag value (Page 2367)

Programming example: Moving data (Page 267)

UMOVE_BLK: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be moved to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

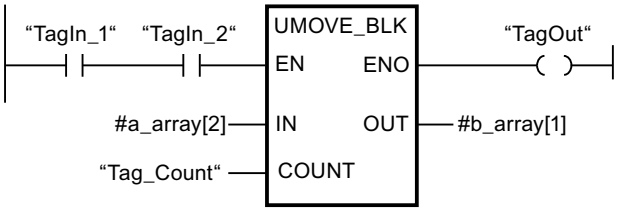
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN [1)]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P, or constant	Number of elements to be moved from the source area to the destination area.

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Move block uninterruptible" instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The move operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the value of the IN input. The destination area is filled beginning with the address specified at the OUT output. The number of repeated move operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is selected and moved to the destination area as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P, or constant	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P, or constant	Number of repeated move operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	Address in destination area from which filling starts.

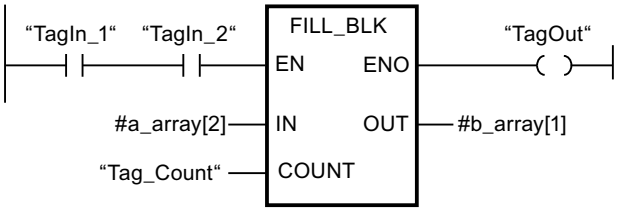
[1] The specified data types can also be used as elements of an ARRAY structure.

[2] The specified data types can only be used as elements of an ARRAY structure.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Fill block" instruction is executed. The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. If the instruction is executed without errors, the ENO and "TagOut" enable outputs are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

UFILL_BLK: Fill block uninterruptible

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (destination area) with the value of the IN input without interruption. The destination area is filled beginning with the address specified at the OUT output. The number of repeated move operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is selected and moved to the destination area as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

You can use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P, or constant	Element used to fill the destination area.
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P, or constant	Number of repeated move operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	Address in destination area from which filling begins.

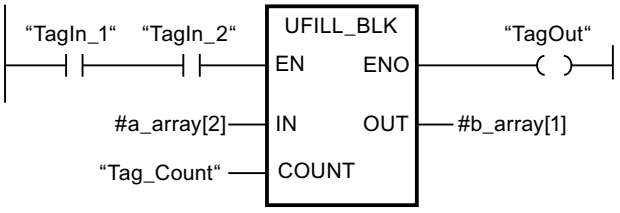
[1] The specified data types can also be used as elements of an ARRAY structure.

[2] The specified data types can only be used as elements of an ARRAY structure.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Fill block uninterruptible" instruction is executed. The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. The move operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the ENO and "TagOut" enable outputs are set to signal state "1".

See also

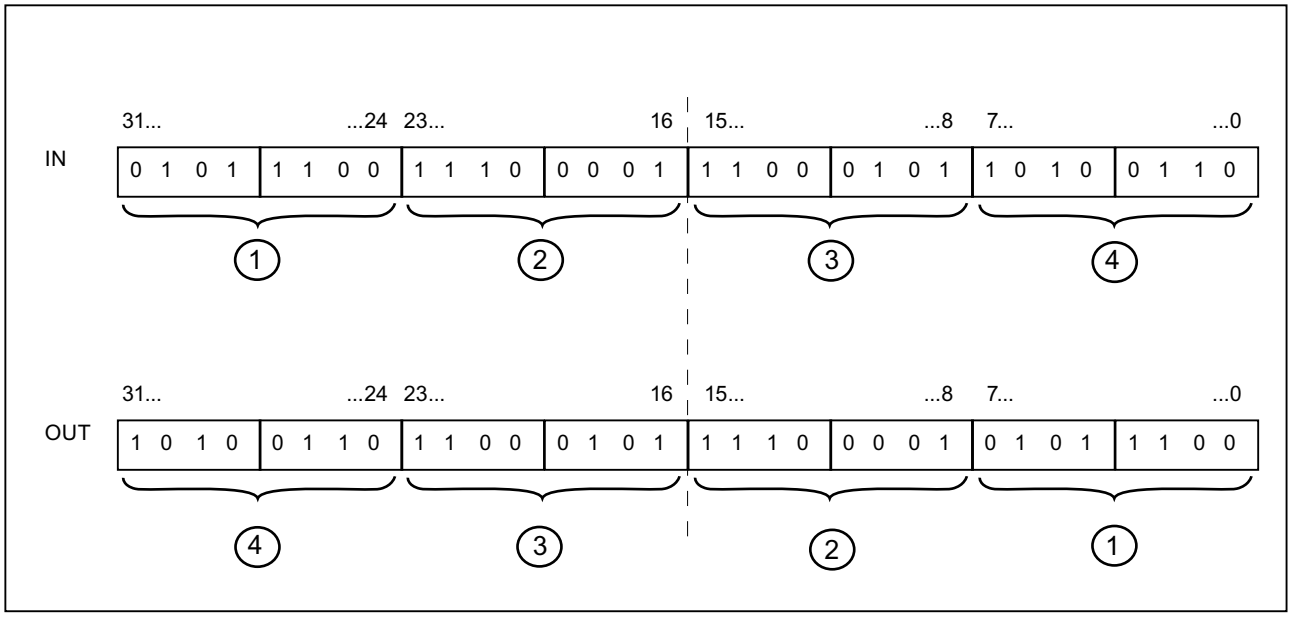
Overview of the valid data types (Page 1908)

SWAP: Swap

Description

You can use the "Swap" instruction to change the order of the bytes at input IN and query the result at output OUT.

The following figure shows how the bytes of an operand of the DWORD data type are swapped using the "Swap" instruction:



Parameter

The following table lists the parameters of the "Swap" instruction:

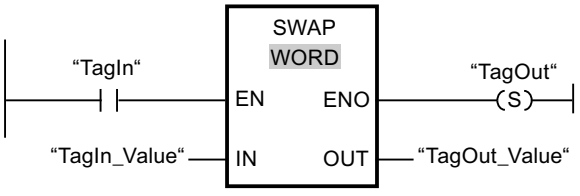
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L or, P constant	Operand whose bytes are swapped.
OUT	Output	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0000 1111 0101 0101
OUT	TagOut_Value	0101 0101 0000 1111

If operand "TagIn" has the signal state "1", the "Swap" instruction is executed. The order of the bytes is changed and stored in operand "TagOut_Value".

See also

Overview of the valid data types (Page 1908)

ARRAY DB

ReadFromArrayDB: Read from array data block

Description

The "Read from array data block" instruction is used to read data from an ARRAY data block and write it to a destination area.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be PLC data type or any other elementary data type. The counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Read from array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P, or constant	Element that is read
VALUE	Output [1)]	VARIANT	I, Q, M, L	Value that is read and output

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
1) The VALUE parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

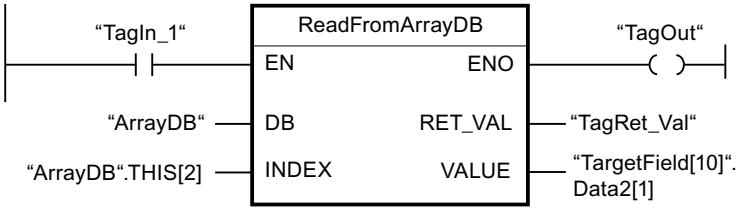
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, write protected, or is located in load memory.
8135	The ARRAY data block contains invalid values.
8154	The data block has the incorrect data type.
8282	The value at parameter INDEX is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8452	Code generation error
8453	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	TargetField[10].Data2[1]	The data type of the "Target-Field" operand is Array [10 to 20] of "MOVE_UDT".

If the "TagIn1" operand has the signal state "1", the "Read from the array data block" instruction is executed. The third element is read from "ArrayDB" and written to the "TargetField[10].Data2[1]" operand. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Using ARRAY data blocks (Page 239)

WriteToArrayDB: Write to array data block

Description

The instruction "Write to array data block" is used to write data to an ARRAY data block.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be PLC data type or any other elementary data type. The counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Write to array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P, or constant	Element in the DB to which data is written

Parameter	Declaration	Data type	Memory area	Description
VALUE	Input	VARIANT	I, Q, M, L	Value to be written
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

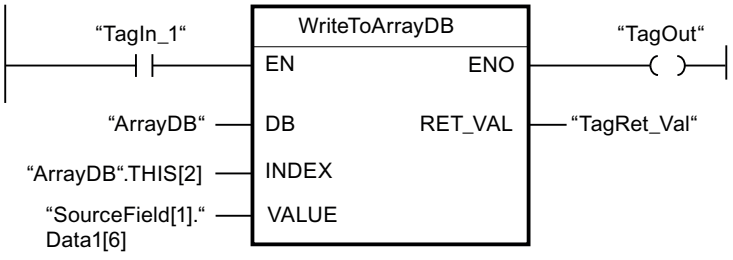
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, or is located in load memory.
8134	The data block is write protected.
8135	The data block is not an ARRAY data block.
8154	The data block has the incorrect data type.
8282	The value at parameter INDEX is outside the limits of the ARRAY.
8350	The data type VARIANT at parameter VALUE provides the value "0".
8352	Code generation error
8353	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceField[1].Data1[6]	The data type of the "Source-Field" operand is Array [0 to 10] of "MOVE_UDT".

If the "TagIn1" operand has the signal state "1", the "Write to ARRAY data block" instruction is executed. From the "SourceField" operand, the "Data1[6]" element from the second element is written to the "ArrayDB". The third element is written to the "ArrayDB". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Using ARRAY data blocks (Page 239)

ReadFromArrayDBL: Read from array data block in load memory

Description

The instruction "Read from array data block in load memory" is used to read data from an ARRAY data block in the load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be PLC data type or any other elementary data type. The counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

Note

The ARRAY data block must be created with the "Optimized" block property.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table lists the parameters of the "Read from array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Begin with the reading of the ARRAY DB
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P, or constant	Element that is read
VALUE	InOut	VARIANT	I, Q, M, L	Value that is read and output No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

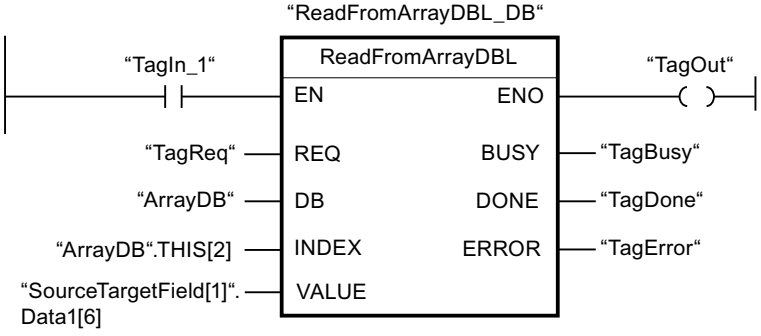
Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.

Error code* (W#16#...)	Explanation
8382	The value at parameter INDEX is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in the load memory" and "WRIT_DBL: Write to data block in the load memory" under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the "SourceTargetField" operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq"operand, the "Read from ARRAY data block in load memory" instruction is executed. The third element is read from "ArrayDB" and written to the "SourceTargetField[1].Data1[6]" operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the

"TagOut" output is set. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 1908)

READ_DBL: Read from data block in the load memory (Page 3351)

WRIT_DBL: Write to data block in the load memory (Page 3353)

Using ARRAY data blocks (Page 239)

WriteToArrayDBL: Write to array data block in load memory

Description

The instruction "Write to array data block in load memory" is used to write data to an ARRAY data block in the load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be PLC data type or any other elementary data type. The counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

Note

The ARRAY data block must be created with the "Optimized" block property.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table lists the parameters of the "Write to array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Start writing to the array DB
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P, or constant	Element in the DB to which data is written
VALUE	Input	VARIANT	I, Q, M, L	Value to be written No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

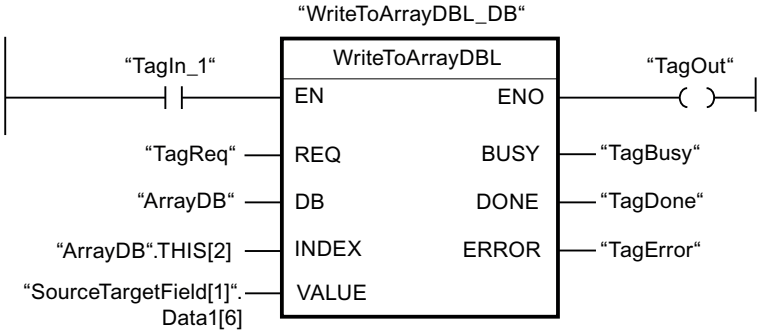
Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8234	The data block is write protected.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at parameter INDEX is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error

Error code* (W#16#...)	Explanation
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in the load memory" and "WRIT_DBL: Write to data block in the load memory" under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the "SourceTargetField" operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq"operand, the "Write to ARRAY data block in load memory" instruction is executed. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written in the third element in the "ArrayDB". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 1908)

READ_DBL: Read from data block in the load memory (Page 3351)

WRIT_DBL: Write to data block in the load memory (Page 3353)

Using ARRAY data blocks (Page 239)

VARIANT

VariantGet: Read out VARIANT tag value

Description

You can use the "Read out VARIANT tag value" instruction to read the value of the tag to which the VARIANT at the SRC parameter points and write it in the tag at the DST parameter.

The SRC parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the DST parameter.

The data type of the tag at the DST parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction. For additional information, refer to "See also".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameter

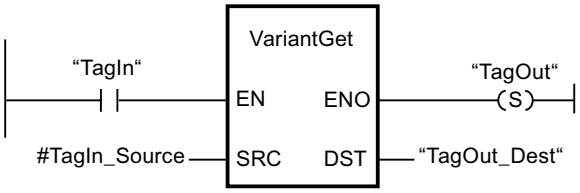
The following table lists the parameters of the instruction "Read out VARIANT tag value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	VARIANT	I, Q, M, L	Tag to be read
DST	Output	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The value of the tag to which the VARIANT at the "#TagIn_Source" operand points, is read and written to the "TagOut_Dest" operand.

See also

Overview of the valid data types (Page 1908)

VARIANT (Page 1951)

MOVE_BLK_VARIANT: Move block (Page 2346)

VariantPut: Write VARIANT tag value

Description

You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter to the tag at the DST parameter to which the VARIANT points.

The DST parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction. For additional information, refer to "See also".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameters

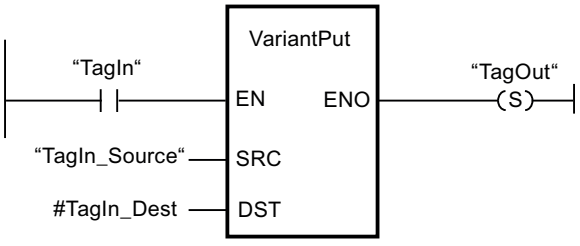
The following table lists the parameters of the instruction "Write VARIANT tag value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	Tag to be read
DST	Input	VARIANT	I, Q, M, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The value of the "TagIn_Source" operand is written to the tag to which the VARIANT at the #TagIn_Dest operand points.

See also

Overview of the valid data types (Page 1908)

VARIANT (Page 1951)

MOVE_BLK_VARIANT: Move block (Page 2346)

CountOfElements: Get number of ARRAY elements

Description

You can use the "Get number of ARRAY elements" instruction to query how many ARRAY elements a tag pointing to an VARIANT has.

If it is a one-dimensional ARRAY, the difference between the high and low limit +1 is output as a result. If it is a multi-dimensional ARRAY, the product of all dimensions is output as the result.

The IN parameter must have the VARIANT data type.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The VARIANT tag is not an ARRAY. (The result is "0".)

If the VARIANT points to an ARRAY of BOOL, the fill elements are also included in the count. (For example, 8 is returned for an ARRAY[0..1] of BOOL)

Parameters

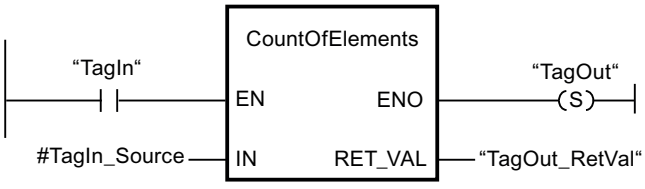
The following table shows the parameters of the "Get number of ARRAY elements" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	VARIANT	I, Q, M, L	Tag to be queried
RET_VAL	Output	UDINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has the signal state "1", the "Get number of ARRAY elements" instruction is executed. The number of ARRAY elements of the tag to which the VARIANT at the "#TagIn_Source" operand points is read and output at the "TagOut_RetVal" operand.

See also

Format of array (16-bit limits) (Page 1941)

Format of array (32-bit limits) (Page 1942)

Overview of the valid data types (Page 1908)

VARIANT (Page 1951)

Legacy

FieldRead: Read field

Description

You can use the "Read field" instruction to read out a specific component from the field specified at input MEMBER and transfer its content to the tag at output VALUE. You specify the index of the field component to be read at input INDEX. Specify the first component of the field from which reading is to occur at input MEMBER.

The data types of the field component at parameter MEMBER, the index and the tags at parameter VALUE must correspond to the data type of the instruction "Read field" because implicit conversion is not possible.

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- The field component specified at input INDEX is not defined in the field specified at output MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Read field":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
INDEX	Input	DINT	DINT	I, Q, M, D, L, P, or constant	Index of field component whose content is read out

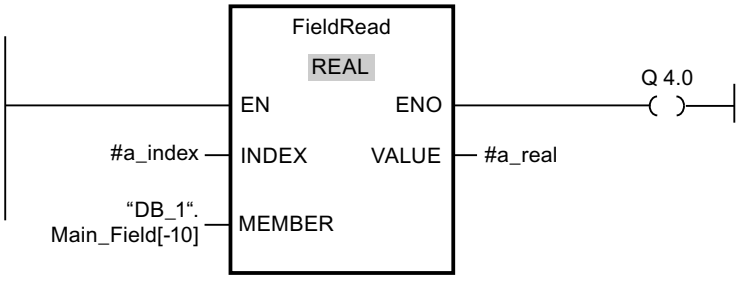
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
MEMBER	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR and WCHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD, CHAR and WCHAR as components of an ARRAY tag	D, L	First component of the field from which reading occurs.
VALUE	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD CHAR, WCHAR	I, Q, M, D, L, P	Operand to which the content of the field component is transferred

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Tag	Value
INDEX	a_index	4
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the data block "DB_1"
VALUE	a_real	Component with index 4 of the "Main_Field[-10..10] of REAL" field

The field component with index 4 is read from the "Main_Field[-10...10] of REAL" field and written to the "a_real" tag. The field component to be read is specified by the value at input INDEX.

See also

Overview of the valid data types (Page 1908)

FieldWrite: Write field

Description

The "Write field" instruction is used to transfer the content of the tag at the VALUE input to a specific component of the field at the MEMBER output. You can use the value at the INDEX input to specify the index of the field component that is described. At the MEMBER output, enter the first component of the field which is to be written to.

The data types of the field component at parameter MEMBER, the index and the tags at parameter VALUE must correspond to the data type of the instruction "Read field" because implicit conversion is not possible.

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- The field component specified at the input INDEX is not defined in the field specified at the output MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Write field":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
INDEX	Input	DINT	DINT	I, Q, M, D, L, P, or constant	Index of the field component that is being written with the content of VALUE.

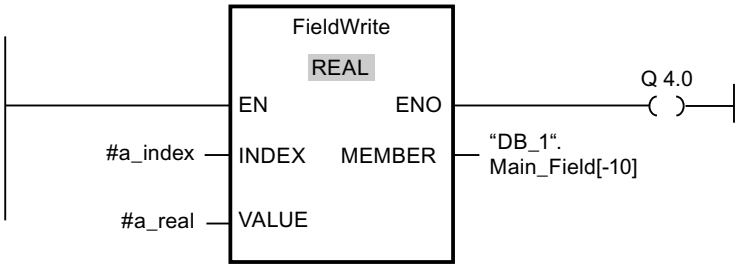
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD, CHAR, WCHAR	I, Q, M, D, L, P, or constant	Operand whose content is copied.
MEMBER	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR and WCHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD, CHAR and WCHAR as components of an ARRAY tag	D, L	First component of the field to which the content of VALUE is written.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
INDEX	a_index	4
VALUE	a_real	10.54
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the data block "DB_1"

The value "10.54" of the "a_real" tag is written to the field component with index 4 of the "Main_Field[-10..10] of REAL" field. The index of the field component to which the content of the tag "a_real" is transferred is specified by the value at the input INDEX.

See also

Overview of the valid data types (Page 1908)

BLKMOV: Move block

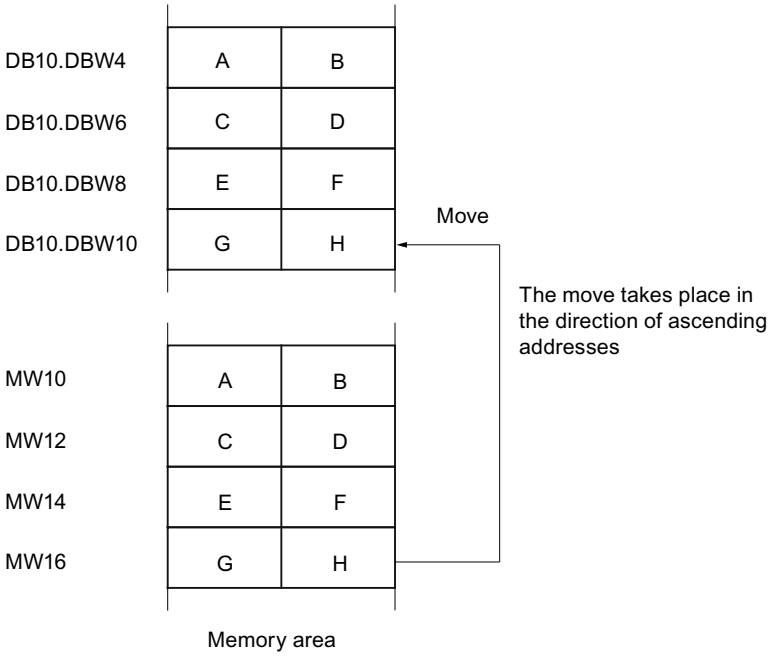
Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

The following figure shows the principle of the move operation:



Consistency of source and destination data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	I, Q, M, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

Parameter	Declaration	Data type	Memory area	Description
DSTBLK	Output 1)	VARIANT	I, Q, M, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

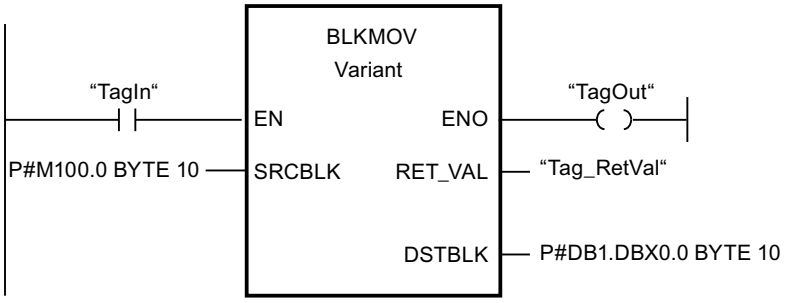
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8152	The WSTRING, WCHAR and BOOL data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR and BOOL data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2417)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	I, Q, M, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1)]	VARIANT	I, Q, M, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

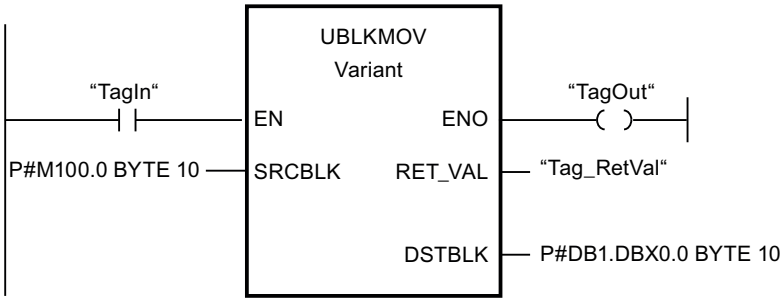
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR and BOOL data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR and BOOL data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2417)

FILL: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

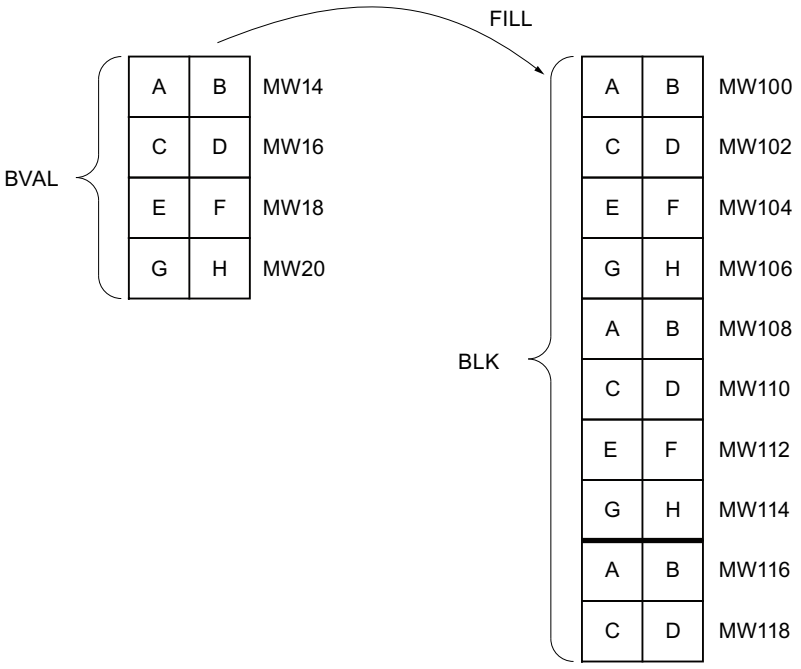
You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK instruction. Fill block" instruction.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the input parameter BVAL, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
BVAL	Input	VARIANT	I, Q, M, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
BLK	Output 1)	VARIANT	I, Q, M, L, P	Specification of the memory area that will be filled with the content of the source area.
1) The BLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

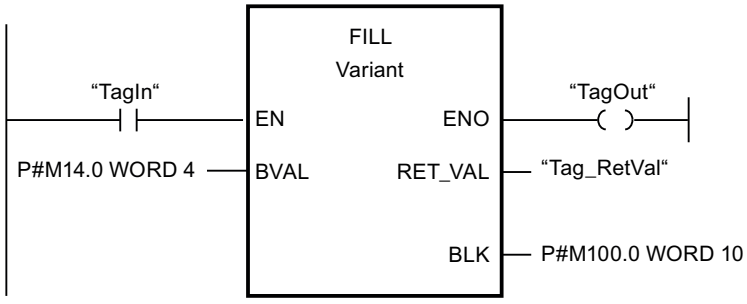
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR and BOOL data types are not supported at the BVAL parameter.
8352	The WSTRING, WCHAR and BOOL data types are not supported at the BLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2417)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types selected in the instruction box. The converted value is provided at output OUT.

For information on possible conversions, refer to the "Explicit conversion" section at "See also".

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Conversion options for bit strings

BYTE and WORD bit strings cannot be selected in the instruction box. However, it is possible to specify an operand of data type DWORD or LWORD at a parameter of the instruction if the lengths of the input and output operands match. The operand is then interpreted by the data type of a bit string according to the data type of the input or output parameter and is implicitly converted. For example, the data type DWORD is interpreted as DINT/UDINT and LWORD is interpreted as LINT/ULINT. These conversion options are also available to you when "IEC check" is activated.

Note

For CPUs of the S7-1500 series: The data types DWORD and LWORD can only be converted to or from data type REAL or LREAL.

Parameters

The following table shows the parameters of the "Convert value" instruction:

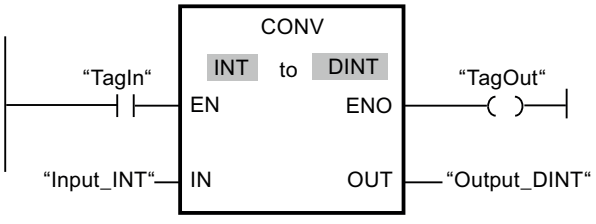
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P, or constant	Value to be converted.
OUT	Output	Bit strings, integers, floating-point numbers, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

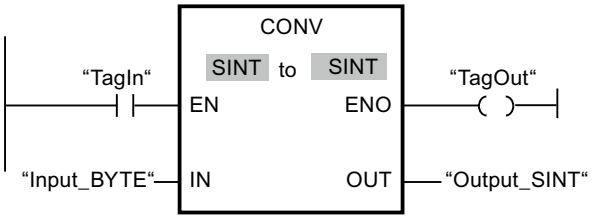
You can find additional information on valid data types under "See also".

Examples

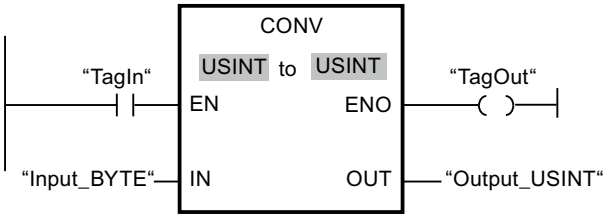
The following example shows the conversion of an integer (16 bit) to another integer (32 bit):



The following example shows the conversion of a byte (8 bit) to the integer SINT (8 bit):



The following example shows the conversion of a byte (8 bit) to the unsigned integer USINT (8 bit):



The conversions are possible because the two operands have the same length.

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to an integer of data type DINT. If the input value is exactly between an even and odd number, the even number is selected. The result of the instruction is sent to the OUT output and can be queried there.

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

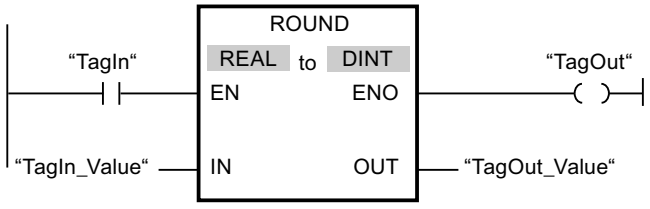
Parameter	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value to be rounded.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result of rounding

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

CEIL: Generate next higher integer from floating-point number

Description

You can use the "Generate next higher integer from floating-point number" instruction to round the value at input IN to the next higher integer. The instruction interprets the value at the IN input as a floating-point number and converts this number to the next higher integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be greater than or equal to the input value.

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Generate next higher integer from floating-point number" instruction:

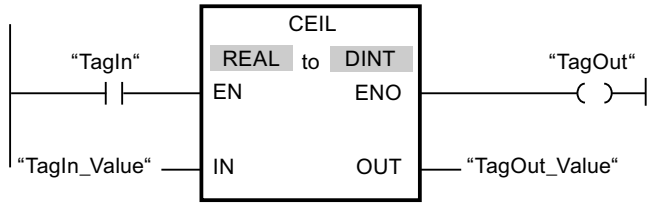
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next higher integer

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at the "TagIn_Value" input is rounded to the next higher integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

FLOOR: Generate next lower integer from floating-point number

Description

You can use the "Generate next lower integer from floating-point number" instruction to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be less than or equal to the input value.

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Generate next lower integer from floating-point number" instruction:

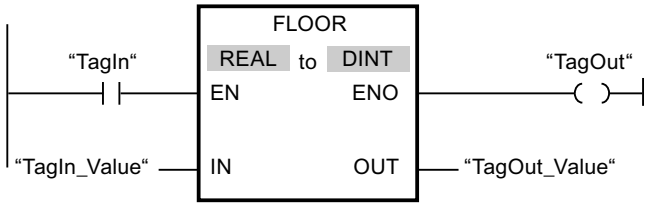
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next lower integer

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at the IN input. The value at the IN input is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to the OUT output without decimal places.

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Truncate numerical value" instruction:

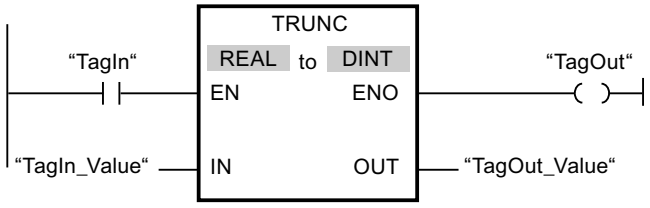
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Integer part of the input value

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The integer part of the floating-point number at the "TagIn_Value" input is converted to an integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

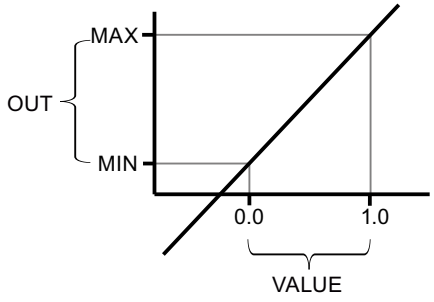
Overview of the valid data types (Page 1908)

SCALE_X: Scale

Description

You can use the "Scale" instruction to scale the value at the VALUE input by mapping it to a specified value range. When the "Scale" instruction is executed, the floating-point value at the VALUE input is scaled to the value range that was defined by the MIN and MAX parameters. The result of the scaling is an integer, which is stored in the OUT output.

The following figure shows an example of how values can be scaled:



The "Scale" instruction works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- The value at the MIN input is greater than or equal to the value at the MAX input.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value at the VALUE input is NaN (Not a Number = result of an invalid arithmetic operation).

Note

For more information on the converting of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range

Parameter	Declaration	Data type	Memory area	Description
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result of scaling

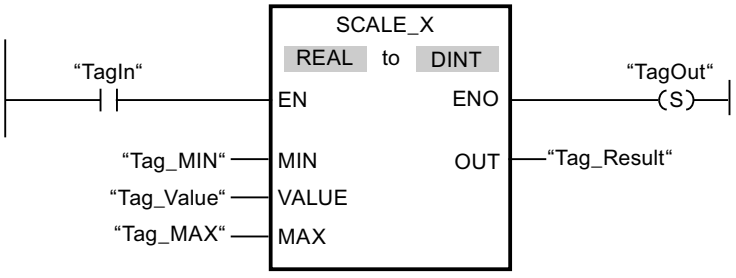
You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

You can find additional information on valid data types under "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5
MAX	Tag_MAX	30
OUT	Tag_Result	20

If the "TagIn" operand has signal state "1", the instruction is executed. The value at the "Tag_Value" input is scaled to the range of values defined by the values at the "Tag_MIN" and "Tag_MAX" inputs. The result is stored in the "Tag_Result" output. If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

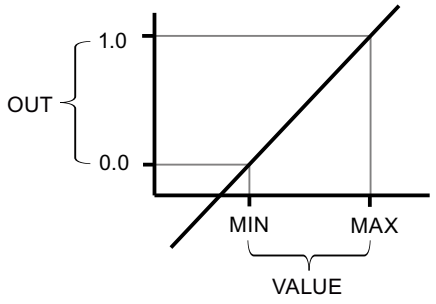
NORM_X: Normalize (Page 2393)

NORM_X: Normalize

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. The result at the OUT output is calculated and stored as a floating-point number depending on the location of the value to be normalized within this value range. If the value to be normalized is equal to the value at the MIN input, the OUT output has the value "0.0". If the value to be normalized is equal to the value at input MAX, output OUT returns the value "1.0".

The following figure shows an example of how values can be normalized:



The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- The value at the MIN input is greater than or equal to the value at the MAX input.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- The value at the VALUE input is NaN (result of an invalid arithmetic operation).

Note

For more information on the converting of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the "Normalize" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MIN [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Value to be normalized.
MAX [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Floating-point numbers	I, Q, M, D, L	Result of the normalization
[1)] If you use constants in these three parameters, you only need to declare one of them.				

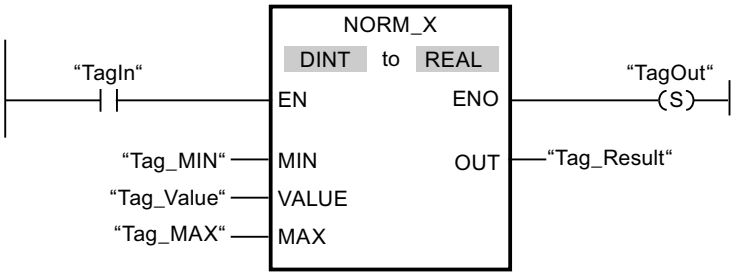
You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

You can find additional information on valid data types under "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	20
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

If the "TagIn" operand has signal state "1", the instruction is executed. The value at the "Tag_Value" input is mapped to the range of values that were defined by the values at the "Tag_MIN" and "Tag_MAX" inputs. The tag value at the "Tag_Value" input is normalized to the defined value range. The result is stored as a floating-point number in the "Tag_Result" output. If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SCALE_X: Scale (Page 2390)

Legacy

SCALE: Scale

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number, which can be scaled in physical units between a low limit value and a high limit value. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output on the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0,0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameter

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit

Parameter	Declaration	Data type	Memory area	Description
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

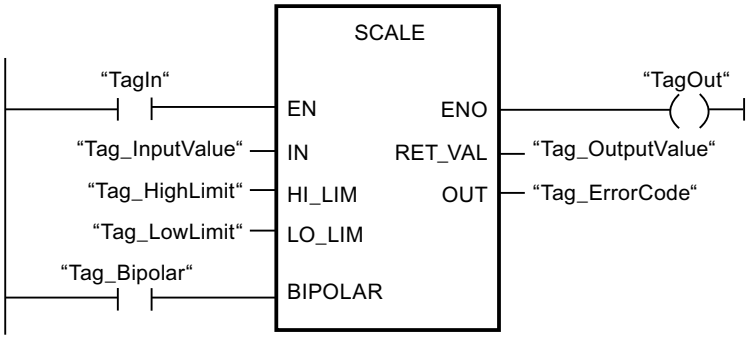
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2417)

UNSCALE: Unscale

Description

The "Unscale" instruction is used to unscale the floating-point number on the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output on the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the constants "K1" and "K2" are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "HI_LIM", the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Input	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

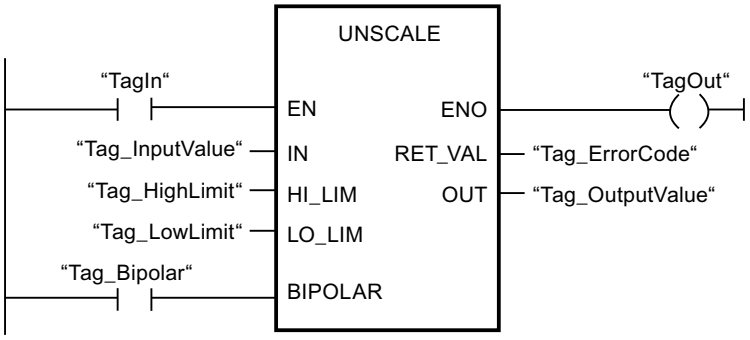
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2417)

Program control operations

---(JMP): Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

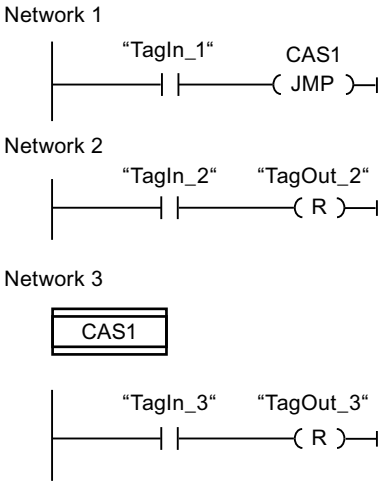
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil is permitted within a network.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

---(JMPN): Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

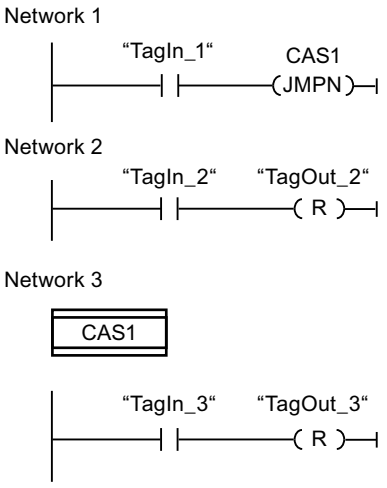
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil is permitted within a network.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of logic operation at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "0", the "Jump if RLO = 0" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

See also

Overview of the valid data types (Page 1908)

LABEL: Jump label

Description

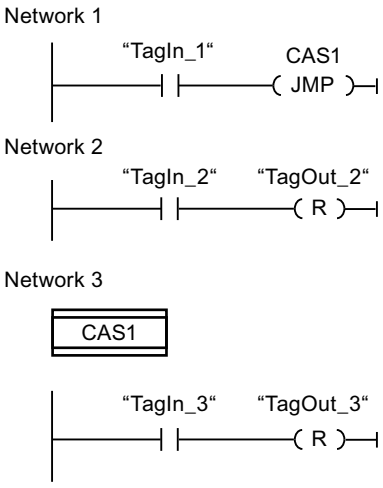
You can use a jump label to identify a destination network, in which the program execution should resume when a jump is executed.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block. You can declare up to 32 jump labels when you use a CPU S7-1200 and a maximum of 256 jump labels when you use a CPU S7-1500.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

See also

Overview of the valid data types (Page 1908)

JMP_LIST: Define jump list

Description

You can use the "Define jump list" instruction to define several conditional jumps and continue the program execution in a specific network depending on the value of the K parameter.

You define the jumps with jump labels (LABEL), which you specify at the outputs of the instruction box. The number of outputs can be expanded in the instruction box. You can declare up to 32 outputs when you use a CPU S7-1200 and a maximum of 99 outputs when you use a CPU S7-1500.

The numbering of the outputs begins with the value "0" and continues in ascending order with each new output. Only jump labels can be specified at the outputs of the instruction. Instructions or operands cannot be specified.

The value of the K parameter specifies the number of the output and thus the jump label where the program execution is to be resumed. If the value in the K parameter is greater than the number of available outputs, the program execution is resumed in the next network of the block.

The "Define jump list" instruction is only executed if the signal state is "1" at the EN enable input.

Parameter

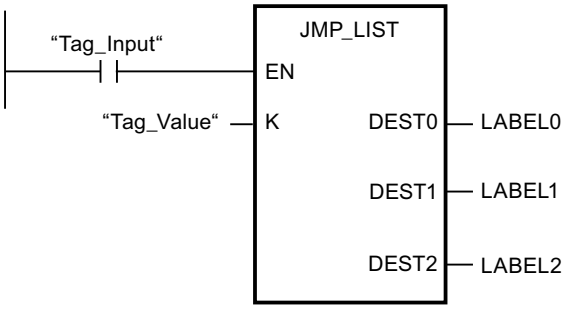
The following table shows the parameters of the "Define jump list" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
K	Input	UINT	I, Q, M, D, L or constant	Specifies the number of the output and thus the jump that is to be made.
DEST0	-	-	-	First jump label
DEST1	-	-	-	Second jump label
DESTn	-	-	-	Optional jump labels

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand/Jump label	Value
K	Tag_Value	1
Dest0	LABEL0	Jump to the network that is identified with the jump label "LABEL0".
Dest1	LABEL1	Jump to the network that is identified with the jump label "LABEL1".
Dest2	LABEL2	Jump to the network that is identified with the jump label "LABEL2".

If operand "Tag_Input" has the signal state "1", the "Define jump list" instruction is executed. The program execution is resumed according to the value of operand "Tag_Value" in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 1908)

SWITCH: Jump distributor

Description

You can use the "Jump distributor" instruction to define multiple program jumps to be executed depending on the result of one or more comparison instructions.

You specify the value to be compared in the K parameter. This value is compared with the values that are provided by the various inputs. You can select the comparison method for each individual input. The availability of the various comparison instructions depends on the data type of the instruction.

The following table shows the comparison instructions that are available depending on the selected data type:

Data type		Instruction	Syntax
S7-1200	S7-1500		
Bit strings	Bit strings	Equal	==
		Not equal	<>
Integers, floating-point numbers, TIME, DATE, TOD	Integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	Equal	==
		Not equal	<>
		Greater or equal	>=
		Less or equal	<=
		Greater than	>
		Less than	<

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. If you select a comparison instruction and the data type of the instruction is not yet defined, the "<???>" drop-down list only offers the data types that are permitted for the selected comparison instruction.

Execution of the instruction begins with the first comparison and runs until a comparison condition is met. If a comparison condition is met, the subsequent comparison conditions are not considered. If none of the specified comparison conditions are met, the jump at the ELSE output is executed. If no program jump is defined at the ELSE output, execution of the program continues in the next network.

The number of outputs can be expanded in the instruction box. The numbering of the outputs begins with the value "0" and continues in ascending order with each new output. Specify jump labels (LABEL) at the outputs of the instruction. Instructions or operands cannot be specified at the outputs of the instruction.

An input is automatically inserted for each additional output. The jump programmed at an output is executed if the comparison condition of the corresponding input is fulfilled.

Parameters

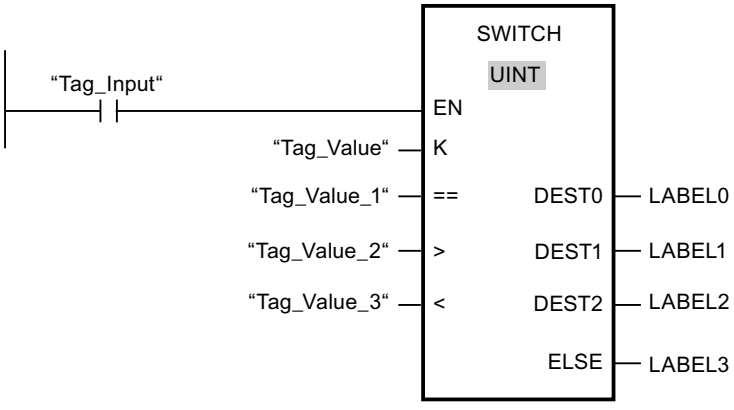
The following table shows the parameters of the "Jump distributor" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
K	Input	UINT	UINT	I, Q, M, D, L or constant	Specifies the value to be compared.
<Comparison values>	Input	Bit strings, integers, floating-point numbers, TIME, DATE, TOD	Bit strings, integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	I, Q, M, D, L or constant	Input value with which the value of the K parameter is compared.
DEST0	-	-	-	-	First jump label
DEST1	-	-	-	-	Second jump label
DEST(n)	-	-	-	-	Optional jump labels(n = 2 to 99)
ELSE	-	-	-	-	Program jump that is executed when none of the comparison conditions are fulfilled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand/Jump label	Value
K	Tag_Value	23
==	Tag_Value_1	20

Parameters	Operand/Jump label	Value
>	Tag_Value_2	21
<	Tag_Value_3	19
Dest 0	LABEL0	Jump to jump label "LABEL0", if the value of the K parameter equals 20.
Dest 1	LABEL1	Jump to jump label "LABEL1" if the value of the K parameter is greater than 21.
Dest 2	LABEL2	Jump to jump label "LABEL2", if the value of the K parameter is less than 19.
ELSE	LABEL 3	Jump to jump label "LABEL3", if the none of the comparison conditions are fulfilled.

If the operand "Tag_Input" changes to signal state "1", the instruction "Jump distributor" is executed. The execution of the program is continued in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 1908)

--(RET): Return

Description

You can use the "Return" instruction to stop the execution of a block. The results is three types through which the block processing can be completed.

- Without call of the "Return" instruction
  The block is exited after execution of the last network. The ENO of the function call is set to the signal state "1".
- Call of the "Return" instruction with preceding logic operation (see example)
  If the left connector has the signal state "1", the block will be exited. The ENO of the function call corresponds to the operand.
- Call of the "Return" instruction without previous logic operation
  The block is exited. The ENO of the function call corresponds to the operand.

Note

Only one jumping coil may be used in a network ("Return", "Jump if RLO=1", "Jump if RLO=0").

If the result of logic operation (RLO) at the input of the "Return" instruction is "1", program execution is terminated in the currently called block and resumed after the call function in the

calling block (for example, in the calling OB). The status (ENO) of the call function is determined by the parameter of the instruction. This can assume the following values:

- RLO
- TRUE/FALSE
- <Operand>

To set the parameter values, double-click the instruction and select the corresponding value in the drop-down list.

The following table shows the status of the call function if the "Return" instruction is programmed in a network within the called block:

RLO	Parameter value	ENO of the call function
1	RLO	1
	TRUE	1
	FALSE	0
	<Operand>	<Operand>
0	RLO	The program execution continues in the next network of the called block.
	TRUE	
	FALSE	
	<Operand>	

If an OB is completed, another block will be selected and started or executed by the priority class system:

- If the program cycle OB was completed, it will be restarted.
- If an OB is completed that has interrupted another block (for example, an alarm OB), then the interrupted block (for example, program cycle OB) will be executed.

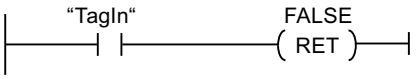
Parameters

The following table shows the parameters of the "Return" instruction:

Parameter	Declaration	Data type	Memory area	Description
Status of the calling function with RLO = 1:				
RLO	-	-		Is set to the signal status of the RLO.
TRUE	-	-		1
FALSE	-	-		0
<Operand>	Input	BOOL	I, Q, M, D, L	Signal state of the specified operand

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Return" instruction is executed. Program execution is terminated in the called block and continues in the calling block. The ENO enable output of the call function is reset to signal state "0".

See also

Overview of the valid data types (Page 1908)

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether configured passwords may be legitimated or not for the CPU. You can prevent legitimated connections, even when the correct password is known.

When you call the command and the REQ parameter has the signal state "0", only the currently set state is displayed at the output parameters. If changes are made to the input parameters, these changes are not transferred to the output parameters.

If the instruction is called and REQ parameter has the signal state "1", the signal state is applied from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD). FALSE means that legitimation per password is not permitted; TRUE means the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited, except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Non-configured passwords must have the TRUE signal state at the input and return the TRUE signal state at the output. The fail-safe password can only be configured for an F-CPU and, therefore, must always be interconnected with theTRUE signal state in a standard CPU. If the instruction return an error, the call remains without effect, i.e. the previous lock remains in force.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".

- Set the operating mode switch to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Inserting an empty memory card (transfer card or program card) into S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access to the HMI system, if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing accidental lockout out with an S7-1500 CPU

The same setting can be made on the front panel of the S7-1500 CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode selector to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing accidental lockout with an S7-1200 CPU

The protection for POWER OFF - POWER ON is disabled because a S7-1200 CPU has no operating mode switch. As a result it is possible and advisable to use certain program sequences within your program to prevent an accidental lockout.

To do this, program a time control using either a cyclic interrupt OB or a timer in the main OB (OB 1). This provides you with the option to call the "Limit and enable password legitimation" instruction again without delay in the corresponding OB (e.g. OB 1 or OB 35) after a transition from POWER OFF - POWER ON. Call the instruction in the startup OB (OB 100) in order to keep as short as possible the time window in which the application is not active and thus ensure that no restrictions exist in the password legitimation. This procedure offers you the largest possible protection against unauthorized access.

If an unintended lockout occurs you can skip the call in the startup OB (for example, by querying an input parameter) and have the set time (for example 10 seconds to 1 minute), to establish a connection to the CPU before the lock becomes active again.

If you have set no timer in your program code and are locked out, insert an empty transfer card or empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then load the user program again from STEP 7 in the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

Warning

Inserting empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

After you remove the transfer card the content of the transfer card is in the internal load memory. Make sure at this point that the card contains no program.

Warning

Inserting empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. The internal load memory is empty after the removal of the formerly empty program card.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200-CPU: the lock is disabled and the instruction must be called again in the program (in the startup OB, for example). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Parameters

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password

Parameter	Declaration	Data type	Memory area	Description
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported
80D0	The password for fail-safe is not configured. With standard CPUs, the signal state must be TRUE.
80D1	The read/write access is not configured
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 1908)

RE_TRIGR: Restart cycle monitoring time

Description

You can use the "Restart cycle monitoring time" instruction to restart the cycle time monitoring of the CPU. The maximum cycle time then starts over again for the duration you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called regardless of the priority in all blocks.

If the instruction is called in a block with a higher priority, such as a hardware interrupt, diagnostic error interrupt, or cyclic interrupt, the instruction is not executed and the ENO enable output is set to signal state "0".

The "Restart cycle monitoring time" instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Parameter

The "Restart cycle monitoring time" instruction has no parameters.

See also

Overview of the valid data types (Page 1908)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

When the result of logic operation (RLO) at the input of the instruction is "1", the CPU changes to STOP mode and program execution is terminated. The signal state at the output of the instruction is not evaluated.

When the RLO is "0" at the input of the instruction, then the instruction will not be executed.

Parameters

The "Exit program" instruction has no parameters.

See also

Overview of the valid data types (Page 1908)

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports an error during the processing of the block, detailed information is stored in the operand at the ERROR output for the first error to occur during the execution of the block since the last execution of the instruction.

Only operands of the "ErrorStruct" system data type can be specified at the ERROR output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the block, the error information about the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The ERROR output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO of the instruction "Get error locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error locally" instruction enables local error handling within a block. If "Get error locally" is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Parameters

The following table lists the parameters of the "Get error locally" instruction:

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	ErrorStruct	D, L	Error information

Data type "ErrorStruct"

The following table lists the structure of the "ErrorStruct" data type:

Structure component		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows if an error occurred during a block call. 16#01: Error during a block call 16#00: No error during a block call
REACTION		BYTE	Default reaction: 0: Ignore (write error) 1: Continue with substitute value "0" (read error) 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of block
	BLOCK_TYPE	BYTE	Type of block where the error occurred: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B Range violations for a directly editable tag of data type DINT: 16#04
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

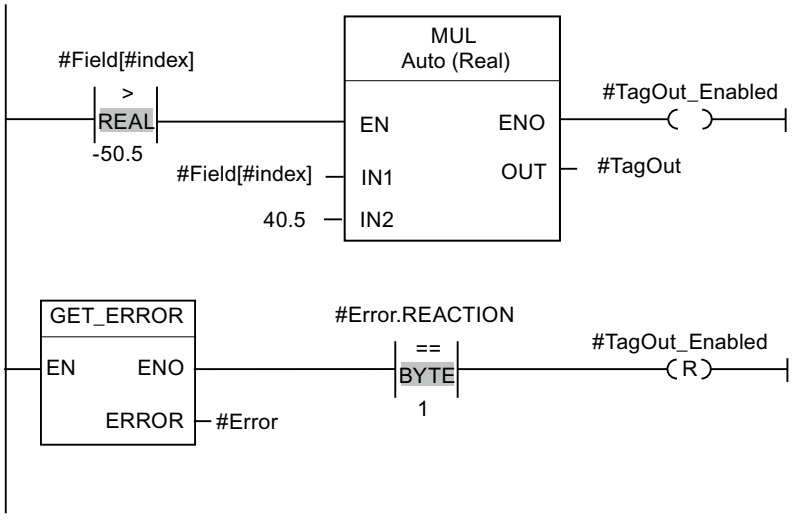
Structure component "ERROR_ID"

The following table lists the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2577	9591	The block property "Parameter passing via registers" is not selected.
2576	9590	Error in the local data distribution
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:



An error occurred accessing the "#Field[#index]" tag. The enable output ENO of the "Multiply" instruction and the "#TagOut_Enabled" operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error locally" instruction is evaluated with the comparison instruction "Equal". If the "#Error.REACTION" structure component has the value "1", this involves a read/access error and the "#TagOut_Enabled" output is reset.

See also

Overview of the valid data types (Page 1908)

Querying and fixing errors in the program code (Page 228)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the error ID of the first error that occurred in the tag is stored at the ID output.

Only operands of the WORD data type can be specified at the ID output. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The ID output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO of the instruction "Get error ID locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Parameters

The following table lists the parameters of the "Get error ID locally" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Output	WORD	I, Q, M, D, L	Error ID

ID parameter

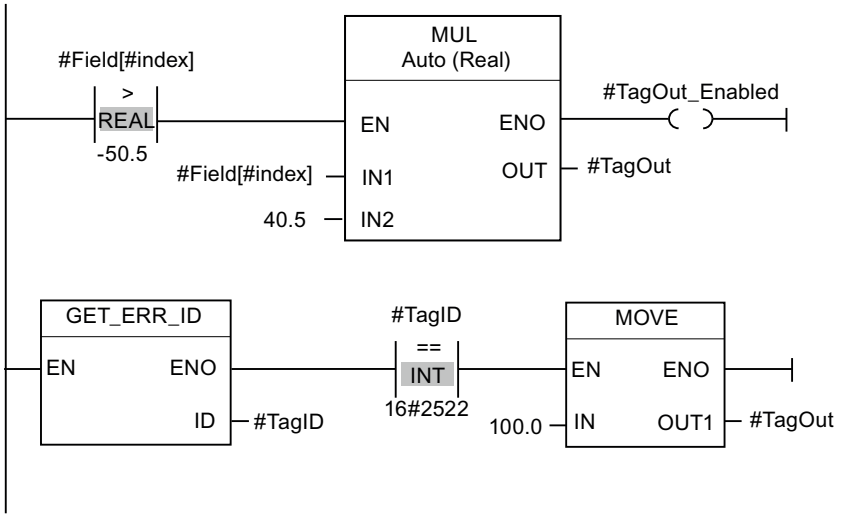
The following table lists the values that can be output at the ID parameter:

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment

ID* (hexadecimal)	ID* (decimal)	Description
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:



An error occurred accessing the "#Field[#index]" tag. The enable output ENO of the "Multiply" instruction and the "#TagOut_Enabled" operand have signal state "1" despite the read/access

error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error ID locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error ID locally" instruction is evaluated with the comparison instruction "Equal". If the operand "#TagID" returns the ID 2522, this involves a read/access error and the value "100.0" is written to the "#TagOut" output.

See also

Overview of the valid data types (Page 1908)

Querying and fixing errors in the program code (Page 228)

INIT_RD: Initialize all retain data

Description

You can use the "Initialize all retain data" instruction to reset the retain data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution would exceed the program cycle duration.

Parameter

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	If the input "REQ" has the signal state "1", all retain data are reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

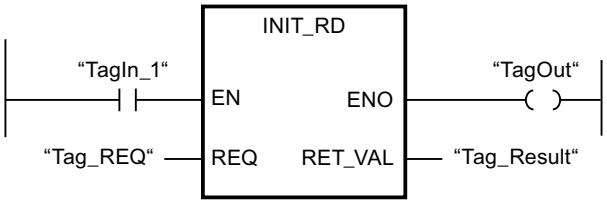
Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "Tag_REQ" have signal state "1", the instruction is executed. The retain data of all data blocks, bit memories and SIMATIC timers and counters are reset If the instruction is executed without errors, the ENO enable output has the signal state "1".

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2417)

WAIT: Configure time delay

Description

The "Configure time delay" instruction pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to +32767 microseconds (µs). The smallest possible delay time depends on the respective CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction returns no error information.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
WT	Input	INT	I, Q, M, D, L, P, or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 1908)

RUNTIME: Measure program runtime

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and write the value to the IN-OUT parameter MEM. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the IN-OUT parameter MEM as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The following applies to S7-1200 CPUs with firmware version earlier than V4.1: The Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overflows (this can occur up to once per minute), the instruction returns values <= 0.0. Ignore these runtime values.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Parameter

The following table shows the parameters of the "Measure program runtime" instruction:

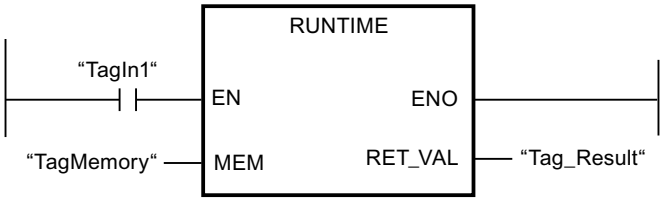
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MEM	InOut	LREAL	I, Q, M, D, L	Saves the starting point of the runtime measurement.
RET_VAL	Output	LREAL	I, Q, M, D, L	Returns the measured runtime in seconds

For additional information on valid data types, refer to "See also".

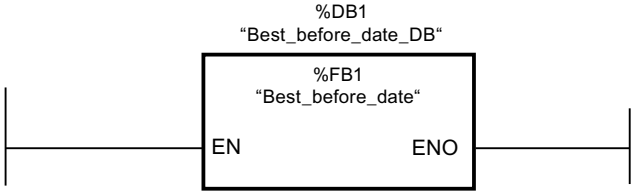
Example

The following example shows how the instruction works based on the runtime calculation of a program block:

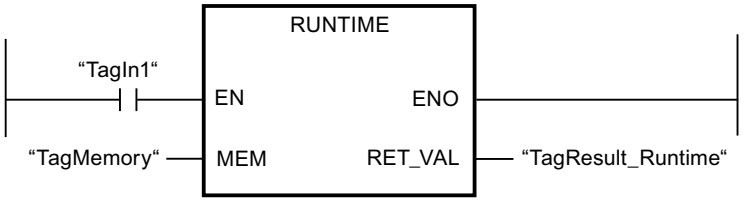
Network 1:



Network 2:



Network 3:



When the "TagIn1" operand in network 1 has the signal state "1", the instruction is executed. The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called in network 2.

When the FB1 program block has been executed and the "TagIn1" operand has the signal state "1", the instruction in network 3 is executed. The second call of the instruction calculates the runtime of the program block and writes the result to the output RET_VAL.

See also

Overview of the valid data types (Page 1908)

Word logic operations

AND: AND logic operation

Description

You can use the "AND logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by AND logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are logically ANDed. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with AND logic (ANDed). The result is stored in the OUT output.

The result bit has signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "AND logic operation" instruction:

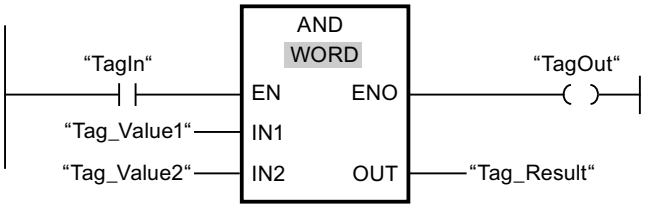
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P, or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P, or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P, or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0000 0000 0000 0101

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ANDed. The result is mapped bit-for-bit and output in operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

OR: OR logic operation

Description

You can use the "OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are combined by OR logic operation. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with OR logic (ORed). The result is stored in the OUT output.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "OR logic operation" instruction:

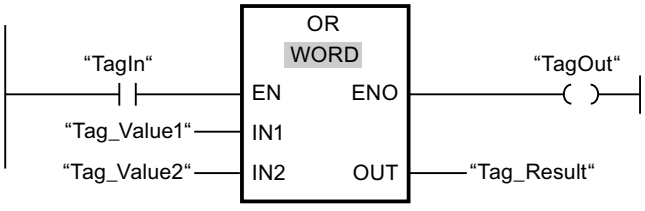
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P, or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P, or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P, or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1111

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ORed. The result is mapped bit-for-bit and output in operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

XOR: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by EXCLUSIVE OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are combined by EXCLUSIVE OR logic operation. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with EXCLUSIVE OR logic. The result is stored in the OUT output.

The result bit has signal state "1" when one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation" instruction:

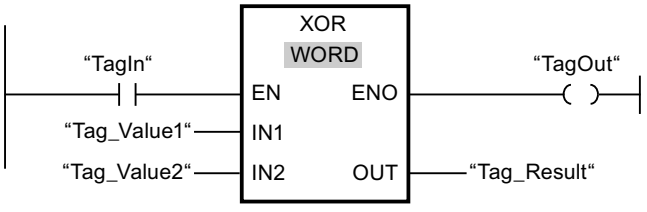
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P, or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P, or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P, or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1010

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are combined by EXCLUSIVE OR logic. The result is mapped bit-for-bit and output in operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

INVERT: Create ones complement

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits at input IN. When the instruction is processed, the value at input IN is linked to EXCLUSIVE OR by a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit number). This inverts the signal state of the individual bits that are then stored at output OUT.

Parameters

The following table shows the parameters of the instruction "Create ones complement":

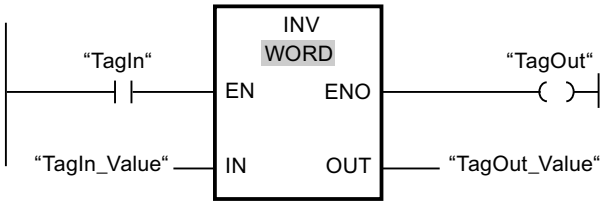
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Bit strings, integers	I, Q, M, D, L, P	Ones complement of the value at input IN

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	W#16#000F	W#16#7E
OUT	TagOut_Value	W#16#FFF0	W#16#81

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction inverts the signal state of the individual bits at input "TagIn_Value" and writes the result to output "TagOut_Value". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value that is specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. When the value at the IN input is greater than 31, a modulo 32 instruction is executed.

Parameters

The following table shows the parameters of the "Decode" instruction:

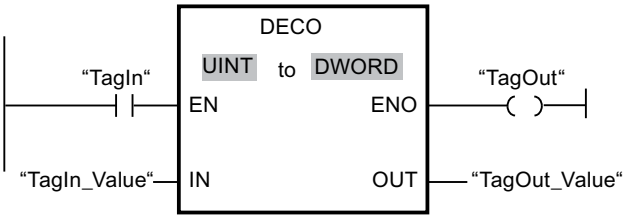
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	UINT	I, Q, M, D, L, P, or constant	Position of the bit in the output value which is set.
OUT	Output	Bit strings	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

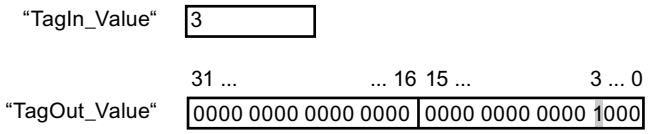
You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction reads bit number "3" from the value of the operand "TagIn_Value" at the input and sets the third bit to the value of the operand "TagOut_Value" at the output.

If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ENCO: Encode

Description

You can use the "Encode" instruction to read the bit number of the least significant bit in the input value and to send it to the OUT output.

The "Encode" instruction selects the least significant bit of the value at the IN input and writes its bit number to the tag in the OUT output.

Parameters

The following table shows the parameters of the "Encode" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

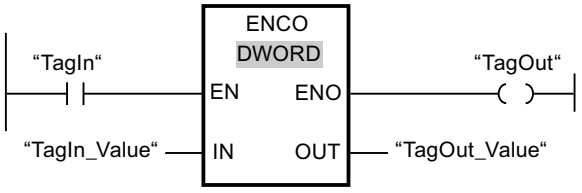
Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P, or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

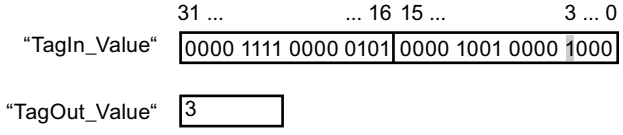
You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction selects the least significant bit at the "TagIn_Value" input and writes bit position "3" to the tag in the "TagOut_Value" output.

If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SEL: Select

Description

Depending on a switch (G input), the "Select" instruction selects one of the inputs, IN0 or IN1 and copies its content to the OUT output. When the input G has signal state "0", the value at the input IN0 is moved. When the input G has signal state "1", the value at the input IN1 is moved to the output OUT.

All tags at all parameters must have the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

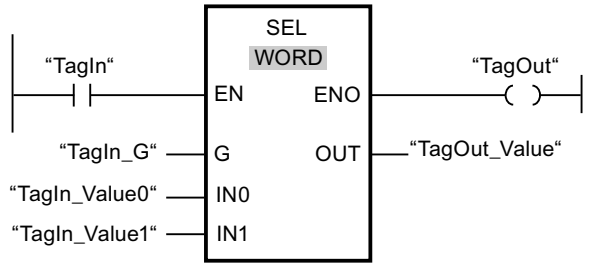
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
G	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, TOD, CHAR, WCHAR, DATE	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, LDT, CHAR, WCHAR, DATE	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, TOD, CHAR, WCHAR, DATE	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, LDT, CHAR, WCHAR, DATE	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, TOD, CHAR, WCHAR, DATE	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, LDT, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If the "TagIn" operand has signal state "1", the instruction is executed. Based on the signal state at the "TagIn_G" input, the value at the "TagIn_Value0" or "TagIn_Value1" input is selected and moved to the "TagOut_Value" output. If the instruction is executed without errors, the enable output ENO has signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 1908)

MUX: Multiplex

Description

You can use the instruction "Multiplex" to copy the content of a selected input to output OUT. The number of selectable inputs of the instruction box can be expanded. You can declare a maximum of 32 inputs.

The inputs are numbered automatically in the box. Numbering starts at IN0 and is incremented continuously with each new input. You can use the K parameter to define the input whose content is to be copied to the OUT output. If the value of the parameter K is greater than the number of available inputs, the content of the ELSE parameter is copied to the OUT output and the ENO enable output is assigned the signal state "0".

The "Multiplex" instruction can only be executed if the tags have the same data type in all inputs and in the OUT output. The K parameter is an exception, since only integers can be specified for it.

The enable output ENO is reset if one of the following conditions applies:

- Enable input EN has signal state "0".
- The input at the K parameter is located outside the available inputs. This applies regardless of whether the ELSE input is used. The value at the OUT output remains changed.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the "Multiplex" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
K	Input	Integers	Integers	I, Q, M, D, L, P, or constant	Specifies the input whose content is to be copied. • If K = 0 => Parameter IN0 • If K = 1 => Parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	Second input value

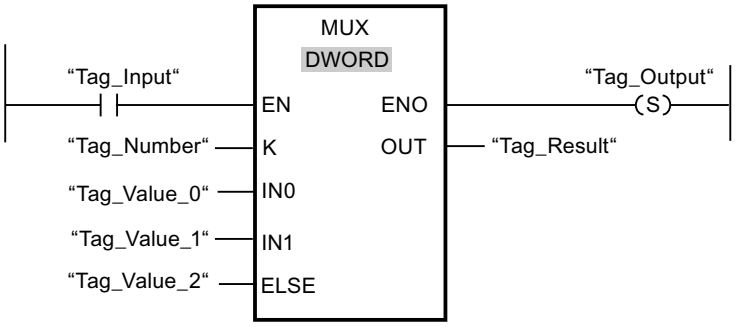
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
INn	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	Optional input values
ELSE	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	Specifies the value to be copied when K > n.
OUT	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Output to which the value is to be copied.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
K	Tag_Number	1
IN0	Tag_Value_0	DW#16#00000000
IN1	Tag_Value_1	DW#16#003E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#003E4A7D

If the "Tag_Input" operand has signal state "1", the instruction is executed. Depending on the value of the operand "Tag_Number", the value at input "Tag_Value_1" is copied and assigned to the operand at output "Tag_Result". If the instruction is executed without errors, the "ENO" and "Tag_Output" enable outputs are set.

See also

Overview of the valid data types (Page 1908)

DEMUX: Demultiplex

Description

You can use the instruction "Demultiplex" to copy the content of the input IN to a selected output. The number of selectable outputs can be extended in the instruction box. The outputs are numbered automatically in the box. Numbering starts at OUT0 and continues consecutively with each new output. You can use the K parameter to define the output to which the content of the IN input is to be copied. The other outputs are not changed. If the value of the parameter K is greater than the number of available outputs, then the content of input IN in the parameter ELSE and the enable output ENO will be assigned to the signal state "0".

The instruction "Demultiplex" can only be executed if the tags at the input IN and at all outputs are of the same data type. The K parameter is an exception, since only integers can be specified for it.

The enable output ENO is reset if one of the following conditions applies:

- Enable input EN has signal state "0".
- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the instruction "Demultiplex":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
K	Input	Integers	Integers	I, Q, M, D, L, P, or constant	Specifies the output to which the input value (IN) is copied. • If K = 0 => Parameter OUT0 • If K = 1 => Parameter OUT1, etc.
IN	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	Input value
OUT0	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	First output

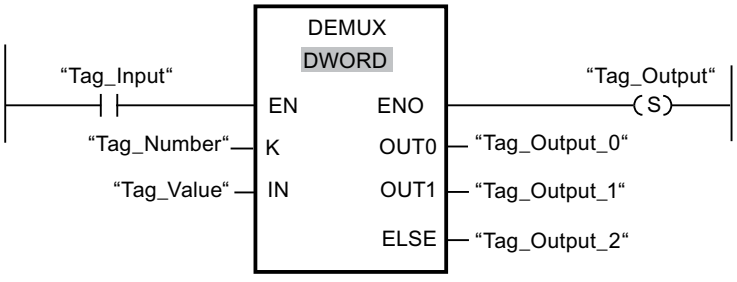
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
OUT1	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Second output
OUTn	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Optional outputs
ELSE	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Output to which the input value (IN) at K > n is copied.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:



The following tables show how the instruction works using specific operand values:

Table 11-24 Input values of the "Demultiplex" instruction before network execution

Parameter	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

Table 11-25 Output values of the "Demultiplex" instruction after network execution

Parameter	Operand	Values	
OUT0	Tag_Output_0	Unchanged	Unchanged
OUT1	Tag_Output_1	DW#16#FFFFFFFF	Unchanged
ELSE	Tag_Output_2	Unchanged	DW#16#003E4A7D

If the "Tag_Input" has signal state "1", the instruction is executed. Depending on the value of the operand "Tag_Number", the value at input "IN" is copied to the corresponding output.

See also

Overview of the valid data types (Page 1908)

Shift and rotate

SHR: Shift right

Description

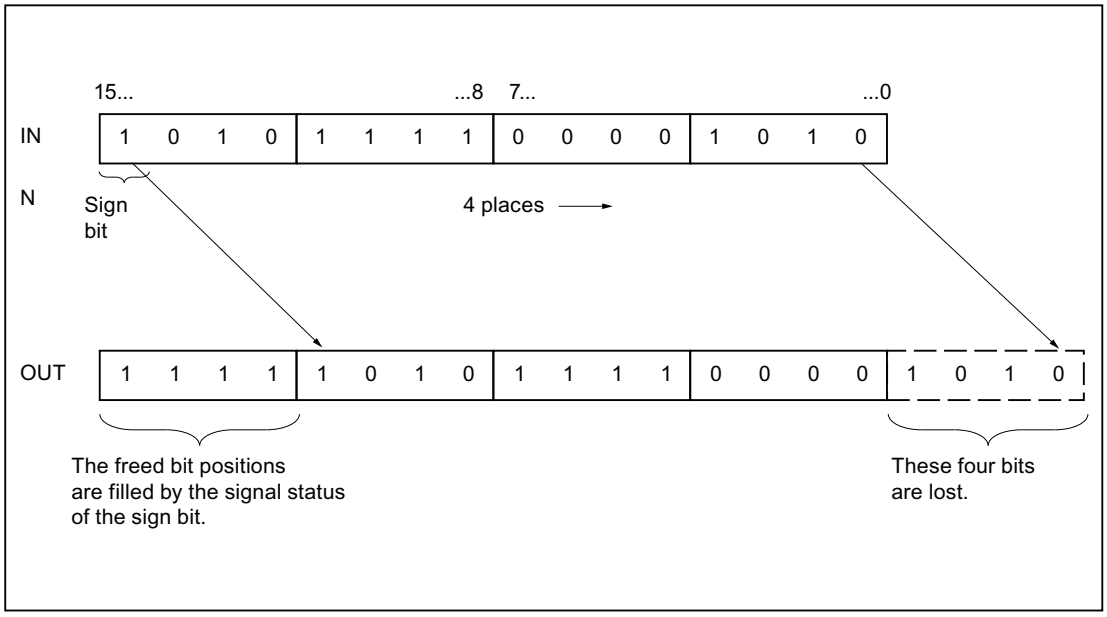
You can use the "Shift right" instruction to shift the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is shifted by the available number of bit positions to the right.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure show how the content of an operand of integer data type is shifted four bit positions to the right:



Parameters

The following table shows the parameters of the instruction "Shift right":

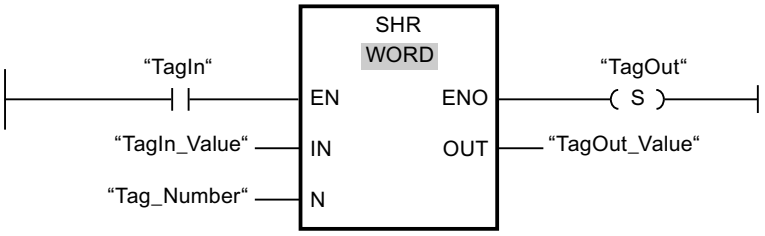
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If operand "TagIn" has the signal state "1", the "Shift right" instruction is executed. The content of operand "TagIn_Value" is shifted three bit positions to the right. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SHL: Shift left

Description

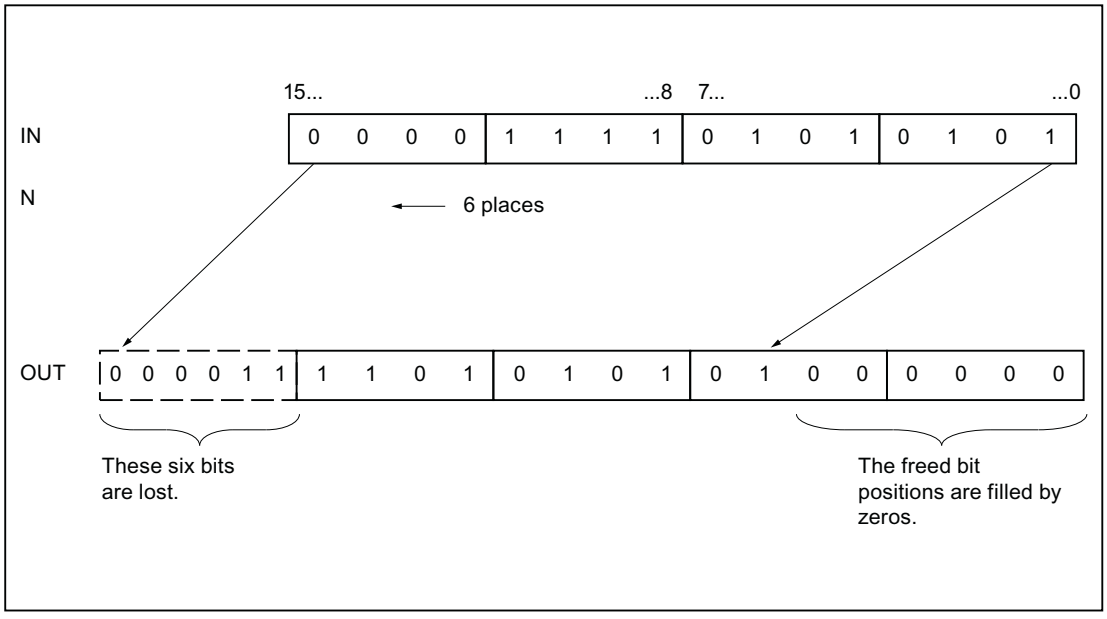
You can use the "Shift left" instruction to shift the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is shifted by the available number of bit positions to the left.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure shows how the content of an operand of the WORD data type is shifted by six bit positions to the left:



Parameters

The following table shows the parameters of the instruction "Shift left":

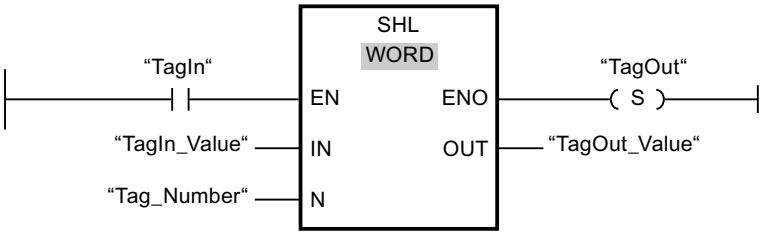
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If operand "TagIn" has the signal state "1", the "Shift left" instruction is executed. The content of operand "TagIn_Value" is shifted four bit positions to the left. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ROR: Rotate right

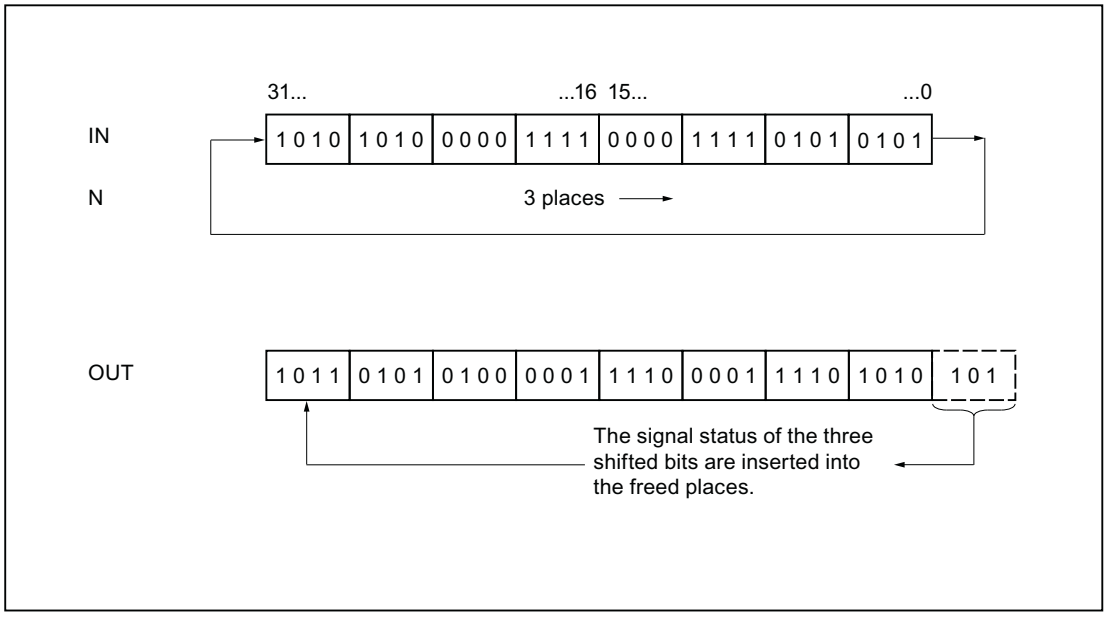
Description

You can use the "Rotate right" instruction to rotate the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

If the value at the parameter N is greater than the number of available bit positions, the operand value at input IN is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of DWORD data type is rotated three positions to the right:



Parameters

The following table shows the parameters of the instruction "Rotate right":

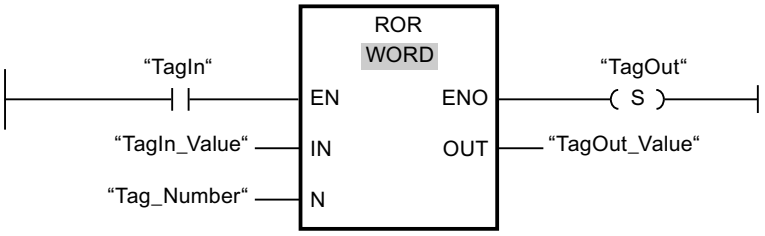
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

If operand "TagIn" has the signal state "1", the "Rotate right" instruction is executed. The content of operand "TagIn_Value" is rotated five bit positions to the right. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ROL: Rotate left

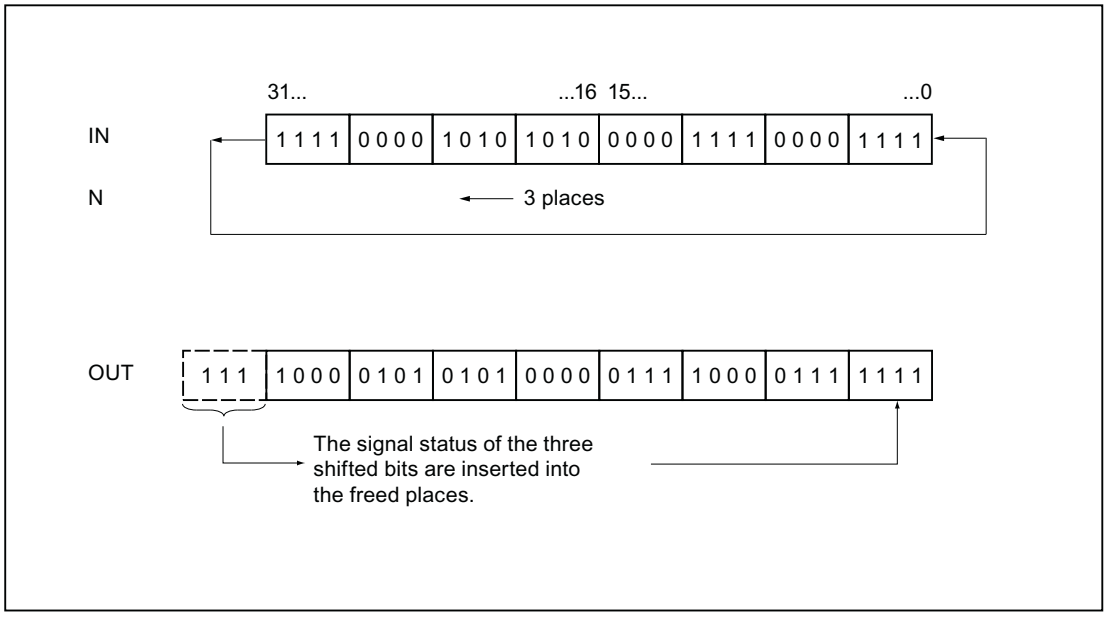
Description

You can use the "Rotate left" instruction to rotate the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

If the value at the parameter N is greater than the number of available bit positions, the operand value at input IN is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of DWORD data type is rotated three positions to the left:



Parameters

The following table shows the parameters of the instruction "Rotate left":

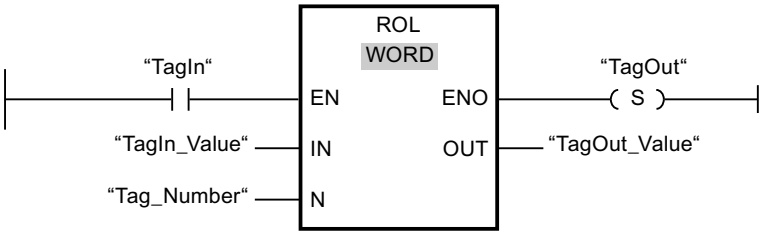
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	1010 1000 1111 0110
N	Tag_Number	5
OUT	TagOut_Value	0001 1110 1101 0101

If input "TagIn" has the signal state "1", the instruction "Rotate left" is executed. The content of operand "TagIn_Value" is rotated five bit positions to the left. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Legacy

DRUM: Implement sequencer

Description

You can use the "Implement sequencer" instruction to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is

"1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK), you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask has signal state "1", the OUT_VAL value sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit at the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit at the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates the reset condition.
JOG	Input	BOOL	I, Q, M, D, L or constant	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" causes the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L or constant	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.

Parameter	Declaration	Data type	Memory area	Description
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P, or constant	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P, or constant	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P, or constant	Current numerical value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P, or constant	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L or constant	Preset count for each step [1 to 16]; where 1 clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

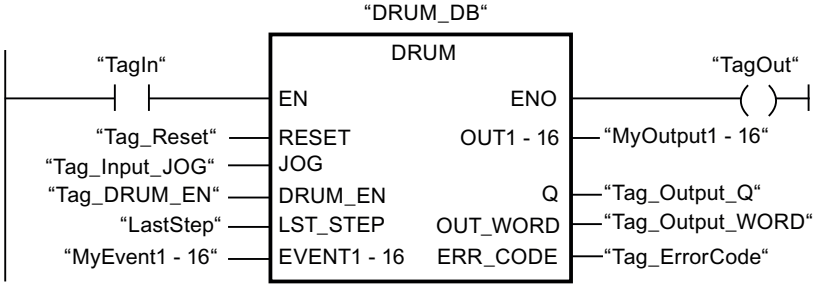
ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameter	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A

Parameter	Address	Value
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE

Parameter	Address	Value
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE

Parameter	Operand	Address	Value
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 1908)

DCAT: Discrete control-timer alarm

Description

The instruction "Discrete control-timer alarm" is used to accumulate the time from the point at which the CMD parameter issued the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET < PT), OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state of the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".
- When the signal state of the parameters CMD and CMD_HIS is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state on the parameter OA is set to "1". If the value of the ET parameter does not exceed the value of the parameter PT, the signal state on the parameter OA is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the CMD, CMD_HIS and O_FB parameters are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the parameter PT, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction returns no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CMD	Input	BOOL	I, Q, M, D, L or constant	The signal state "0" indicates a "close" command. The signal state "1" indicates the "open" command.

Parameter	Declaration	Data type	Memory area	Description
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
CMD_HIS	Static	BOOL	D, L or constant	CMD history bit

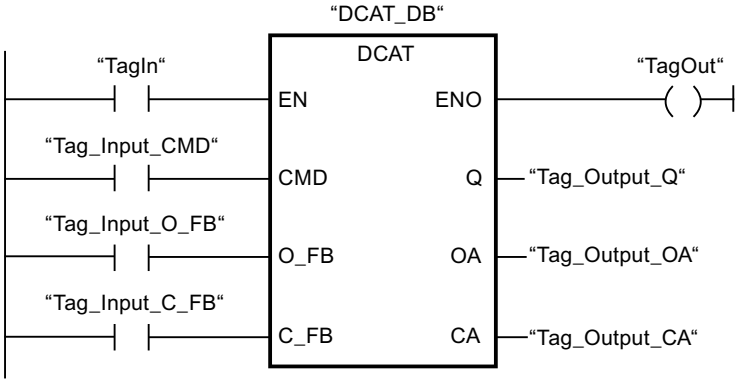
For additional information on valid data types, refer to "See also".

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 1908)

MCAT: Motor control-timer alarm

Description

The instruction "Motor control-timer alarm" is used to accumulate the time from the point in time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single

instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

The "Motor control-timer alarm" instruction returns no error information.

Parameters

The following table shows the reactions of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
O_CMD	Input	BOOL	I, Q, M, D, L or constant	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L or constant	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L or constant	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition.
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
O_HIS	Static	BOOL	D, L or constant	"Open" history bit
C_HIS	Static	BOOL	D, L or constant	"Close" history bit

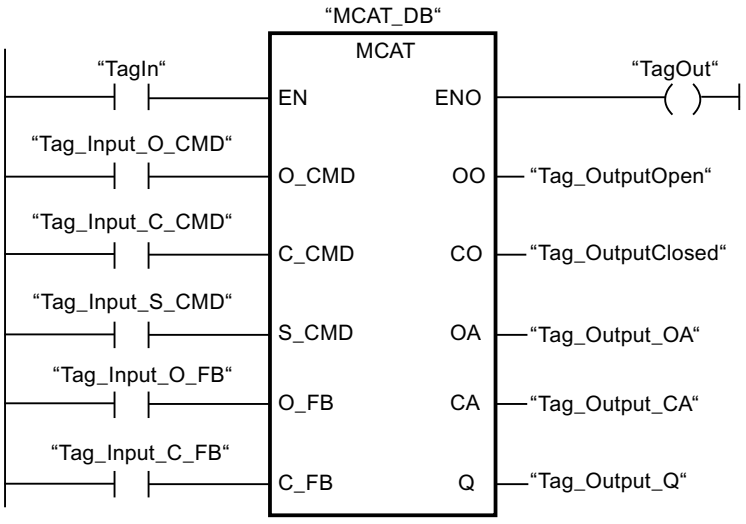
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE

Parameter	Operand	Value
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 1908)

IMC: Compare input bits with the bits of a mask

Description

The instruction "Compare input bits with the bits of a mask" is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. On the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise, the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that a match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 1908)

SMC: Compare scan matrix

Description

The "Compare scan matrix" instruction is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value of the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.

Parameter	Declaration	Data type	Memory area	Description
LAST	Static	BYTE	I, Q, M, D, L, P or constant	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 1908)

LEAD_LAG: Lead and lag algorithm

Description

The "Lead and lag algorithm" instruction is used to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the instruction "Lead and lag algorithm" is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The

"Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two operations together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed. Constants can also be specified on the IN parameter.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time Constants can also be specified on the SAMPLE_T parameter.
OUT	Output	REAL	I, Q, M, D, L	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in in the same unit as sample time.
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).

Parameter	Declaration	Data type	Memory area	Description
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

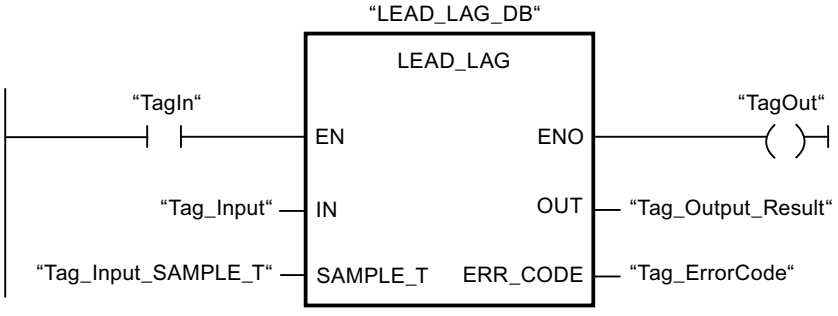
Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

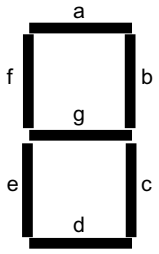
Overview of the valid data types (Page 1908)

SEG: Create bit pattern for seven-segment display

Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

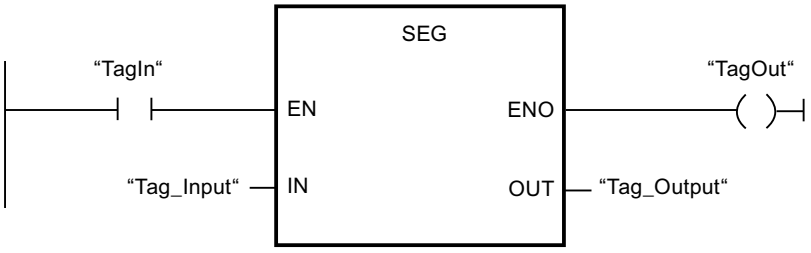
Parameter

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 Display: 1234

See also

Overview of the valid data types (Page 1908)

BCDCPL: Create tens complement

Description

You can use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

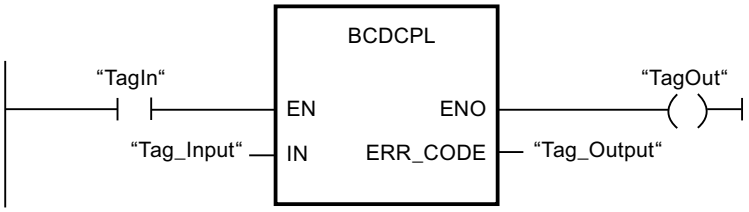
The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 1908)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output on the RET_VAL parameter.

Parameters

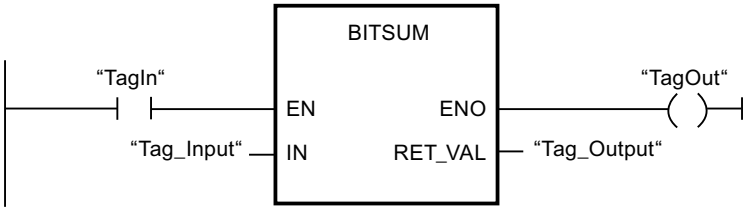
The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of bits to be set

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 1908)

11.6.2.2 FBD

Bit logic operations

&: AND logic operation

Description

You can use the instruction "AND logic operation" to query the signal states of two or more specified operands and evaluate them according to the AND truth table.

If the signal state of all the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of one of the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

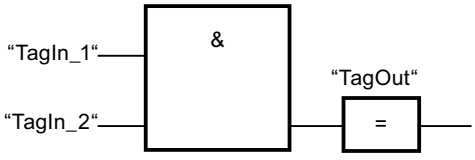
Parameters

The following table shows the parameters of the instruction "AND logic operation":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set, when the signal state of the operands "TagIn_1" and "TagIn_2" is "1" and reset when the state of the operands "TagIn_1" and "TagIn_2" is "0".

See also

AND truth table (Page 2472)

Example of detecting the direction of a conveyor belt (Page 3872)

Example of controlling room temperature (Page 3879)

Overview of the valid data types (Page 1908)

Insert input (Page 2476)

Example of detecting the fill level of a storage area (Page 3874)

AND truth table

Results of the logic operation

The following table shows the results that arise from the AND logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	1	1
0	1	0
1	0	0
0	0	0

See also

&: AND logic operation (Page 2471)

Overview of the valid data types (Page 1908)

>=1: OR logic operation

Description

You can use the instruction "OR logic operation" to query the signal states of two or more specified operands and evaluate them according to the OR truth table.

If the signal state of one of the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of all the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

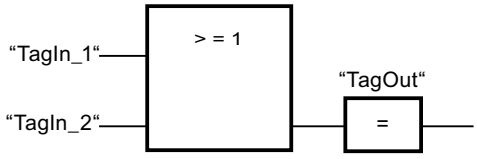
Parameters

The following table shows the parameters of the instruction "OR logic operation":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set, when the signal state of the operands "TagIn_1" or "TagIn_2" is "1".

See also

OR truth table (Page 2474)

Example of controlling a conveyor belt (Page 3871)

Overview of the valid data types (Page 1908)

Insert input (Page 2476)

OR truth table

Results of the logic operation

The following table shows the results that arise from the OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	1
0	0	0

See also

>=1: OR logic operation (Page 2473)

Overview of the valid data types (Page 1908)

X: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" instruction to query the result of a signal state query according to the EXCLUSIVE OR truth table.

With an "EXCLUSIVE OR logic operation" instruction, the signal state is "1" when the signal state of one of the two specified operands is "1". When more than two operands are queried, the common RLO is "1" if an odd number of the queried operands returns the result "1".

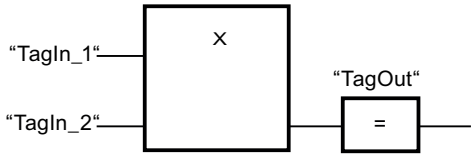
Parameters

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the signal state of the operands "TagIn_1" and "TagIn_2" is "1". When both operands return the signal state "1" or "0", the output "TagOut" is reset.

See also

EXCLUSIVE OR truth table (Page 2475)

Overview of the valid data types (Page 1908)

Insert input (Page 2476)

EXCLUSIVE OR truth table

Results of the logic operation

The following table shows the results that arise from the EXCLUSIVE OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	0
0	0	0

The following table shows the results that arise from the EXCLUSIVE OR logic operation of three operands:

Signal state of the first operand	Signal state of the second operand	Signal state of the third operand	Result of the logic operation
1	0	0	1
0	1	1	0
0	1	0	1
1	0	1	0
0	0	1	1
1	1	0	0
1	1	1	1
0	0	0	0

See also

X: EXCLUSIVE OR logic operation (Page 2474)

Overview of the valid data types (Page 1908)

Insert input

Description

The "Insert input" instruction is used to add an input to the box of one of the following instructions:

- "AND logic operation"
- "OR logic operation"
- "EXCLUSIVE OR logic operation"

You can query the signal state of several operands by the extension of an instruction box.

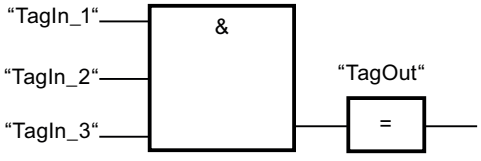
Parameters

The following table shows the parameters of the "Insert input" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



The box of the "AND logic operation" instruction was extended by an additional input at which the signal state of the operand "TagIn_3" is queried. The "TagOut" output is set, when the operands "TagIn_1" and "TagIn_2" and "TagIn_3" return signal state "1".

See also

&: AND logic operation (Page 2471)

>=1: OR logic operation (Page 2473)

X: EXCLUSIVE OR logic operation (Page 2474)

Overview of the valid data types (Page 1908)

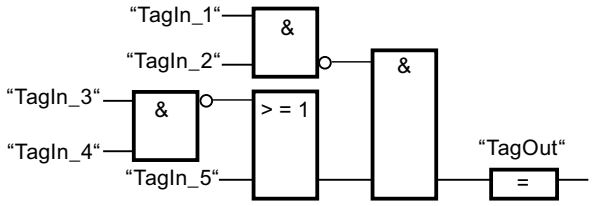
Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO).

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The input "TagIn_1" and/or "TagIn_2" has signal state "0".
- The input "TagIn_3" and/or "TagIn_4" has signal state "0" or the input "TagIn_5" has signal state "1".

See also

Overview of the valid data types (Page 1908)

Example of detecting the direction of a conveyor belt (Page 3872)

Example of detecting the fill level of a storage area (Page 3874)

Example of controlling room temperature (Page 3879)

=: Assignment

Description

You can use the instruction "Assignment" to set the bit of a specified operand. If the result of logic operation (RLO) at the box input has the signal state "1", the specified operand is set to signal state "1". If the signal state at the box input is "0", the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the box input is assigned directly to the operand above the assignment box.

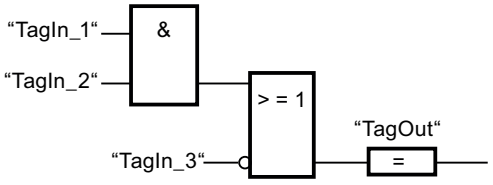
Parameters

The following table shows the parameters of the instruction "Assignment":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



The operand "TagOut" is set at the output of the "Assignment" instruction when one of the following conditions is fulfilled:

- The inputs "TagIn_1" and "TagIn_2" have the signal state "1".
- The signal state at the input "TagIn_3" is "0".

See also

Overview of the valid data types (Page 1908)

Example of detecting the fill level of a storage area (Page 3874)

Example of controlling room temperature (Page 3879)

/=: Negate assignment

Description

The instruction "Negate assignment" inverts the result of logic operation (RLO) and assigns this to the operand above the box. If the RLO at the input of the box is "1", the binary operand is reset. If the RLO at the input of the box is "0", the operand is set to signal state "1".

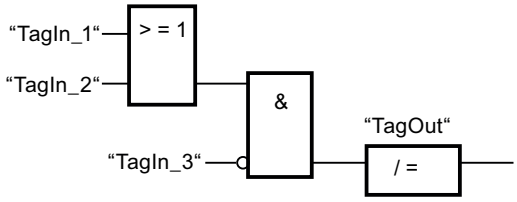
Parameters

The following table shows the parameters of the instruction "Negate assignment":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the negated RLO is assigned.

Example

The following example shows how the instruction works:



The operand "TagOut" is reset when the following conditions are fulfilled:

- The operand "TagIn_1" or "TagIn_2" has the signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 1908)

R: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is reset to "0". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

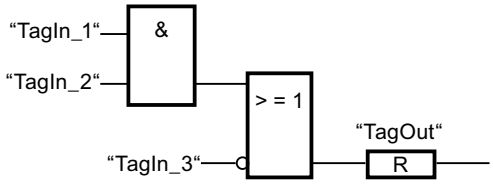
Parameters

The following table shows the parameters of the "Reset output" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Operand that is reset if RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Example of controlling a conveyor belt (Page 3871)

Example of detecting the direction of a conveyor belt (Page 3872)

Overview of the valid data types (Page 1908)

S: Set output

Description

You can use the instruction "Set output" to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is set to "1". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

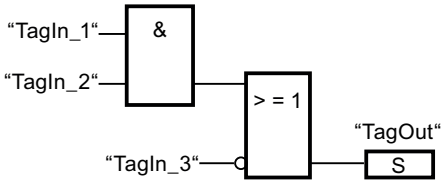
Parameters

The following table shows the parameters of the instruction "Set output":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 1908)

Example of detecting the direction of a conveyor belt (Page 3872)

SET_BF: Set bit field

Description

You can use the instruction "Set bit field" to set multiple bits starting from a certain address.

You can use the input N to define the number of bits to be set. The address of the first bit to be set is defined by (<Operand>). If the value of the N input is greater than the number of bits in a selected byte, the bits of the next byte are set. The bits remain set until they are explicitly reset, for example, by another instruction.

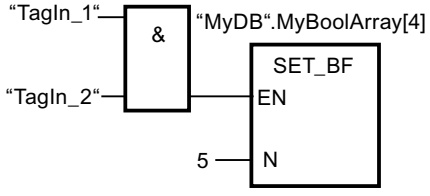
Parameters

The following table shows the parameters of the instruction "Set bit field":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
N	Input	UINT	Constant	Constant	Number of bits to be set
<Operand>	Output	BOOL	I, Q, M With a DB or an IDB, an element of an ARRAY[..] of BOOL	I, Q, M With a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be set.

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are set starting at the address of the operand "MyDB".MyBoolArray[4].

See also

Overview of the valid data types (Page 1908)

RESET_BF: Reset bit field

Description

You can use the instruction "Reset bit field" to reset several bits starting from a certain address.

You can use the value of the N input to define the number of bits to be reset. The address of the first bit to be reset is defined by (<Operand>). If the value of the input N is greater than the number of bits in a selected byte, the bits of the next byte are reset. The bits remain reset until they are explicitly set, for example, by another instruction.

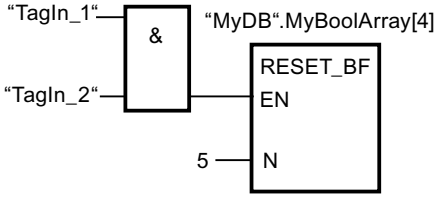
Parameters

The following table shows the parameters of the instruction "Reset bit field":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
N	Input	UINT	Constant	Constant	Number of bits to be reset.
<Operand>	Output	BOOL	I, Q, M With a DB or an IDB, an element of an ARRAY[..] of BOOL	I, Q, M With a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be reset.

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are reset starting at the address of the operand "MyDB".MyBoolArray[4].

See also

Overview of the valid data types (Page 1908)

SR: Set/reset flip-flop

Description

You can use the "Set/reset flip-flop" instruction to set or reset the bit of a specified operand based on the signal state of the inputs S and R1. If the signal state is "1" at input S and "0" at input R1, the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

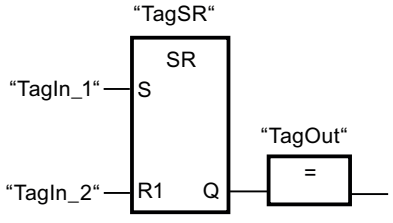
Parameters

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
S	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable setting
R1	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable resetting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is set or reset
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 1908)

RS: Reset/set flip-flop

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

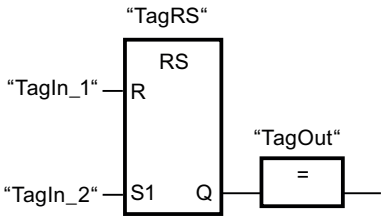
Parameters

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
R	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable resetting
S1	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable setting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is reset or set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 1908)

P: Scan operand for positive signal edge

Description

You can use the "Scan operand for positive signal edge" instruction to determine whether there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

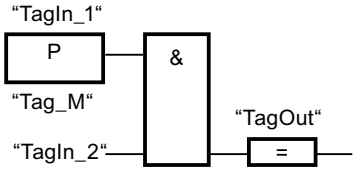
Parameters

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved.

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- There is a rising edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 1908)

Example of detecting the direction of a conveyor belt (Page 3872)

N: Scan operand for negative signal edge

Description

You can use the "Scan operand for negative signal edge" instruction to determine whether there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

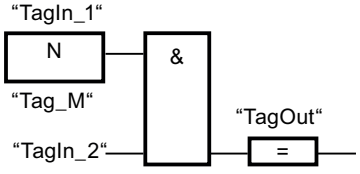
Parameters

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved.

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- There is a falling edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 1908)

P=: Set operand on positive signal edge

Description

You can use the instruction "Set operand on positive signal edge" to set a specified operand (<Operand2>) when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand1>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

When a positive edge is detected, <Operand2> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

You specify the operand (<Operand2>) to be set in the operand placeholder above the instruction. Specify the edge memory bit (<Operand1>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

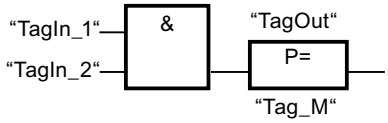
Parameters

The following table shows the parameters of the instruction "Set operand on positive signal edge":

Parameters	Declaration	Data type	Memory area	Description
<Operand2>	Output	BOOL	I, Q, M, D, L	Operand which is set when there is a positive signal edge.
<Operand1>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows the parameters of the instruction:



The "TagOut" output is set for one program cycle, when the signal state at the input of the instruction box switches from "0" to "1" (positive signal edge). In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 1908)

N=: Set operand on negative signal edge

Description

You can use the instruction "Set operand on negative signal edge" to set a specified operand (<Operand1>) when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

When a negative edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

You specify the operand (<Operand1>) to be set in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

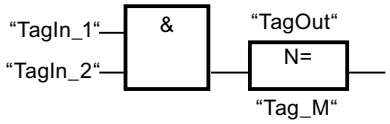
Parameters

The following table shows the parameters of the instruction "Set operand on negative signal edge":

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set when there is a negative signal edge.
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows how the instruction works:



The operand "TagOut" is set for one program cycle if the signal state at the input of the instruction box changes from "1" to "0" (negative signal edge). In all other cases, the operand "TagOut" has the signal state "0".

See also

Overview of the valid data types (Page 1908)

P_TRIG: Scan RLO for positive signal edge

Description

Use the instruction "Scan RLO for positive signal edge" to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<Operand>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

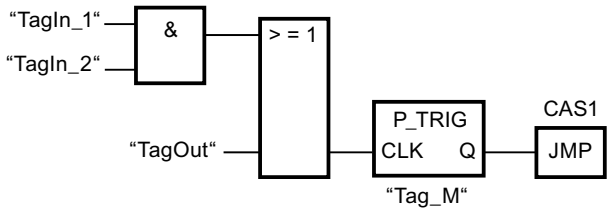
Parameters

The following table shows the parameters of the instruction "Scan RLO for positive signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 1908)

N_TRIG: Scan RLO for negative signal edge

Description

Use the instruction "Scan RLO for negative signal edge" to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query saved in the edge memory bit (<Operand>). If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

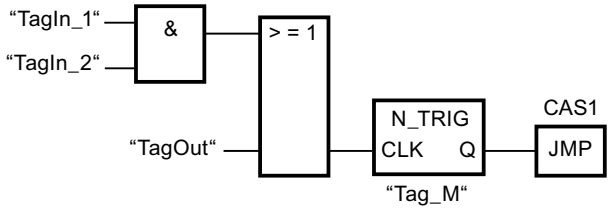
Parameters

The following table shows the parameters of the instruction "Scan RLO for negative signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 1908)

R_TRIG:Detect positive signal edge

Description

With the "Detect positive signal edge" instruction, you can detect a state change from "0" to "1" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "0" to "1", a positive signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface.

Parameters

The following table shows the parameters of the "Detect positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

The previous state of the tag at the CLK input is stored in the "R_TRIG_DB" tag. If a change in the signal state from "0" to "1" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 1908)

F_TRIG:Detect negative signal edge

Description

With the "Detect negative signal edge" instruction, you can detect a state change from "1" to "0" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "1" to "0", a negative signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface.

Parameters

The following table shows the parameters of the "Detect negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

The previous state of the tag at the CLK input is stored in the "F_TRIG_DB" tag. If a change in the signal state from "1" to "0" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 1908)

Timer operations

TP: Generate pulse

Description

You can use the instruction "Generate pulse" to set output Q for the duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The configured time duration PT begins when the instruction starts. Output Q is set for the time duration PT, regardless of the subsequent course of the input signal (positive edge). Even when a new positive signal edge is detected, the signal state of the Q output is not affected as long as the PT duration is running.

The current time value can be queried at the output ET. The time value starts at T#0s and ends when the value of the time duration PT is reached. If the configured time duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Each call of the "Generate pulse" instruction must be assigned an IEC Timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TP_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

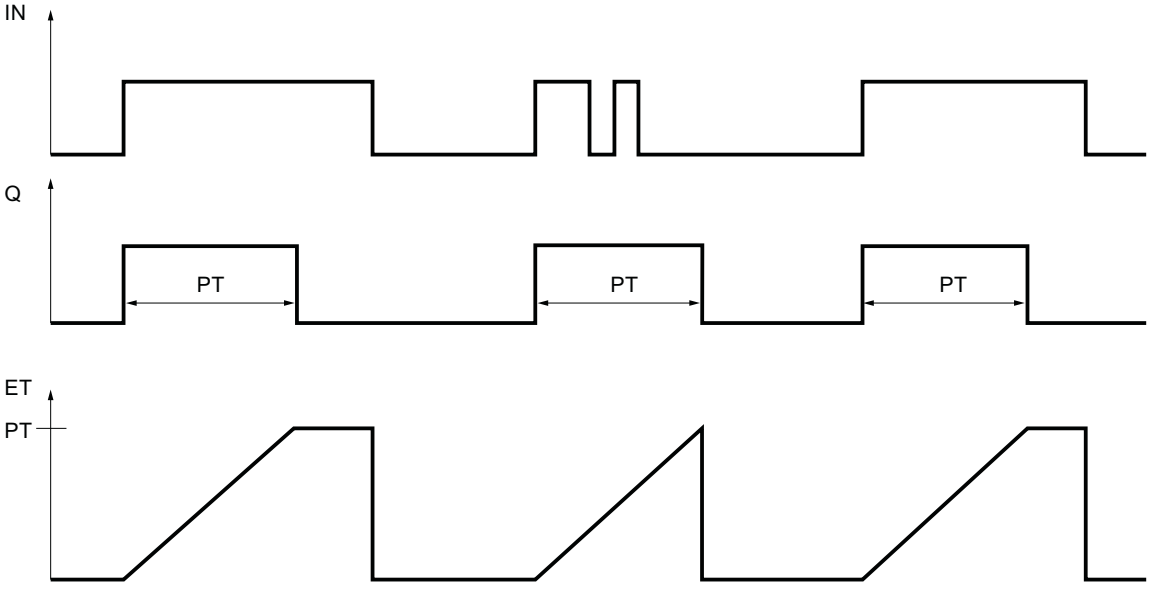
The following table shows the parameters of the "Generate pulse" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Pulse output
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:



See also

Overview of the valid data types (Page 1908)

Example of controlling room temperature (Page 3879)

TON: Generate on-delay

Description

You can use the instruction "Generate on-delay" to delay setting of the Q output by the time configured with the PT time. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the duration PT expires, output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

The current time value can be queried at the output ET. The time value starts at T#0s and ends when the value of the time duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned an IEC Timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TON_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

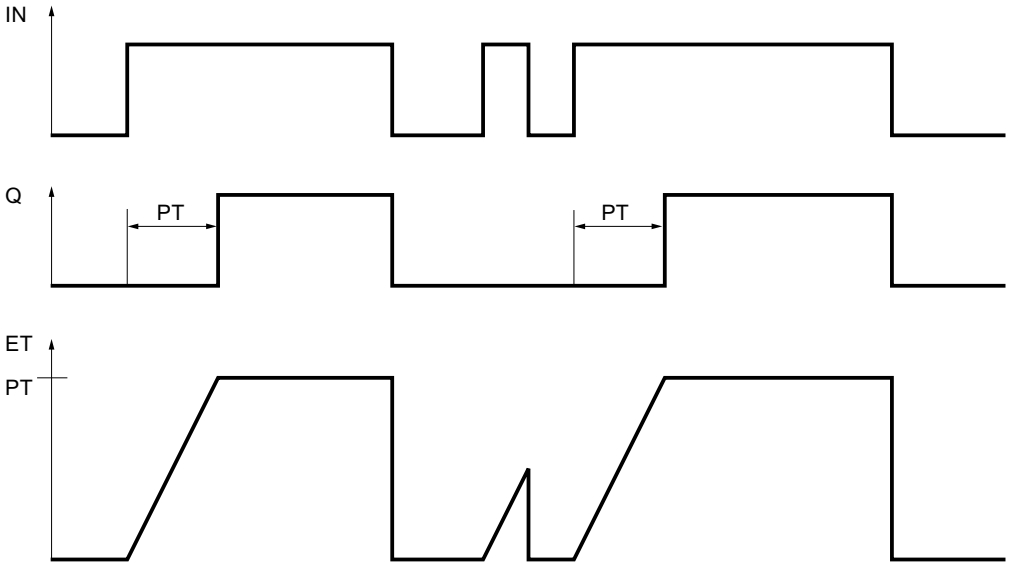
The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:



See also

Overview of the valid data types (Page 1908)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay setting of the Q output by the time configured with the PT time. The output Q is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). When the signal state at input IN changes back to "0" (positive signal edge), the configured time duration PT starts. Output Q remains set as long as the time duration PT is running. When the PT time duration expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q will continue to be "1".

The current time value can be queried at the output ET. The time value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET output remains set to the current value until input IN changes back to "1". If input IN changes to "1" before the time duration PT has expired, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC Timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TOF_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

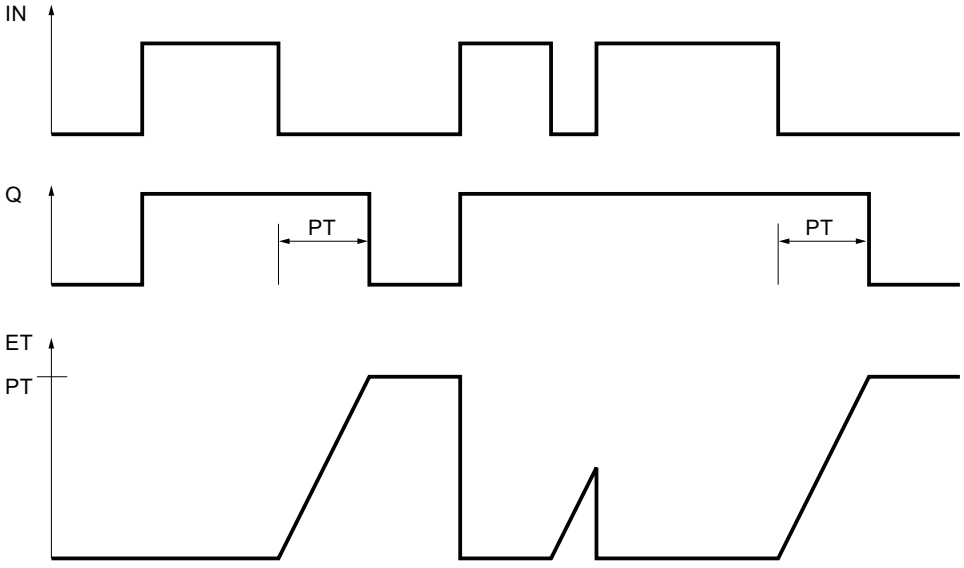
The following table shows the parameters of the "Generate off-delay" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is reset when the timer PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate off-delay" instruction:



See also

Overview of the valid data types (Page 1908)

TONR: Time accumulator

Description

The instruction "Time accumulator" is used to accumulate time values within a period set by the parameter PT. The instruction is executed and the configured time duration PT is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive edge). While the time set at PT is running, the time values are accumulated that are recorded at signal state "1" at input IN. The accumulated time is written to output ET and can be queried there. When the current time value PT is reached, the output Q has the signal state "1". Output Q remains set at "1", even when the signal state at input IN changes to "0".

The R input resets the outputs ET and Q regardless of the signal state at the start input.

Each call of the "Time accumulator" instruction must be assigned to an IEC Timer in which the instruction data is stored.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TONR_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

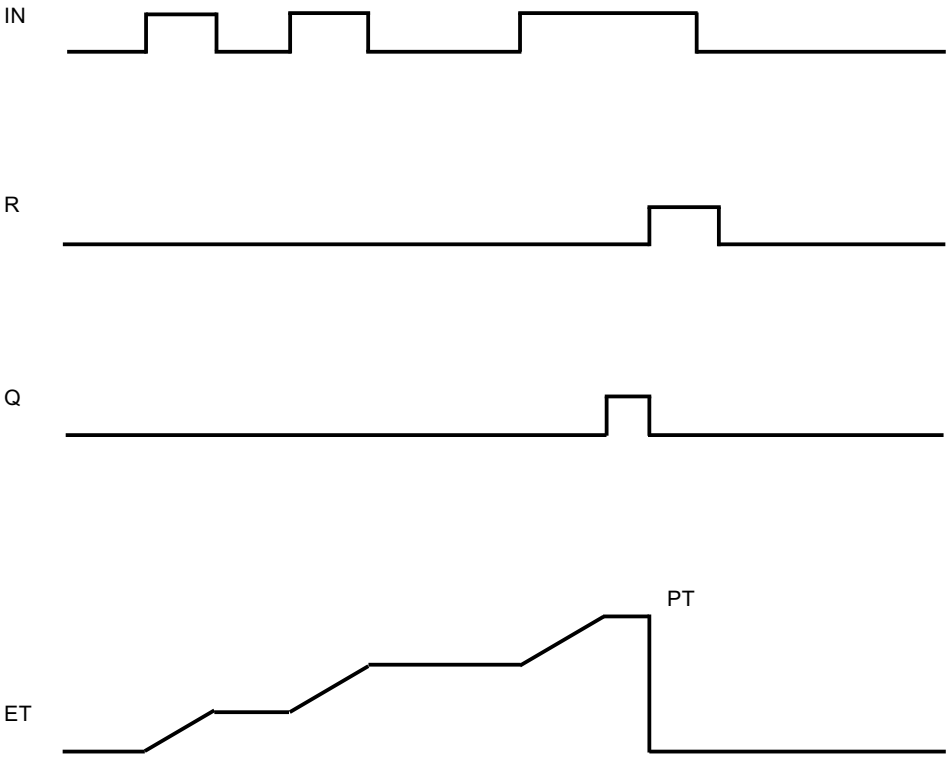
The following table shows the parameters of the "Time accumulator" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C, P	Start input
R	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Reset input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Maximum duration of time recording. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:



See also

Overview of the valid data types (Page 1908)

TP: Start pulse timer

Description

Use the instruction "Start pulse timer" to start an IEC Timer with a specified duration as pulse. The IEC Timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC Timer runs for the specified time duration regardless of the subsequent course of the RLO. The expiry of the IEC Timer is also not affected by the detection of a new rising edge. As long as the IEC Timer is running, the querying of the timer status for "1" returns the signal state "1". When the IEC Timer has expired, the timer status returns the signal state "0".

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER or TP_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the specified timer is accessed.

The current timer status is saved in the structure components Q of the IEC Timer. You can query the timer status with the help of a binary logic operation. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the instruction "Start pulse timer" assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

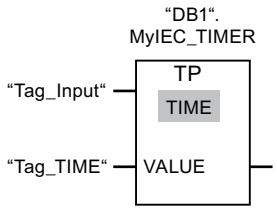
The following table shows the parameters of the instruction "Start pulse timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TP_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME	D, L	IEC Timer which is started.

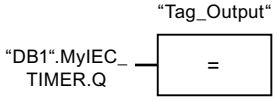
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The instruction "Start pulse timer" is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The timer "DB1".MyIEC_TIMER is started for the time stored in the operand "TagTime".



As long as the timer "DB1".MyIEC_TIMER is running, the timer status ("DB1".MyIEC_TIMER.Q) has signal state "1" and the operand "Tag_Output" is set. When the IEC Timer has expired, the signal state of the time status changes back to "0" and the "Tag_Output" operand is reset.

See also

Overview of the valid data types (Page 1908)

TON: Start on-delay timer

Description

Use the "Start on-delay timer" instruction to start an IEC Timer with a specified duration as on-delay. The IEC Timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC Timer runs for the specified time duration. The output returns the signal state "1" if the RLO at the input of the instruction has the signal state "1". If the RLO changes to "0" before the end of the timer, the running IEC Timer is reset. The query of the timer status for "1" returns the signal state "0". The IEC Timer restarts when the next rising signal edge is detected at the input of the instruction.

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start on-delay timer" stores its data in a structure of the data type IEC_TIMER or TON_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start on-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the specified timer is accessed.

The current timer status is saved in the structure components ET of the IEC Timer. You can query the timer status with the help of a binary logic operation. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Start on-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

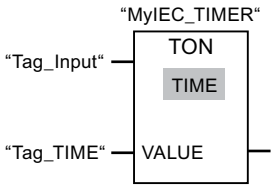
The following table shows the parameters of the instruction "Start on-delay timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TON_TIME	IEC_TIMER, IEC_LTIMER, TON_TIME, TON_LTIME	D, L	IEC Timer which is started.

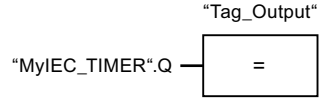
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start on-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The "MyIEC_TIMER" timer is started for the time stored in the "Tag_TIME" operand.



If the timer "MyIEC_TIMER" has expired and the operand "Tag_Input" has the signal state "1", querying the timer status ("MyIEC_TIMER".Q) returns signal state "1" and the "Tag_Output" operand is set. When the signal state of the operand "Tag_Input" changes to "0", the querying of the timer status returns the signal state "0" and the operand "Tag_Output" is reset.

See also

Overview of the valid data types (Page 1908)

TOF: Start off-delay timer

Description

Use the "Start off-delay timer" instruction to start an IEC Timer with a specified duration as off-delay. The query of the timer status for "1" returns the signal state "0" if the result of the logic operation (RLO) at the input of the instruction has the signal state "1". When the RLO changes from "1" to "0" (negative signal edge), the IEC Timer starts with the specified time duration. The timer status remains at signal state "1" as long as the IEC Timer is running. When the timer has run out and the RLO at the input of the instruction has the signal state "0", the timer status is set to the signal state "0". If the RLO changes to "1" before the end of the timer, the running IEC Timer is reset and the timer status remain at the signal state "1".

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start off-delay timer" stores its data in a structure of the data type IEC_TIMER or TOF_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start off-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the specified timer is accessed.

The current timer status is saved in the structure components ET of the IEC Timer. You can query the timer status with the help of a binary logic operation. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Start off-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

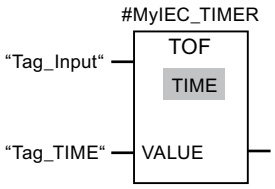
The following table shows the parameters of the instruction "Start off-delay timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TOF_TIME	IEC_TIMER, IEC_LTI-MER, TOF_TIME, TOF_LTIME	D, L	IEC Timer which is started.

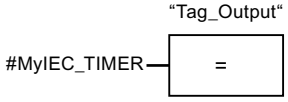
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start off-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "1" to "0". The #MyIEC_TIMER timer is started for the time stored in the operand "Tag_TIME".



As long as timer #MyIEC_TIMER is running, the query of the time status (#MyIEC_TIMER.Q) returns the signal state "1" and operand "Tag_Output" is set. If the timer has expired and the operand "Tag_Input" has the signal state "0", the query of the timer status returns the signal state "0". If the signal state of the operand "Tag_Input" changes to "1" before timer #MyIEC_TIMER expires, the timer is reset. When the signal state of the operand "Tag_Input" is "1", the query of the timer status returns the signal state "1".

See also

Overview of the valid data types (Page 1908)

TONR: Time accumulator

Description

You can use the "Time accumulator" instruction to record how long the signal is at the input of instruction "1". The instruction is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The time is recorded as long at the RLO is "1". If the RLO changes to "0", the instruction is halted. If the RLO changes back to "1", the time recording is continued. The query of the timer status for "1" returns the signal state "1" if the recorded time exceeds the value of the specified time duration and the RLO at the input of coil is "1".

The timer status and the currently expired timer can be reset to "0" using the "Reset timer" instruction.

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER or TONR_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the specified timer is accessed.

The current timer status is saved in the structure components ET of the IEC Timer. You can query the timer status with the help of a binary logic operation. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed only at the end of the network.

Parameters

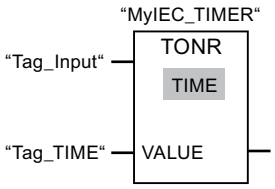
The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TONR_TIME	IEC_TIMER, IEC_LTIMER, TONR_TIME, TONR_LTIME	D, L	IEC Timer which is started.

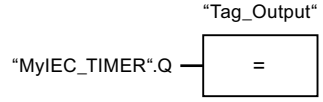
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Time accumulator" instruction is executed if there is a positive signal edge in the RLO. The time is recorded as long as the operand "Tag_Input" has the signal state "1".



If the recorded time exceeds the value of the operand "Tag_TIME", then the query of the timer status ("MyIEC_TIMER".Q) will return the signal state "1" and the operand "Tag_Output" will be set.

See also

Overview of the valid data types (Page 1908)

RT: Reset timer (Page 2511)

RT: Reset timer

Description

You can use the "Reset timer" instruction to reset an IEC Timer to "0". You specify the IEC Timer to be reset by entering the name of the data block that contains the structure of the IEC Timer in the placeholder above the instruction.

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". When the instruction is executed the structure components of the IEC Timer are reset to "0" in the specified data block. If the RLO at box input is "0", the instruction is not executed.

The instruction does not influence the RLO. The RLO at the box input is transferred directly to the box output.

You must assign a IEC Timer declared in the program to the "Reset timer" instruction.

The instruction data is updated only when the instruction is called and not each time the assigned IEC Timer is accessed. The query of the data is only identical from the call of the instruction to the next call of the instruction.

Parameters

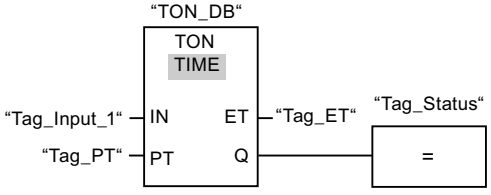
The following table shows the parameters of the instruction "Reset timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<IEC Timer>	InOut	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC Timer, which is reset.

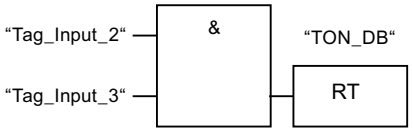
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC Timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".



If the operands "Tag_Input_2" and "Tag_Input_3" have the signal state "1", the "Reset timer" instruction is executed and the IEC Timer stored in the data block "TON_DB" is reset.

See also

Overview of the valid data types (Page 1908)

PT: Load time duration

Description

Use the "Load time duration" instruction to set the time duration of an IEC Timer. The instruction is executed in every cycle when the result of logic operation (RLO) at the input of the instruction has the signal state "1". The instruction writes the specified time duration to the structure of the specified IEC Timer.

Note

If the specified IEC Timer is running during the execution, the instruction overwrites the current time duration of the specified IEC Timer. As a result, the timer status of the IEC Timer can change.

You must assign a IEC Timer declared in the program to the "Load time duration" instruction.

The instruction data is updated when the instruction is called and each time the assigned IEC Timer is accessed. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Parameters

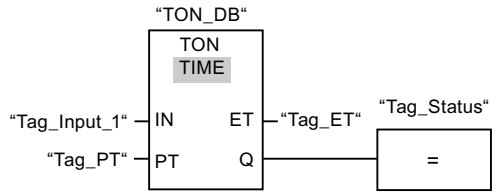
The following table shows the parameters of the instruction "Load time duration":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Time duration
<IEC Timer>	InOut	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC Timer, the duration of which is set.

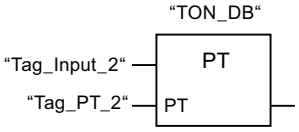
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC Timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".



The "Load time duration" instruction is executed when the operand "Tag_Input_2" has the signal state "1". The instruction writes the time duration "Tag_PT_2" in the instance data block "TON_DB" and at the same time overwrites the value of the operand "Tag_PT" within the data block. As a result, the signal state of the timer status can change at the next query or upon access to "MyTimer".Q or "MyTimer".ET.

Note

The "Tag_Input_2" is executed as pulse flag in order that the time duration is loaded only throughout one program cycle.

See also

Overview of the valid data types (Page 1908)

Legacy

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as long as the signal state at input S is "1". If the signal state at input S changes to "0" before the programmed duration expires, the timer is stopped. In this case, the signal state at output Q is "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down

to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1", the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign pulse timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

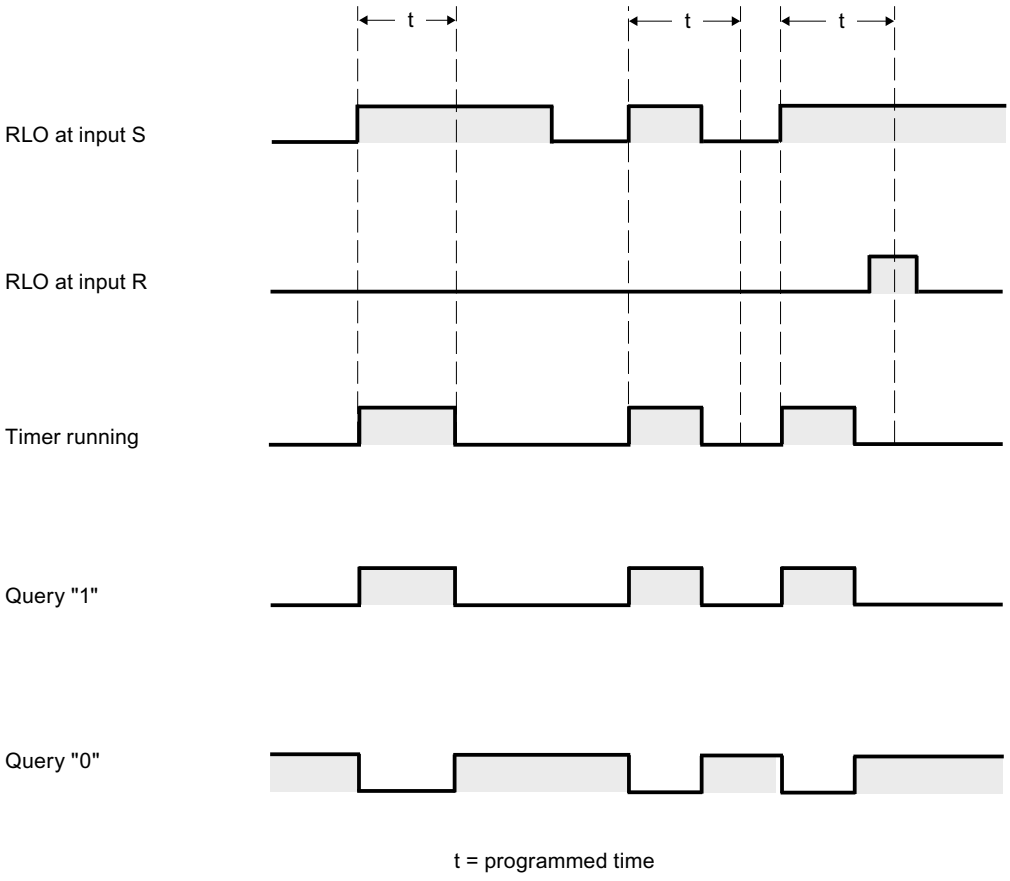
The following table shows the parameters of the "Assign pulse timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

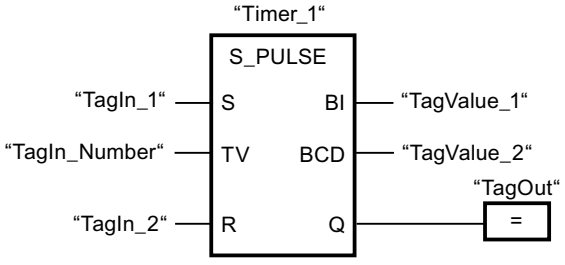
The following figure shows the pulse timing diagram of the "Assign pulse timer parameters and start" instruction:



t = programmed time

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer runs for the time value of the "TagIn_Number" operand as long as the "TagIn_1" operand has the signal state "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer "Timer_1" is stopped. The "TagOut" operand is reset to signal state "0".

The "TagOut" operand has the signal state "1" as long as the timer is running and the "TagIn_1" operand has the signal state "1". When the time has expired or is reset, the "TagOut" operand is reset to "0".

See also

Overview of the valid data types (Page 1908)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". As long as the timer is running, output Q has the signal state "1". When the timer has expired, output Q is reset to "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input TV.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1", the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign extended pulse timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

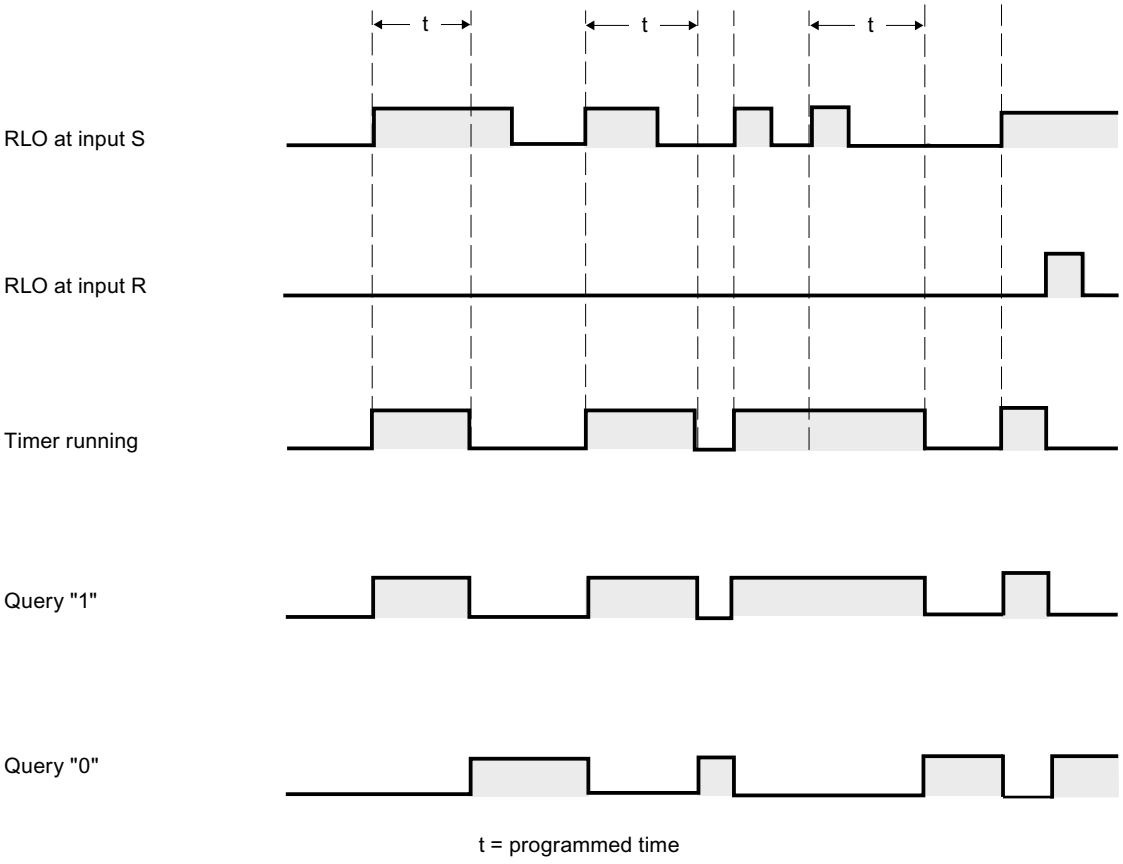
The following table shows the parameters of the "Assign extended pulse timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

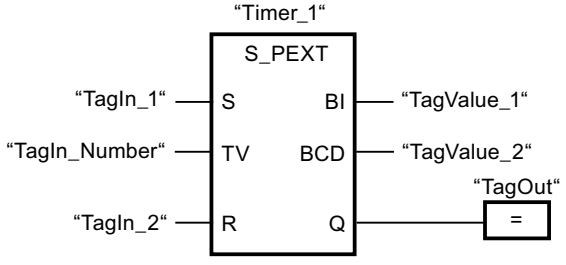
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer runs for the time value of the "TagIn_Number" operand without being affected by a negative edge at the S input. If the signal state at the "TagIn_1" operand changes from "0" to "1" before the timer expires, the timer is restarted.

The "TagOut" operand has the signal state "1" as long as the timer is running. When the time has expired or is reset, the "TagOut" operand is reset to "0".

See also

Overview of the valid data types (Page 1908)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as long as the signal state at input S is "1". If the timer expires correctly and input S still has signal state "1", output Q returns signal state "1". If the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the time is running and the signal state at input R changes from "0" to "1", the current time value and the time base are also set to zero. In this case, the signal state at output Q is "0". The timer is reset if the signal state is "1" at the R input even if the timer is not running and the RLO at input S is "1".

The "Assign on-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign on-delay timer parameters and start" instruction:

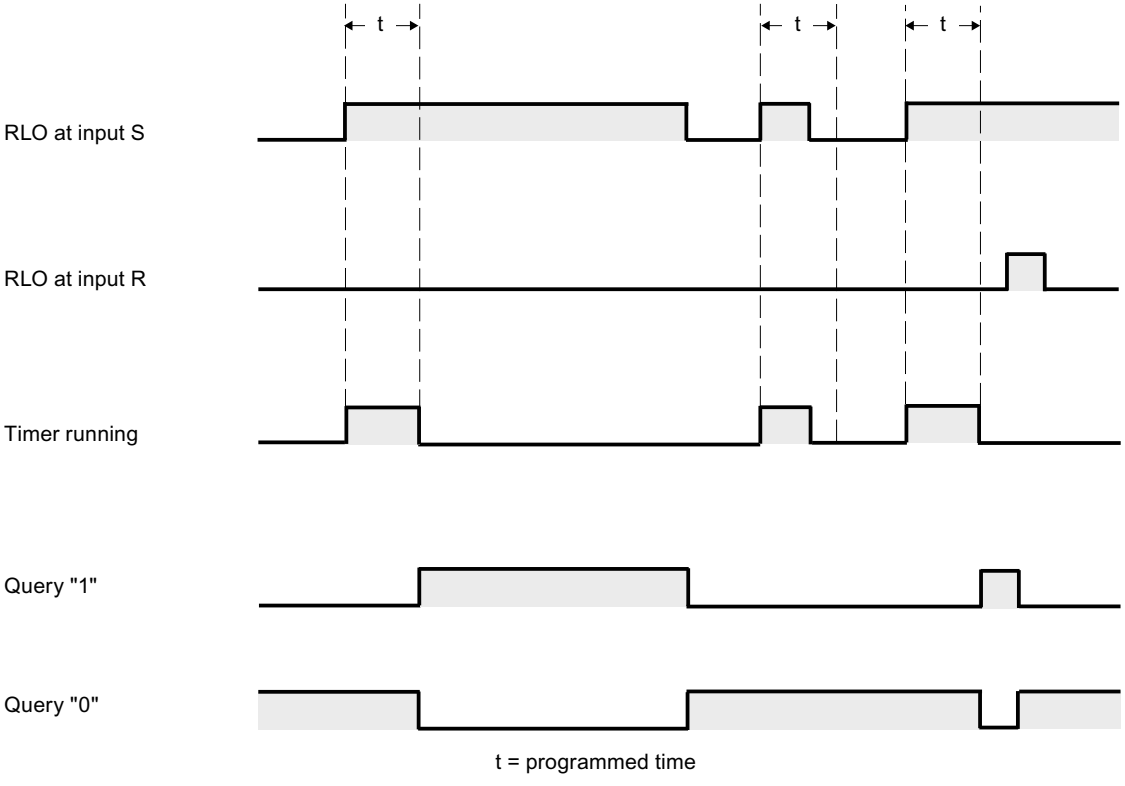
Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input

Parameters	Declaration	Data type	Memory area	Description
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

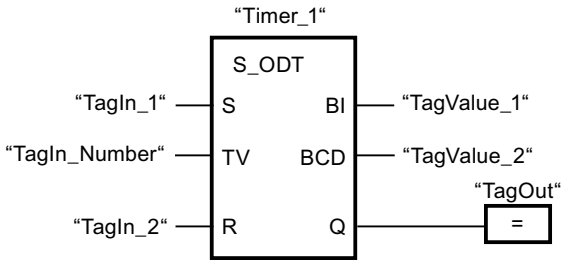
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". If the timer expires and the signal state of the operand is "1", the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. The "TagOut" operand has the signal state "0".

See also

Overview of the valid data types (Page 1908)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". If the timer expires, the "Q" output returns signal state "1" regardless of the signal state at input "S". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input (TV).

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0" regardless of the signal state at start input S. In this case, the signal state at output Q is "0".

The "Assign retentive on-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

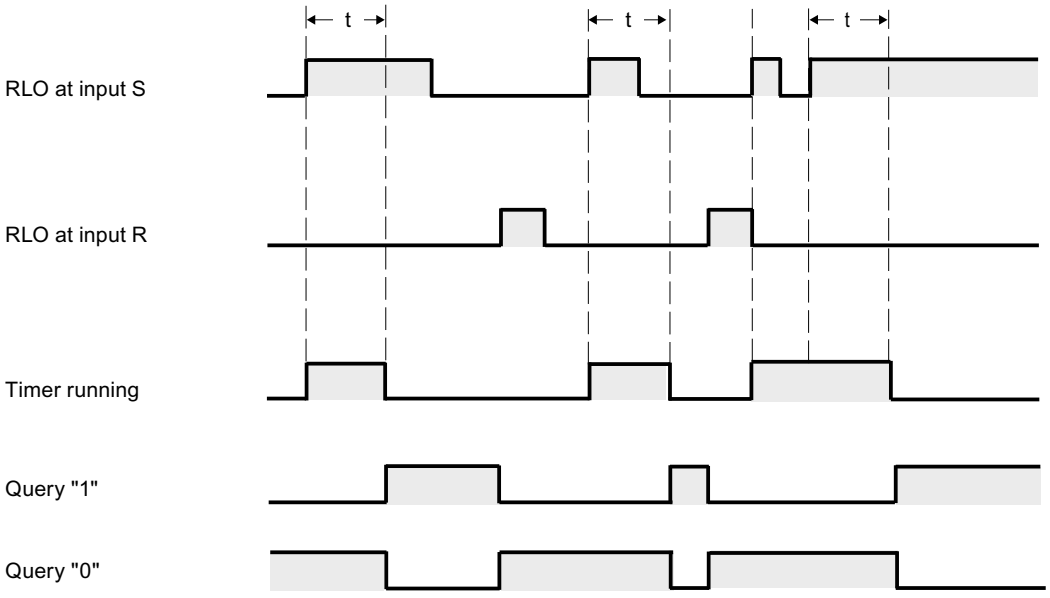
The following table shows the parameters of the "Assign retentive on-delay timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

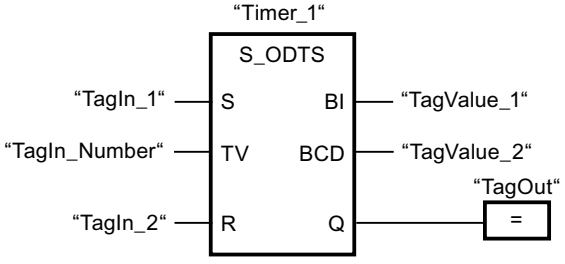
The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:



t = programmed time

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand, even when the signal state of the "TagIn_1" operand changes to "0". When the time expires, the "TagOut" operand is reset to "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is restarted.

See also

Overview of the valid data types (Page 1908)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a transition from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV). As long as the timer is running or input S returns signal state "1", output Q has signal state "1". If the timer expires and the signal state is "0", output Q is reset to signal state "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at input S.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0". In this case, the signal state at output Q is "0".

The "Assign off-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign off-delay timer parameters and start" instruction:

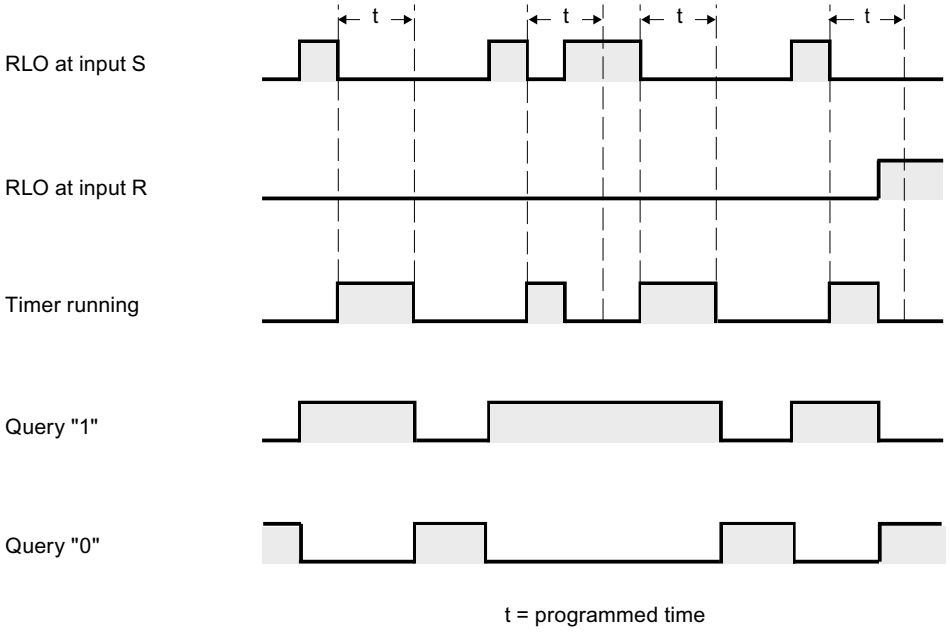
Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input

Parameters	Declaration	Data type	Memory area	Description
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

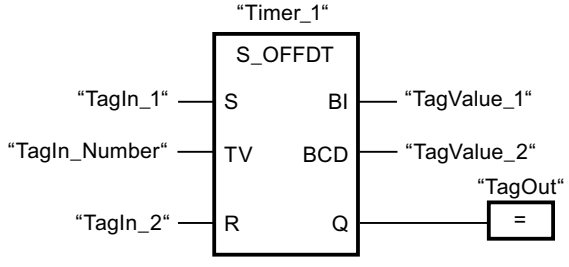
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "1" to "0". The timer expires with the value of operand "TagIn_Number". The "TagOut" operand is set to "1"

if the timer is running or the "TagIn_1" operand has the signal state "0". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is reset.

See also

Overview of the valid data types (Page 1908)

SP: Start pulse timer

Description

The instruction "Start pulse timer" starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs with the specified duration as long as the RLO has the signal state "1". As long as the timer is running, the query of timer status "1" returns the signal state "1". If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer stops. In this case, the query for timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start pulse timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

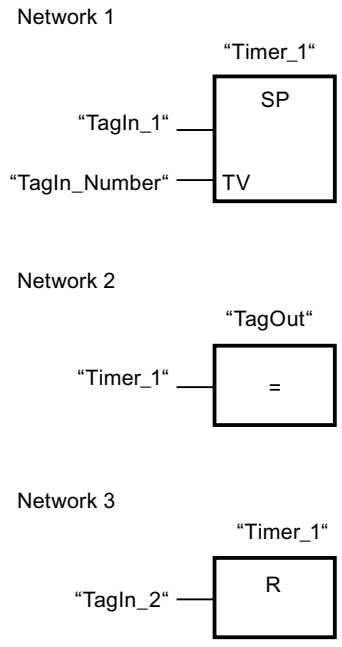
The following table shows the parameters of the instruction "Start pulse timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand as long as the signal state of the "TagIn_1" operand is "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. As long as the timer is running, the "TagOut" operand returns signal state "1". If the signal state of the "TagIn_1" operand changes from "0" to "1", the timer is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 1908)

SE: Start extended pulse timer

Description

The "Start extended pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period even when the RLO changes to signal state "0". As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is restarted with the programmed time period. When the timer expires, the query for timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start extended pulse timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

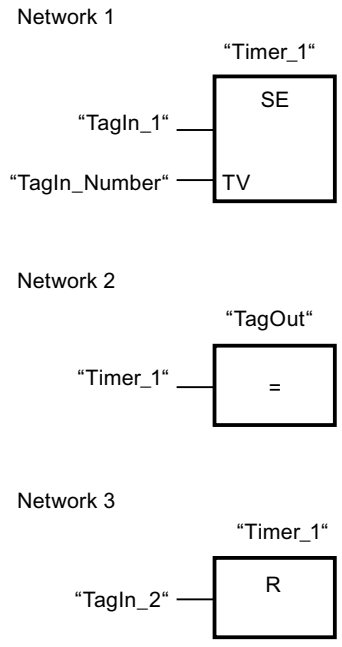
The following table shows the parameters of the instruction "Start extended pulse timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand without being affected by a negative edge at the RLO. As long as the timer is running, the "TagOut" operand returns signal state "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" before the timer expires, the timer is restarted.

See also

Overview of the valid data types (Page 1908)

SD: Start on-delay timer

Description

The "Start on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified period of time as long as the RLO is "1". If the timer expires and the RLO has the signal state "1", the query timer status "1" returns the signal state "1". If the RLO changes from "1" to "0" while the timer is running, the timer is stopped. In this case, querying the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start on-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

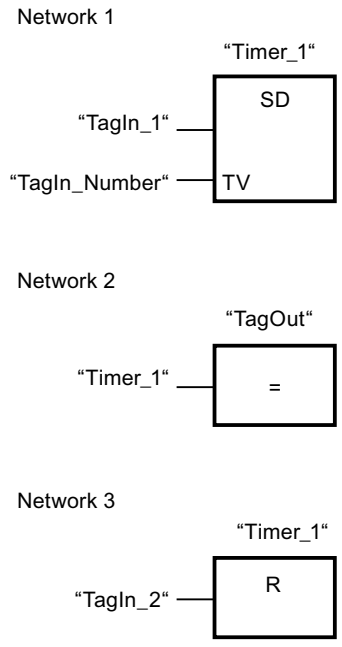
The following table shows the parameters of the instruction "Start on-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". If the timer expires and the RLO has the signal state "1", the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 1908)

SS: Start retentive on-delay timer

Description

The "Start retentive on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period even when the RLO changes to signal state "0". When the timer expires, the query for timer status for "1" returns the signal state "1". When the timer expires, the timer can only be restarted if it is explicitly reset.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start retentive on-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

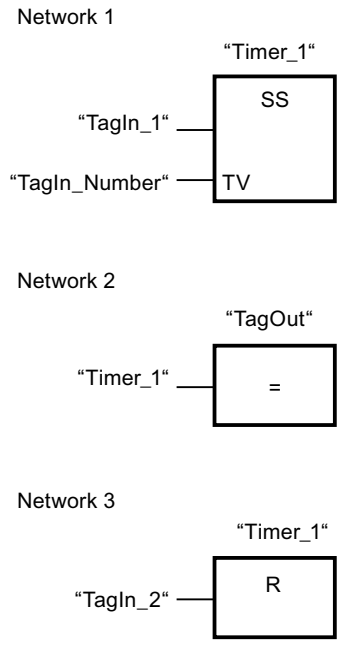
The following table shows the parameters of the instruction "Start retentive on-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". When the time expires, the "TagOut" operand is reset to "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is restarted. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 1908)

SF: Start off-delay timer

Description

The "Start off-delay timer" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period. As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is reset. The timer is always restarted when the RLO changes from "1" to "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start off-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

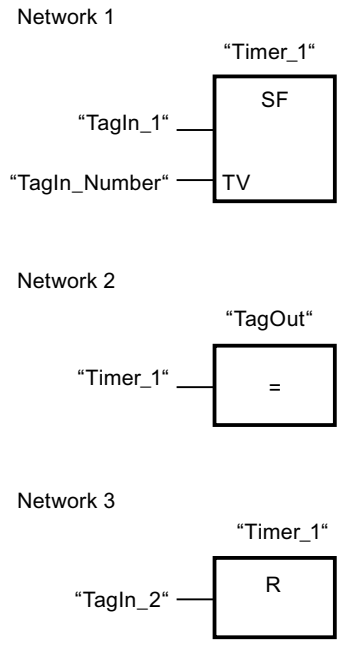
The following table shows the parameters of the instruction "Start off-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "1" to "0". The timer expires with the value of operand "TagIn_Number". As long as the timer is running, the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" while the timer is running, the timer is restarted. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 1908)

Counter operations

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the output CV. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can query the counter status in the Q output. The signal state at the Q output is determined by the PV parameter. If the current counter value is greater than or equal to the value of the

PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is reset to "0" and saved to an edge memory bit when the signal state at input R changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will

find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction "Count up":

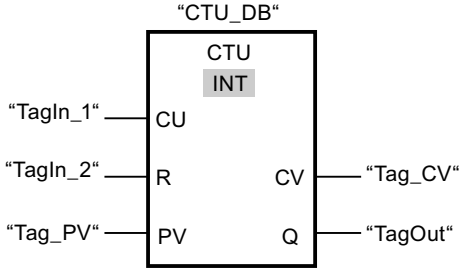
Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, D, L, P, or constant	I, Q, M, T, C, D, L, P or constant	Reset input
PV	Input	Integers	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CV" operand is incremented

by one. With each additional positive signal edge, the counter value is incremented until the high limit of the specified data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output returns the signal state "0".

See also

Overview of the valid data types (Page 1908)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can query the counter status in the Q output. If the current counter value is less than or equal to "0", the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is set to the value of the PV parameter and saved to a edge memory bit when the signal state at the LD input changes from "0" to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
LD	Input	BOOL	I, Q, M, D, L, P, or constant	I, Q, M, T, C, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Value to which the CV output is set with LD = 1.

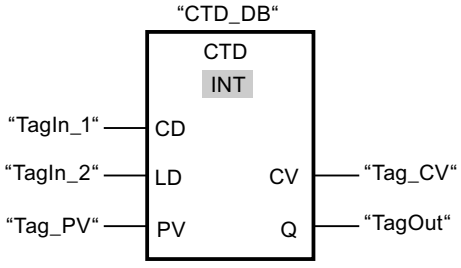
Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the instruction is executed and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the counter value is decremented until the low limit of the specified data type (INT = -32768) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is less than or equal to "0". In all other cases, the "TagOut" output returns the signal state "0".

See also

Overview of the valid data types (Page 1908)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the current counter value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV output. When the high limit is reached, the counter value is no longer incremented

on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter and stored in a edge memory bit. As long as the LD input has signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The counter value is set to "0" and stored in an edge memory bit when the signal state at input R changes to "1". As long as the R input has signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0". You can also specify a constant for the PV parameter.

You can query the status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction "Count up and down":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P, or constant	I, Q, M, T, C, D, L, P or constant	Reset input

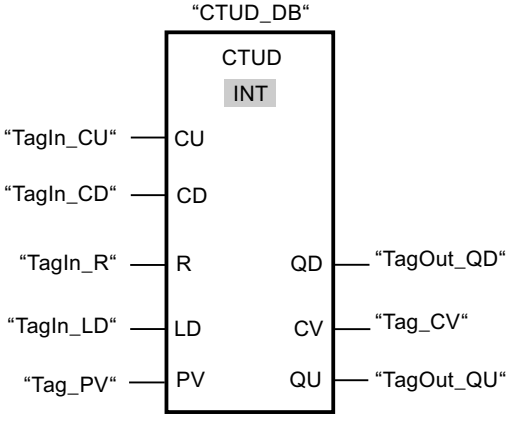
Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
LD	Input	BOOL	I, Q, M, D, L, P, or constant	I, Q, M, T, C, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Value at which the output QU is set. / Value to which the CV output is set with LD = 1.
QU	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Up-counter status
QD	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Down-counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_CU" or "TagIn_CD" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_CU" input, the current counter value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_CD" input, the counter value is decremented by one and stored at the "Tag_CV" output. When there is a positive edge at the CU input, the counter value is incremented until it reaches the high limit (INT = 32767). If input CD has a positive signal edge, the counter value is decremented until it reaches the low limit (INT = -32768).

The "TagOut_GU" output has signal state "1" as long as the current counter value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut_QU" output returns the signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current counter value is less than or equal to "0". In all other cases, the "TagOut_QD" output has signal state "0".

See also

Overview of the valid data types (Page 1908)

Example of detecting the fill level of a storage area (Page 3874)

Legacy

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The count is incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CU is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

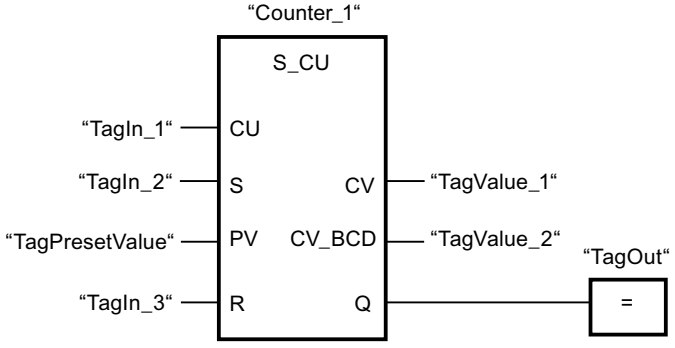
The following table shows the parameters of the "Assign parameters and count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L, T, C	Count up input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. When the signal state at the "TagIn_2" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 1908)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The count is decremented until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CD is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count down" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count down" instruction:

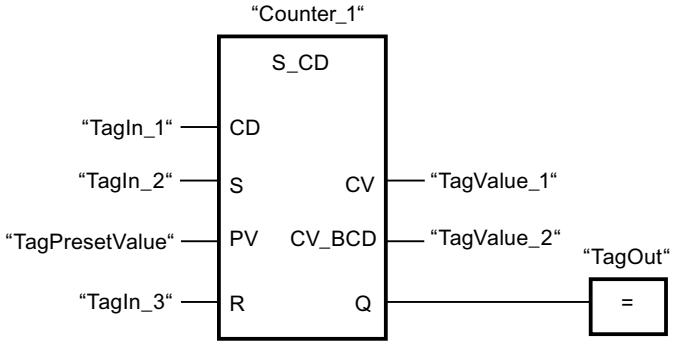
Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input

Parameters	Declaration	Data type	Memory area	Description
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decremented by one. When the signal state at the "TagIn_2" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 1908)

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment or decrement the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. If there is a positive signal edge at the CU and CD inputs in one program cycle, the counter value remains unchanged.

The counter value is incremented until the high limit of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of "0" is reached, the counter value is no longer decremented.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO is "1" at the inputs CU and CD, the counter counts accordingly in the next scan cycle, even if no change in the signal edge is detected.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU, CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up / down" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count up / down" instruction:

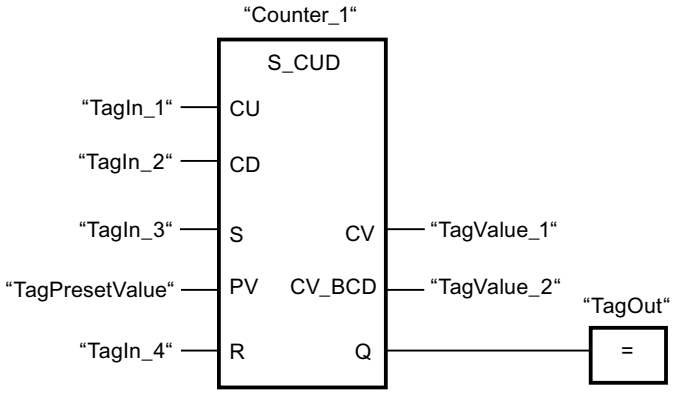
Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L, T, C	Count up input
CD	Input	BOOL	I, Q, M, D, L, T, C or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)

Parameters	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_1" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Assign parameters and count up / down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input and the current counter value is less than "999", the counter value is incremented by one. When there is a positive signal edge at the "TagIn_2" input and the current counter value is greater than "0", the counter value is decremented by one.

When the signal state at the "TagIn_3" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_4" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 1908)

SC: Set counter value

Description

You can use the "Set counter value" instruction to set the value of a counter. The instruction is executed when the result of logic operation (RLO) at the start input of the instruction changes from "0" to "1". When the instruction is executed, the counter is set to the specified counter value.

The "Set counter value" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

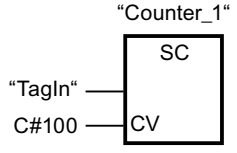
The following table lists the parameters of the "Set counter value" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
CV	Input	WORD	I, Q, M, D, L or constant	Value with which the counter is preset in the BCD format. (C#0 to C#999)
<Counter>	InOut/Input	COUNTER	C	Counter that is preset.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



the counter "Counter_1" starts with the value "100" when the signal state of the "TagIn" operand changes from "0" to "1".

See also

Overview of the valid data types (Page 1908)

CU: Count up

Description

You use the "Count up" instruction to increment the value of the specified counter by the count of one on a rising edge at the start input. The count is incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

The "Count up" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

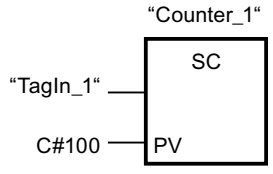
Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
<Counter>	InOut/Input	COUNTER	C	Counter whose value is incremented.

For additional information on valid data types, refer to "See also".

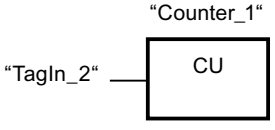
Example

The following example shows how the instruction works:

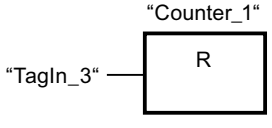
Network 1



Network 2



Network 3



When the signal state of the operand "TagIn_1" changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of "Counter_1" is incremented by the count of one when the signal state of the "TagIn_2" operand changes from "0" to "1".

If the "TagIn_3" operand returns signal state "1", the value of "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 1908)

CD: Count down

Description

You use the "Count down" instruction to decrement the value of the specified counter by the count of one on a rising edge at the start input. The count is decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer changed on a positive signal edge.

The "Count down" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

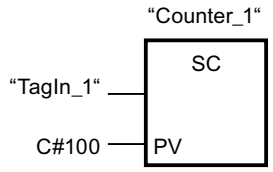
Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
<Counter>	InOut/Input	COUNTER	C	Counter whose value is decremented.

For additional information on valid data types, refer to "See also".

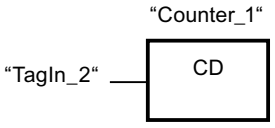
Example

The following example shows how the instruction works:

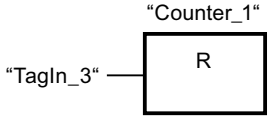
Network 1



Network 2



Network 3



When the signal state of the operand "TagIn_1" changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of "Counter_1" is incremented by the count of one when the signal state of the "TagIn_2" operand changes from "0" to "1".

If the "TagIn_3" operand returns signal state "1", the value of "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 1908)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to query whether the value at input IN1 is equal to the value at input IN2.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table lists examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0

The "Equal" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width (length) of the operands must be the same. If the floating-point numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

Note

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, instead of the instruction "CMP ==: Equal", use the instruction "IN_RANGE: Value within range".

Note

Comparison of the data type PORT

To be able to compare the operands of the PORT data type with the "Equal" instruction, you need to select the WORD data type from the drop-down list of the instructions box.

Parameters

The following table lists the parameters of the "Equal" instruction:

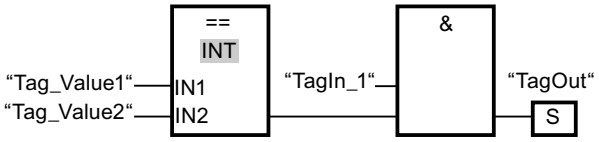
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" = "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to query whether the value at input IN1 is not equal to the value at input IN2.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table lists examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0

The "Not equal" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width (length) of the operands must be the same. If the floating-point

numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

Note

Comparison of the data type PORT

To be able to compare the operands of the PORT data type with the "Not equal" instruction, you need to select the WORD data type from the drop-down list of the instructions box.

Parameters

The following table lists the parameters of the "Not equal" instruction:

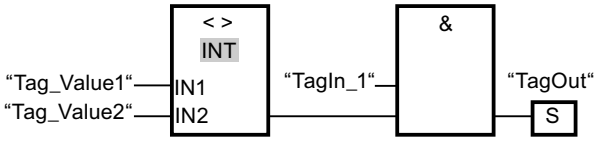
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <> "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to query whether the value at input IN1 is greater or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table lists examples of string comparisons:

IN1	IN2	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'Hello World'	'Hello World'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'AAA'	'a'	0

The "Greater or equal" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is greater (more recent) than or equal to the timer at input IN2.

Parameters

The following table lists the parameters of the "Greater or equal" instruction:

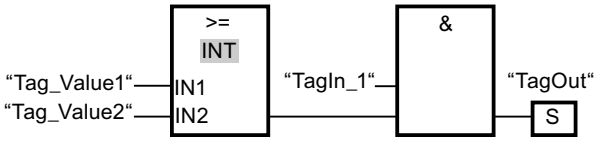
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" >= "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

Example of detecting the fill level of a storage area (Page 3874)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to query whether the value at input IN1 is less or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table lists examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'Hello World'	'Hello World'	1
'HelloWorld'	'Hello World'	0
'BB'	'AA'	0
'AAA'	'AA'	0

The "Less or equal" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is smaller (less recent) than or equal to the timer at input IN2.

Parameters

The following table lists the parameters of the "Less or equal" instruction:

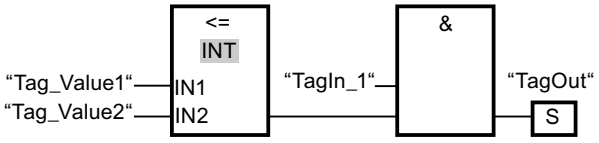
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <= "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

CMP >: Greater than

Description

You can use the "Greater than" instruction to query whether the value at input IN1 is greater than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table lists examples of string comparisons:

IN1	IN2	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'AA'	'aa'	0
'AAA'	'a'	0

The "Greater than" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is greater (more recent) than the timer at input IN2.

Parameters

The following table lists the parameters of the "Greater than" instruction:

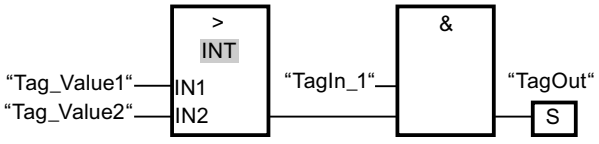
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" > "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

CMP <: Less than

Description

You can use the "Less than" instruction to query whether the value at input IN1 is less than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'BB'	'AA'	0
'AAA'	'AA'	0

The "Less than" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is less (less recent) than the timer at input IN2 .

Parameters

The following table lists the parameters of the "Less than" instruction:

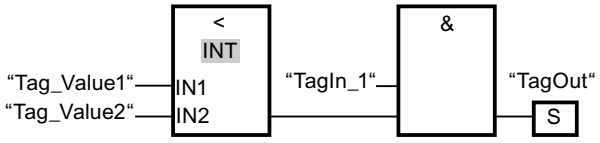
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" < "Tag_Value2".

See also

Overview of the valid data types (Page 1908)

Example of detecting the fill level of a storage area (Page 3874)

IN_RANGE: Value within range

Description

You can use the "Value within range" instruction to query whether of the value at input VAL is within a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value within range" instruction compares the value at input VAL with the values of the inputs MIN and MAX and sends the result to the box output. If the value at input VAL satisfies the comparison MIN <= VAL or VAL <=MAX, the box output has the signal state "1". If the comparison is not fulfilled, the signal state is "0" at the box output.

The comparison function can only execute if the values to be compared are of the same data type.

Parameters

The following table shows the parameters of the "Value within range" instruction:

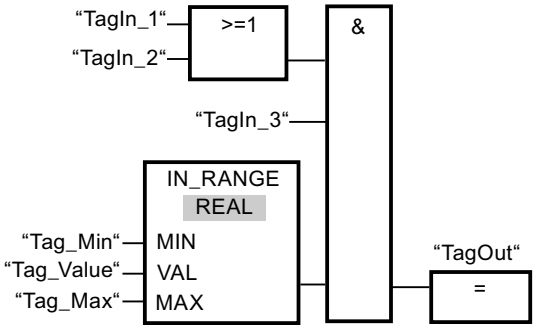
Parameters	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1" or "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".
- The value of the operand "Tag_Value" is within the value range that is specified by the current values of the operands "Tag_Min" and "Tag_Max" (MIN <= VAL or VAL <= MAX).

See also

Overview of the valid data types (Page 1908)

OUT_RANGE: Value outside range

Description

You can use the "Value outside range" instruction to query whether of the value at input VAL is outside a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value outside range" instruction compares the value at input VAL with the values of the inputs MIN and MAX and sends the result to the box output. If the value at input VAL satisfies the comparison MIN > VAL or VAL >MAX, the box output has the signal state "1". The box output has the signal state "1" if a specified operand of data type REAL has an invalid value.

The box output returns the signal state "0", if the value at input VAL does not satisfy the MIN > VAL or VAL > MAX condition.

The comparison function can only execute if the values to be compared are of the same data type.

Parameters

The following table shows the parameters of the "Value outside range" instruction:

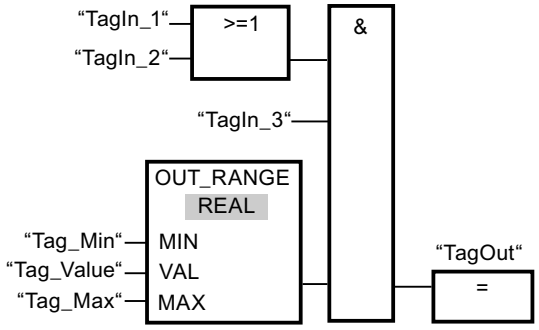
Parameters	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".
- The value of the operand "Tag_Value" is outside the value range that is specified by the values of the operands "Tag_Min" and "Tag_Max" (MIN > VAL or VAL > MAX).

See also

Overview of the valid data types (Page 1908)

OK: Check validity

Description

You can use the "Check validity" instruction to check if the value of an operand (<Operand>) is a valid floating-point number. The check is performed in every program cycle. If the operand value at the time of the query is a valid floating-point number, the output box will return the signal state"1". In all other cases, the signal state at the output of the "Check validity" instruction is "0".

You can use the "Check validity" instruction together with the EN mechanism. If you connect the instruction box to an EN enable input, the enable input is set only when the result of the validity query of the value is positive. You can use this function to ensure that an instruction is enabled only when the value of the specified operand is a valid floating-point number.

Parameters

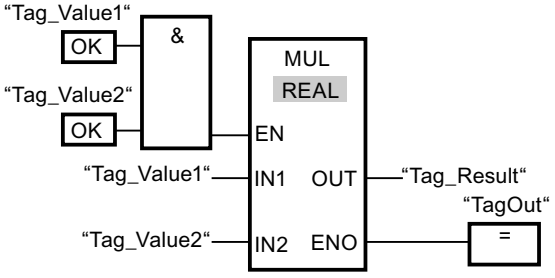
The following example shows how the "Check validity" instruction works:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be checked.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



When the values of the operands "Tag_Value1" and "Tag_Value2" show valid floating-point numbers, the "Multiply" (MUL) instruction is activated and the ENO output is set. During the execution of the "Multiply" (MUL) instruction, the value of the operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The product of the multiplication is then stored in the operand "Tag_Result". If no errors occur during the execution of the instruction, the outputs ENO and "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

NOT_OK: Check invalidity

Description

You can use the "Check invalidity" instruction to check if the value of an operand (<Operand>) is an invalid floating-point number. The check is performed in every program cycle. If the operand value at the time of the query is a valid floating-point number, then the output box will return the signal state "1". In all other cases, the signal state on the output box is "0".

Parameters

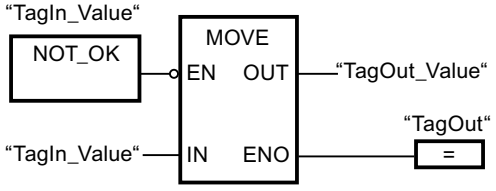
The following table shows the parameters of the instruction "Check invalidity":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be checked.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



When the value of operand "TagIn_Value" is an invalid floating-point number, the instruction "Move value" (MOVE) is not executed. The "TagOut" output is reset to signal state "0".

See also

Overview of the valid data types (Page 1908)

VARIANT

EQ_Type: Compare data type for EQUAL with the data type of a tag

Description

You can use the "Compare data type for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Equal to".

The tag at the IN1 parameter must be VARIANT data type. The tag at the IN2 parameter may be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

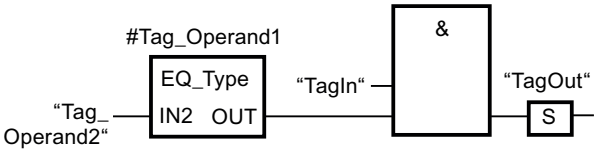
The following table lists the parameters of the "Compare data type for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	I, Q, M, L	First operand
IN2	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L, P	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand1 operand is equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 1908)

NE_Type: Compare data type for UNEQUAL with the data type of a tag

Description

You can use the "Compare data type for UNEQUAL with the data type of a tag" instruction to query which data type a tag to which a VARIANT points does not have. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Not equal to".

The tag at the IN1 parameter must be VARIANT data type. The tag at the IN2 parameter may be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

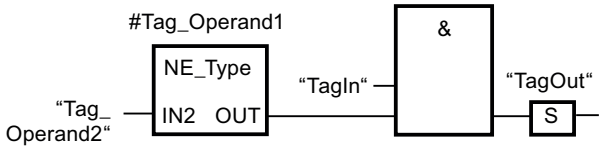
The following table lists the parameters of the "Compare data type for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	I, Q, M, L	First operand
IN2	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L, P	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand1 operand is not equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 1908)

EQ_ElemType: Compare data type of an ARRAY element for EQUAL with the data type of a tag

Description

You can use the "Compare data type of an ARRAY element for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Equal to".

The tag at the IN1 parameter must be VARIANT data type. The tag at the IN2 parameter may be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

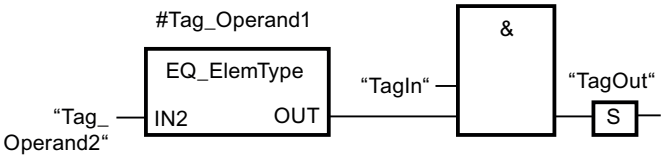
The following table lists the parameters of the "Compare data type of an ARRAY for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	I, Q, M, L	First operand
IN2	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand1 operand is equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 1908)

NE_ElemType: Compare data type of an ARRAY element for UNEQUAL with the data type of a tag

Description

You can use the "Compare data type of an ARRAY element for UNEQUAL with the data type of a tag" instruction to query which data type a tag does not have to which a VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Not equal to".

The tag at the IN1 parameter must be VARIANT data type. The tag at the IN2 parameter may be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

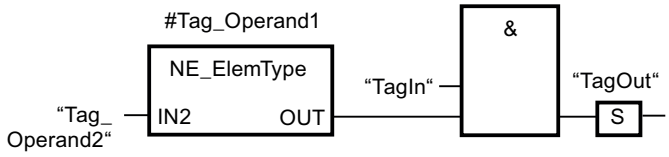
The following table lists the parameters of the "Compare data type of an ARRAY for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	I, Q, M, L	First operand
IN2	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand1 operand is not equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 1908)

IS_NULL: Check for EQUALS NULL pointer

Description

You can use the instruction "Check for EQUALS NULL pointer" to query whether the VARIANT points to a NULL pointer and therefore does not point to an object.

<Operand> must have the VARIANT data type.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "0" even if the ANY pointer is NULL.

Parameters

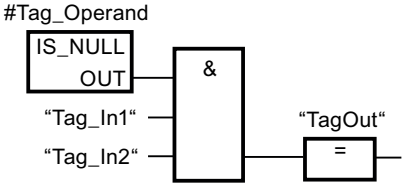
The following table shows the parameters of the "Check for EQUALS NULL pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	I, Q, M, L	Operand that is compared for EQUALS NULL

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "Tag_In1" and "Tag_In2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand operand does not point to an object.

See also

Overview of the valid data types (Page 1908)

NOT_NULL: Check for UNEQUALS NULL pointer

Description

You can use the instruction "Check for UNEQUALS NULL pointer" to query whether the VARIANT does not point to a NULL pointer and therefore points to an object.

<Operand> must have the VARIANT data type.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "1" even if the ANY pointer is NULL.

Parameters

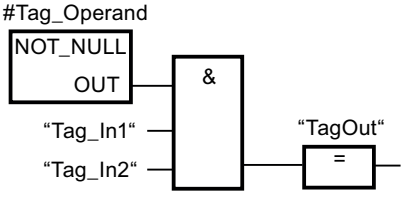
The following table shows the parameters of the "Check for UNEQUALS NULL pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	I, Q, M, L	Operand that is compared for UNEQUALS NULL

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "Tag_In1" and "Tag_In2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand operand points to an object.

See also

Overview of the valid data types (Page 1908)

IS_ARRAY: Check for ARRAY

Description

You can use the "Check for ARRAY" instruction to query whether the VARIANT points to a tag of the ARRAY data type.

<Operand> must have the VARIANT data type.

Parameters

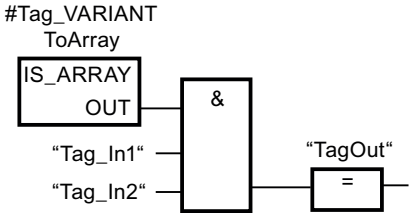
The following table shows the parameters of the "Check for ARRAY" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	I, Q, M, L	Operand that is queried for ARRAY

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "Tag_In1" and "Tag_In2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_VARIANTToArray" operand is of the ARRAY data type.

See also

Overview of the valid data types (Page 1908)

Math functions

CALCULATE: Calculate

Description

The "Calculate" instruction is used to define and execute an expression for the calculation of mathematical operations or complex logic operations depending on the selected data type.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. Depending on the selected data type, you can combine the functionality of specific instructions to execute a complex calculation. The expression to be calculated is specified via a dialog you can open via the "Calculator" icon at the top of the instruction box. The expression can contain the names of the input parameters and the syntax of the instructions. It is not permitted to specify operand names or operand addresses.

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box.

The values of the inputs are use to execute the specified expression. Not all defined inputs have to be used in the expression. The result of the instruction is transferred to the box output OUT.

Note

If one of the mathematical operations fails in the expression, then no result is transferred to the OUT output and the ENO enable output has the signal state "1".

If, in the expression, you use inputs that are not available in the box, these inputs are automatically inserted. Provided that there are no gaps in the numbering of the inputs that are to be newly defined in the expression. You cannot, for example, use the input IN4 in the expression if the input IN3 is not defined.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result or an interim result of the "Calculate" instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.
- An error occurred during the execution of one of the instructions specified in the expression.

The following table shows the instructions that, depending on the selected data type, can be combined and executed in the expression of the "Calculate" instruction:

Data type	Instruction	Syntax	Example
Bit strings	AND: AND logic operation	AND	IN1 AND IN2 OR IN3
	OR: OR logic operation	OR	
	XOR: EXCLUSIVE OR logic operation	XOR	
	INV: Create ones complement	NOT	
	SWAP: Swap [1)]	SWAP	
Integers	ADD: Add	+	(IN1 + IN2) * IN3; (ABS(IN2)) * (ABS(IN1))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	MOD: Return remainder of division	MOD	
	INV: Create ones complement	NOT	
	NEG: Create twos complement	-(in1)	
	ABS: Form absolute value	ABS( )	

Data type	Instruction	Syntax	Example
Floating-point numbers	ADD: Add	+	((SIN(IN2) * SIN(IN2) + (SIN(IN3) * SIN(IN3)) / IN3)); (SQR(SIN(IN2)) + (SQR(COS(IN3)) / IN2))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	EXPT: Exponentiate	**	
	ABS: Form absolute value	ABS( )	
	SQR: Form square	SQR( )	
	SQRT: Form square root	SQRT( )	
	LN: Form natural logarithm	LN( )	
	EXP: Form exponential value	EXP( )	
	FRAC: Return fraction	FRAC( )	
	SIN: Form sine value	SIN( )	
	COS: Form cosine value	COS( )	
	TAN: Form tangent value	TAN( )	
	ASIN: Form arcsine value	ASIN( )	
	ACOS: Form arccosine value	ACOS( )	
	ATAN: Form arctangent value	ATAN( )	
	NEG: Create twos complement	-(in1)	
	TRUNC: Truncate numerical value	TRUNC( )	
	ROUND: Round numerical value	ROUND( )	
	CEIL: Generate next higher integer from floating-point number	CEIL( )	
	FLOOR: Generate next lower integer from floating-point number	FLOOR( )	

[1] Not possible for data type BYTE.

Parameters

The following table shows the parameters of the instruction "Calculate":

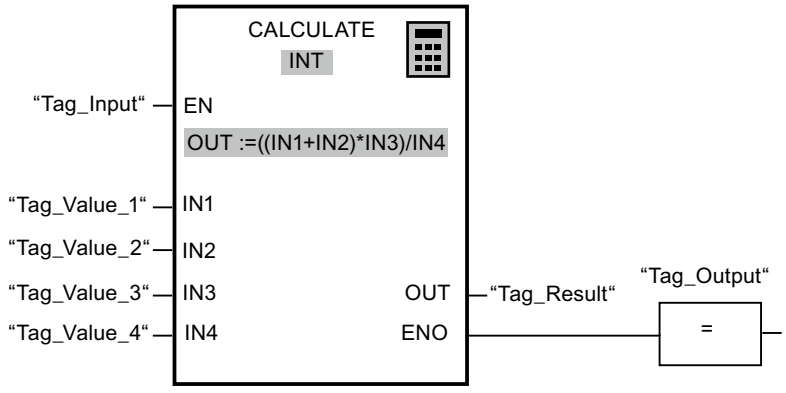
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First available input
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second available input

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
INn	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Additionally inserted inputs
OUT	Output	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Output to which the end result is to be transferred.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	Tag_Value_1	4
IN2	Tag_Value_2	4
IN3	Tag_Value_3	3
IN4	Tag_Value_4	2
OUT	Tag_Result	12

If the "Tag_Input" has the signal state "1", the instruction is executed. The value of operand "Tag_Value_1" is added to the value of operand "Tag_Value_2". The sum is multiplied with the value of the operand "Tag_Value_3". The product is divided by the value of the operand "Tag_Value_4". The quotient is transferred as end result to the operand "Tag_Result" at the OUT output of the instruction. If no errors occur during the execution of the individual instructions, output ENO and the operand "Tag_Output" are set to "1".

See also

Overview of the valid data types (Page 1908)

Example of calculating an equation (Page 3877)

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 to the value at input IN2 and query the sum at output OUT (OUT := IN1+IN2).

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are added. The sum is stored at output OUT.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Add":

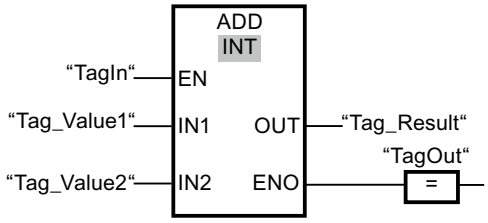
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second number to be added
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values, which are added.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Sum

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at output OUT (OUT := IN1-IN2).

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Subtract":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Minuend

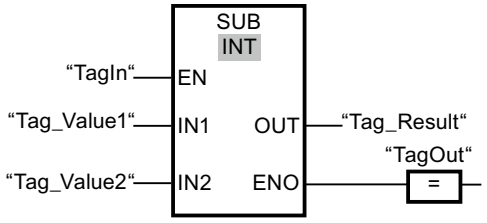
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Subtrahend
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Difference

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at input IN1 with the value at input IN2 and query the total at output OUT (OUT := IN1*IN2).

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are multiplied. The product is stored at the OUT output.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Multiply":

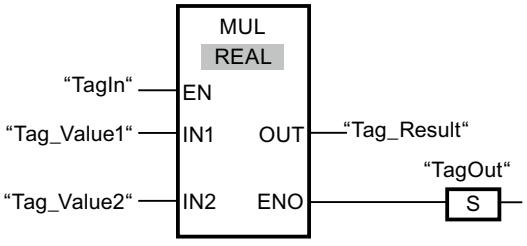
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for multiplication
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for multiplication
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values, which are multiplied.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Product

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is multiplied with the value of operand "Tag_Value2". The result of the multiplication is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT (OUT := IN1/IN2).

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Divide":

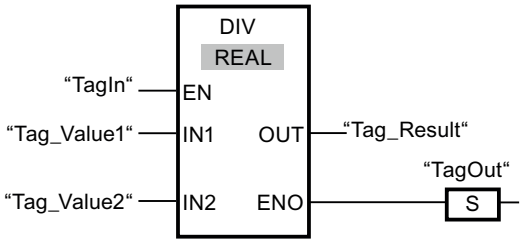
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Quotient value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The result of the division is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at input IN1 by the value at input IN2 and query the remainder of division at output OUT.

Parameters

The following table shows the parameters of the "Return remainder of division" instruction:

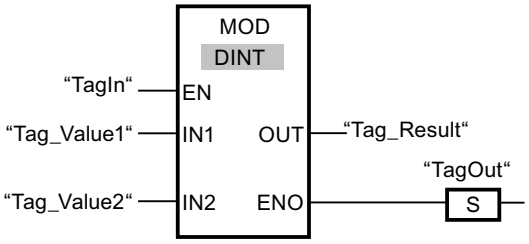
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers	I, Q, M, D, L, P	I, Q, M, D, L, P	Remainder of division

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder of division is stored in operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at input IN and query the result at output OUT. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

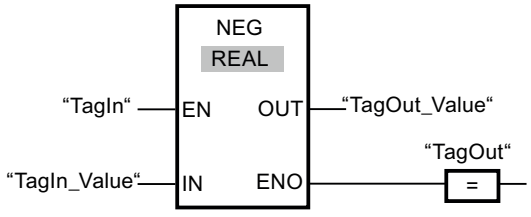
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is stored at output "TagOut_Value". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

INC: Increment

Description

You can use the "Increment" instruction to change the value of the operand at parameter IN/OUT to the next higher value and query the result.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Increment" instruction:

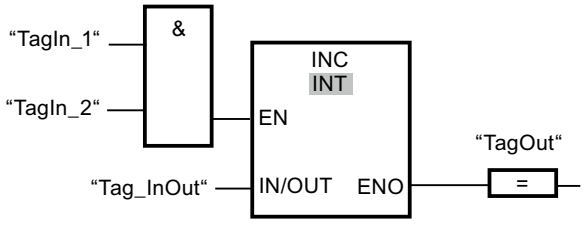
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	I, Q, M, D, L	Value to be incremented.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operands TagIn_1 and TagIn_2 have the signal state "1", the value of the operand "Tag_InOut" is incremented by one and the output "TagOut" is set.

See also

Overview of the valid data types (Page 1908)

DEC: Decrement

Description

You can use the "Decrement" instruction to change the value of the operand at parameter IN/OUT to the next lower value and query the result.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Decrement":

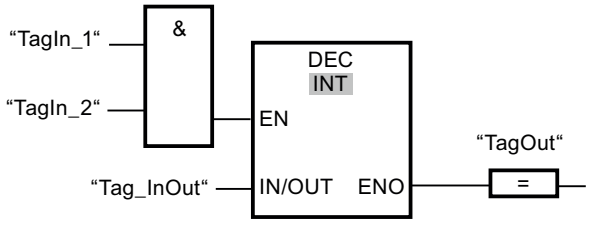
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	I, Q, M, D, L	Value to be decremented.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of the operand "Tag_InOut" is decremented by one and the output "TagOut" is set.

See also

Overview of the valid data types (Page 1908)

ABS: Form absolute value

Description

You can use the "Form absolute value" instruction to calculate the absolute value of the value specified at input IN. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Form absolute value":

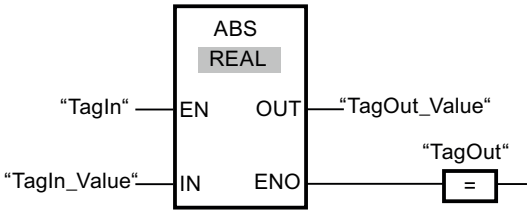
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Absolute value of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the available inputs and writes the lowest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

In its initial state the instruction contains at least two inputs (IN1 and IN2) and no more than 100 inputs.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

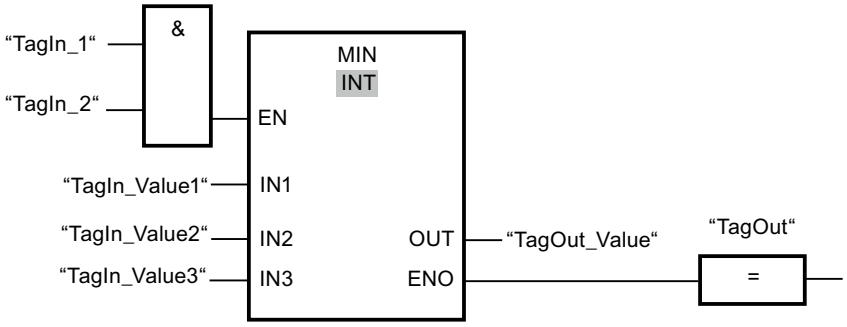
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Additionally inserted inputs whose values are to be compared
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result
If the IEC check is not enabled, you can also use tags of the data type TIME, LTIME, TOD, LTOD, DATE and LDT by selecting a bit string of the same length as the data type. (e.g. instead of TIME => DINT, UDINT or DWORD = 32 bits)					

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the available inputs and writes the highest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

In its initial state the instruction contains at least two inputs (IN1 and IN2) and no more than 100 inputs.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

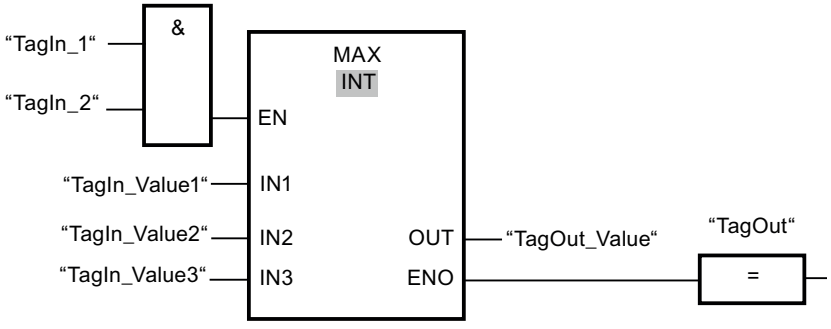
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Additionally inserted inputs whose values are to be compared.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result
If the IEC check is not enabled, you can also use tags of the data type TIME, LTIME, TOD, LTOD, DATE and LDT by selecting a bit string of the same length as the data type. (e.g. instead of TIME => DINT, UDINT or DWORD = 32 bits)					

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the MN condition <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN, output OUT is set to the value of the input MN. If the high limit MX is exceeded, output OUT is set to the value of the input MX.

If the value at input MN is greater than that at input MX, the result is undefined and the enable output ENO is "0".

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN input is greater than the value at the MX input.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
MN	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Low limit

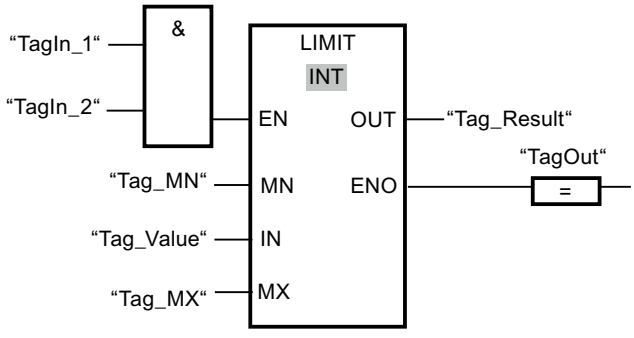
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
MX	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	High limit
OUT	Output	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.						

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value of the operand "Tag_Value" is less than the low limit, the value of the operand "Tag_MN" is copied to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SQR: Form square

Description

The instruction "Form square" is used to square the value of a floating-point number at input IN and to write the result to output OUT.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form square":

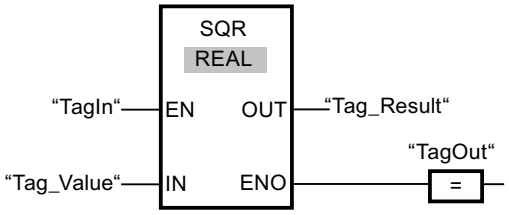
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Square of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction squares the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SQRT: Form square root

Description

The instruction "Form square root" is used form the square root from the value of the floating-point number at input IN and to write the result to output OUT. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number. If the value at input IN is "0", the result is also "0".

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form square root":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

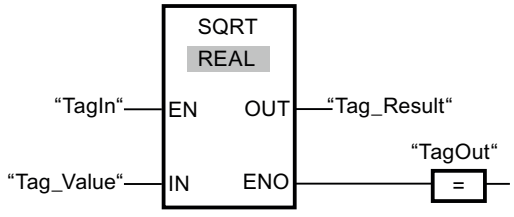
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	I, Q, M, D, L	Square root of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the square root of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) of the value at input IN. The result is sent to output OUT and can be queried there. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form natural logarithm":

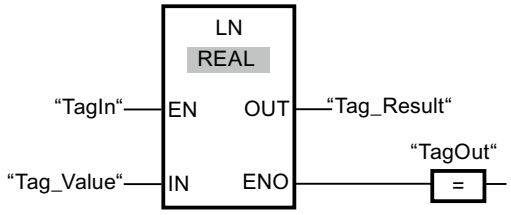
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Natural logarithm of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction forms the natural logarithm of the value at input "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

EXP: Form exponential value

Description

You can use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value specified at input IN. The result is provided at output OUT and can be queried there (OUT = e^{IN}).

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form exponential value":

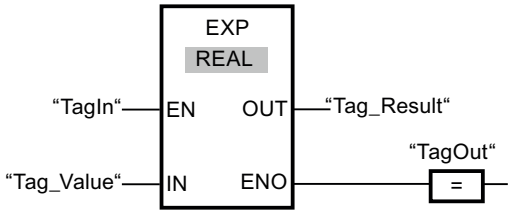
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Exponential value of input value IN

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction forms the exponent from base e and the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SIN: Form sine value

Description

You can use the "Form sine value" instruction to calculate the sine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form sine value":

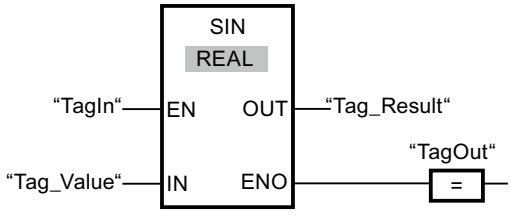
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Size of the angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Sine of the specified angle

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the sine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

COS: Form cosine value

Description

You can use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form cosine value":

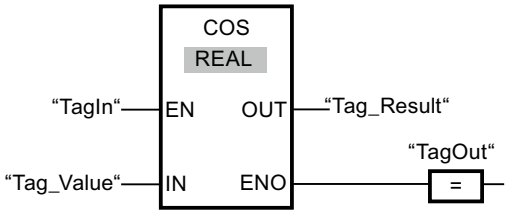
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Size of the angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Cosine of the specified angle

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the cosine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

TAN: Form tangent value

Description

You can use the "Form tangent value" instruction to calculate the tangent of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form tangent value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

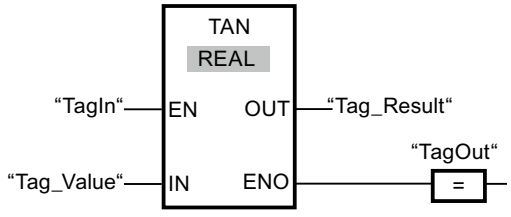
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Tangent of the specified angle

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

If the operand "TagIn" has the signal state "1", the "Form tangent value" instruction is executed. The instruction calculates the tangent of the angle specified at input "Tag_Value" and stores the result at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 1908)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from the sine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

The following table shows the parameters of the instruction "Form arcsine value":

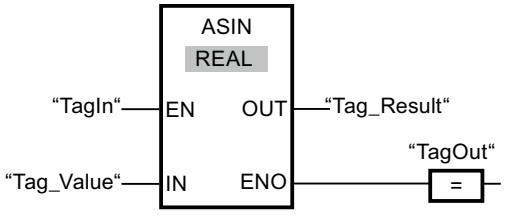
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Size of the angle in radians

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from the cosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is output in radians at output OUT and can range in value from 0 to +π.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

The following table shows the parameters of the instruction "Form arccosine value":

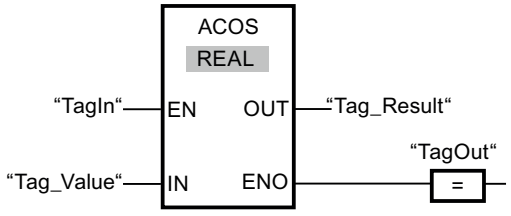
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Size of the angle in radians

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from the tangent value specified at input IN, which corresponds to this value. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) at input IN. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form arctangent value":

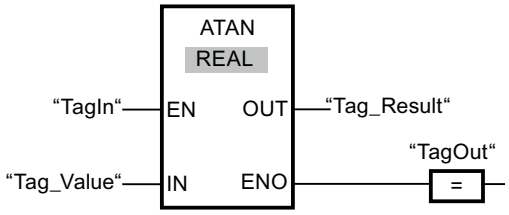
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Tangent value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Size of the angle in radians

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398 (π/4)

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Invalid floating-point numbers (Page 1927)

FRAC: Return fraction

Description

You can use the "Return fraction" instruction to determine the decimal places of the value at the IN input. The result of the query is stored at output OUT and can be queried there. If the input IN has e.g. the value 123.4567, the output OUT has the value 0.4567.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors occur during the execution of the instruction, for example there is no valid floating-point number at the input.

Parameters

The following table shows the parameters of the "Return fraction" instruction:

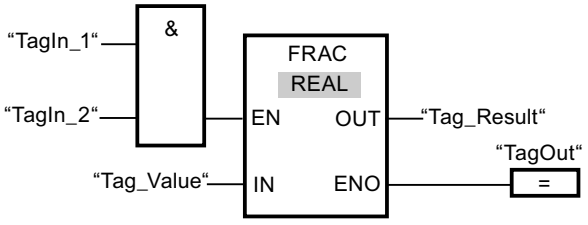
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value, whose decimal places are to be determined.
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Decimal places of the input value at input IN

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	2.555
OUT	Tag_Result	0.555

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is started. The decimal places from the value at the operand "Tag_Value" are copied to the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

EXPT: Exponentiate

Description

You can use the "Exponentiate" instruction to raise the value at the input IN1 by a power specified with the value at input IN2. The result of the instruction is stored at output OUT and can be queried there (OUT = $IN1^{IN2}$).

The value at input IN1 must be a valid floating-point number. Integers are also allowed for setting the input IN2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors occur during the instruction processing, for example, if there is an overflow.

Parameters

The following table shows the parameters of the instruction "Exponentiate":

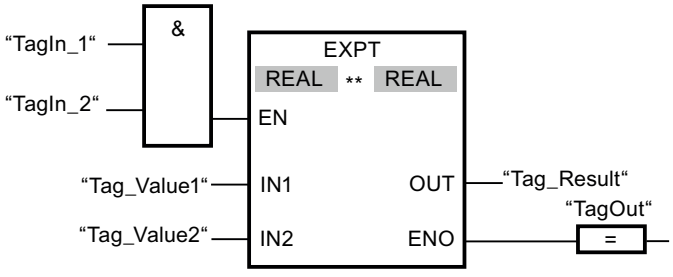
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Base value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value with which the base value is exponentiated
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Exponentiate" instruction is executed. The value of operand "Tag_Value1" is raised by the power of the value of the operand "Tag_Value2". The result is stored at output "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of the higher address.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data type at the IN parameter does not correspond to the specified data type at the OUT1 parameter.

The following table lists the possible transfers for the S7-1200 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE ,TOD, CHAR
WORD	WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
USINT	USINT, UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
INT	INT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UINT	UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UDINT	UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
LREAL	LREAL	LREAL
TIME	TIME	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME
DATE	DATE	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, DATE
TOD	TOD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, CHAR, Character of a string[1)]
WCHAR	WCHAR	BYTE, WORD, DWORD, CHAR, WCHAR, character of a character string[1)]
Character of a string[1)]	Character of a string	CHAR, WCHAR, character of a string
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER

The following table lists the possible transfers for the S7-1500 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE ,TOD, LTOD, CHAR
WORD	WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, S5TIME, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
DWORD	DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
LWORD	LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LREAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
SINT	SINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
USINT	USINT, UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
INT	INT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UINT	UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD. LTOD
DINT	DINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UDINT	UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
LINT	LINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
ULINT	ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LWORD, LREAL
S5TIME	S5TIME	WORD, S5TIME
TIME	TIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME
LTIME	LTIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTIME
DATE	DATE	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, DATE
DT	DT	DT
LDT	LDT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LDT
TOD	TOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TOD
LTOD	LTOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, LWORD, CHAR, Character of a string[1)]
WCHAR	WCHAR	BYTE, WORD, DWORD, LWORD, CHAR, WCHAR, Character of a string[1)]
Character of a string[1)]	Character of a string	CHAR, WCHAR, character of a string

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
COUNTER	COUNTER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	TIMER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_LTIMER	IEC_LTIMER	IEC_LTIMER
IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNT-ER	IEC_LCOUNTER	IEC_LCOUNTER
IEC_UL-COUNTER	IEC_ULCOUNTER	IEC_ULCOUNTER

[1)] You can also use the "Move value" instruction to transfer individual characters of a string to operands of the CHAR or WCHAR data type. The number of the character to be transferred is specified in square brackets next to the operand name. "MyString[2]", for example, transfers the second character of the "MyString" string. The transfer of the operand of the data type CHAR or WCHAR to the individual character of a string is also possible. You can also replace a specific character of a string with the character of another string.

[2)] Transferring entire arrays (ARRAY) is possible only when the array components of the operands at input IN and at output OUT1 are of the same data type.

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at input IN is less than the bit length of the data type at output OUT1, the higher-order bits of the destination value will be overwritten with zeros.

In its initial state, the instruction box contains 1 output (OUT1). The number of outputs can be extended. The added outputs are numbered in ascending order on the box. During the execution of the instruction, the content of the operand at the input IN is transferred to all available outputs. The instruction box cannot be extended if structured data types (DTL, STRUCT, ARRAY) or characters of a string are transferred.

You can also use the "Move block" (MOVE_BLK) and "Move block uninterruptible" (UMOVE_BLK) instructions to move operands of the ARRAY data type. Operand of the data type STRING or WSTRING can be copied with the instruction "Move character string" (S_MOVE).

Parameters

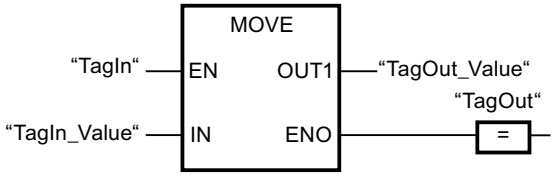
The following table shows the parameters of the "Move value" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, timers, date and time, CHAR, WCHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, timers, date and time, CHAR; WCHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data type, PLC data type (UDT)	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Element used to overwrite the destination address.
OUT1	Output	Bit strings, integers, floating-point numbers, timers, date and time, CHAR, WCHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, timers, date and time, CHAR; WCHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data type, PLC data type (UDT)	I, Q, M, D, L	I, Q, M, D, L	Destination address

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies the content of the "TagIn_Value" operand to the "TagOut_Value" operand. If the instruction is executed without errors, the ENO and "TagOut" enable outputs are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

MOVE_BLK: Move block (Page 2621)

UMOVE_BLK: Move block uninterruptible (Page 2627)

S_MOVE: Move character string (Page 3001)

Deserialize: Deserialize

Description

You can use the "Deserialize" instruction to convert the sequential representation of a PLC data type (UDT) back to a PLC data type and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type is located must have the ARRAY of BYTE data type and be declared with standard access. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion.

The instruction enables you to convert multiple sequential representations of converted PLC data types back to their original state step-by-step.

If you only want to convert back a single sequential representation of a PLC data type (UDT), you can also use the instruction "TRCV: Receive data via communication connection".

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", the bit has a length of 1 byte.

Parameters

The following table lists the parameters of the "Deserialize" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
SRC_ARRAY	Input	VARIANT	I, Q, M, L	Global data block in which the generated data stream is stored
DEST_VARIABLE	InOut	VARIANT	I, Q, M, L	Tag in which the returned converted PLC data type (UDT) is stored
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

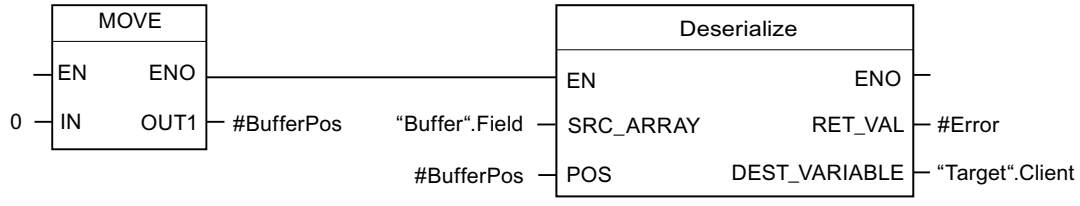
The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The data block at the DEST_VARIABLE parameter is not a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains no value.
8151	Code generation error at the SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8250	The VARIANT data type at the DEST_VARIABLE parameter contains no value.
8251	Code generation error at the DEST_VARIABLE parameter
8254	Invalid data type at the DEST_VARIABLE parameter
8382	The value at parameter POS is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

The following example shows how the instruction works:

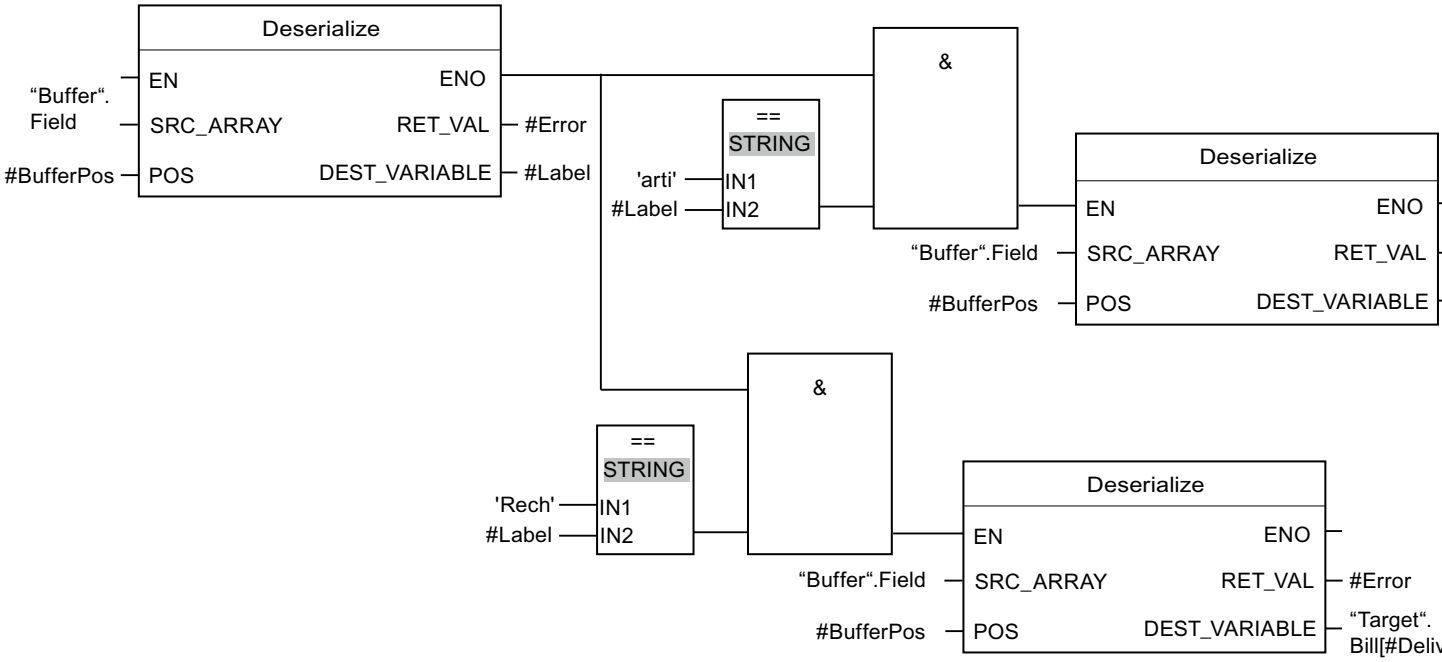
Network 1:



The "Move value" instruction moves the value "0" to the "#BufferPos" operand. The "Deserialize" instruction deserializes the sequential representation of the customer data from

the "Buffer" data block and writes it to the "Target" data block. The number of bytes used by the converted customer data is stored in the "#BufferPos" operand.

Network 2:



The "Deserialize" instruction deserializes the sequential representation of the separator sheet, which was saved with the sequential representation after the customer data, from the "Buffer" data block and writes the characters to the "#Label" operand. The characters are compared using comparison instructions "arti" and "Bill". If the comparison for "arti" = TRUE, these are article data that have been deserialized and written to the "Target" data block. If the comparison for "Bill" = TRUE, these are billing data that have been deserialized and written to the "Target" data block.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT

Name of the PLC data types	Name	Data type
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 1908)

PLC data types (Page 1954)

Serialize: Serialize

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT) to a sequential representation without losing parts of their structure.

You can use the instruction to cache multiple structured data from your program in a buffer, which is preferably in a global data block, and send it to another CPU. The memory area in which the converted PLC data types are stored must have the ARRAY of BYTE data type and be declared with standard access. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion.

The operand at the POS parameter contains information about the number of bytes used by the converted PLC data types.

If you want to send a single PLC data type (UDT), you can directly call the instruction "TSEND: Send data via communication connection".

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", the bit has a length of 1 byte.

Parameters

The following table lists the parameters of the instruction "Serialize":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_VARIABLE	Input	VARIANT	I, Q, M, L	PLC data type (UDT) that is converted to sequential representation.
DEST_ARRAY	InOut	VARIANT	I, Q, M, L	Data block in which the generated data stream is stored.
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

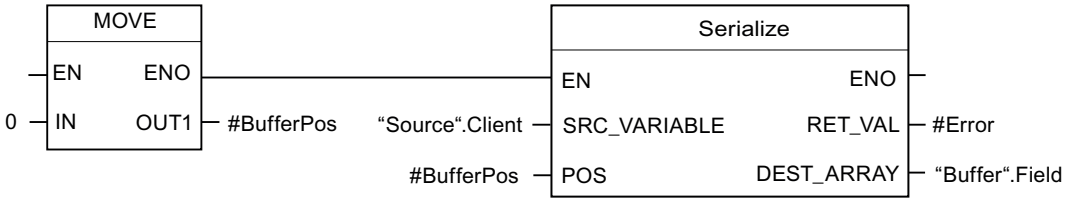
The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains no value.
8152	Code generation error at the SRC_VARIABLE parameter
8236	The data block at the DEST_ARRAY parameter is not a block with standard access.
8250	The VARIANT data type at the DEST_ARRAY parameter contains no value.
8252	Code generation error at the DEST_ARRAY parameter
8253	There is not enough free memory available at the DEST_ARRAY parameter.
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at parameter POS is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

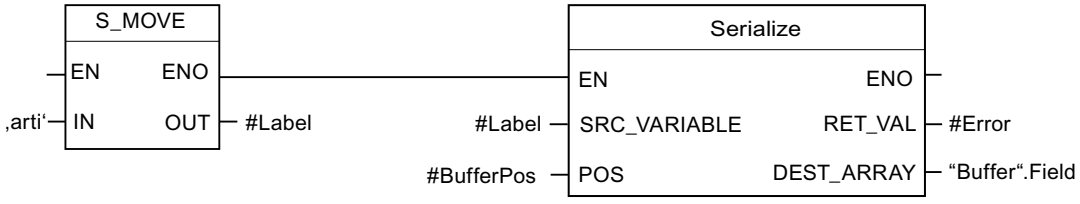
The following example shows how the instruction works:

Network 1:



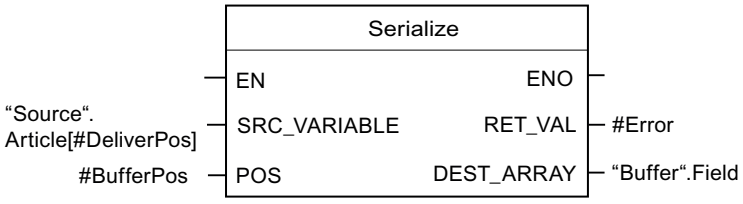
The "Move value" instruction moves the value "0" to the "#BufferPos" operand. The "Serialize" instruction serializes the customer data from the "Source" data block and writes it in sequential representation to the "Buffer" data block. The number of bytes used by the sequential representation is stored in the "#BufferPos" operand.

Network 2:



A kind of separator sheet is inserted now to make it easier to deserialize the sequential representation later. The "Move string" instruction moves the "arti" characters to the "#Label" operand. The "Serialize" instruction writes these characters after the customer data to the "Buffer" data block. The number of bytes that the characters need is added to the number already stored in the "#BufferPos" operand.

Network 3:



The "Serialize" instruction serializes the data of a specific article, which is calculated in runtime, from the "Source" data block and writes it in sequential representation to the "Buffer" data block after the "arti" characters.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 1908)

PLC data types (Page 1954)

MOVE_BLK: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be moved to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source area and the destination area have the same data type.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Parameters

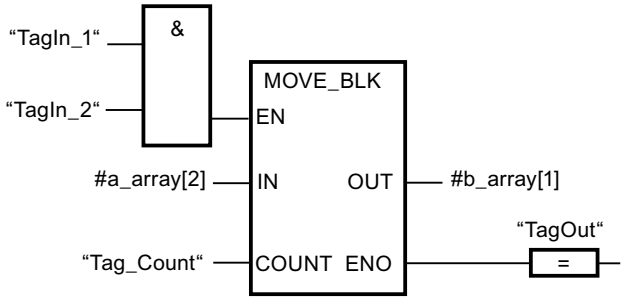
The following table lists the parameters of the "Move block" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.						

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	a_array[2]	The data type of the a_array operand is ARRAY [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is ARRAY [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MOVE_BLK_VARIANT: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). You can copy a complete ARRAY or elements of an ARRAY to another ARRAY of the same data type. The size (number of elements) of source and destination ARRAY may be different. You can copy multiple or single elements within an ARRAY.

When you use the instruction at the time to the block is created, the ARRAY does not yet have to be known, as the source and the designation are transferred per VARIANT.

Counting at the parameters SRC_INDEX and DEST_INDEX always begins with the low limit "0", regardless of the later declaration of the ARRAY.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is copied than is made available.

Note

VARIANT in connection with BOOL data type

If you want to interconnect a parameter of data type VARIANT (source or destination area) with a tag of data type BOOL or an ARRAY of BOOL, you have the following options:

1. You can address it symbolically
   Example: Parameter SRC: "Data_block".myArray
2. You can address it with help of ANY pointer. However, you must note that the specified length of the area must be dividable by 8, or the instruction will otherwise not be executed.
   Example: Parameter SRC: P#DB123.DBX456.0 BOOL 1000

Parameters

The following table lists the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	VARIANT (which points to an ARRAY or an individual ARRAY element), ARRAY of <Data_type>	I, Q, M, L	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L or constant	Number of elements which are copied Assign the value "1" to the parameter COUNT, if no ARRAY is specified at parameter SRC or at Parameter DEST.

Parameters	Declaration	Data type	Memory area	Description
SRC_INDEX	Input	DINT	I, Q, M, D, L or constant	• The SRC_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter SRC, the integer at parameter SRC_INDEX specifies the first element within the source area from which it is to be copied. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter SRC or only one single element of an ARRAY is specified, then assign the value "0" at parameter SRC_INDEX.
DEST_INDEX	Input	DINT	I, Q, M, D, L or constant	• The DEST_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter DEST, the integer at parameter DEST_INDEX specifies the first element within the destination area to it is to be copied. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter DEST, then assign the value "0" at parameter DEST_INDEX.
DEST	Output [1)]	VARIANT	I, Q, M, L	Destination area into which the contents of the source block are copied.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
1) The DEST parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

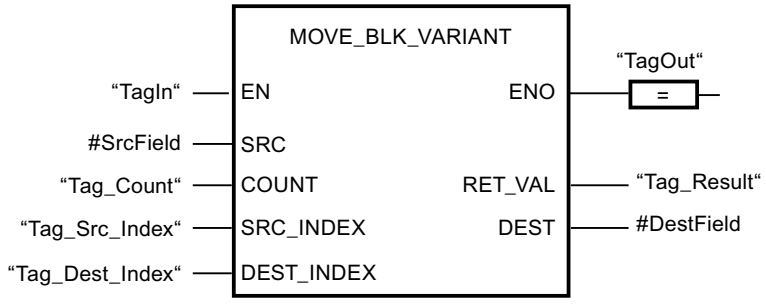
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at the SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value
8382	The value at the SRC_INDEX parameter is outside the limits of the VARIANT.
8383	The value at parameter SRC_INDEX is outside the high limit of the ARRAY.
8482	The value at the DEST_INDEX parameter is outside the limits of the VARIANT.
8483	The value at parameter DEST_INDEX is outside the high limit of the ARRAY.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at the DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Declaration in the block interface	Operand	Value
SRC	Input	#SrcField	The local operand #SrcField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[0..10] of "MOVE_UDT"
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	The local operand #DestField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[10..20] of "MOVE_UDT"

If the "TagIn" operand has signal state "1", the instruction is executed. Two elements are copied from the source area, beginning from the fourth element of the ARRAY of UDT, to the destination area. The copies are inserted in the ARRAY of UDT starting from the fourth element. If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

VariantGet: Read out VARIANT tag value (Page 2645)

Programming example: Moving data (Page 267)

UMOVE_BLK: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be moved to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

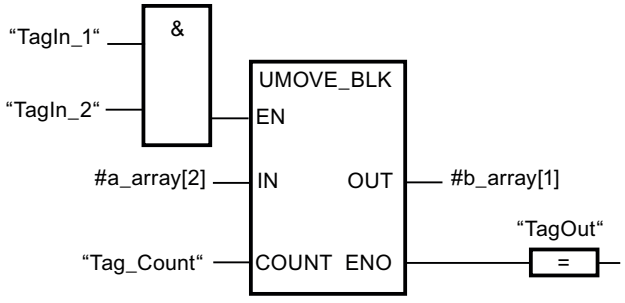
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN [1)]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Number of elements to be copied from the source area to the destination area

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an array structure.						

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	a_array[2]	The data type of the a_array operand is ARRAY [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is ARRAY [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The move operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the value of the IN input. The destination area is filled beginning with the address specified at the OUT output. The number of repeated move operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is selected and moved to the destination area as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Parameters

The following table shows the parameters of the "Fill block" instruction:

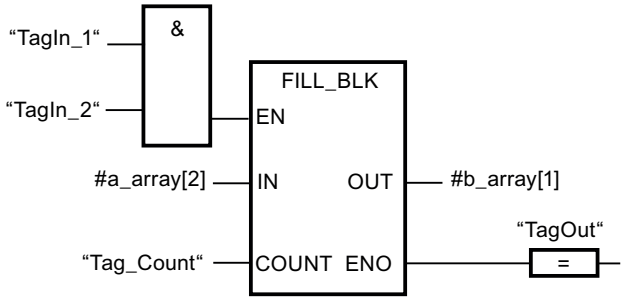
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN [1)]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Number of repeated move operations

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
OUT [2]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	Address in destination area from which filling starts.
[1] The specified data types can also be used as elements of an Array structure. [2] The specified data types can only be used as elements of an Array structure.						

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	a_array[2]	The data type of the a_array operand is ARRAY [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is ARRAY [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. If no errors occur during the execution of the instruction, the outputs ENO and "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

UFILL_BLK: Fill block uninterruptible

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (destination area) with the value of the IN input without interruption. The destination area is filled beginning with the address specified at the OUT output. The number of repeated move operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is selected and moved to the destination area as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

You can use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

The following table shows the parameters of the "Fill block uninterruptible" instruction:

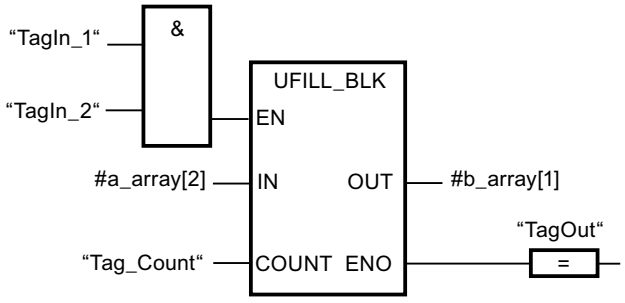
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN [1)]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Element used to fill the destination area

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Number of repeated move operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	Address in destination area from which filling starts.
[1] The specified data types can also be used as elements of an Array structure. [2] The specified data types can only be used as elements of an Array structure.						

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	a_array[2]	The data type of the a_array operand is ARRAY [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is ARRAY [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. The move operation cannot be interrupted by other operating system activities. If

no errors occur during the execution of the instruction, the outputs ENO and "TagOut" are set to signal state "1".

See also

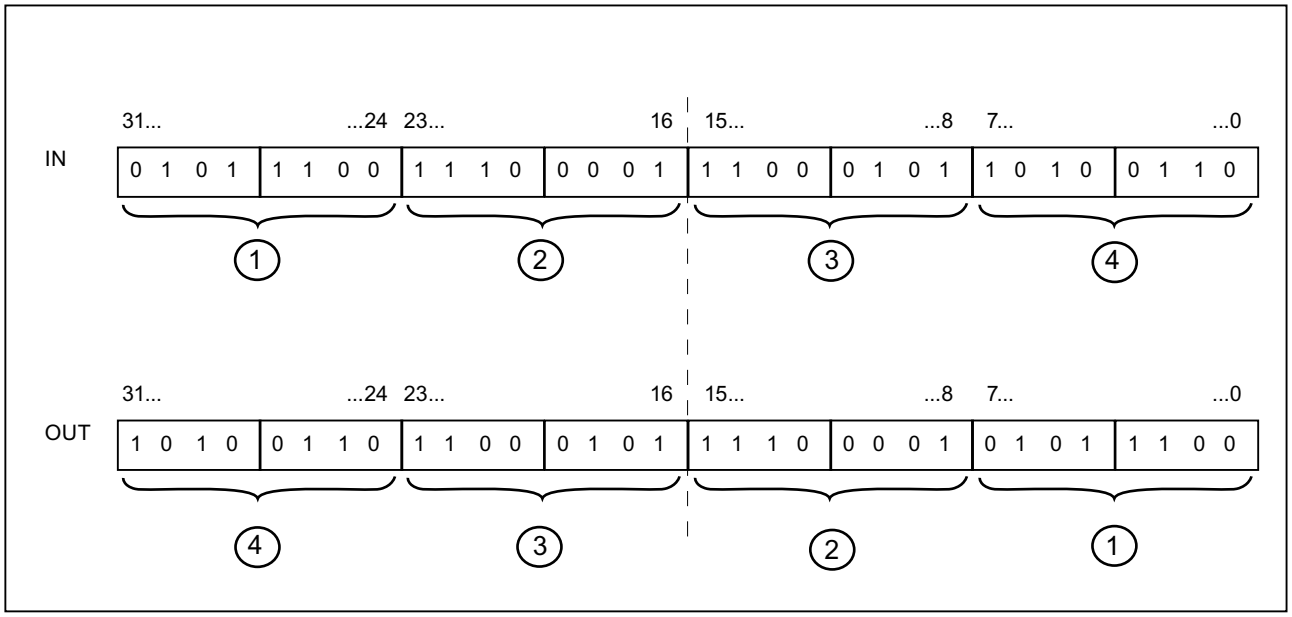
Overview of the valid data types (Page 1908)

SWAP: Swap

Description

You can use the instruction "Swap" to change the order of the bytes within the tag at input IN and query the result at output OUT.

The following figure shows how the bytes of a DWORD data type operand are swapped using the "Swap" instruction:



Parameters

The following table shows the parameters of the "Swap" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

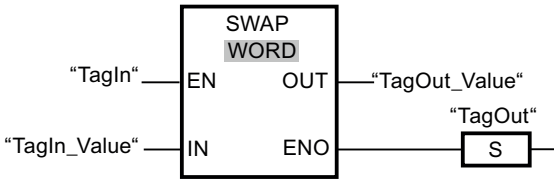
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Operand whose bytes are swapped.
OUT	Output	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 0101 0101
OUT	TagOut_Value	0101 0101 0000 1111

If the "TagIn" operand has signal state "1", the instruction is executed. The arrangement of the bytes is changed and stored in the operand "TagOut_Value".

See also

Overview of the valid data types (Page 1908)

ARRAY DB

ReadFromArrayDB: Read from array data block

Description

The "Read from array data block" instruction is used to read data from an ARRAY data block and write it to a destination area.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be PLC data type or any other elementary data type. The counting

at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Read from array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P, or constant	Element that is read
VALUE	Output 1)	VARIANT	I, Q, M, L	Value that is read and output
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
1) The VALUE parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

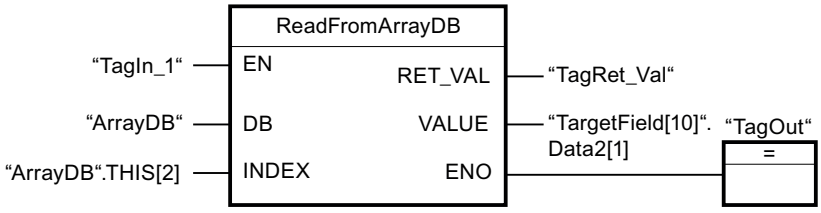
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, write protected, or is located in load memory.
8135	The ARRAY data block contains invalid values.
8154	The data block has the incorrect data type.
8282	The value at parameter INDEX is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8452	Code generation error
8453	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	TargetField[10].Data2[1]	The data type of the "Target-Field" operand is Array [10 to 20] of "MOVE_UDT".

If the "TagIn1" operand has the signal state "1", the "Read from the array data block" instruction is executed. The third element is read from "ArrayDB" and written to the "TargetField[10].Data2[1]" operand. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Using ARRAY data blocks (Page 239)

WriteToArrayDB: Write to array data block

Description

The instruction "Write to array data block" is used to write data to an ARRAY data block.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be PLC data type or any other elementary data type. The counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Write to array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P, or constant	Element in the DB to which data is written
VALUE	Input	VARIANT	I, Q, M, L	Value to be written
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

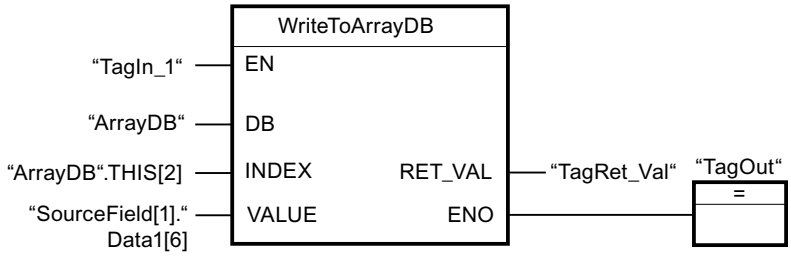
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, or is located in load memory.
8134	The data block is write protected.
8135	The data block is not an ARRAY data block.
8154	The data block has the incorrect data type.
8282	The value at parameter INDEX is outside the limits of the ARRAY.
8350	The data type VARIANT at parameter VALUE provides the value "0".
8352	Code generation error
8353	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceField[1].Data1[6]	The data type of the "Source-Field" operand is Array [0 to 10] of "MOVE_UDT".

If the "TagIn1" operand has the signal state "1", the "Write to ARRAY data block" instruction is executed. From the "SourceField" operand, the "Data1[6]" element from the second element is written to the "ArrayDB". The third element is written to the "ArrayDB". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Using ARRAY data blocks (Page 239)

ReadFromArrayDBL: Read from array data block in load memory

Description

The instruction "Read from array data block in load memory" is used to read data from an ARRAY data block in the load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be PLC data type or any other elementary data type. The counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1"

for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

Note

The ARRAY data block must be created with the "Optimized" block property.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table lists the parameters of the "Read from array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Begin with the reading of the array DB
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P, or constant	Element that is read
VALUE	InOut	VARIANT	I, Q, M, L	Value that is read and output No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

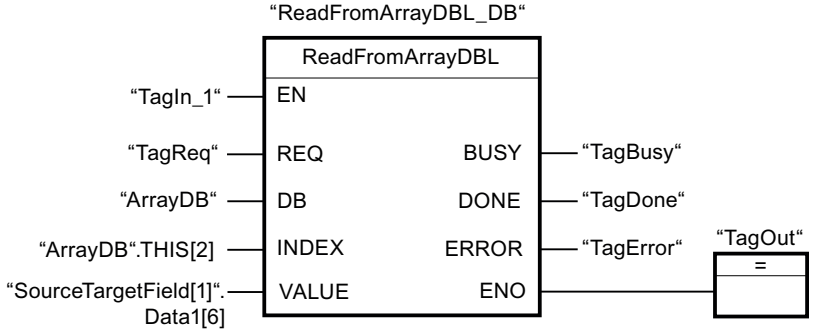
The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at parameter INDEX is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in the load memory" and "WRIT_DBL: Write to data block in the load memory" under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"

Parameters	Operand	Value
VALUE	SourceTargetField[1].Data1[6]	The data type of the "SourceTargetField" operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq"operand, the "Read from ARRAY data block in load memory" instruction is executed. The third element is read from "ArrayDB" and written to the "SourceTargetField[1].Data1[6]" operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 1908)

WRIT_DBL: Write to data block in the load memory (Page 3353)

READ_DBL: Read from data block in the load memory (Page 3351)

Using ARRAY data blocks (Page 239)

WriteToArrayDBL: Write to array data block in load memory

Description

The instruction "Write to array data block in load memory" is used to write data to an ARRAY data block in the load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be PLC data type or any other elementary data type. The counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate

data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

Note

The ARRAY data block must be created with the "Optimized" block property.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table lists the parameters of the "Write to array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Start writing to the array DB
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P, or constant	Element in the DB to which data is written
VALUE	Input	VARIANT	I, Q, M, L	Value to be written No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

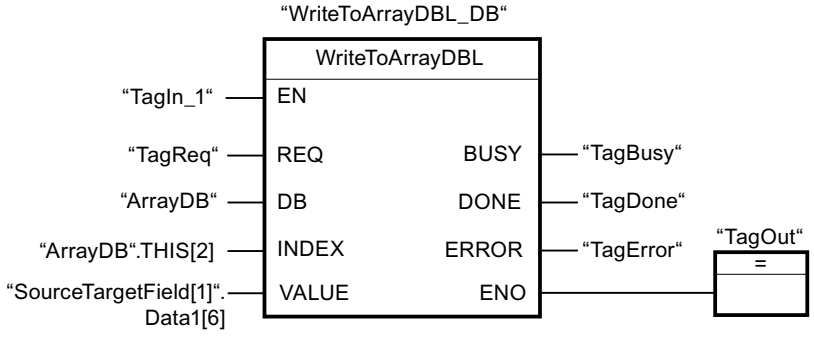
The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8234	The data block is write protected.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at parameter INDEX is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in the load memory" and "WRIT_DBL: Write to data block in the load memory" under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.

Parameters	Operand	Value
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the "SourceTargetField" operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq"operand, the "Write to ARRAY data block in load memory" instruction is executed. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written in the third element in the "ArrayDB". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 1908)

WRIT_DBL: Write to data block in the load memory (Page 3353)

READ_DBL: Read from data block in the load memory (Page 3351)

Using ARRAY data blocks (Page 239)

VARIANT

VariantGet: Read out VARIANT tag value

Description

You can use the "Read out VARIANT tag value" instruction to read the value of the tag to which the VARIANT at the SRC parameter points and write it in the tag at the DST parameter.

The SRC parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the DST parameter.

The data type of the tag at the DST parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction. For additional information, refer to "See also".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameters

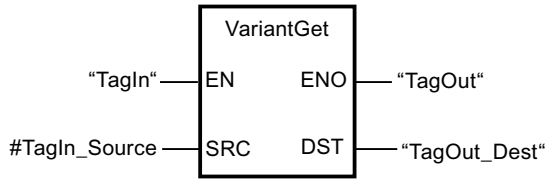
The following table lists the parameters of the instruction "Read out VARIANT tag value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	VARIANT	I, Q, M, L	Tag to be read
DST	Output	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The value of the tag to which the VARIANT at the "#TagIn_Source" operand points, is read and written to the "TagOut_Dest" operand.

See also

Overview of the valid data types (Page 1908)

VARIANT (Page 1951)

MOVE_BLK_VARIANT: Move block (Page 2623)

VariantPut: Write VARIANT tag value

Description

You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter to the tag at the DST parameter to which the VARIANT points.

The DST parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction. For additional information, refer to "See also".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameters

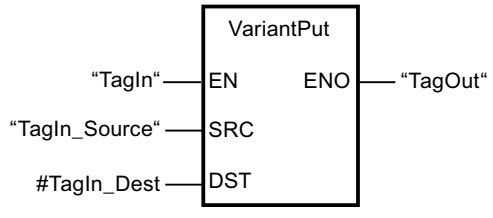
The following table lists the parameters of the instruction "Write VARIANT tag value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	Tag to be read
DST	Input	VARIANT	I, Q, M, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The value of the "TagIn_Source" operand is written to the tag to which the VARIANT at the #TagIn_Dest operand points.

See also

Overview of the valid data types (Page 1908)

VARIANT (Page 1951)

MOVE_BLK_VARIANT: Move block (Page 2623)

CountOfElements: Get number of ARRAY elements

Description

You can use the "Get number of ARRAY elements" instruction to query how many ARRAY elements a tag pointing to an VARIANT has.

If it is a one-dimensional ARRAY, the difference between the high and low limit +1 is output as a result. If it is a multi-dimensional ARRAY, the product of all dimensions is output as the result.

The IN parameter must have the VARIANT data type.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The VARIANT tag is not an ARRAY. (The result is "0".)

If the VARIANT points to an ARRAY of BOOL, the fill elements are also included in the count. (For example, 8 is returned for an ARRAY[0..1] of BOOL)

Parameters

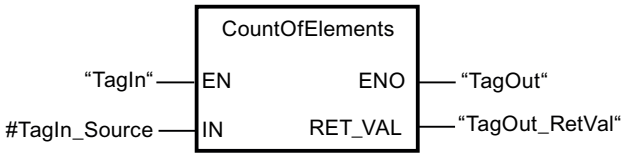
The following table shows the parameters of the "Get number of ARRAY elements" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	VARIANT	I, Q, M, L	Tag to be queried
RET_VAL	Output	UDINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has the signal state "1", the "Get number of ARRAY elements" instruction is executed. The number of ARRAY elements of the tag to which the VARIANT at the "#TagIn_Source" operand points is read and output at the "TagOut_RetVal" operand.

See also

Format of array (16-bit limits) (Page 1941)

Format of array (32-bit limits) (Page 1942)

Overview of the valid data types (Page 1908)

VARIANT (Page 1951)

Legacy

FieldRead: Read field

Description

You can use the "Read field" instruction to read out a specific component from the field specified in the MEMBER parameter and transfer its content to the tag in the VALUE parameter. You use the parameter INDEX to define the index of the field components that are to be read. At the parameter MEMBER you specify the first component of the field to be read.

The data types of the field component at parameter MEMBER, the index and the tags at parameter VALUE must correspond to the data type of the instruction "Read field" because implicit conversion is not possible.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The field component specified at the parameter INDEX is not defined in the field specified at the parameter MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Read field":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

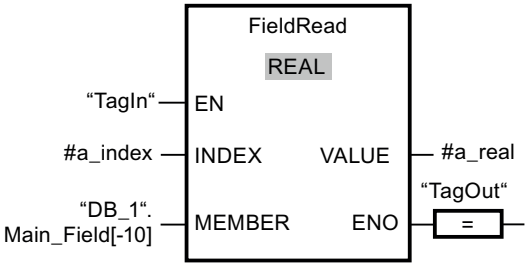
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
INDEX	Input	DINT	DINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Index of field components whose content is read out
MEMBER	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR and WCHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD, CHAR and WCHAR as components of an ARRAY tag	D, L	D, L	First component of the field from which will be read
VALUE	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD CHAR, WCHAR	I, Q, M, D, L, P	I, Q, M, D, L, P	Operand to which the contents of the field component are transferred.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Tag	Value
INDEX	a_index	4
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the data block "DB_1"
VALUE	a_real	Component with index 4 of the "Main_Field[-10..10] of REAL" field

The field component with index 4 is read from the "Main_Field[-10...10] of REAL" field and written to the "a_real" tag. The field component to be read is defined by the value at the parameter INDEX.

See also

Overview of the valid data types (Page 1908)

FieldWrite: Write field

Description

The "Write field" instruction is used to transfer the content of the tag at the VALUE input to a specific component of the field at the MEMBER output. You use the value at the INDEX input to specify the index of the field component that is described. At the MEMBER output, enter the first component of the field which is to be written to.

The data types of the field component at parameter MEMBER, the index and the tags at parameter VALUE must correspond to the data type of the instruction "Read field" because implicit conversion is not possible.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The field component specified at the input INDEX is not defined in the field specified at the output MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Write field":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

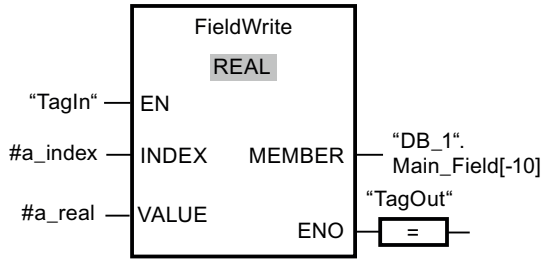
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
INDEX	Input	DINT	DINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Index of field component that is written with the content of VALUE
VALUE	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD CHAR, WCHAR	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Operand whose contents are copied
MEMBER	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR and WCHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD, CHAR and WCHAR as components of an ARRAY tag	D, L	D, L	First component of the field to which the content of VALUE is written.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
INDEX	a_index	4
VALUE	a_real	10.54
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the data block "DB_1"

The value "10.54" of the "a_real" tag is written to the field component with index 4 of the "Main_Field[-10 ... 10] of REAL" field. The index of the field component to which the content of the tag "a_real" is transferred is specified by the value at the input INDEX.

See also

Overview of the valid data types (Page 1908)

BLKMOV: Move block

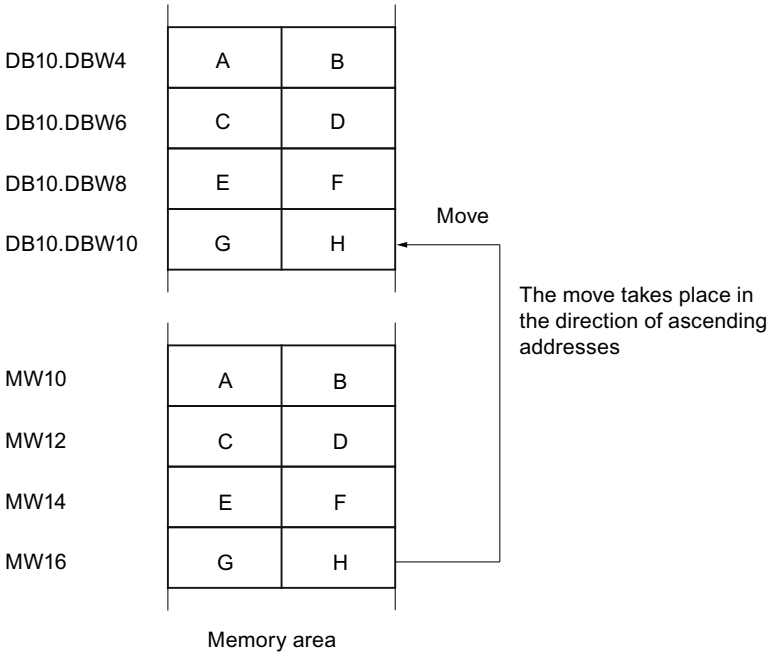
Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

The following figure shows the principle of the move operation:



Consistency of source and destination data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	I, Q, M, L, P	I, Q, M, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1)]	VARIANT	I, Q, M, L, P	I, Q, M, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.					

Parameter RET_VAL

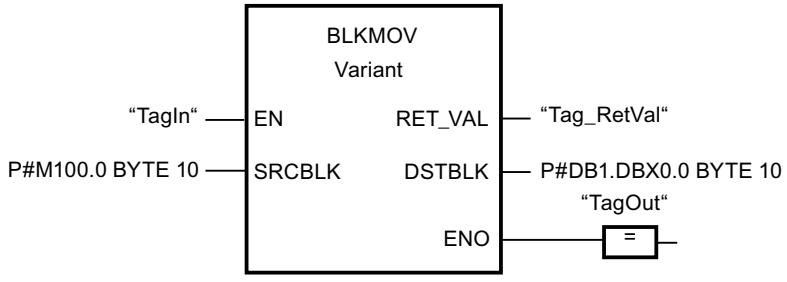
The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.

Error code* (W#16#...)	Explanation
8152	The WSTRING, WCHAR and BOOL data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR and BOOL data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2697)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

The move operation cannot be interrupted by other operating system activities. As a result the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	I, Q, M, L, P	I, Q, M, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
DSTBLK	Output [1]	VARIANT	I, Q, M, L, P	I, Q, M, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.					

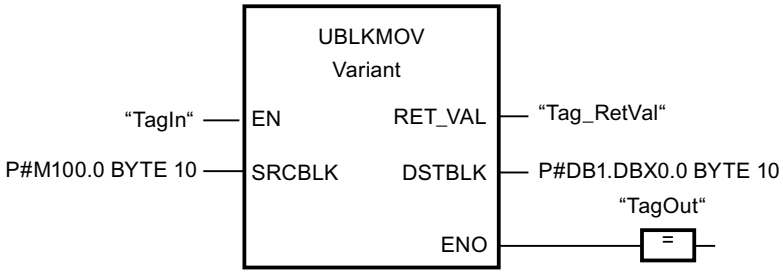
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
8091	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR and BOOL data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR and BOOL data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2697)

FILL: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

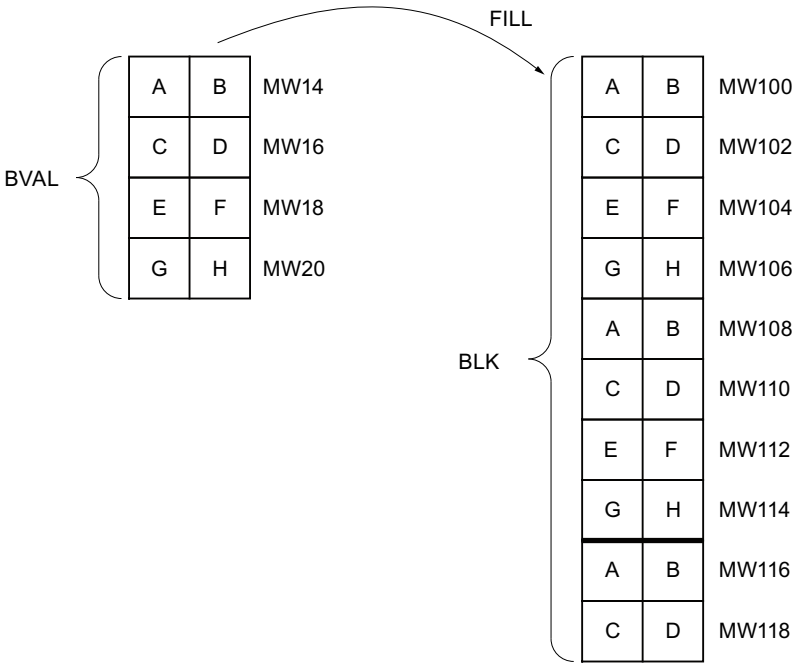
You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK instruction. Fill block" instruction.

The following figure shows the principle of the move operation:



Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is data type STRING, the instruction writes the entire string including the administration information.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
BVAL	Input	VARIANT	I, Q, M, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
BLK	Output 1)	VARIANT	I, Q, M, L, P	Specification of the memory area that will be filled with the content of the source area.
1) The BLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

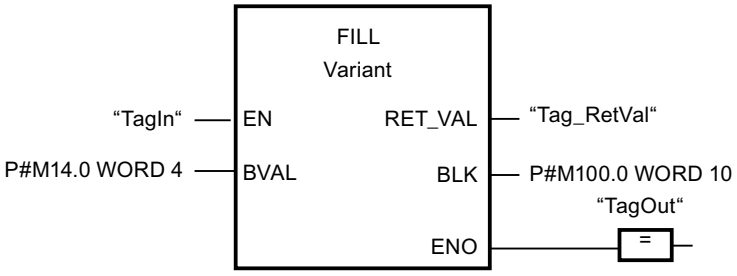
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR and BOOL data types are not supported at the BVAL parameter.
8352	The WSTRING, WCHAR and BOOL data types are not supported at the BLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

The following example shows how the instruction works:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2697)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types set in the instruction box. The converted value is output at the OUT output.

For information on possible conversions, refer to the "Explicit conversion" section at "See also".

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Conversion options for bit strings

The bit strings BYTE and WORD cannot be selected in the instruction box. It is, however, possible to specify an operand of the data type DWORD or LWORD for a parameter of the instruction if the length of the input and output operand match. The operand will be interpreted and implicitly converted from the data type of a bit string according to the data type of the input or output parameter. The data type DWORD is for example interpreted as DINT/UDINT and LWORD as LINT/ULINT. These conversion options are available even if the "IEC check" is enabled.

Note

For CPUs of the S7-1500 series: The data types DWORD and LWORD can only be converted from or to data type REAL or LREAL.

Parameters

The following table shows the parameters of the "Convert value" instruction:

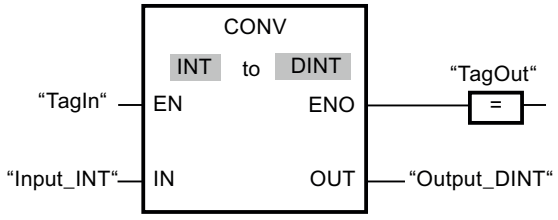
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value to be converted.
OUT	Output	Bit strings, integers, floating-point numbers, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the conversion

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

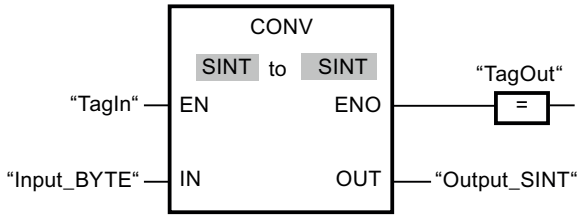
For additional information on valid data types, refer to "See also".

Example

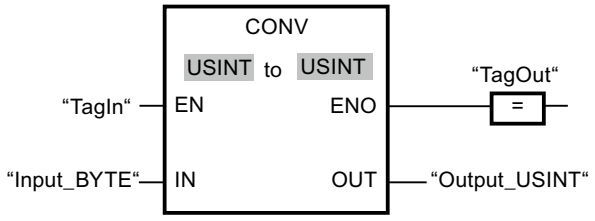
The following example shows conversion of an integer (16 bit) to another integer (32 bit):



The following example shows conversion of a byte (8 bit) to the integer SINT (8 bit):



The following example shows conversion of a byte (8 bit) to the unsigned integer USINT (8 bit):



The conversions are possible, because the length of the two operands is the same.

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to the nearest integer. If the input value is exactly between an even and odd number, then the even number is converted. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

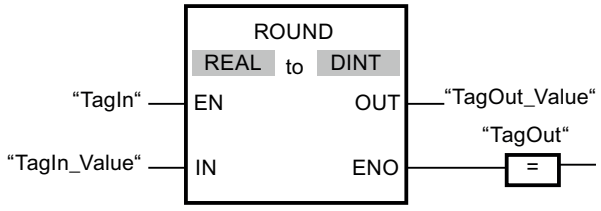
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value to be rounded.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the rounding

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Explicit conversion (Page 2113)

CEIL: Generate next higher integer from floating-point number

Description

You can use the "Generate next higher integer from floating-point number" instruction to round the value at input IN to the next higher integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next higher integer. The result of the instruction is output at the OUT output and can be queried there. The output value can be greater than or equal to the input value.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next higher integer from floating-point number":

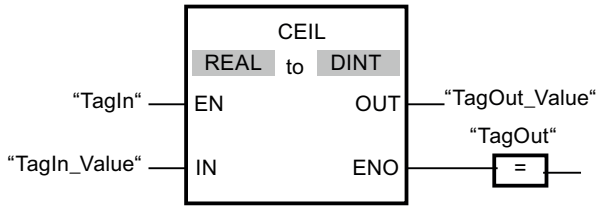
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result with the next higher integer

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next higher integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Explicit conversion (Page 2113)

FLOOR: Generate next lower integer from floating-point number

Description

You can use the "Generate next lower integer from floating-point number" instruction to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is provided at output OUT and can be queried there. The output value can be less than or equal to the input value.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next lower integer from floating-point number":

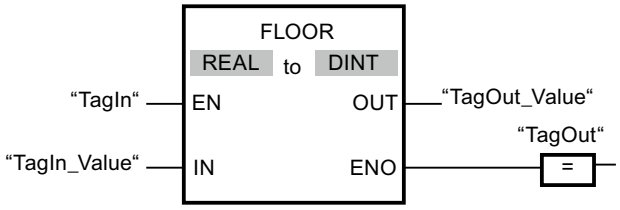
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result with the next lower integer

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and displayed at output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Explicit conversion (Page 2113)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at input IN. The value at input IN is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to output OUT without decimal places.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Truncate numerical value":

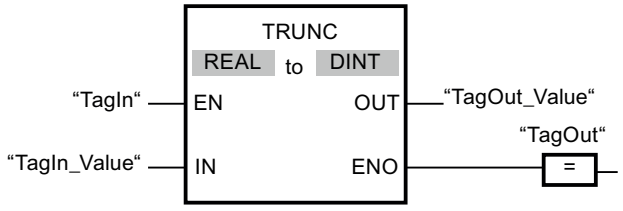
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result with integer part of the floating-point number

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The integer part of the floating-point number at input "TagIn_Value" is converted to an integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

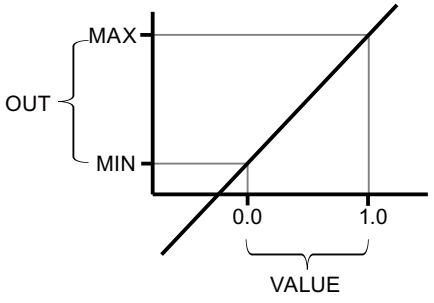
Explicit conversion (Page 2113)

SCALE_X: Scale

Description

You can use the "Scale" instruction to scale the value at the VALUE input by mapping it to a specified value range. When the instruction "Scale" is executed, the floating-point value at input VALUE is scaled to the value range, which is defined by the parameters MIN and MAX. The result of the scaling is an integer, which is stored at output OUT.

The following figure shows an example of how values can be scaled:



The "Scale" instruction works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input MIN is greater than or equal to the value at input MAX.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value at input VALUE is NaN (Not a number = result of an invalid arithmetic operation).

Note

For more information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	I, Q, M, D, L	Result of scaling

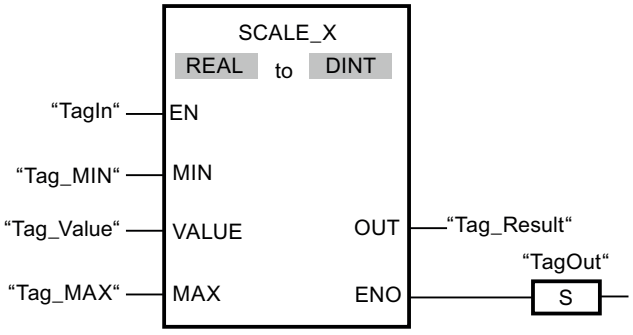
You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5

Parameters	Operand	Value
MAX	Tag_MAX	30
OUT	Tag_Result	20

If the "TagIn" operand has signal state "1", the instruction is executed. The value at input "Tag_Value" is scaled to the range of values defined by the values at inputs "Tag_MIN" and "Tag_MAX". The result is stored at output "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

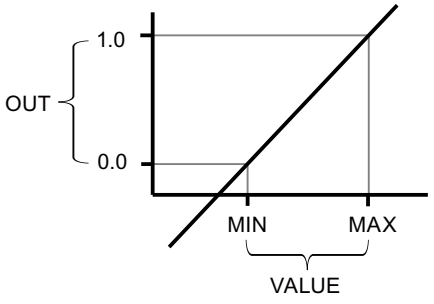
NORM_X: Normalize (Page 2672)

NORM_X: Normalize

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the parameters MIN and MAX to define the limits of a value range that is applied to the scale. Depending on the location of the normalized value in this value range, the result at output OUT is calculated and stored as a floating-point number. If the value to be normalized is equal to the value at input MIN, output OUT returns the value "0.0". If the value to be normalized is equal to the value at input MAX, output OUT returns the value "1.0".

The following figure shows an example of how values can be normalized:



The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input MIN is greater than or equal to the value at input MAX.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- The value at input VALUE is NaN (result of an invalid arithmetic operation).

Note

For more information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the instruction "Normalize":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
MIN [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Low limit of the value range
VALUE [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be normalized.
MAX [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Floating-point numbers	I, Q, M, D, L	I, Q, M, D, L	Result of the normalization
[1)] If you use constants in these three parameters, you only need to declare one of them.					

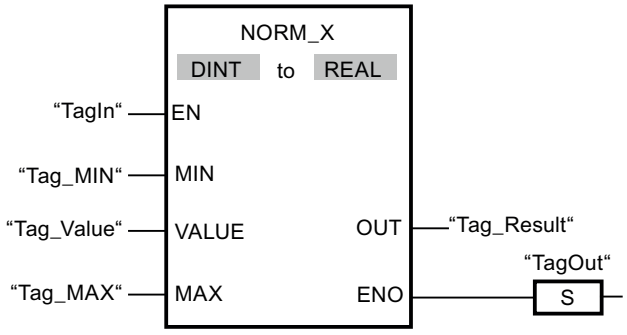
You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	20
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

If the "TagIn" operand has signal state "1", the instruction is executed. The value at input "Tag_Value" is assigned to the range of values defined by the values at inputs "Tag_MIN" and "Tag_MAX". The tag value at input "Tag_Value" is normalized corresponding to the defined value range. The result is stored as a floating-point number at output "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SCALE_X: Scale (Page 2670)

Legacy

SCALE: Scale

Description

Use the "Scale" instruction to convert the integer in the IN parameter into a floating-point number, which can be scaled in physical units between a low limit value and a high limit value. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible in the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case, the constant "K1" has the value -27648.0 and the constant "K2" the value +27648.0.
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case, the constant "K1" has the value 0.0 and the constant "K2" the value +27648.0.

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	INT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Indicates if the value at IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	I, Q, M, D, L, P	Error information

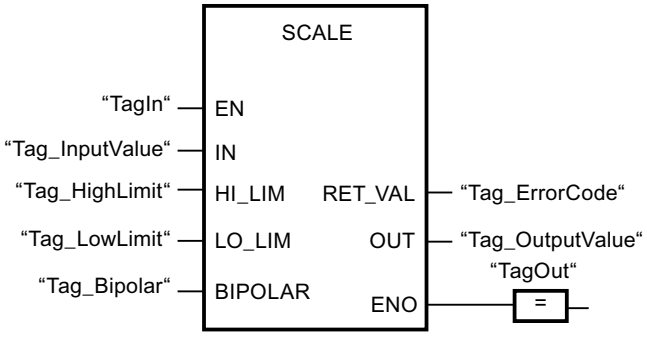
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the "K2" constant or less than the value of the "K1" constant.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2697)

UNSCALE: Unscale

Description

The "Unscale" instruction unscales the floating-point number at the IN parameter into physical units between a low limit value and a high limit and converts them into integers. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible in the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case, the constant "K1" has the value -27648.0 and the constant "K2" the value +27648.0.
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case, the constant "K1" has the value 0.0 and the constant "K2" the value +27648.0.

When the value at the IN parameter is greater than the value of the constant "HI_LIM", the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Low limit

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	I, Q, M, D, L, P	Error information

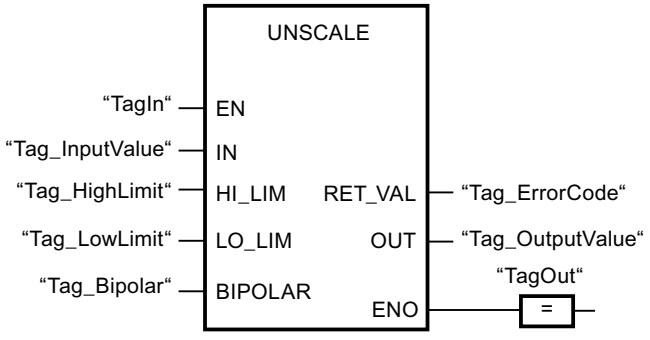
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2697)

Program control operations

JMP: Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

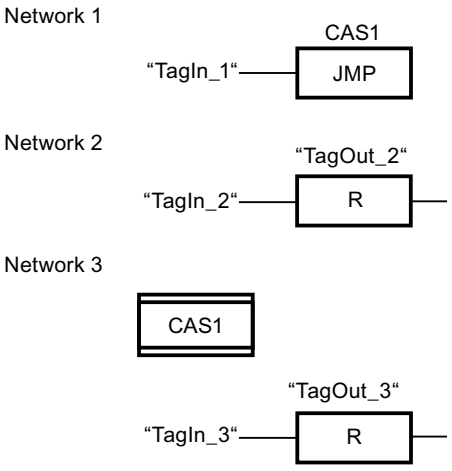
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil can occur in a network.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If the "TagIn_1" operand has signal state "1", the instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 1908)

JMPN: Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

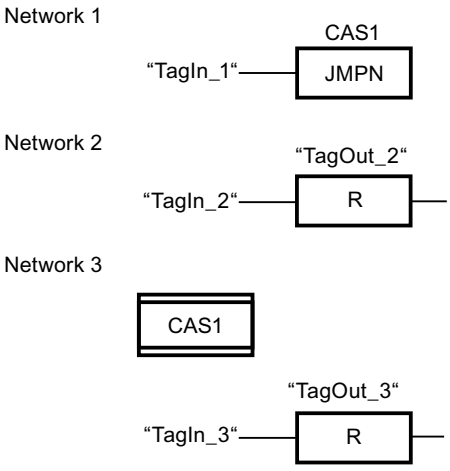
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil can occur in a network.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of the logic operation RLO at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If the "TagIn_1" operand has signal state "0", the instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 1908)

LABEL: Jump label

Description

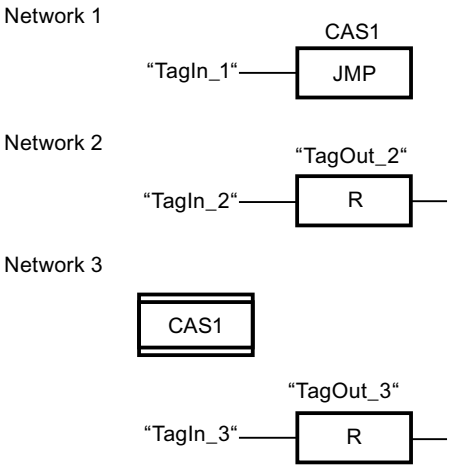
The jump label identifies a destination network in which the execution of the program can be resumed after the execution of a jump instruction.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block. You can declare up to 32 jump labels when you use a CPU S7-1200 and a maximum of 256 jump labels when you use a CPU S7-1500.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

Example

The following example shows how the instruction works:



If the "TagIn_1" operand has signal state "1", the instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 1908)

JMP_LIST: Define jump list

Description

You can use the "Define jump list" instruction to define several conditional jumps and continue the program execution in a specific network depending on the value of the K parameter.

You define the jumps with jump labels (LABEL), which you specify at the outputs of the instruction box. The number of outputs can be expanded in the instruction box. You can declare up to 32 outputs when you use a CPU S7-1200 and a maximum of 99 outputs when you use a CPU S7-1500.

The numbering of the outputs begins with the value "0" and is continued in ascending order with each new output. Only jump labels can be specified at the outputs of the instruction. It is not permitted to specify instructions or operands.

The value of the K parameter specifies the number of the output and thus the jump label where the program execution is to be resumed. If the value in the K parameter is greater than the number of available outputs, the program execution is resumed in the next network of the block.

The "Define jump list" instruction is only executed if the signal state is "1" at the EN enable input.

Parameters

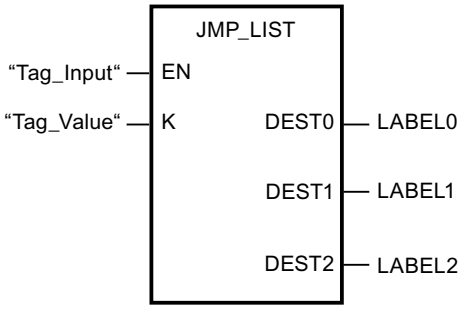
The following table shows the parameters of the "Define jump list" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, L, D	Enable input
K	Input	UINT	I, Q, M, L, D or constant	Specifies the number of the output and thus the jump that is executed.
DEST0	-	-	-	First jump label
DEST1	-	-	-	Second jump label
DESTn	-	-	-	Optional jump labels

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand/jump label	Value
K	Tag_Value	1
DEST0	LABEL0	Jump to the network that is identified with the jump label "LABEL0".
DEST1	LABEL1	Jump to the network that is identified with the jump label "LABEL1".
DEST2	LABEL2	Jump to the network that is identified with the jump label "LABEL2".

If the "Tag_Input" operand has signal state "1", the instruction is executed. The execution of the program is continued according to the value of the operand "Tag_Value" in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 1908)

SWITCH: Jump distributor

Description

You can use the "Jump distributor" instruction to define multiple program jumps to be executed depending on the result of one or more comparison instructions.

At the parameter K, you specify the value to be compared. This value is compared with the values that the individual inputs return. You select the type of comparison for each input. The availability of various comparison instructions depends on the data type of the instruction.

The following table shows the comparison instructions that are available depending on the selected data type:

Data type		Instruction	Syntax
S7-1200	S7-1500		
Bit strings	Bit strings	Equal	==
		Not equal	<>
Integers, floating-point numbers, TIME, DATE, TOD	Integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	Equal	==
		Not equal	<>
		Greater or equal	>=
		Less or equal	<=
		Greater than	>
		Less than	<

You can select the data type of the instruction from the "???" drop-down list of the instruction box. If you select a comparison instruction and the data type of the instruction is not yet defined, then the "???" drop-down list will only list those data types that are permitted for the selected comparison instruction.

The execution of the instruction begins with the first comparison and is executed until a comparison condition is fulfilled. When a comparison condition is fulfilled the subsequent comparison conditions are not considered. If none of the specified comparison conditions are fulfilled, the jump is executed at output ELSE. If not jump label is defined at output ELSE, the linear execution of the program is not interrupted but instead continued in the next network.

In its initial state the instruction box contains at least 2 outputs (DEST0 and DEST1). The number of outputs can be extended. The numbering of the outputs begins with the value "0" and is continued in ascending order with each new output. Specify jump labels (LABEL) at the outputs of the instruction. It is not permitted to specify instructions or operands at the outputs of the instruction.

An input is automatically inserted to each additional output. The jump programmed at an output is executed when the comparison condition of the corresponding is fulfilled.

Parameters

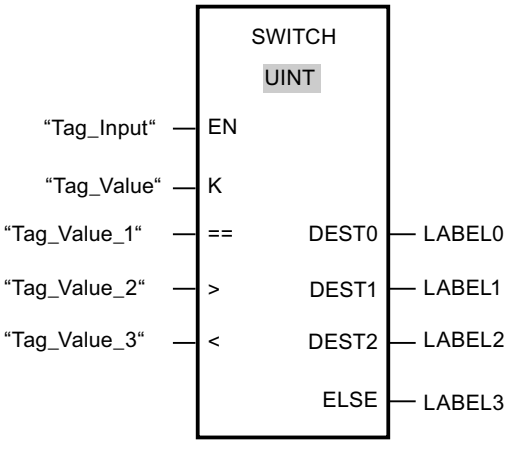
The following table shows the parameters of the instruction "Jump distributor":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
K	Input	UINT	UINT	I, Q, M, D, L or constant	Specifies the value to be compared.
<Comparison values>	Input	Bit strings, integers, floating-point numbers, TIME, DATE, TOD	Bit strings, integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	I, Q, M, D, L or constant	Input values with which the value of the parameter K is compared.
DEST0	-	-	-	-	First jump label
DEST1	-	-	-	-	Second jump label
DEST(n)	-	-	-	-	Optional jump labels (n = 2 to 99)
ELSE	-	-	-	-	Program jump which is executed if none of the comparison conditions are fulfilled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand/jump label	Value
K	Tag_Value	23
==	Tag_Value_1	20
>	Tag_Value_2	21
<	Tag_Value_3	19
DEST0	LABEL0	Jump to jump label "LABEL0", if the value of parameter K is equal to 20.
DEST1	LABEL1	Jump to jump label "LABEL1", if the value of parameter K is greater than 21.
DEST2	LABEL2	Jump to jump label "LABEL2", if the value of parameter K is less than 19.
ELSE	LABEL 3	Jump to jump label "LABEL3", if none of the comparison conditions are fulfilled.

If the operand "Tag_Input" changes to signal state "1", the instruction is executed. The execution of the program is continued in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 1908)

RET: Return

Description

You can use the instruction "Return" to stop the execution of a block. This results in three types, in which the block processing can be completed:

- Without call of the "Return" instruction
  The block is exited after the execution of the last network. The ENO of the call function is set to the signal state "1".
- Call of the "Return" instruction with logic operation (see example)
  If the left connector has the signal state "1", then the block is exited. The ENO of the function call corresponds to the operand.
- Call of the "Return" instruction without logic operation
  The block is exited. The ENO of the function call corresponds to the operand.

Note

Only one jumping coil may be used in a network ("Return", "Jump if RLO = 1", "Jump if RLO = 0").

If the result of logic operation (RLO) at the input of the instruction "Return" is "1", the execution of the program is terminated in the currently called block and continued after the call function in the calling block (for example, in the calling OB). The status (ENO) of the call function is determined by the parameter of the instruction. This can assume the following values:

- RLO
- TRUE/FALSE
- <Operand>

To set the parameter values, double-click the instruction and select the corresponding value in the drop-down list.

The following table shows the status of the call function when the instruction "Return" is programmed in a network within the called block:

RLO	Parameter value	ENO of the call function
1	RLO	1
	TRUE	1
	FALSE	0
	<Operand>	<Operand>
0	RLO	In this case, the execution of the program continues in the next network of the called block.
	TRUE	
	FALSE	
	<Operand>	

If an OB is completed, another block is selected by the priority class system and started or re-executed.

- If the OB program cycle was completed, it is restarted.
- If an OB, which interrupted another block (e.g. an Alarm OB), is completed, then the interrupted block (e.g. OB program cycle) is executed.

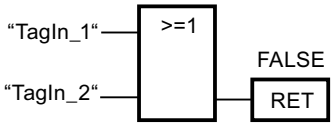
Parameters

The following table shows the parameters of the instruction "Return":

Parameters	Declaration	Data type	Memory area	Description
Status of the calling function when RLO = 1:				
RLO	-	-	-	Is set to the signal state of the RLO.
TRUE	-	-	-	1
FALSE	-	-	-	0
<Operand>	Input	BOOL	I, Q, M, D, L	Signal state of the specified operand

Example

The following example shows how the instruction works:



If one of the operands "TagIn_1" or "TagIn_2" have signal state "1", the instruction is executed. Program execution in the called block is terminated and continues in the calling block. The enable output ENO of the call function is reset to signal state "0".

See also

Overview of the valid data types (Page 1908)

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether passwords configured for the CPU are legitimized or not. In this way, you can prevent legitimized connections, even if the correct password is known.

When the instruction is called and the REQ parameter has the signal state "0", the currently set state is displayed at the output parameters. If changes have been made to the input parameters, these are not transferred to the output parameters.

When the instruction is called and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD). FALSE means that legitimation per password is not permitted; TRUE means the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Passwords that are not configured must have the signal state TRUE at the input and return the signal state TRUE at the output. The fail-safe password can only be configured for an F-CPU and must therefore always be interconnected with the signal state TRUE in a standard CPU. If the instruction returns an error, the call has no effect and the previous lock is still in effect.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.

- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the mode selector to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Plugging an empty memory card (transfer module or program card) into an S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called once again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access from HMI systems if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing unintentional lockout on an S7-1500 CPU

The settings can be made on the front panel of the CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode selector to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing unintentional lockout on an S7-1200 CPU

Because the S7-1200 CPU does not have a mode switch, protection is disabled with POWER OFF - POWER ON. This means that it is possible and advisable to prevent accidental lockout with the help of specific program sequences within your program.

To do so, program a time control either by means of a cyclic interrupt OB or a timer in the Main OB (OB 1). This gives you the option of calling the "Limit and enable password legitimation" instruction in the respective OB (e.g., OB 1 or OB 35) relatively quickly after a transition from POWER OFF to POWER ON and the associated disabling of the protection. Call the instruction in the startup OB (OB 100) to keep the time window in which the instruction is inactive and there are no limitations for password legitimation relatively small. This procedure will give you the best possible protection from unauthorized access.

If there has been an accidental lockout, you can skip the call in the startup OB (by querying an input parameter, for example) and you have the set time (for example, 10 seconds up to 1 minute) to establish a connection to the CPU before the lock becomes active once again.

If there is no timer in your program code and a lockout has occurred, insert an empty transfer card or an empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then download the user program once again from STEP 7 to the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

Warning

Inserting an empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Once you have pulled the transfer card, the content of the transfer card is available in the internal load memory. Make sure that the card does not include a program at this point.

Warning

Inserting an empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. Once you have pulled the previously empty program card, the internal load memory is empty.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200 CPU: The lock is disabled and the instruction must be called once again in the program (e.g., in the startup OB). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Parameters

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password

Parameters	Declaration	Data type	Memory area	Description
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported
80D0	The password for fail-safe is not configured. The signal state must be TRUE for standard CPUs.
80D1	The read/write access is not configured
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

See also

Overview of the valid data types (Page 1908)

RE_TRIGR: Restart cycle monitoring time

Description

You can use the "Restart cycle monitoring time" instruction to restart the cycle time monitoring of the CPU. The cycle monitoring time then starts again for the time you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called in all blocks regardless of the priority .

If the instruction is called in a block with a higher priority, for example a hardware interrupt, diagnostics interrupt or a cyclic interrupt, the instruction is not executed and the enable output ENO is set to signal state "0".

The "Restart cycle monitoring time" instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Parameters

The instruction "Restart cycle monitoring time" has no parameters.

See also

Overview of the valid data types (Page 1908)

STP: Exit program

Description

The instruction "Exit program" instruction is used to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

When the (RLO) at the input of the instruction is "1", the CPU changes to STOP mode and the execution of the program is terminated. The signal state at the output of the instruction is not evaluated.

If the RLO at the input of the instruction is "0", then the instruction will not be executed.

Parameters

The instruction "Exit program" has no parameters.

See also

Overview of the valid data types (Page 1908)

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports an error during the processing of the block, detailed information is stored in the operand at the ERROR output for the first error to occur during the execution of the block since the last execution of the instruction.

Only operands of the "ErrorStruct" system data type can be specified at the ERROR output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the block, the error information about the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The ERROR output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO of the instruction "Get error locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error locally" instruction enables local error handling within a block. If "Get error locally" is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Parameters

The following table lists the parameters of the "Get error locally" instruction:

Parameters	Declaration	Data type	Memory area	Description
ERROR	Output	ErrorStruct	D, L	Error information

Data type "ErrorStruct"

The following table lists the structure of the "ErrorStruct" data type:

Structure component		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows if an error occurred during a block call. 16#01: Error during a block call 16#00: No error during a block call
REACTION		BYTE	Default reaction: 0: Ignore (write error), 1: Continue with substitute value "0" (read error), 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of block
	BLOCK_TYPE	BYTE	Type of block where the error occurred: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B Range violations for a directly editable tag of data type DINT: 16#04

Structure component		Data type	Description
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

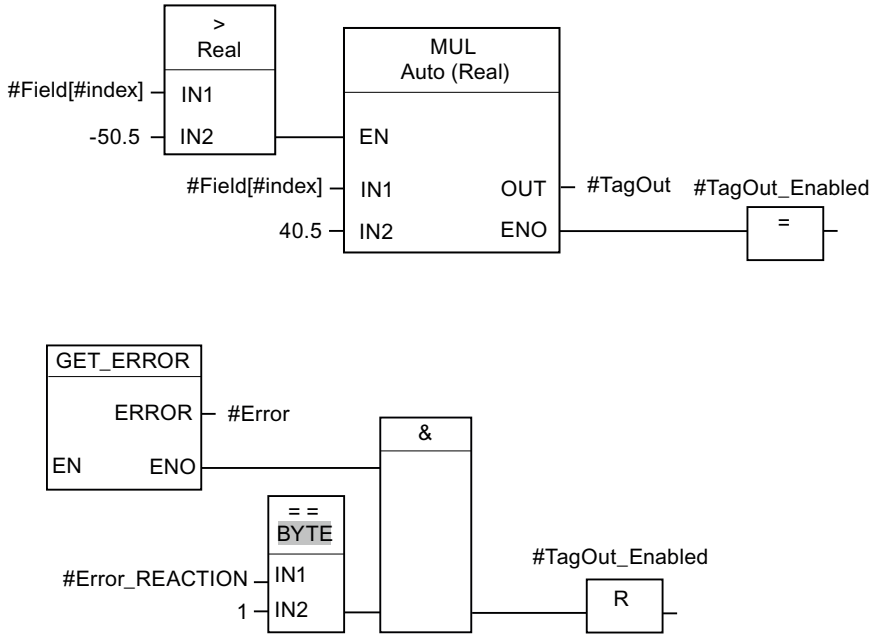
Structure component "ERROR_ID"

The following table lists the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".		

Example

The following example shows how the instruction works:



An error occurred accessing the #Field[#index] tag. The enable output ENO of the "Multiply" instruction and the #TagOut_Enabled operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error locally" instruction is evaluated with the comparison instruction "Equal". If the "#Error.REACTION" structure component has the value "1", this involves a read/access error and the #TagOut_Enabled output is reset.

See also

Overview of the valid data types (Page 1908)

Querying and fixing errors in the program code (Page 228)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the error ID of the first error that occurred in the tag is stored at the ID output.

Only operands of the WORD data type can be specified at the ID output. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The ID output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO of the instruction "Get error ID locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Parameters

The following table lists the parameters of the "Get error ID locally" instruction:

Parameters	Declaration	Data type	Memory area	Description
ID	Output	WORD	I, Q, M, D, L	Error ID

Parameter ID

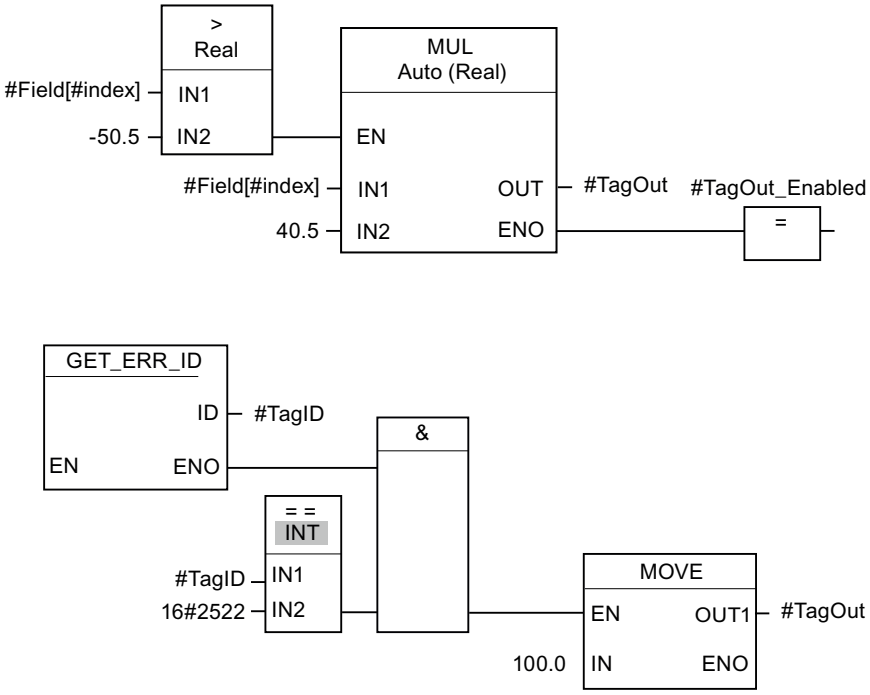
The following table lists the values that can be output at the ID parameter:

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment

ID* (hexadecimal)	ID* (decimal)	Description
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".		

Example

The following example shows how the instruction works:



An error occurred accessing the #Field[#index] tag. The enable output ENO of the "Multiply" instruction and the #TagOut_Enabled operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error ID locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error ID locally" instruction is evaluated with the comparison instruction "Equal". If the #TagID operand returns the ID 2522, this involves a read/access error and the value "100.0" is written to the #TagOut output.

See also

Overview of the valid data types (Page 1908)

Querying and fixing errors in the program code (Page 228)

INIT_RD: Initialize all retain data

Description

The "Initialize all retain data" instruction is used to reset the retentive data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution exceeds the program cycle duration.

Parameters

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	If the input "REQ" has the signal state "1", all retentive data are reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

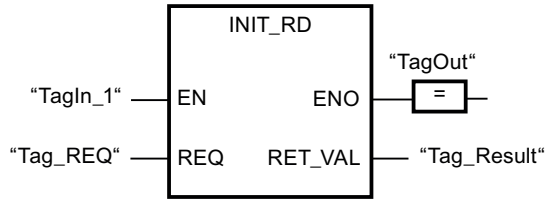
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "Tag_REQ" have signal state "1", the instruction is executed. All retentive data of all data blocks, bit memories and SIMATIC timers and counters are reset. If the instruction is executed without errors, the ENO enable output has the signal state "1".

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2697)

WAIT: Configure time delay

Description

The "Configure time delay" instruction pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to +32767 microseconds (µs). The smallest possible delay time depends on the respective CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction returns no error information.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 1908)

RUNTIME: Measure program runtime

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and write the value to the IN-OUT parameter MEM. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the IN-OUT parameter MEM as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The following applies to S7-1200 CPUs with firmware version earlier than V4.1: The "Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overflows (this can occur up to once per minute), the instruction returns values <= 0.0. Ignore these runtime values.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Parameters

The following table lists the parameters of the "Measure program runtime" instruction:

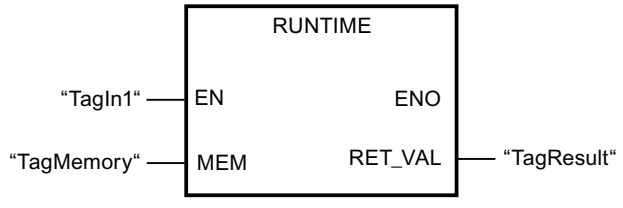
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MEM	InOut	LREAL	I, Q, M, D, L	Saves the starting point of the runtime measurement.
RET_VAL	Output	LREAL	I, Q, M, D, L	Returns the measured runtime in seconds

For additional information on valid data types, refer to "See also".

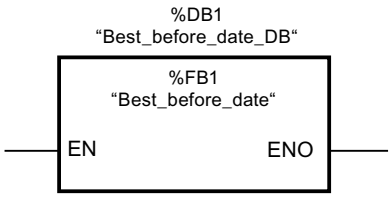
Example

The following example shows how the instruction works based on the runtime calculation of a program block:

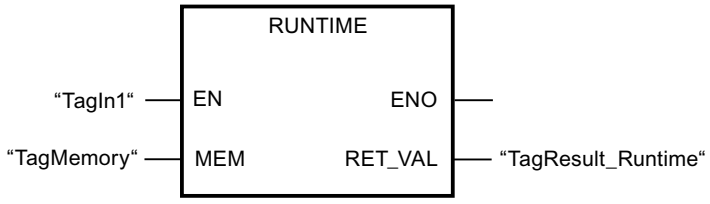
Network 1:



Network 2:



Network 3:



When the "TagIn1" operand in network 1 has the signal state "1", the instruction is executed. The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called in network 2.

When the FB1 program block has been executed and the "TagIn1" operand has the signal state "1", the instruction in network 3 is executed. The second call of the instruction calculates the runtime of the program block and writes the result to the output RET_VAL.

See also

Overview of the valid data types (Page 1908)

Word logic operations

AND: AND logic operation

Description

You can use the instruction "AND logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by AND logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by AND logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by AND logic. The result is stored at output "OUT".

The result bit has the signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "AND logic operation":

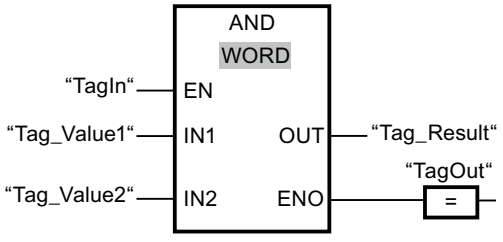
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0000 0000 0000 0101

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is linked by AND to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

OR: OR logic operation

Description

You can use the instruction "OR logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended in the instruction box. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by OR logic. The result is stored at output "OUT".

The result bit has the signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "OR logic operation":

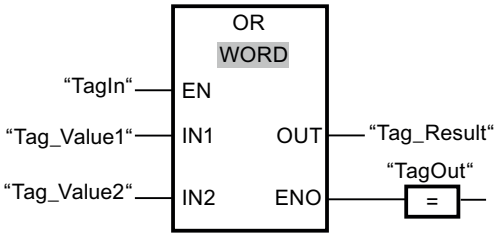
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1111

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is linked by OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

XOR: EXCLUSIVE OR logic operation

Description

You can use the instruction "EXCLUSIVE OR logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by EXCLUSIVE OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by EXCLUSIVE OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended in the instruction box. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by EXCLUSIVE OR logic. The result is stored at output "OUT".

The result bit has the signal state "1" when one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

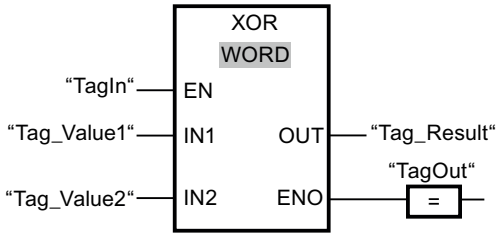
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1010

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is linked by EXCLUSIVE OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

INVERT: Create ones complement

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits at input IN. When the instruction is processed, the value at input IN is linked to EXCLUSIVE OR by a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit number). This inverts the signal state of the individual bits that are then stored at output OUT.

Parameters

The following table shows the parameters of the instruction "Create ones complement":

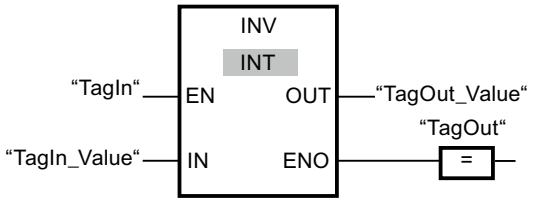
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Bit strings, integers	I, Q, M, D, L, P	I, Q, M, D, L, P	Ones complement of the value at input IN

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	W#16#000F	W#16#7E
OUT	TagOut_Value	W#16#FFF0	W#16#81

If operand "TagIn" has the signal state "1", the "Create ones complement" instruction is executed. The instruction inverts the signal state of the individual bits at input "TagIn_Value" and writes the result to output "TagOut_Value". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 1908)

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. If the value at input IN is greater than 31, a modulo 32 instruction is executed.

Parameters

The following table shows the parameters of the "Decode" instruction:

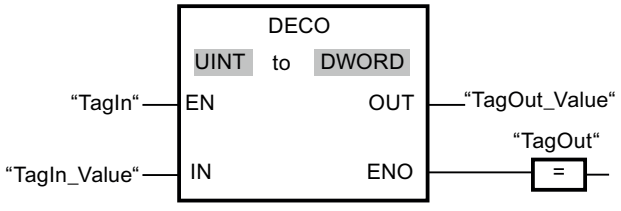
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	UINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set.
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

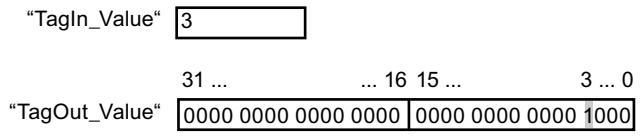
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using concrete operand values:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction reads bit number "3" from the value of the operand "TagIn_Value" and sets the third bit to the value of the operand "TagOut_Value".

If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 1908)

ENCO: Encode

Description

The instruction "Encode" is used to read the bit number of the lowest bit in the input value and output it to the output OUT.

The instruction "Encode" selects the least significant bit of the value at the IN input and writes its bit number to the tag in the output OUT.

Parameters

The following table shows the parameters of the instruction "Encode":

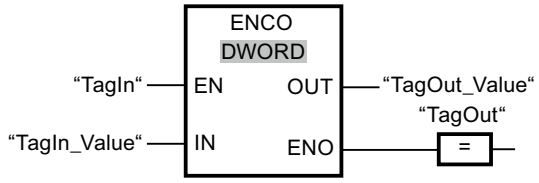
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

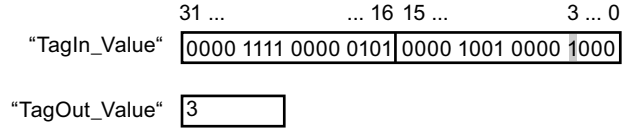
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using concrete operand values:



If the "TagIn" operand has signal state "1", the instruction is executed. The instruction selects bit position "3" as the least significant bit at input "TagIn_Value" and writes the value "3" to the tag at the output "TagOut_Value.

If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SEL: Select

Description

Depending on the signal state at switch (input G), the "Select" instruction selects one of the inputs IN0 or IN1 and moves its content to the output OUT. When the input G has the signal state "0", the value at the input IN0 is moved. When the input G has the signal state "1", the value at the input IN1 is moved to the output OUT.

The instruction can only be executed if the enable input EN has the signal state "1" and the tags of all parameters are of the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

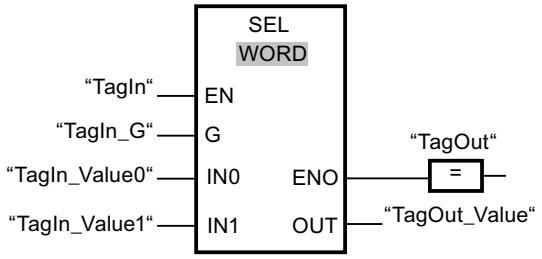
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
G	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, DATE, TDT, CHAR, WCHAR	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, DATE, TDT, CHAR, WCHAR	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, DATE, TDT, CHAR, WCHAR	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If the "TagIn" operand has signal state "1", the instruction is executed. Based on the signal state at the input "TagIn_G", the value at input "TagIn_Value0" or "TagIn_Value1" is selected and copied to output "TagOut_Value". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

MUX: Multiplex

Description

You can use the instruction "Multiplex" to copy the content of a selected input to output OUT. In its initial state the instruction box contains at least 2 inputs (IN0 and IN1). The number of selectable inputs of the instruction box can be expanded. You can declare a maximum of 32 inputs.

The inputs are numbered automatically in the box. Numbering starts at IN0 and is incremented continuously with each new input. You use the K parameter to define the input whose content is to be copied to the OUT output. If the value of the parameter K is greater than the number of available inputs, the content of the parameter ELSE is copied to output OUT and enable output ENO is assigned the signal state "0".

The "Multiplex" instruction can only be executed if the tags have the same data type in all inputs and in the OUT output. The K parameter is an exception, since only integers can be specified for it.

The enable output ENO is reset if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The input at the K parameter is located outside the available inputs. This is the reaction regardless of whether the ELSE input will be used or not. The value at the OUT output remains unchanged.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the "Multiplex" instruction:

Parameters	Declaring	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
K	Input	Integers	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Specifies the input whose content is to be copied. • If K = 0 => Parameter IN0 • If K = 1 => Parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value

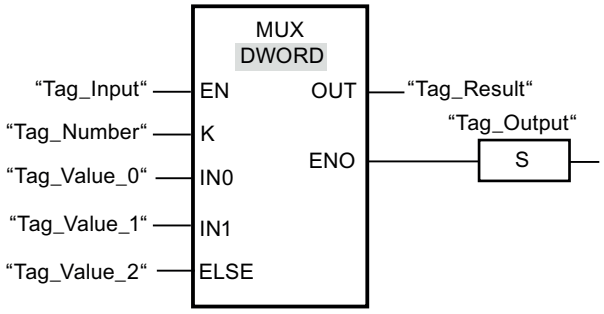
Parameters	Declaring	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN1	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
INn	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
ELSE	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Specifies the value to be copied when K > n
OUT	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Output to which the value is to be copied

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
K	Tag_Number	1
IN0	Tag_ValueI_0	DW#16#00000000
IN1	Tag_Value_1	DW#16#003E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#003E4A7D

If the "Tag_Input" operand has signal state "1", the instruction is executed. Depending on the value of the operand "Tag_Number", the value at input "Tag_Value_1" is copied and assigned to the operand at output "Tag_Result". If no errors occur during the execution of the instruction, the outputs ENO and "Tag_Output" are set.

See also

Overview of the valid data types (Page 1908)

DEMUX: Demultiplex

Description

You can use the instruction "Demultiplex" to copy the content of the input IN to a selected output. In its initial state the instruction box contains at least 2 outputs (OUT0 and OUT1). The number of selectable outputs can be extended in the instruction box. The outputs are numbered automatically in the box. Numbering starts at OUT0 and continues consecutively with each new output. You use the K parameter to define the output to which the content of the IN input is to be copied. The other outputs are not changed. If the value of the parameter K is greater than the number of available outputs, then the content of input IN in the parameter ELSE and the enable output ENO will be assigned to the signal state "0".

The instruction "Demultiplex" can only be executed if the tags at the input IN and at all outputs are of the same data type. The K parameter is an exception, since only integers can be specified for it.

The enable output ENO is reset if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the instruction "Demultiplex":

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
K	Input	Integers	Integers	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Specifies the output to which the input value (IN) is copied. • If K = 0 => Parameter OUT0 • If K = 1 => Parameter OUT1, etc.
IN	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Input value
OUT0	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	First output

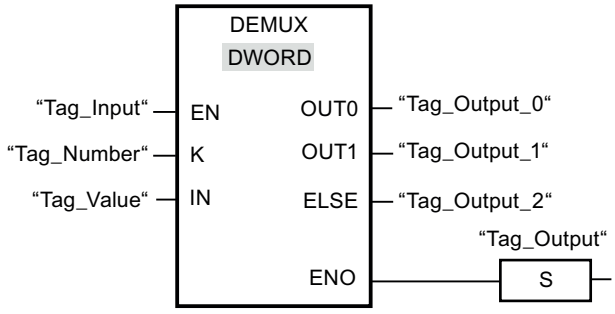
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
OUT1	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Second output
OUTn	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Optional outputs
ELSE	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Output to which the input value (IN) at K > n is copied.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:



The following tables show how the instruction works using specific operand values:

Table 11-26 Input values of the "Demultiplex" instruction before network execution

Parameter	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

Table 11-27 Output values of the "Demultiplex" instruction after network execution

Parameter	Operand	Values	
OUT0	Tag_Output_0	Unchanged	Unchanged
OUT1	Tag_Output_1	DW#16#FFFFFFFF	Unchanged
ELSE	Tag_Output_2	Unchanged	DW#16#003E4A7D

If the "Tag_Input" input has the signal state"1", the "Demultiplex" instruction is executed. Depending on the value of the operand "Tag_Number", the value at input "IN" is copied to the corresponding output.

See also

Overview of the valid data types (Page 1908)

Shift and rotate

SHR: Shift right

Description

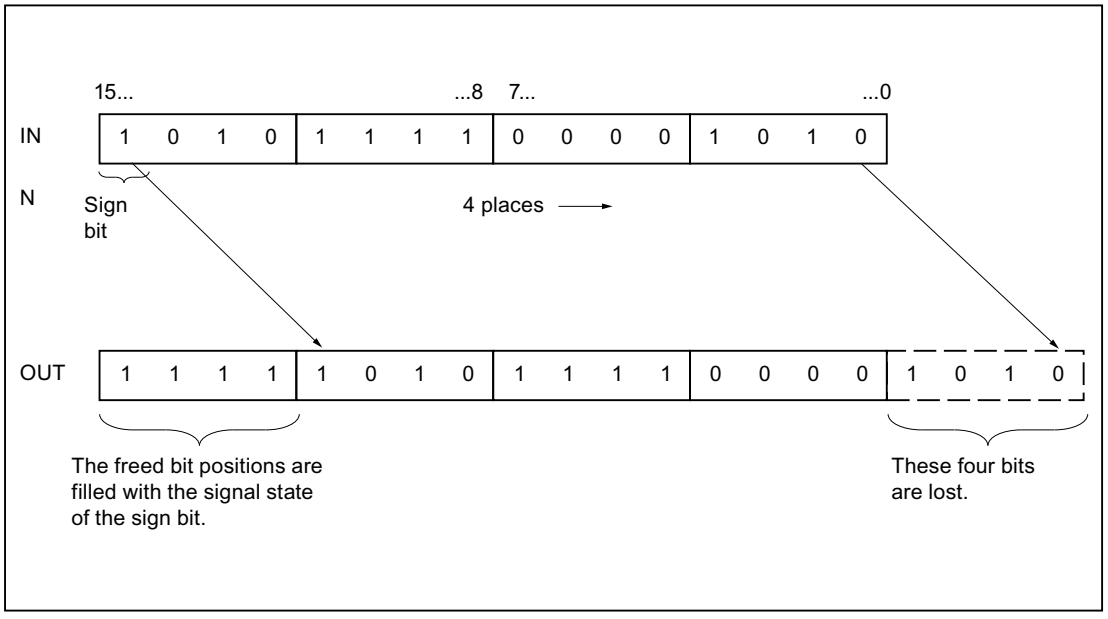
You can use the "Shift right" instruction to shift the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be moved.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value at the input N is greater than the number of available bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure show how the content of an operand of integer data type is shifted four bit positions to the right:



Parameters

The following table shows the parameters of the instruction "Shift right":

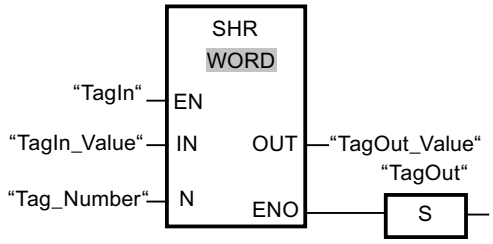
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If the "TagIn" operand has signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is shifted three bit positions to the right. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

SHL: Shift left

Description

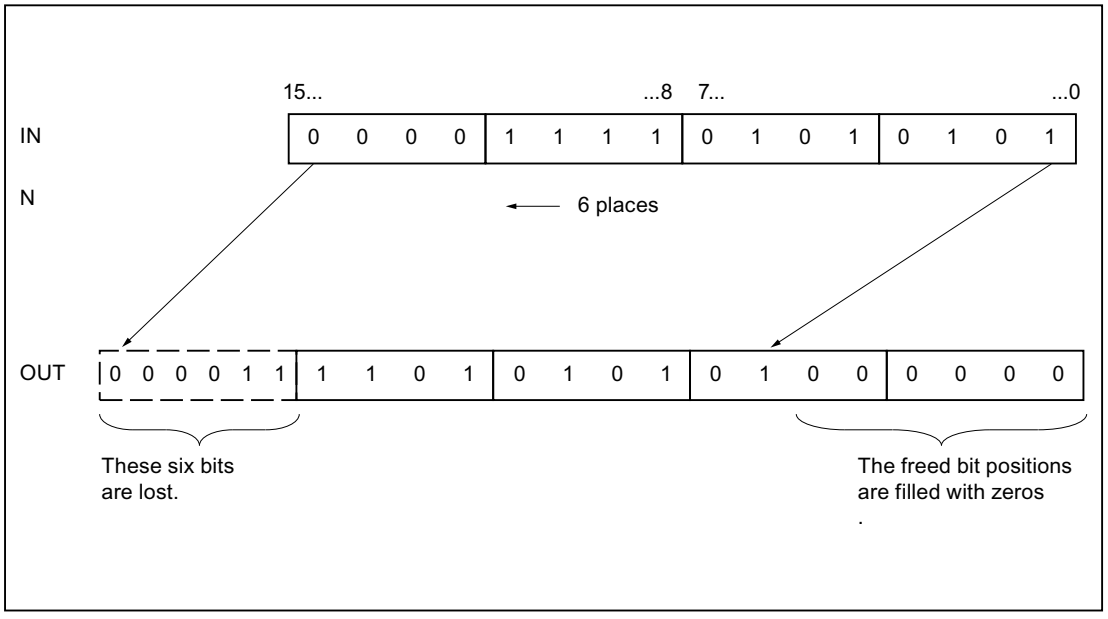
You can use the "Shift left" instruction to shift the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be moved.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value at the input N is greater than the number of available bit positions, the operand value at input IN is shifted to the left by the available number of bit positions.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure shows how the content of an operand of the WORD data type is shifted by six bit positions to the left:



Parameters

The following table shows the parameters of the instruction "Shift left":

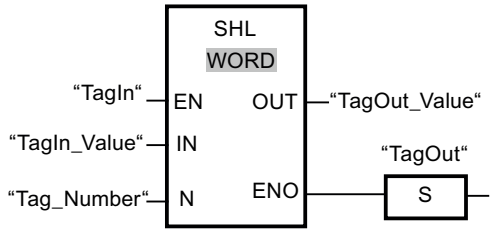
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be shifted.
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted.
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If the "TagIn" operand has signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is shifted four bit positions to the left. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ROR: Rotate right

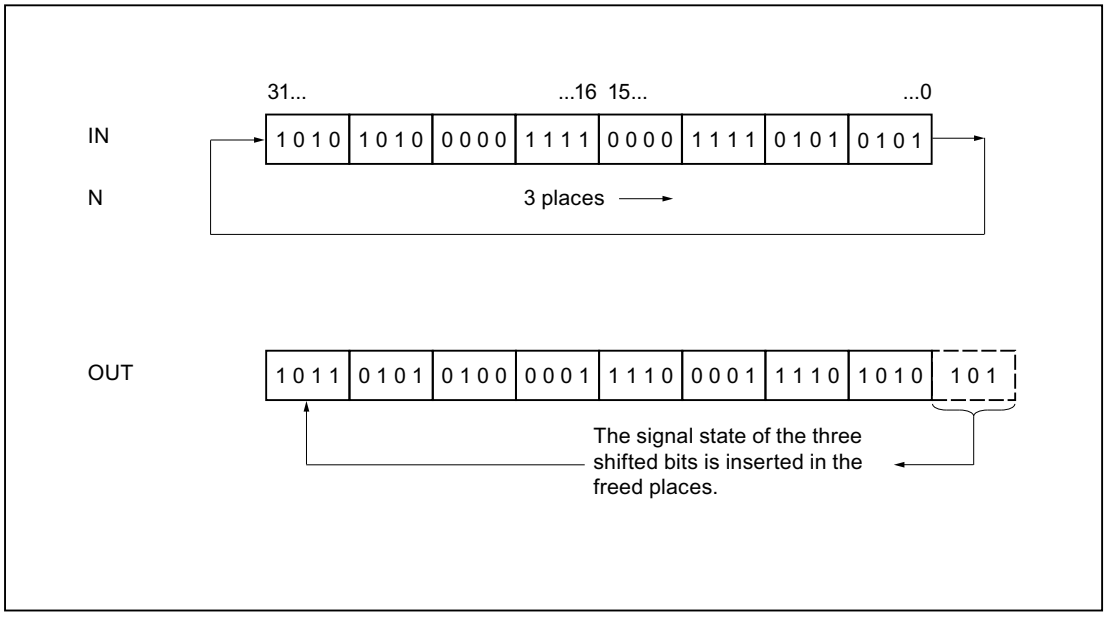
Description

You can use the "Rotate right" instruction to rotate the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating on the left-hand side are filled true-to-position with the bit positions that are pushed out from the left-hand side.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value of the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:



Parameters

The following table shows the parameters of the instruction "Rotate right":

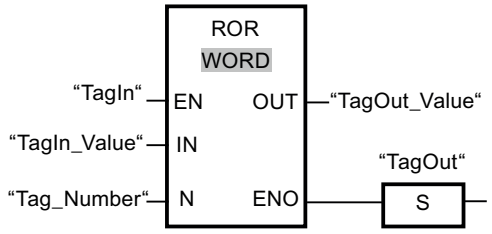
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

If the "TagIn" operand has signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the right. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

ROL: Rotate left

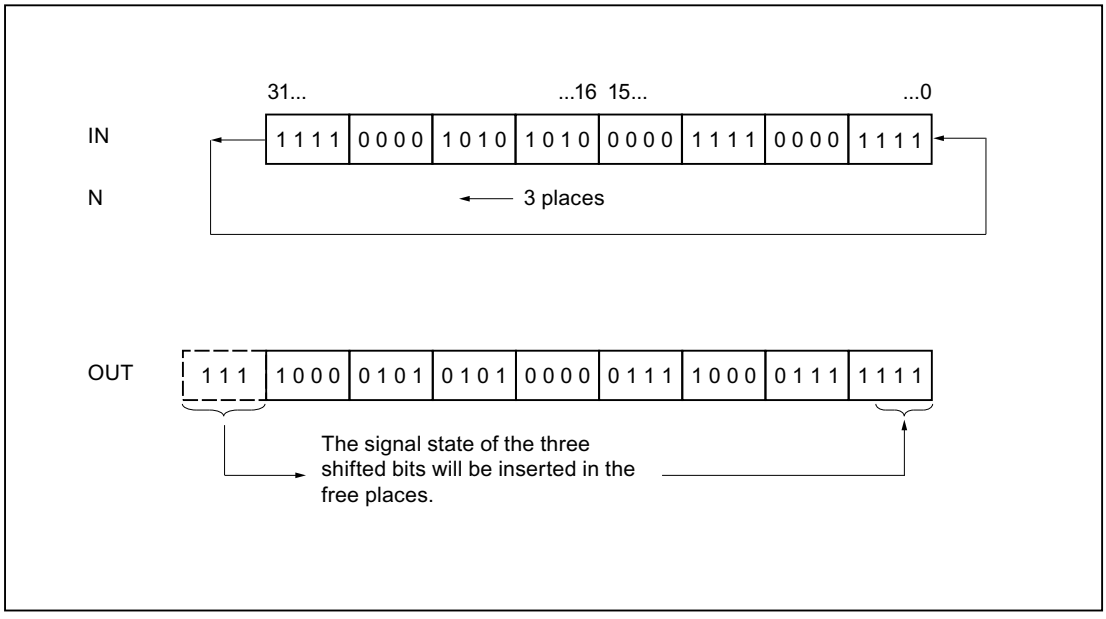
Description

You can use the "Rotate left" instruction to rotate the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating on the right-hand side are filled true-to-position with the bit positions that are pushed out from the left-hand side.

If the value at the input N is "0", the value at input IN is copied to the operand at output OUT.

If the value of the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:



Parameters

The following table shows the parameters of the instruction "Rotate left":

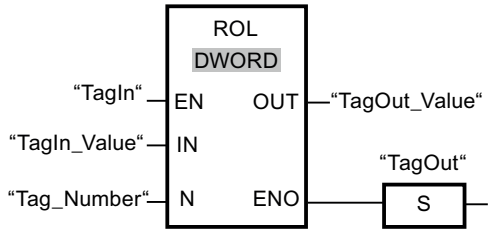
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be rotated.
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated.
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	1010 1000 1111 0110
N	Tag_Number	5
OUT	TagOut_Value	0001 1110 1101 0101

If the "TagIn" has the signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the left. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 1908)

Legacy

DRUM: Implement sequencer

Description

You can use the "Implement sequencer" instruction to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK), you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask has signal state "1", the OUT_VAL value sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit at the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit at the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates the reset condition.
JOG	Input	BOOL	I, Q, M, D, L or constant	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" causes the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L or constant	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)

Parameter	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L or constant	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P, or constant	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P, or constant	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P, or constant	Current numerical value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P, or constant	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L or constant	Preset count for each step [1 to 16]; where 1 clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

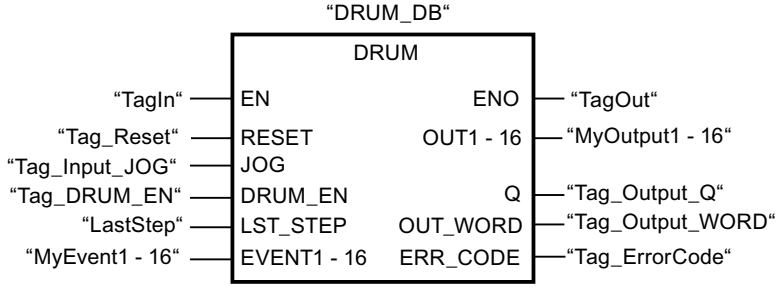
ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A

Parameter	Address	Value
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE

Parameter	Address	Value
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE

Parameter	Operand	Address	Value
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 1908)

DCAT: Discrete control-timer alarm

Description

The "Discrete control-timer alarm" instruction is used to accumulate the time from the point at which the CMD parameter issued the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET is < PT) OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state of the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".
- When the signal state of the CMD and CMD_HIS parameters is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state at the parameter OA is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state at the OA parameter is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the CMD, CMD_HIS and O_FB parameters are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the CMD, CMD_HIS and C_FB parameters return signal state "0", the time difference (ms) since the last execution of the instruction is added to the value of the ET parameter. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction returns no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L or constant	The signal state "0" indicates a "close" command. The signal state "1" indicates the "open" command.
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L or constant	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	D, L or constant	Preset timer value, where one count = 1 ms.
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
CMD_HIS	Static	BOOL	D, L or constant	CMD history bit

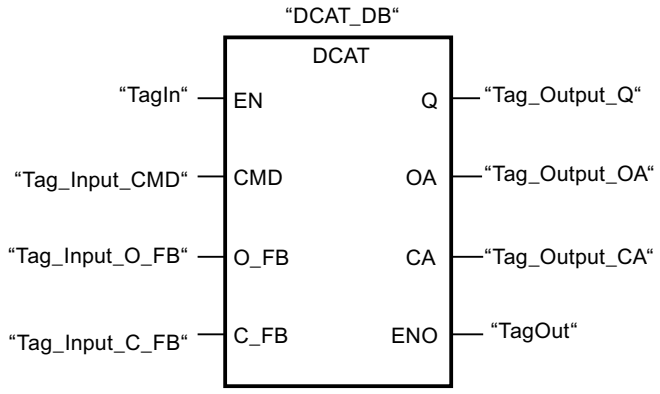
For additional information on valid data types, refer to "See also".

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 1908)

MCAT: Motor control-timer alarm

Description

You can use the "Motor control-timer alarm" instruction to accumulate the time from the point in time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction "Motor control-timer alarm" returns no error information.

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the instruction "Motor control-timer alarm" to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped
Legend:																
INC	Add the time difference (ms) since the last processing of the FB to ET															
PT	PT is set to the same value as ET															

Input parameters	Output parameters
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT
If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row of the above-named table (X) is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case: OO = FALSE CO = FALSE OA = FALSE CA = FALSE ET = PT Q = TRUE	

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
O_CMD	Input	BOOL	I, Q, M, D, L or constant	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L or constant	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L or constant	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	A signal state of "0" indicates an error condition.
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	D, L or constant	Preset time value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
O_HIS	Static	BOOL	D, L or constant	"Open" history bit
C_HIS	Static	BOOL	D, L or constant	"Close" history bit

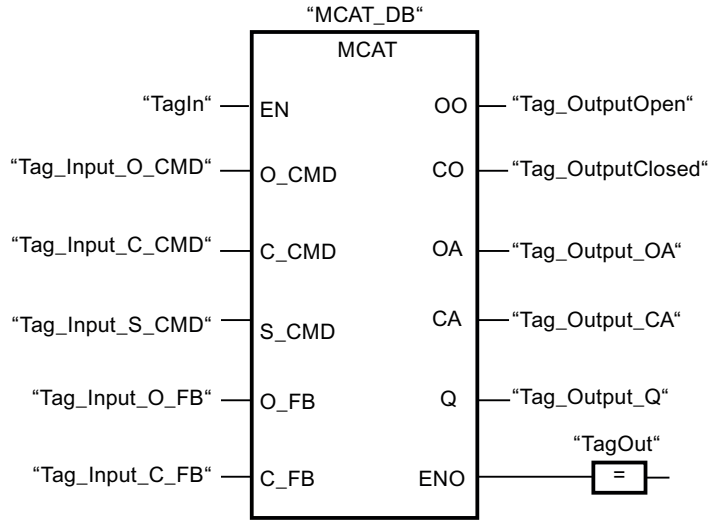
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works based on concrete values:

Before processing

In this example the following values are used for the input and output parameters:

Parameters	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameters	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 1908)

IMC: Compare input bits with the bits of a mask

Description

The instruction "Compare input bits with the bits of a mask" is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. At the CMP_STEP parameter, specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise, the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.

Parameter	Declaration	Data type	Memory area	Description
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 1908)

SMC: Compare scan matrix

Description

The instruction "Compare scan matrix" is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found, the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written to the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value of the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information

Parameter	Declaration	Data type	Memory area	Description
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P or constant	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 1908)

LEAD_LAG: Lead and lag algorithm

Description

The "Lead and lag algorithm" instruction is used to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the instruction "Lead and lag algorithm" is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).

Parameter	Declaration	Data type	Memory area	Description
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

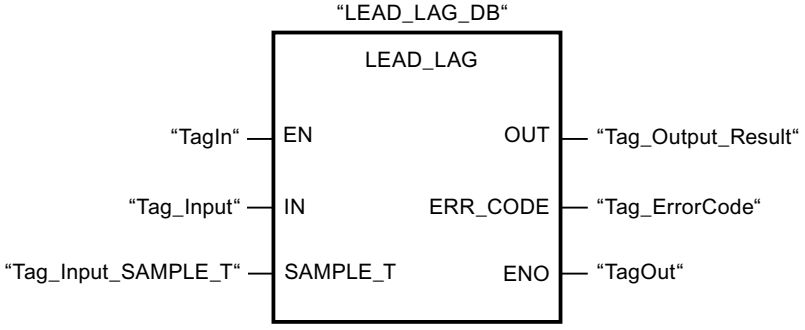
Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input parameters:

Parameters	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameters	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameters	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

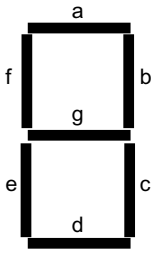
Overview of the valid data types (Page 1908)

SEG: Create bit pattern for seven-segment display

Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	**Assignment of the segments - g f e d c b a**	**Display** (Hexadecimal)	**Seven-segment display**
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

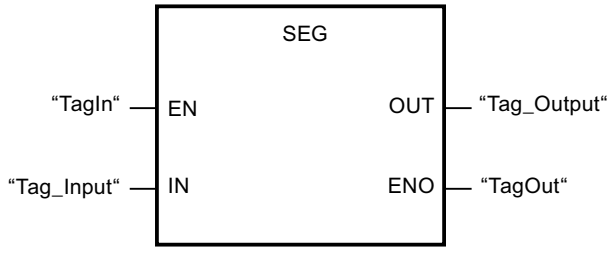
Parameters

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameters	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 1908)

BCDCPL: Create tens complement

Description

You can use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

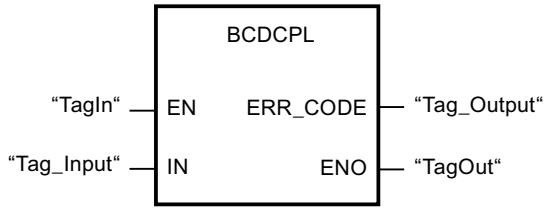
The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 1908)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output at the RET_VAL parameter.

Parameters

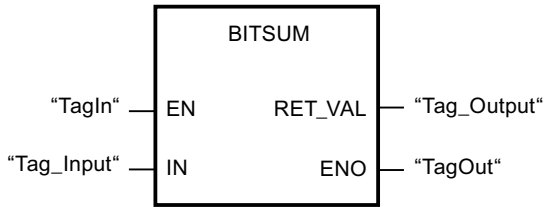
The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted.
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of bits to be set

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 1908)

11.6.2.3 SCL

Bit logic operations

R_TRIG:Detect positive signal edge

Description

With the "Detect positive signal edge" instruction, you can detect a state change from "0" to "1" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "0" to "1", a positive signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface.

Syntax

The following syntax is used for the "Detect positive signal edge" instruction:

```
SCL
<Instance>(CLK := <Operand>,
           Q => <Operand>)
```

Parameters

The following table shows the parameters of the "Detect positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Incoming signal, the edge of which is to be queried
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

```
SCL
"R_TRIG_DB"(CLK := "TagIn",
            Q => "TagOut");
```

The previous state of the tag at the CLK input is stored in the "R_TRIG_DB" tag. If a change is the signal state from "0" to "1" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 1908)

F_TRIG:Detect negative signal edge

Description

With the "Detect negative signal edge" instruction, you can detect a state change from "1" to "0" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "1" to "0", a negative signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface.

Syntax

The following syntax is used for the "Detect negative signal edge" instruction:

SCL

```
<Instance>(CLK := <Operand>,
           Q => <Operand>)
```

Parameters

The following table shows the parameters of the "Detect negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Incoming signal, the edge of which is to be queried
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

SCL

```
"F_TRIG_DB"(CLK := "TagIn",
            Q => "TagOut");
```

The previous state of the tag at the CLK input is stored in the "F_TRIG_DB" tag. If a change in the signal state from "1" to "0" is detected in the "TagIn" operand, the "TagOut" output has signal state "1".

See also

Overview of the valid data types (Page 1908)

Timer operations

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the Q parameter for the duration PT. The instruction starts when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. The Q parameter is set for the time PT, regardless of the subsequent changes in the input signal. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the time duration PT is active.

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT is reached and the signal state at the IN parameter is "0", the ET parameter is reset.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TP_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TP_TIME type in the "Static" section of a block (for example, #MyTP_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME or TP_LTIME in the "Static" section of a block (for example #MyTP_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

Syntax

Use the following syntax for the "Generate pulse" instruction:

- Data block of system data type IEC_Timer (global DB):

```
SCL
<IEC_Timer_DB> TP(IN := <Operand>,
                  PT := <Operand>,
                  Q => <Operand>,
                  ET => <Operand>)
```

- Local tag:

```
SCL
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

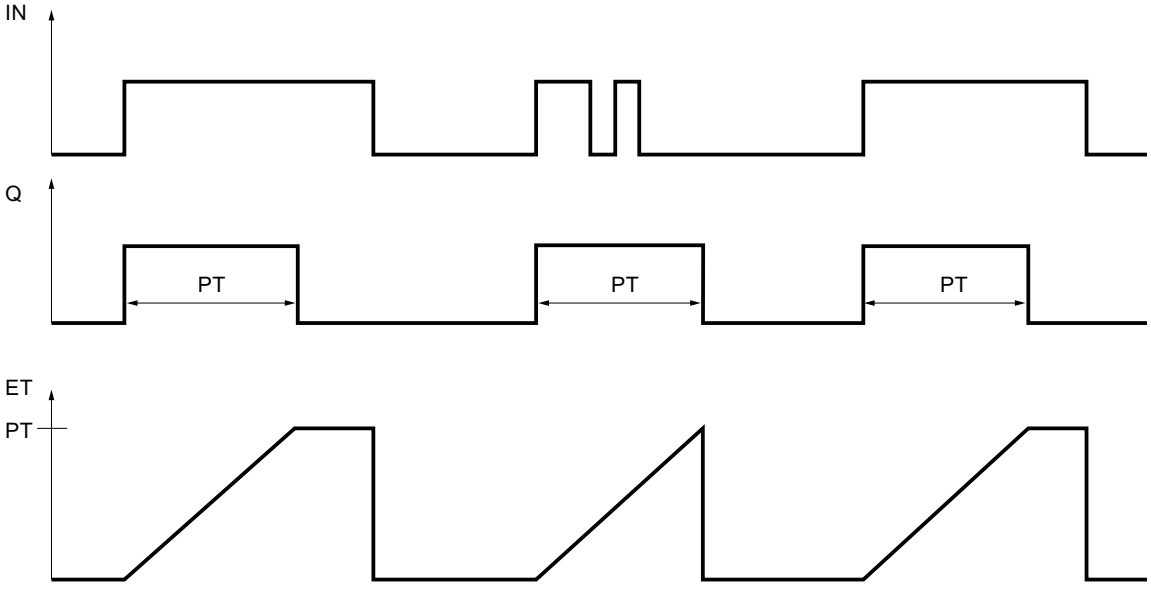
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L, P	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L, P	Operand that is set for the PT duration.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:



Example

The following example shows how the instruction works:

```
SCL
"TP_DB".TP(IN := "Tag_Start",
           PT := "Tag_PresetTime",
           Q => "Tag_Status",
           ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time period programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 1908)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay the setting of the Q parameter for the programmed duration PT. The instruction starts when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the duration PT has expired, the Q parameter returns signal state "1". The Q parameter remains set as long as the start input is still "1". If the signal state of the IN parameter changes from "1" to "0", the parameter Q will be reset. The timer function is restarted when a new positive signal edge is detected at the IN parameter.

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. The ET parameter is reset as soon as the signal state of the IN parameter changes to "0".

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TON_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TON_TIME type in the "Static" section of a block (for example, #MyTON_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or TON_LTIME in the "Static" section of a block (for example #MyTON_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

Syntax

Use the following syntax for the "Generate on-delay" instruction:

- Data block of system data type IEC_Timer (global DB):

```
SCL
<IEC_Timer_DB> TON(IN := <Operand>,
                   PT := <Operand>,
                   Q => <Operand>,
                   ET => <Operand>)
```

- Local tag:

```
SCL
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

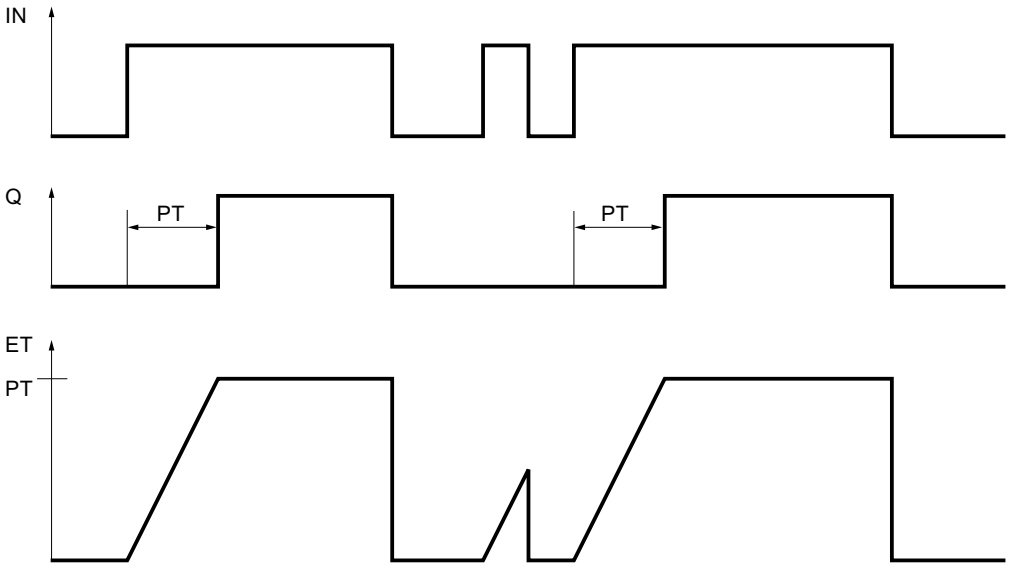
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L, P	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L, P	Operand that is set when the timer PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:



Example

The following example shows how the instruction works:

```
SCL
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. After the end of the period the "Tag_Status" operand is set to the signal state "1". The operand "Tag_Status" remains set to signal state "1" as long as the operand "Tag_Start" has signal state "1". The current time value is stored in the operand "Tag_ElapsedTime". When the signal state at the operand "Tag_Start" changes from "1" to "0", the "Tag_Status" operand is reset.

See also

Overview of the valid data types (Page 1908)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay the resetting of the Q parameter for the programmed duration PT. The Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state of the IN parameter changes back to "0", the programmed time PT starts. The Q parameter remains set as long the time duration PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before the time duration PT has expired, the timer is reset. The signal state of the Q parameter remains set to "1".

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1". If the IN parameter changes to "1" before the time PT has expired, the ET parameter is reset to the value T#0s.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate off-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TOF_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TOF_TIME type in the "Static" section of a block (for example, #MyTOF_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or TOF_LTIME in the "Static" section of a block (for example #MyTOF_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

Syntax

Use the following syntax for the "Generate off-delay" instruction:

- Data block of system data type IEC_Timer (global DB):

```
SCL
<IEC_Timer_DB> TOF(IN := <Operand>,
                   PT := <Operand>,
                   Q => <Operand>,
                   ET => <Operand>)
```

- Local tag:

```
SCL
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

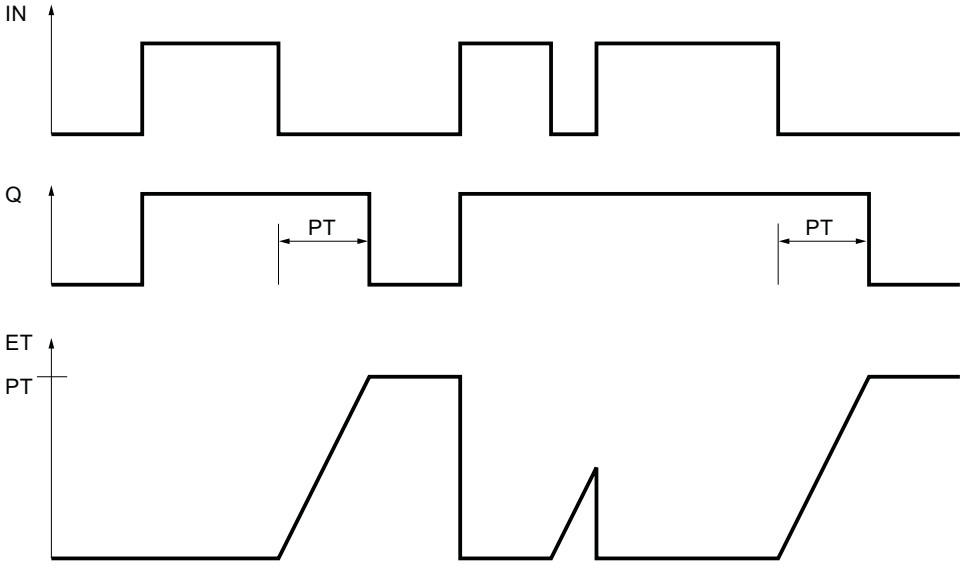
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L, P	Duration of the off delay. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L, P	Operand that is reset when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction "Generate off-delay":



Example

The following example shows how the instruction works:

```
SCL
"TOF_DB".TOF(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

With a change in the signal state of the "Tag_Start" operand from "0" to "1", the "Tag_Status" operand is set. When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set. When the time has expired, the "Tag_Status" operand is reset. The current time value is stored in the operand "Tag_ElapsedTime".

See also

Overview of the valid data types (Page 1908)

TONR: Time accumulator

Description

The "Time accumulator" instruction is used to accumulate time values within a period set by the PT parameter. When the signal state of the IN parameter changes to "1", the instruction executes and the time duration PT starts. While the time duration PT is running, the time values that are recorded when the IN parameter has signal state "1" are accumulated. The accumulated time is output in the ET parameter and can be queried there. When the time duration PT is reached, the Q parameter has signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes to "0".

The R parameter resets the ET and Q parameters regardless of the signal state at the IN parameter.

Each call of the "Time accumulator" instruction must be assigned to an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TONR_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TONR_TIME type in the "Static" section of a block (for example, #MyTONR_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or TONR_LTIME in the "Static" section of a block (for example #MyTONR_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

Syntax

Use the following syntax for the "Time accumulator" instruction:

- Data block of system data type IEC_Timer (global DB):

SCL

```
<IEC_Timer_DB> TONR(IN := <Operand>,
                    R := <Operand>,
                    PT := <Operand>,
                    Q => <Operand>,
                    ET => <Operand>)
```

- Local tag:

SCL

```
#myLocal_timer(IN := <Operand>,
               R := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

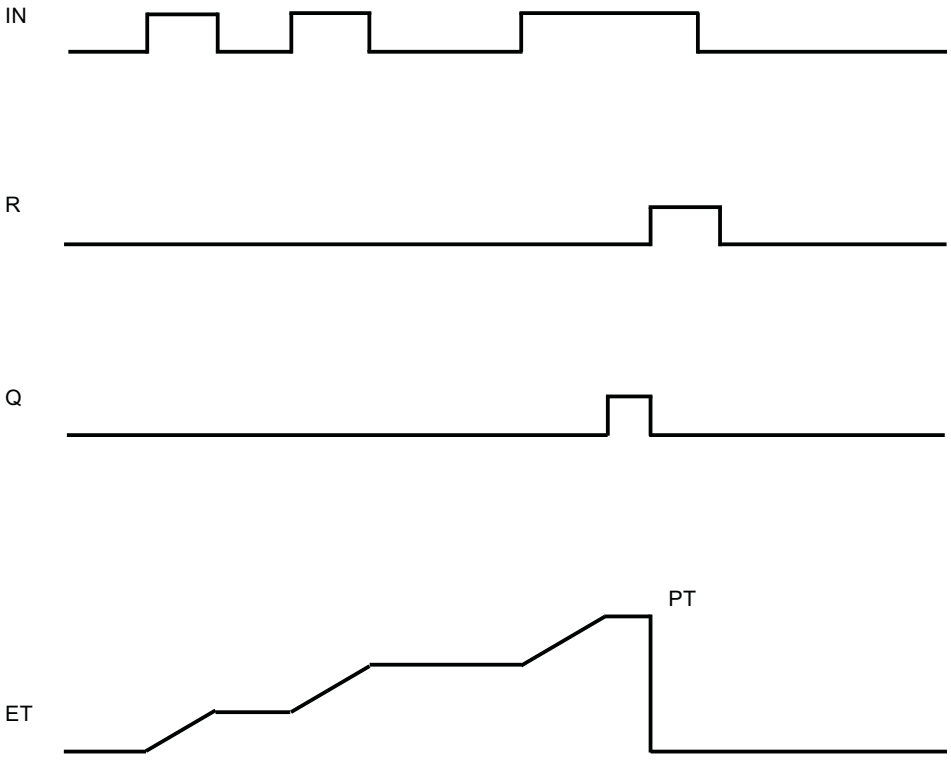
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I, Q, M, D, L, P	Start input
R	Input	BOOL	BOOL	I, Q, M, D, L, P	Reset of the ET and Q parameters
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L, P	Maximum duration of time recording. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L, P	Operand that remains set when the timer PT has expired.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L, P	Accumulated time

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:



Example

The following example shows how the instruction works:

SCL

```
"TONR_DB".TONR(IN := "Tag_Start",
               R := "Tag_Reset",
               PT := "Tag_PresetTime",
               Q => "Tag_Status",
               ET => "Tag_Time");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. While the timer is running, the time values that are recorded at signal state "1" of the operand "Tag_Start" is accumulated. The accumulated times is stored in the "Tag_Time" operand. When the time value displayed at the PT parameter is reached, the "Tag_Status" operand is set to the signal state "1". The current time value is stored in the operand "Tag_Time".

See also

Overview of the valid data types (Page 1908)

RESET_TIMER: Reset timer

Description

You can use the "Reset timer" instruction to reset an IEC timer to "0". The structure components of the timer in the specified data block are reset to "0".

The instruction does not influence the RLO. At the TIMER parameter, the "Reset timer" instruction is assigned an IEC timer declared in the program. The instruction must be programmed in an IF instruction. The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

Syntax

Use the following syntax for the "Reset timer" instruction:

SCL

```
RESET_TIMER(TIMER := <IEC timer>)
```

Parameters

The following table shows the parameters of the "Reset timer" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer that is reset

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF #started = false THEN
```

SCL

```
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
#started := true;
END_IF;

IF "TON_DB".ET < T#25s THEN
RESET_TIMER(TIMER := "TON_DB");
#started := false;
END_IF;
```

When the tag #started has the signal state "0", the instruction "Generate on-delay" is executed when there is positive signal edge on the operand "Tag_Start". The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PresetTime". The operand "Tag_Status" is set if the time duration specified by the operand "Tag_PresetTime" has expired. The parameter Q remains set as long as the operand "Tag_Start" still has the signal state "1". When the signal state of the start input changes from "1" to "0", the operand on the Q parameter is reset.

If the expired time of the IEC timer "TON_DB" is less than 25s, the "Reset timer" instruction is executed and the timer stored in the "TON_DB" instance data block is reset.

See also

Overview of the valid data types (Page 1908)

PRESET_TIMER: Load time duration

Description

You can use the "Load time duration" instruction to set the time for an IEC timer. The instruction is executed in every cycle when the result of logic operation (RLO) at the input of the instruction has the signal state "1". The instruction writes the specified time to the structure of the specified IEC timer.

Note

If the specified IEC timer is running while the instruction executes, the instruction overwrites the current time of the specified IEC timer. This can change the timer status of the IEC timer.

You assign an IEC timer declared in the program to the "Load time duration" instruction.

The instruction data is updated only when the instruction is called and each time the assigned IEC timer is accessed. The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Syntax

Use the following syntax for the "Load time duration" instruction:

SCL

```
PRESET_TIMER(PT := <Operand>,
             TIMER := <IEC timer)
```

Parameter

The following table shows the parameters of the "Load time duration" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L	Duration with which the IEC timer runs.
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer whose duration of which is set.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF #started = false THEN
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
#started := true;
#preset = true
END_IF;

IF "TON_DB".ET < T#10s AND #preset = true THEN
```

SCL

```
PRESET_TIMER(PT := T#25s,
             TIMER := "TON_DB");
#preset := false;
END_IF;
```

When the tag #started has the signal state "0" and the operand "Tag_Start" has a positive signal edge, the instruction "Generate on-delay" is executed. The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PresetTime". The operand "Tag_Status" is set if the time duration PT specified by the operand "Tag_PresetTime" has expired. The parameter Q remains set as long as the operand "Tag_Start" still has the signal state "1". When the signal state of the start input changes from "1" to "0", the operand on the Q parameter is reset.

The instruction "Load time duration" is executed when the expired time of the IEC timer "TON_DB" is less than 10s and the tag #preset has the signal state "1". The instruction writes the time duration that is specified at the parameter PT in the instance data block "TON_DB", thereby overwriting the time value of the operand "Tag_PresetTime" within the instance data block. The signal state of the timer status may therefore change at the next query or when "TON_DB".Q or "TON_DB".ET are accessed.

See also

Overview of the valid data types (Page 1908)

Legacy

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts the time programmed in the T_NO parameter when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) of the S parameter. The timer runs for the programmed time (TV) as long as the signal state of the S parameter is "1".

When the signal state of the S parameter changes to "0" before the programmed time has expired, the timer is stopped and the Q parameter is reset to "0".

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

Parameter Q returns signal state "1" as long as the timer is running and the signal state at parameter S is "1". When the signal state of the S parameter changes to "0" before the

programmed time has expired, the Q parameter returns signal state "0". If the timer is reset by parameter R or if the timer has expired then parameter Q also returns signal state "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign pulse timer parameters and start" instruction:

```
SCL
S_PULSE(T_NO := <Operand>,
        S := <Operand>,
        TV := <Operand>,
        R := <Operand>,
        Q => <Operand>,
        BI => <Operand>)
```

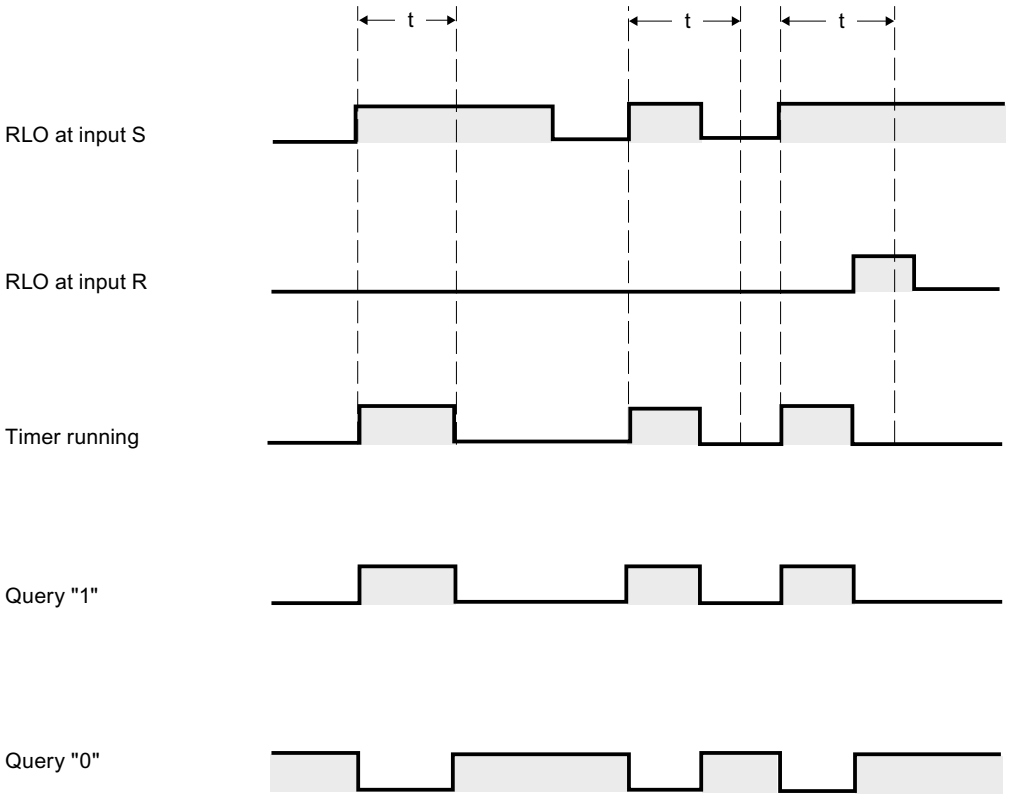
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign pulse timer parameters and start" instruction:



t = programmed time

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_PULSE(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q := "Tag_Status",
                        BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer counts down with the time value of the operand "Tag_Number" until operand "Tag_1" returns signal state "1".

If the signal state at the S parameter changes to "0" before the programmed time has elapsed, the "Tag_Status" operand is reset to "0". If the timer is reset by the R parameter or if the timer has expired, the "Tag_Status" operand also returns signal state "0".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 1908)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) even if the signal state at the S parameter changes to "0". As long as the timer runs, parameter Q returns the signal state "1".

When the timer has expired, parameter Q is reset to "0". If the signal state at the S parameter changes from "0" to "1" while the timer is running, the timer is restarted with the time programmed in the TV parameter.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the timer is running and the signal state at parameter R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running then signal state "1" at parameter R has no effect.

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign extended pulse timer parameters and start" instruction:

SCL

```
S_PEXT(T_NO := <Operand>,
       S := <Operand>,
       TV := <Operand>,
       R := <Operand>,
       Q => <Operand>,
       BI =><Operand>)
```

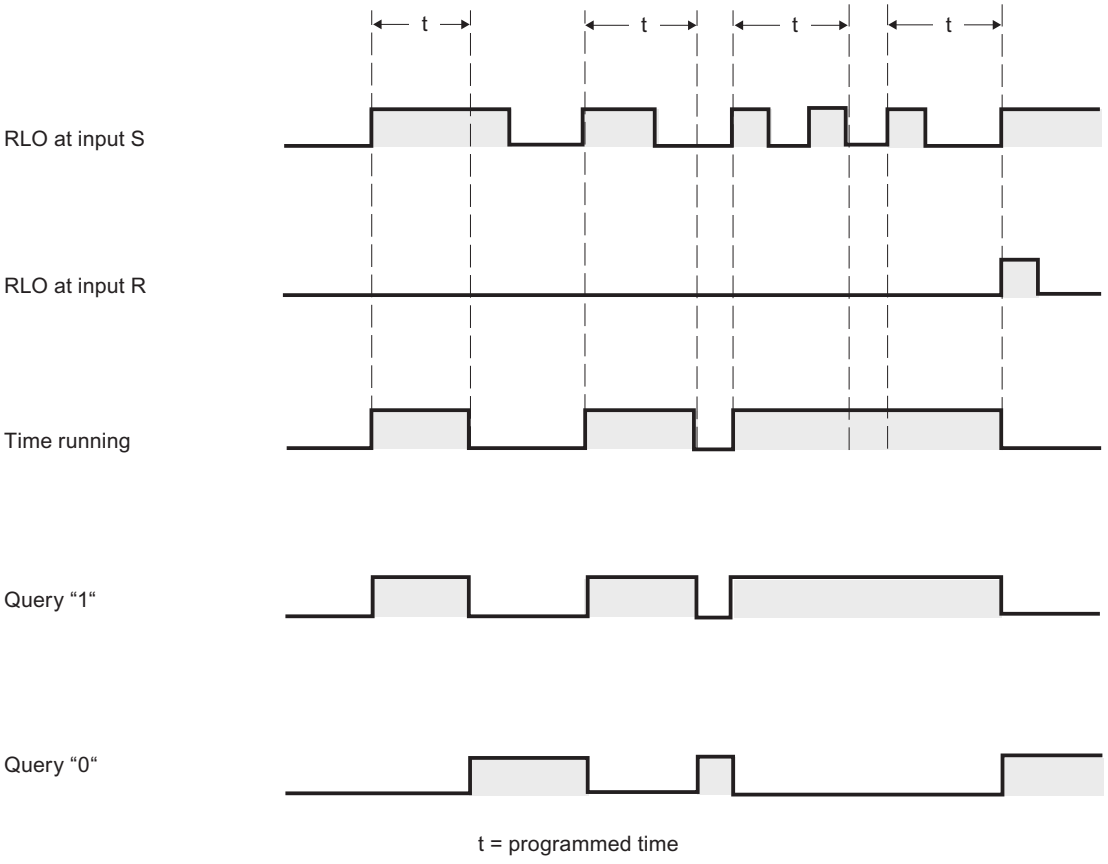
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:



t = programmed time

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_PEXT(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q := "Tag_Status",
                       BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". As long as the timer runs, operand "Tag_Status" returns the signal state "1". When the timer has expired, operand "Tag_Status" is reset to "0". If the signal state at the S input changes from "0" to "1" while the timer is running, the timer is restarted with the time "Tag_Number".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 1908)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer as on-delay when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) as long as the signal state of the S parameter is "1".

If the timer expires correctly and parameter S still has signal state "1" then parameter Q returns signal state "1". If the signal state at the S parameter changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the time is running and the signal state at input R changes from "0" to "1" then the current time value and the time base are also set to zero. In this case, the signal state at parameter Q is "0". The timer is reset if the signal state at the R parameter is "1", even if the timer is not running and the result of logic operation (RLO) at the S parameter is "1".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign on-delay timer parameters and start" instruction:

SCL

```
S_ODT(T_NO := <Operand>,
      S := <Operand>,
      TV := <Operand>,
      R := <Operand>,
      Q => <Operand>,
      BI =><Operand>)
```

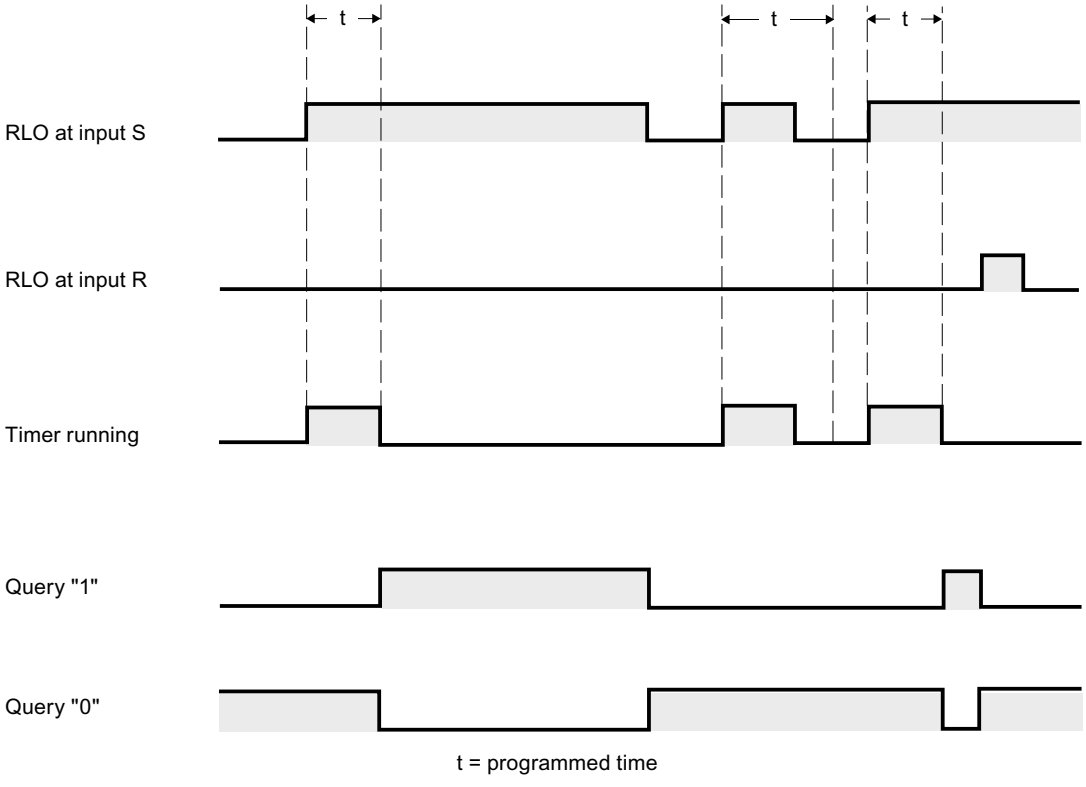
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := S_ODT(T_NO := "Timer_1",
                      S := "Tag_1",
                      TV := "Tag_Number",
                      R := "Tag_Reset",
                      Q := "Tag_Status",
                      BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number" as long as the signal state of operand "Tag_1" is "1".

If the timer expires correctly and operand "Tag_Status" has signal state "1" then operand "Tag_Status" is reset to "1". If the signal state of the "Tag_1" operand changes from "1" to "0" while the timer is running, the timer is stopped. In this case, operand "Tag_Status" returns signal state "0".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 1908)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) even if the signal state at the S parameter changes to "0".

If the timer expires, the Q parameter returns signal state "1" regardless of the signal state of the S parameter. If the signal state at the S parameter changes from "0" to "1" while the timer is running, the timer is restarted with the programmed time TV.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

Signal state "1" at parameter R resets the current time value and time base to "0", independent of the signal state at parameter S. In this case, the signal state at parameter Q is "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign retentive on-delay timer parameters and start" instruction:

SCL

```
S_ODTS(T_NO := <Operand>,
       S := <Operand>,
       TV := <Operand>,
       R := <Operand>,
       Q => <Operand>,
       BI =><Operand>)
```

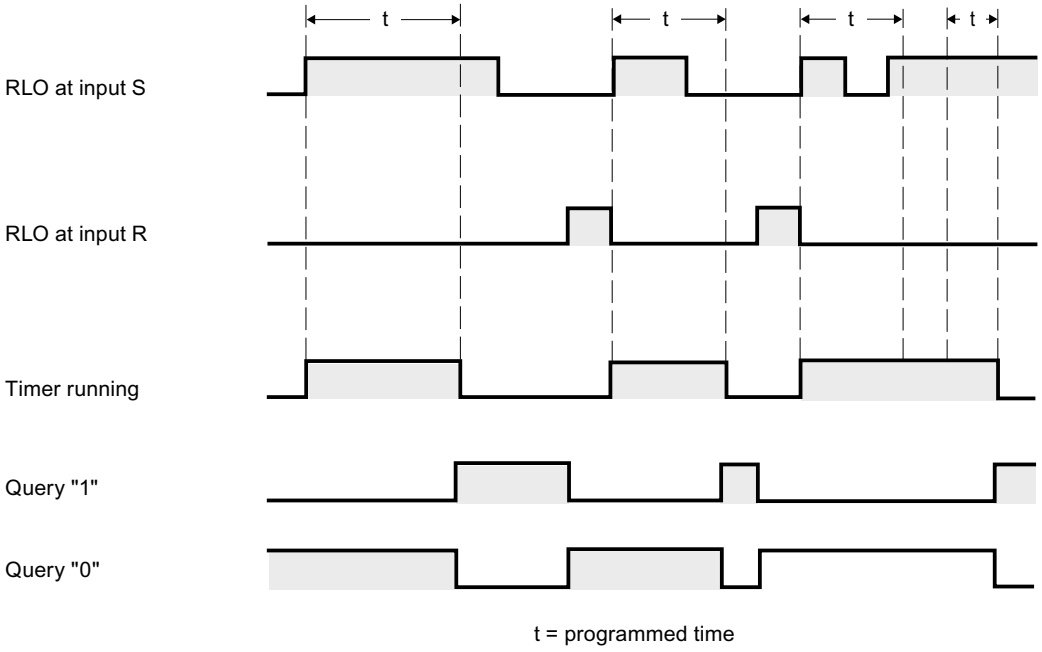
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_ODTS(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q := "Tag_Status",
                       BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number".

If the timer expires, operand "Tag_Status" returns signal state "1" independent of the signal state of operand "Tag_1". If the signal state of the "Tag_1" operand changes from "0" to "1" while the timer is running, the timer is restarted with the time "Tag_Number".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 1908)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a negative signal edge is detected at the S parameter. The timer runs for the programmed time (TV). As long as the timer is running or parameter S returns signal state "1", then parameter Q has signal state "1".

If the timer expires and the signal state is "0" then parameter Q is reset to signal state "0". If the signal state at parameter S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at parameter S.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

Signal state "1" at parameter R resets the current time value and time base to "0". In this case, the signal state at parameter Q is "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign off-delay timer parameters and start" instruction:

SCL

```
S_OFFDT(T_NO := <Operand>,
        S := <Operand>,
        TV := <Operand>,
        R := <Operand>,
        Q => <Operand>,
        BI =><Operand>)
```

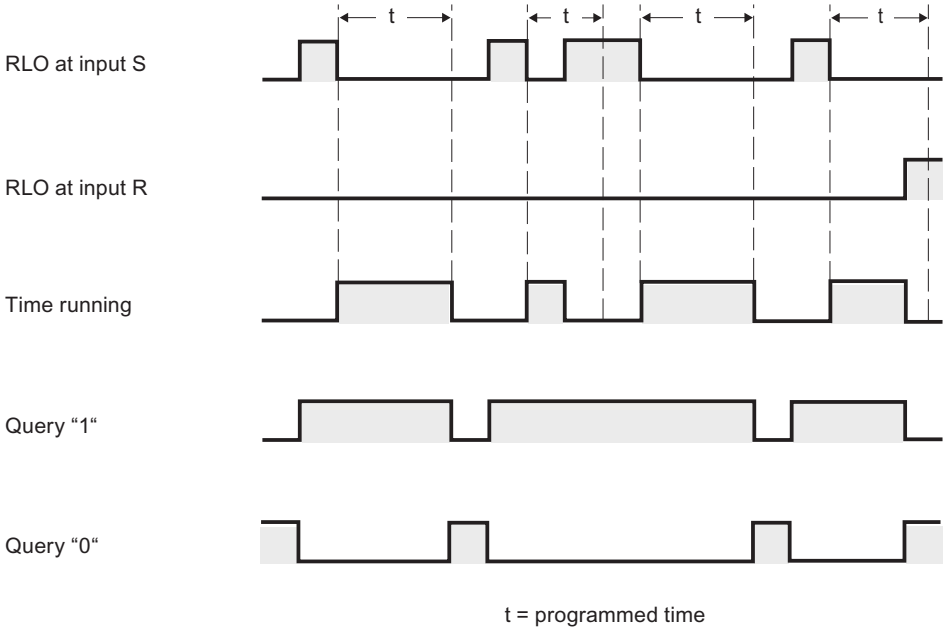
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := S_OFFDT(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q := "Tag_Status",
                        BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number". As long as the timer is running or operand "Tag_1" returns signal state "1", then operand "Tag_Status" has signal state "1".

If the timer expires and the signal state of operand "Tag_1" is "0" then operand "Tag_Status" is reset to signal state "0". If the signal state of the "Tag_1" operand changes from "0" to "1" while the timer is running, the timer is reset. The timer is only restarted after a falling edge is detected at parameter S.

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 1908)

Counter operations

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at the CV parameter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the parameter CV is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit of the data type specified at the CV parameter. When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current counter value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the signal state of the R parameter is "1", the signal state of the CU parameter has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data type> in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

Syntax

The following syntax is used for the "Count up" instruction:

Table 11-28 Data block of system data type IEC_counter (global DB)

SCL

```
<IEC_Counter_DB>.CTU(CU := <Operand>,
                     R := <Operand>,
                     PV := <Operand>,
                     Q => <Operand>,
                     CV => <Operand>)
```

Table 11-29 Local tag

SCL

```
#myLocal_counter(CU := <Operand>,
                 R := <Operand>,
                 PV := <Operand>,
                 Q => <Operand>,
                 CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, P	Reset input
PV	Input	Integers	I, Q, M, D, L, P	Value at which the Q output is set
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

SCL

```
"IEC_COUNTER_DB".CTU(CU := "Tag_Start",
                     R := "Tag_Reset",
                     PV := "Tag_PresetValue",
                     Q => "Tag_Status",
                     CV => "Tag_CounterValue");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the specified data type (INT = 32767) is reached.

The "Tag_Status" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" operand. In all other cases, the

"Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 1908)

CTD: Count down

Description

The "Count down" instruction is used to decrement the value at the parameter CV. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the signal state of the LD parameter is "1", the signal state of the CD parameter has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER	• CTD_SINT / CTD_USINT
• IEC_COUNTER / IEC_UCOUNTER	• CTD_INT / CTD_UINT
• IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_DINT / CTD_UDINT
• IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_LINT / CTD_ULINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data type> in the "Static" section of a block (for example, #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

Syntax

The following syntax is used for the "Count down" instruction:

Table 11-30 Data block of system data type IEC_counter (global DB)

SCL

```
<IEC_Counter_DB>.CTD(CD := <Operand>,
                     LD := <Operand>,
                     PV := <Operand>,
                     Q => <Operand>,
                     CV => <Operand>)
```

Table 11-31 Local tag

SCL

```
#myLocal_counter(CD := <Operand>,
                 LD := <Operand>,
                 PV := <Operand>,
                 Q => <Operand>,
                 CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	Integers	I, Q, M, D, L, P	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

```
SCL
"IEC_SCOUNTER_DB".CTD(CD := "Tag_Start",
                      LD := "Tag_Load",
                      PV := "Tag_PresetValue",
                      Q => "Tag_Status",
                      CV => "Tag_CounterValue");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the instruction is executed and the value of the "Tag_CounterValue" operand is decremented by one. With each additional positive signal edge, the counter value is decremented until it reaches the low limit of the specified data type (-128).

The operand "Tag_Status" has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 1908)

CTUD: Count up and down

Description

Use the "Count up and down" instruction to increment or decrement the counter value at the CV parameter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the counter value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in a program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV parameter. When the high limit is reached, the counter value is no longer incremented

on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state of the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has signal state "1", the signal state of the CU and CD parameters has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0". You can also specify a constant for the PV parameter.

You can query the status of the down counter at the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER	• CTUD_SINT / CTUD_USINT
• IEC_COUNTER / IEC_UCOUNTER	• CTUD_INT / CTUD_UINT
• IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_DINT / CTUD_UDINT

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER	• CTUD_SINT / CTUD_USINT
• IEC_COUNTER / IEC_UCOUNTER	• CTUD_INT / CTUD_UINT
• IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_DINT / CTUD_UDINT
• IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_LINT / CTUD_ULINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data type> in the "Static" section of a block (for example, #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

Syntax

The following syntax is used for the "Count up and down" instruction:

Table 11-32 Data block of system data type IEC_counter (global DB)

SCL

```
<IEC_Counter_DB>.CTUD(CU := <Operand>,
                      CD := <Operand>,
                      R := <Operand>,
                      LD := <Operand>,
                      PV := <Operand>,
                      QU=> <Operand>,
                      QD := <Operand>,
                      CV => <Operand>)
```

Table 11-33 Local tag

SCL

```
myLocal_counter(CU := <Operand>,
                CD := <Operand>,
                R := <Operand>,
                LD := <Operand>,
                PV := <Operand>,
                QU=> <Operand>,
                QD := <Operand>,
                CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	Integers	I, Q, M, D, L, P	Value at which the QU output is set. / Value to which the CV output is set with LD = 1.

Parameter	Declaration	Data type	Memory area	Description
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

```
SCL
"IEC_COUNTER_DB".CTUD(CU := "Tag_Start1",
                      CD := "Tag_Start2",
                      LD := "Tag_Load",
                      R := "Tag_Reset",
                      PV := "Tag_PresetValue",
                      QU => "Tag_CU_Status",
                      QD => "Tag_CD_Status",
                      CV => "Tag_CounterValue");
```

If the "Tag_Start1" operand has a positive signal edge in the signal state, the current counter value is incremented by one and stored in the "Tag_CounterValue" operand. If the "Tag_Start2" operand has a positive signal edge in the signal state, the counter value is decremented by one and is also stored in the "Tag_CounterValue" operand. The counter value is incremented on the positive signal edge of the CU parameter until it reaches the high limit of the specified data type (INT). If the CD parameter has a positive signal edge, the counter value is decremented until it reaches the low limit of the specified data type (INT).

The operand "Tag_CU_Status" has signal state "1" as long as the current counter value is greater than or equal to the value of the operand "Tag_PresetValue". In all other cases, the "Tag_CU_Status" output has signal state "0".

The operand "Tag_CD_Status" has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_CD_Status" output has signal state "0".

See also

Overview of the valid data types (Page 1908)

Legacy

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is provided at the parameter CV. The counter value is incremented until the limit of "999" is reached. When the limit value is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state of the parameter S changes from "0" to "1", the counter value is set to the value of the parameter PV. If the counter is set and if the result of logic operation (RLO) at the CU input is "1", the counter counts once in the next cycle, even when no signal edge change is detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the parameter R has the signal state "1", a change in the signal state of the parameters CU and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

To avoid counting errors, only use a counter at a single location in the program.

Syntax

The following syntax is used for the "Assign parameters and count up" instruction:

SCL

```
S_CU(C_NO := <Operand>,
     CU := <Operand>,
     S := <Operand>,
     PV := <Operand>,
     R := <Operand>,
     Q => <Operand>,
     CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
C_NO	Input	COUNTER, INT	C	Counters The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
S	Input	BOOL	I, Q, M, D, L	Input for presetting the counter

Parameter	Declaration	Data type	Memory area	Description
PV	Input	WORD	I, Q, M, D, L	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	I, Q, M, D, L	Reset input
Q	Output	BOOL	I, Q, M, D, L	Status of the counter
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value
Function value		WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := S_CU(C_NO := "Counter_1",
                     CU := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

If the signal state of the "Tag_Start" parameter changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. If the signal state of the input "Tag_1" changes from "0" to "1", the counter value in BCD format is set to the value of the operand "Tag_PresetValue". The counter value is reset to "0" when the operand "Tag_Reset" has signal state "1".

The current counter value is stored in hexadecimal form in the operand "Tag_Value".

The output "Tag_Status" has the signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 1908)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. When the signal state of the parameter CD changes from "0" to "1" (positive signal edge), the current counter value is decreased by one. The current counter value is provided at the parameter CV. The counter value is decreased until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decreased on a positive signal edge.

When the signal state of the parameter S changes from "0" to "1", the counter value is set to the value of the parameter PV. If the counter is set and if the result of logic operation (RLO) at the parameter CD is "1", the counter counts once in the next cycle, even when no signal edge change is detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the parameter R has the signal state "1", a change in the signal state of the parameters CD and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

To avoid counting errors, only use a counter at a single location in the program.

Syntax

The following syntax is used for the "Assign parameters and count down" instruction:

SCL

```
S_CD(C_NO:= <Operand>,
     CD := <Operand>,
     S := <Operand>,
     PV := <Operand>,
     R := <Operand>,
     Q => <Operand>,
     CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
C_NO	Input	COUNTER, INT	C	Counters The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L	Count down input
S	Input	BOOL	I, Q, M, D, L	Input for presetting the counter

Parameter	Declaration	Data type	Memory area	Description
PV	Input	WORD	I, Q, M, D, L	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	I, Q, M, D, L	Reset input
Q	Output	BOOL	I, Q, M, D, L	Status of the counter
CV	Output	WORD, S5TIME, WORD	I, Q, M, D, L	Current counter value
Function value		WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CD(C_NO := "Counter_1",
                     CD := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

When the signal state of the operand "Tag_Start" changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decreased by one. If the signal state of the operand "Tag_1" changes from "0" to "1", the counter value is set in BCD format to the value of the operand "Tag_PresetValue". The counter value is reset to "0" when the operand "Tag_Reset" has signal state "1".

The current counter value is stored in the operand "Tag_Value".

The operand "Tag_Status" returns signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 1908)

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment and decrement the value of a counter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. When the signal state of the parameter CD changes from "0" to "1" (positive signal edge), the counter value is decreased by one. The current counter value is provided at the parameter CV. If there is a positive signal edge at the parameters CU and CD in one program cycle, the counter value remains unchanged.

The counter value is incremented until the high limit of "999" is reached. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit "0" is reached, the counter value is not decremented any further.

When the signal state of the parameter S changes from "0" to "1", the counter value is set to the value of the parameter PV. If the counter is set and if the result of logic operation (RLO) of the parameters CU and CD is "1", the counter will count once in the next cycle, even if no signal edge change was detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the parameter R has the signal state "1", processing of the signal state of the parameters CU, CD and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

To avoid counting errors, only use a counter at a single location in the program.

Syntax

The following syntax is used for the "Assign parameters and count up / down" instruction:

SCL

```
S_CUD(C_NO:= <Operand>,
      CU := <Operand>,
      CD := <Operand>,
      S := <Operand>,
      PV := <Operand>,
      R := <Operand>,
      Q => <Operand>,
      CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
C_NO	Input	COUNTER, INT	C	Counters The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
S	Input	BOOL	I, Q, M, D, L	Input for presetting the counter
PV	Input	WORD	I, Q, M, D, L	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	I, Q, M, D, L	Reset input
Q	Output	BOOL	I, Q, M, D, L	Status of the counter
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
Function value		WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CD(C_NO := "Counter_1",
                     CU := "Tag_CU",
                     CD := "Tag_CD",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

When a positive signal edge is detected in the signal state of the operand "Tag_CU" and the current counter value is less than "999", the counter value is incremented by one. When a positive signal edge is detected in the signal state of the operand "Tag_CD" and the current counter value is greater than "0", the counter value is decremented by one.

If the signal state of the operand "Tag_1" changes from "0" to "1", the counter value is set in BCD format to the value of the operand "Tag_PresetValue". The counter value is reset to "0" when the operand "Tag_Reset" has signal state "1".

The current counter value is stored in the operand "Tag_Value".

The operand "Tag_Status" returns signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 1908)

Comparator operations

TypeOf: Check data type of a VARIANT tag

Description

You can use the "Check data type of a VARIANT tag" instruction to query the data type of a tag to which a VARIANT tag is pointing. You can compare the data type of the <Operand> tag that you have declared in the block interface either to the data type of another tag or directly with a data type to determine whether they are "Equal" or "Not equal".

The operand must have the VARIANT data type. The comparison operand can be an elementary data type or a PLC data type.

You can only use the "Check data type of a VARIANT tag" instruction within an IF instruction.

Syntax

The following syntax is used for the "Check data type of a VARIANT tag" instruction:

SCL

```
TypeOf(<Operand>)
```

Parameters

The following table shows the parameters of the "Check data type of a VARIANT tag" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	I, Q, M, L	First operand

For additional information on valid data types, refer to "See also".

Example

The following example shows the comparison with another tag:

SCL

```
IF TypeOf(#TagVARIANT) = TypeOf("TagBYTE") THEN
...;
END_IF;
```

The following example shows the comparison with a data type:

SCL

```
IF TypeOf(#TagVARIANT) = BYTE THEN
...;
END_IF;
```

If the operand to which the VARIANT #TagVARIANT is pointing has the BYTE data type, the comparison condition has been met.

See also

Overview of the valid data types (Page 1908)

Programming example: Programming queues (FIFO) (Page 270)

TypeOfElements: Check data type of an ARRAY element of a VARIANT tag

Description

You can use the "Check data type of an ARRAY element of a VARIANT tag" instruction to query the data type of a tag to which a VARIANT tag is pointing. You compare the data type of the tag to the data type of a tag that you have declared in the block interface to determine whether they are "Equal" or "Not equal".

The operand must have the VARIANT data type. The comparison operand can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

You can only use the instruction within an IF instruction.

Syntax

The following syntax is used for the "Check data type of an ARRAY element of a VARIANT tag" instruction:

SCL

```
TypeOfElements(<Operand>)
```

Parameters

The following table shows the parameters of the "Check data type of an ARRAY element of a VARIANT tag" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	I, Q, M, L	First operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF TypeOfElements(#Tag_VARIANT) = TypeOf("GlobalDB".Product[1]) THEN
"Tag_Result" := "GlobalDB".Product[1] * 3;
END_IF;
```

The following table shows how the instruction works using specific operand values:

Operand	Value
#Tag_VARIANT	1.5
"GlobalDB".Product[1]	3.5

If the tag to which the VARIANT points and the "GlobalDB".Product[1] operand have the REAL data type, the "GlobalDB".Product[1] operand is then multiplied by 3 and the result is written to the "Tag_Result" operand.

See also

Overview of the valid data types (Page 1908)

Programming example: Programming queues (FIFO) (Page 270)

IS_ARRAY: Check for ARRAY

Description

You can use the "Check for ARRAY" instruction to query whether the VARIANT points to a tag of the ARRAY data type.

The <Operand> must have the VARIANT data type.

You can only use the "Check for ARRAY" instruction within an IF instruction.

Syntax

The following syntax is used for the "Check for ARRAY" instruction:

SCL

```
IS_ARRAY(<Operand>)
```

Parameters

The following table shows the parameters of the "Check for ARRAY" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	I, Q, M, L	Operand that is queried for ARRAY
Function value		UDINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF IS_ARRAY(#Tag_VARIANTToArray) THEN
"Tag_Result" := CountOfElements(#Tag_VARIANT);
END_IF;
```

If the tag to which the VARIANT points is an ARRAY, the number of ARRAY elements is output.

See also

Overview of the valid data types (Page 1908)

Programming example: Programming queues (FIFO) (Page 270)

Math functions

ABS: Form absolute value

Description

Use the "Form absolute value" instruction to calculate the absolute value of an input value and to save the result in the specified operands.

Syntax

Use the following syntax for the "Form absolute value" instruction:

SCL

```
ABS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Expression>	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Input value
Function value		SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := ABS("Tag_Value");
"Tag_Result2" := ABS("Tag_Value1"*"Tag_Value2");
```

The absolute value of the input value is returned in the format of the input value as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	-2
Tag_Result1	2
Tag_Value1	4
Tag_Value2	-1
Tag_Result2	4

See also

Overview of the valid data types (Page 1908)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values of the available inputs and returns the lowest value as the result.

A minimum of two and a maximum of 32 inputs can be specified at the instruction.

The result is invalid if any of the following conditions are met:

- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Syntax

The following syntax is used for the "Get minimum" instruction:

```
SCL
MIN(IN1:= <Operand>,
    IN2 := <Operand>)
```

Parameter

The following table shows the parameters of the "Get minimum" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Second input value
INn	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Additionally inserted inputs whose values are to be compared
Function value		Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Result of the instruction
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := MIN(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	12222
IN2	Tag_Value2	14444
IN3	Tag_Value3	13333
Function value	Tag_Result	12222

The instruction compares the values of the available inputs and copies the lowest value ("Tag_Value1") to operand "Tag_Result".

See also

Overview of the valid data types (Page 1908)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values of the available inputs and returns the greatest value as the result.

A minimum of two and a maximum of 32 inputs can be specified at the instruction.

The result is invalid if any of the following conditions are met:

- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Syntax

The following syntax is used for the "Get maximum" instruction:

```
SCL
MAX(IN1:= <Operand>,
    IN2 := <Operand>)
```

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Second input value
INn	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Additionally inserted inputs whose values are to be compared
Function value		Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Result of the instruction
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MAX(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	12 222
IN2	Tag_Value2	14 444
IN3	Tag_Value3	13 333
Function value	Tag_Result	14 444

The instruction compares the values of the specified operands and copies the greatest value ("Tag_Value2") to the operand "Tag_Result".

See also

Overview of the valid data types (Page 1908)

LIMIT: Set limit value

Description

The "Set limit value" instruction limits the value of the parameter IN to the values of the parameters MN and MX. The value of the parameter MN may not be greater than the value of the parameter MX.

If the value of the IN parameter fulfills the MN condition <=IN <= MX, it is returned as the result of the instruction. If the condition is not fulfilled and the IN input value is less than the MN low limit, the value of the MN parameter is returned as the result. If the high limit MX is exceeded, the value of the MX parameter is returned as the result.

If the value of the MN input is greater than the value of the MX input, the result is undefined.

The instruction is only executed if the operands of all parameters are of the same data type.

Syntax

The following syntax is used for the "Set limit value" instruction:

SCL

```
LIMIT(MN := <Operand>,
      IN := <Operand>,
      MX := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
MN	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Low limit
IN	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Input value
MX	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	High limit
Function value		Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Result of the instruction
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := LIMIT(MN := "Tag_Minimum",
                      IN := "Tag_Value",
                      MX := "Tag_Maximum");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_Minimum	12 000
IN	Tag_Value	8 000
MX	Tag_Maximum	16 000
Function value	Tag_Result	12 000

The value of operand "Tag_Value" is compared with the values of operands "Tag_Minimum" and "Tag_Maximum". Because the value of the operand "Tag_Value" is less than the low limit value, the value of operand "Tag_Minimum" will be copied to operand "Tag_Result".

See also

Overview of the valid data types (Page 1908)

SQR: Form square

Description

Use the "Form square" instruction to square the input value and save the result in the specified operand.

Syntax

Use the following syntax for the instruction "Form square":

SCL

```
SQR(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := SQR("Tag_Value");
"Tag_Result2" := SQR((SQR("Tag_Value1"))*"Tag_Value2");
```

The square of the input value is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	6.25
Tag_Value1	6.0
Tag_Value2	2.0
Tag_Result2	5184.0

See also

Overview of the valid data types (Page 1908)

SQRT: Form square root

Description

Use the "Form square root" instruction to calculate the square root of the input value and save the result in the specified operand. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number. If the input value is "0", the result is also "0".

Syntax

The following syntax is used for the "Form square root" instruction:

SCL

```
SQRT(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := SQRT("Tag_Value");
"Tag_Result2" := SQRT((SQR("Tag_Value1"))+"Tag_Value2");
```

The square root of the input value is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	4.0
Tag_Result1	2.0
Tag_Value1	3.0
Tag_Value2	16.0
Tag_Result2	5.0

See also

Overview of the valid data types (Page 1908)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e=2.718282) from the input value. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number.

Syntax

The following syntax is used for the "Form natural logarithm" instruction:

SCL

```
LN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := LN("Tag_Value");
"Tag_Result2" := LN("Tag_Value1"+"Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	0.916
Tag_Value1	1.5
Tag_Value2	3.2
Tag_Result2	1.548

See also

Overview of the valid data types (Page 1908)

EXP: Form exponential value

Description

The "Form exponential value" instruction calculates the exponent from the base e (e = 2,718282) and the input value and saves the result in the specified operand.

Syntax

The following syntax is used for the "Form exponential value" instruction:

SCL

```
EXP(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Exponential value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := EXP("Tag_Value");
"Tag_Result2" := EXP("Tag_Value1"/"Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	20.5
Tag_Result1	799 902 200
Tag_Value1	15.5
Tag_Value2	30.2
Tag_Result2	1.671

See also

Overview of the valid data types (Page 1908)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the input value. The input value must be given in radians.

Syntax

Use the following syntax for the "Form sine value" instruction:

SCL

```
SIN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value (size of an angle in radians)
Function value		Floating-point numbers	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SIN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796 (π/2)
Tag_Result	1.0

See also

Overview of the valid data types (Page 1908)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the input value. The input value must be given in radians.

Syntax

Use the following syntax for the "Form cosine value" instruction:

SCL

```
COS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value (size of an angle in radians)
Function value		Floating-point numbers	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := COS("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796 (π/2)
Tag_Result	0

See also

Overview of the valid data types (Page 1908)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the sine of the input value. The input value must be given in radians.

Syntax

Use the following syntax for the "Form tangent value" instruction:

SCL

```
TAN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value (size of an angle in radians)
Function value		Floating-point numbers	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := TAN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+3.141593 (π)
Tag_Result	0

See also

Overview of the valid data types (Page 1908)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from a sine value, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in radians and can range in value from -π/2 to +π/2.

Syntax

Use the following syntax for the "Form arcsine value" instruction:

SCL

```
ASIN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Sine value
Function value		Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ASIN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+1.570796 (π/2)

See also

Overview of the valid data types (Page 1908)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from a cosine value, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in radians and can range in value from 0 to +π.

Syntax

Use the following syntax for the "Form arccosine value" instruction:

SCL

```
ACOS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Cosine value
Function value		Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ACOS("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	0
Tag_Result	+1.570796 (π/2)

See also

Overview of the valid data types (Page 1908)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from a tangent value, which corresponds to this value. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) as input value. The calculated angle size is given in radians and can range in value from -π/2 to +π/2.

Syntax

The following syntax is used for the "Form arctangent value" instruction:

SCL

```
ATAN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Tangent value
Function value		Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ATAN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+0.785398 (π/4)

See also

Overview of the valid data types (Page 1908)

Invalid floating-point numbers (Page 1927)

FRAC: Return fraction

Description

The result of the instruction "Return fraction" returns the decimal places of a value. Input value 123.4567, for example, returns the value 0.4567.

Syntax

The following syntax is used for the instruction "Return fraction":

SCL

```
FRAC(<Expression>)
FRAC_<Data type>(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
_<Data type>		Floating-point numbers Default: REAL	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Floating-point numbers	I, Q, M, D, L, P	Decimal places of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := FRAC("Tag_Value");
"Tag_Result2" := FRAC_LREAL("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	2.555	-1.4421
Tag_Result1	0.555	-0.4421

See also

Overview of the valid data types (Page 1908)

Move operations

Deserialize: Deserialize

Description

You can use the "Deserialize" instruction to convert the sequential representation of a PLC data type (UDT) back to a PLC data type and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type is located must have the ARRAY of BYTE data type and be declared with standard access. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion.

The instruction enables you to convert multiple sequential representations of converted PLC data types back to their original state step-by-step.

If you only want to convert back a single sequential representation of a PLC data type (UDT), you can also use the instruction "TRCV: Receive data via communication connection" directly.

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", the bit has a length of 1 byte.

Syntax

The following syntax is used for the "Deserialize" instruction:

SCL

```
Deserialize(SRC_ARRAY := <Operand>,
            DEST_VARIABLE:= <Operand>,
            POS := <Operand>)
```

Parameters

The following table shows the parameters of the "Deserialize" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC_ARRAY	Input	VARIANT	I, Q, M, L	Global data block in which the generated data stream is stored
DEST_VARIABLE	InOut	VARIANT	I, Q, M, L	Tag in which the returned converted PLC data type (UDT) is stored
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
Function value		INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The data block at the DEST_VARIABLE parameter is not a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains no value.
8151	Code generation error at SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8250	The VARIANT data type at the DEST_VARIABLE parameter contains no value.
8251	Code generation error at DEST_VARIABLE parameter
8254	Invalid data type at the DEST_VARIABLE parameter
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                           DEST_VARIABLE := "Target".Client,
                           POS := #BufferPos);

#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                           DEST_VARIABLE := #Label,
```

SCL

```
                              POS := #BufferPos);

IF #Label = 'arti' THEN
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                              DEST_VARIABLE := "Target".Article[#DeliverPos],
                              POS := #BufferPos);
ELSIF #Label = 'Bill' THEN
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                              DEST_VARIABLE := "Target".Bill[#DeliverPos],
                              POS := #BufferPos);
;
ELSE
;
END_IF;
```

The "Deserialize" instruction deserializes the sequential representation of the customer data from the "Buffer" data block and writes it to the "Target" data block. The number of bytes occupied by the converted customer data is stored in the #BufferPos operand.

The "Deserialize" instruction deserializes the sequential representation of the separator sheet, which was stored in the sequential representation after the customer data, from the "Buffer" data block and writes the characters to the #Label operand. The characters are compared using comparison instructions for "arti" and "Bill". If the comparison for "arti" = TRUE, these are article data that have been deserialized and written to the "Target" data block. If the comparison for "Bill" = TRUE, these are billing data that have been deserialized and written to the "Target" data block.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 1908)

PLC data types (Page 1954)

Serialize: Serialize

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT) to a sequential representation without losing parts of their structure.

You can use the instruction to temporarily save multiple structured data items from your program in a buffer, which should preferably be in a global data block, and send them to another CPU. The memory area in which the converted PLC data types are stored must have the ARRAY of BYTE data type and be declared with standard access. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion.

The operand at the POS parameter contains information about the number of bytes used by the converted PLC data types.

If you want to send a single PLC data type (UDT), you can directly call the instruction "TSEND: Send data via communication connection".

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", the bit has a length of 1 byte.

Syntax

The following syntax is used for the "Serialize" instruction:

SCL

```
Serialize(SRC_VARIABLE := <Operand>,
          DEST_ARRAY := <Operand>,
          POS := <Operand>)
```

Parameters

The following table shows the parameters of the "Serialize" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC_VARIABLE	Input	VARIANT	I, Q, M, L	PLC data type (UDT) that is converted to sequential representation.
DEST_ARRAY	InOut	VARIANT	I, Q, M, L	Data block in which the generated data stream is stored.
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
Function value		INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains no value.
8152	Code generation error at SRC_VARIABLE parameter
8236	The data block at the DEST_ARRAY parameter is not a block with standard access.
8250	The VARIANT data type at the DEST_ARRAY parameter contains no value.
8252	Code generation error at DEST_ARRAY parameter
8253	There is not enough free memory available at the DEST_ARRAY parameter.
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
#Tag_RetVal := Serialize(SRC_VARIABLE := "Source".Client,
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);

#Label := STRING_TO_WSTRING('arti');
```

SCL

```
#Tag_RetVal := Serialize(SRC_VARIABLE := #Label,
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);

#Tag_RetVal := Serialize(SRC_VARIABLE := "Source".Article[#DeliverPos],
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);
```

The "Serialize" instruction serializes the customer data from the "Source" data block and writes them in sequential representation to the "Buffer" data block. The number of bytes occupied by the sequential representation is stored in the #BufferPos operand.

A kind of separator sheet is inserted now to make it easier to deserialize the sequential representation later. The "Move character string" instruction moves the "arti" characters to the #Label operand. The "Serialize" instruction writes these characters after the customer data to the "Buffer" data block. The number of bytes that the characters need is added to the number already stored in the #BufferPos operand.

The "Serialize" instruction serializes the data of a specific article, which is calculated in runtime, from the "Source" data block and writes them in sequential representation to the "Buffer" data block after the "arti" characters.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 1908)

PLC data types (Page 1954)

MOVE_BLK: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the first element in the source area.

The instruction can only be executed if the source area and the destination area have the same data type.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at the IN parameter or OUT parameter.

Syntax

The following syntax is used for the "Move block" instruction:

SCL

```
MOVE_BLK(IN := <Operand>,
         COUNT := <Operand>,
         OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
MOVE_BLK(IN := #a_array[2],
         COUNT := "Tag_Count",
         OUT => #b_array[1]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is ARRAY [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is ARRAY [0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element.

See also

Overview of the valid data types (Page 1908)

MOVE_BLK_VARIANT: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). You can copy a complete ARRAY or elements of an ARRAY to another ARRAY of the same data type. The size (number of elements) of source and destination ARRAY may be different. You can copy multiple or single elements within an ARRAY.

When you use the instruction at the time to the block is created, the ARRAY does not yet have to be known, as the source and the designation are transferred per VARIANT.

Counting at the parameters SRC_INDEX and DEST_INDEX always begins with the low limit "0", regardless of the later declaration of the ARRAY.

The instruction is not executed if more data is copied than is made available.

Note

VARIANT in connection with the BOOL data type

If you want to interconnect a parameter of VARIANT data type (source or destination area) with a tag of BOOL data type or an ARRAY of BOOL, you have the following options:

1. You can address it symbolically
   Example: Parameter SRC: "Data_block".myArray
2. You can address it with help of ANY pointer. However, you must note that the specified length of the area must be dividable by 8, or the instruction will otherwise not be executed.
   Example: Parameter SRC: P#DB123.DBX456.0 BOOL 1000

Syntax

The following syntax is used for the "Move block" instruction:

SCL

```
MOVE_BLK_VARIANT(SRC := <Operand>,
                 COUNT := <Operand>,
                 SRC_INDEX := <Operand>,
                 DEST_INDEX := <Operand>,
                 DEST => <Operand>)
```

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC	Input	VARIANT (which points to an ARRAY or an individual ARRAY element), ARRAY of <Data_type>	I, Q, M, L	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L	Number of elements which are copied Assign the value "1" to the parameter COUNT, if no SRC is specified at parameter DEST or at Parameter ARRAY.
SRC_INDEX	Input	DINT	I, Q, M, D, L	• The SRC_INDEX parameter is calculated zero-based. If an SRC is specified at parameter ARRAY, the integer at parameter SRC_INDEX specifies the first element within the source range from which it is to be copied. Independent of the declared ARRAY limits. • If no SRC is specified at parameter ARRAY or only one single element of an ARRAY is specified, then assign the value "0" at parameter SRC_INDEX.

Parameter	Declaration	Data type	Memory area	Description
DEST_INDEX	Input	DINT	I, Q, M, D, L	• The DEST_INDEX parameter is calculated zero-based. If an DEST is specified at parameter ARRAY, the integer at parameter DEST_INDEX specifies the first element within the target range to it is to be copied. Independent of the declared ARRAY limits. • If no DEST is specified at parameter ARRAY, then assign the value "0" at parameter DEST_INDEX.
DEST	Output 1)	VARIANT	I, Q, M, L	Destination area into which the contents of the source block are copied.
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information
1) The DEST parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value
8382	The value at the SRC_INDEX parameter is outside the limits of the VARIANT.
8383	The value at parameter SRC_INDEX is outside the high limit of the ARRAY.
8482	The value at the DEST_INDEX parameter is outside the limits of the VARIANT.
8483	The value at parameter DEST_INDEX is outside the high limit of the ARRAY.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at DEST parameter

Error code* (W#16#...)	Explanation
8554	The operand at the DEST parameter has the data type BOOL.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MOVE_BLK_VARIANT(SRC := #SrcField,
                                 COUNT := "Tag_Count",
                                 SRC_INDEX := "Tag_Src_Index",
                                 DEST_INDEX := "Tag_Dest_Index",
                                 DEST => #DestField);
```

The following table shows how the instruction works using specific operand values:

Parameter	Declaration in the block interface	Operand	Value
SRC	Input	#SrcField	The local operand #SrcField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[0..10] of "MOVE_UDT"
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	The local operand #DestField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[10..20] of "MOVE_UDT"

Two elements are moved from the source area, starting from the fourth element of the ARRAY of UDT, to the destination area. The copies are inserted in the array of UDT starting from the fourth element.

See also

Overview of the valid data types (Page 1908)

VariantGet: Read out VARIANT tag value (Page 2866)

Programming example: Moving data (Page 267)

UMOVE_BLK: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The instruction cannot be interrupted. The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the first element in the source area.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at the IN parameter or OUT parameter.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Syntax

The following syntax is used for the "Move block uninterruptible" instruction:

SCL

```
UMOVE_BLK(IN := <Operand>,
          COUNT := <Operand>,
          OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
UMOVE_BLK(IN := #a_array[2],
          COUNT := "Tag_Count",
          OUT => #b_array[1]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is ARRAY [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is ARRAY [0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The copy operation cannot be interrupted by other operating system activities.

See also

Overview of the valid data types (Page 1908)

FILL_BLK: Fill block

Description

The "Fill block" instruction is used to copy the content of a memory area (source area) to a selected memory area (target area). The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the source area is selected and moved to the destination area as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at input IN or output OUT.

Syntax

The following syntax is used for the "Fill block" instruction:

SCL

```
FILL_BLK(IN := <Operand>,
         COUNT := <Operand>,
         OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P	Number of repeated copy operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	Address in destination area from which filling starts
[1] The specified data types can also be used as elements of an ARRAY structure.					
[2] The specified data types can only be used as elements of an ARRAY structure.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
FILL_BLK(IN := #a_array[2],
         COUNT := "Tag_Count",
         OUT => #b_array[1]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is ARRAY [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is ARRAY [0..6] of INT. It consists of seven elements of data type INT.

The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag.

See also

Overview of the valid data types (Page 1908)

UFILL_BLK: Fill block uninterruptible

Description

The instruction "Fill block uninterruptible" is used to copy the content of a memory area (source area) to a selected memory area (target area). The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the IN input is selected and copied to the destination area as often as specified by the value in the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at input IN or output OUT.

You can use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Syntax

The following syntax is used for the "Fill block uninterruptible" instruction:

SCL

```
UFILL_BLK(IN := <Operand>,
          COUNT := <Operand>,
          OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P	Number of repeated copy operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	Address in destination area from which filling starts
[1] The specified data types can also be used as elements of an ARRAY structure.					
[2] The specified data types can only be used as elements of an ARRAY structure.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
UFILL_BLK(IN := #a_array[2],
          COUNT := "Tag_Count",
          OUT => #b_array[1]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is ARRAY [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is ARRAY [0..6] of INT. It consists of seven elements of data type INT.

The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. The copy operation cannot be interrupted by other operating system activities.

See also

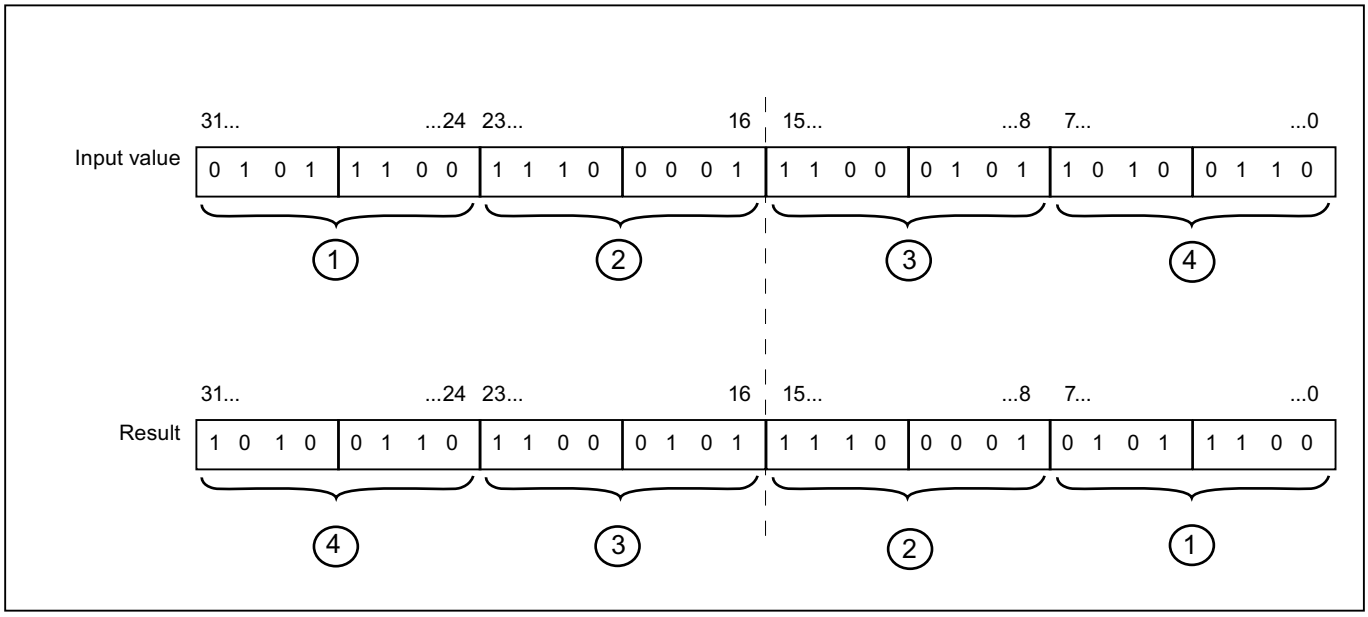
Overview of the valid data types (Page 1908)

SWAP: Swap

Description

You can use the "Swap" instruction to change the arrangement of the bytes of an input value and save the result in the specified operand.

The following figure shows how the bytes of an operand of the DWORD data type are swapped using the "Swap" instruction:



Syntax

The following syntax is used for the "Swap" instruction:

SCL

```
SWAP(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Expression>	Input	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P	Input value
Function value		WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SWAP("Tag_Value");
```

The result of the instruction is returned as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	0000 1111 0101 0101
Tag_Result	0101 0101 0000 1111

See also

Overview of the valid data types (Page 1908)

ARRAY DB

ReadFromArrayDB: Read from array data block

Description

The "Read from array data block" instruction is used to read data from an ARRAY data block and write it to a destination area.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be of data type PLC or any other elementary data type. The

counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

Syntax

The following syntax is used for the "Read from array data block" instruction:

SCL

```
ReadFromArrayDB(DB := <Operand>,
                INDEX := <Operand>,
                VALUE => <Operand>)
```

Parameters

The following table shows the parameters of the "Read from array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P	Element that is read
VALUE	Output [1)]	VARIANT	I, Q, M, L	Value that is read and output
Function value (RET_VAL)		INT	I, Q, M, D, L	Result of the instruction
1) The VALUE parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, write protected, or is located in load memory.
8135	The ARRAY data block contains invalid values.
8154	The data block has the incorrect data type.
8282	The value at parameter INDEX is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8452	Code generation error
8453	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"TagResult" := ReadFromArrayDB(DB := "ArrayDB",
                               INDEX := "ArrayDB"."THIS"[2],
                               VALUE => "TargetField[10]".Data2[1]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB."THIS"[2]	Third element of the "ArrayDB"
VALUE	"TargetField[10]".Data2[1]	The data type of the "Target-Field" operand is Array [10 to 20] of "MOVE_UDT".

The third element is read from "ArrayDB" and written to the "TargetField[10]".Data2[1] operand.

See also

Overview of the valid data types (Page 1908)

Using ARRAY data blocks (Page 239)

WriteToArrayDB: Write to array data block

Description

The instruction "Write to array data block" is used to write data to an ARRAY data block.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be of data type PLC or any other elementary data type. The counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

Syntax

The following syntax is used for the "Write to array data block" instruction:

SCL

```
WriteToArrayDB(DB := <Operand>,
               INDEX := <Operand>,
               VALUE := <Operand>)
```

Parameters

The following table shows the parameters of the "Write to array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P	Element in the DB to which data is written
VALUE	Input	VARIANT	I, Q, M, L	Value to be written
Function value (RET_VAL)		INT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, or is located in load memory.
8134	The data block is write protected.
8135	The data block is not an ARRAY data block.
8154	The data block has the incorrect data type.
8282	The value at parameter INDEX is outside the limits of the ARRAY.
8350	The data type VARIANT at parameter VALUE provides the value "0".
8352	Code generation error
8353	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"TagResult" := WriteToArrayDB(DB := "ArrayDB",
                              INDEX := "ArrayDB"."THIS"[2],
                              VALUE := "SourceField[1]".Data1[6]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB."THIS"[2]	Third element of the "ArrayDB"
VALUE	"SourceField[1]".Data1[6]	The data type of the "SourceField" operand is Array [0 to 10] of "MOVE_UDT".

From the "SourceField" operand, the "Data1[6]" element from the second element is written to the "ArrayDB". The third element is written to the "ArrayDB".

See also

Overview of the valid data types (Page 1908)

Using ARRAY data blocks (Page 239)

ReadFromArrayDBL: Read from array data block in load memory

Description

The instruction "Read from array data block in load memory" is used to read data from an ARRAY data block in the load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be of data type PLC or any other elementary data type. The counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Note

The ARRAY data block must be created with the "Optimized" block property.

Syntax

The following syntax is used for the "Read from array data block in load memory" instruction:

SCL

```
"Instance DB"(REQ := <Operand>,
              DB := <Operand>,
              INDEX := <Operand>,
              VALUE := <Operand>,
              BUSY => <Operand>,
              DONE => <Operand>,
              ERROR => <Operand>)
```

Parameters

The following table shows the parameters of the "Read from array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = "1": Begin with the reading out of the ARRAY DB
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P	Element that is read
VALUE	InOut	VARIANT	I, Q, M, L	Value that is read and output No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.

Error code* (W#16#...)	Explanation
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at parameter INDEX is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in load memory" and "WRIT_DBL: Write to data block in the load memory" under "See also".

Example

The following example shows how the instruction works:

SCL

```
"ReadFromArrayDBL_DB"(REQ := "TagReg",
                      DB := "ArrayDB",
                      INDEX := "ArrayDB"."THIS"[2],
                      VALUE := "SourceTargetField[1]".Data1[6],
                      BUSY => "TagBusy",
                      DONE => "TagDone",
                      ERROR => "TagError");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB."THIS"[2]	Third element of the "ArrayDB"
VALUE	"SourceTargetField[1]".Data1[6]	The data type of the "SourceTargetField" operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If a positive edge is detected at the "TagReq" operand, the "Read from array data block in load memory" instruction is executed. The third element is read from "ArrayDB" and output at the "SourceTargetField[1]".Data1[6] operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter

is no longer changed. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 1908)

READ_DBL: Read from data block in the load memory (Page 3351)

WRIT_DBL: Write to data block in the load memory (Page 3353)

Using ARRAY data blocks (Page 239)

WriteToArrayDBL: Write to array data block in load memory

Description

The instruction "Write to array data block in load memory" is used to write data to an ARRAY data block in the load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be of data type PLC or any other elementary data type. The counting at the ARRAY always begins with the low limit "0", regardless of the later declaration of the ARRAY.

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Note

The ARRAY data block must be created with the "Optimized" block property.

Syntax

The following syntax is used for the "Write to array data block in load memory" instruction:

SCL

```
"Instance DB"(REQ := <Operand>,
              DB := <Operand>,
              INDEX := <Operand>,
              VALUE := <Operand>,
```

SCL

```
BUSY => <Operand>,
DONE => <Operand>,
ERROR => <Operand>)
```

Parameters

The following table shows the parameters of the "Write to array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = "1": Start writing to the array DB
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P	Element in the DB to which data is written
VALUE	Input	VARIANT	I, Q, M, L	Value to be written No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8234	The data block is write protected.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.

Error code* (W#16#...)	Explanation
8382	The value at parameter INDEX is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in load memory" and "WRIT_DBL: Write to data block in the load memory" under "See also".

Example

The following example shows how the instruction works:

SCL

```
"WriteToArrayDBL_DB"(REQ := "TagReg",
                     DB := "ArrayDB",
                     INDEX := "ArrayDB"."THIS"[2],
                     VALUE := "SourceTargetField[1]".Data1[6],
                     BUSY => "TagBusy",
                     DONE => "TagDone",
                     ERROR => "TagError");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB."THIS"[2]	Third element of the "ArrayDB"
VALUE	"SourceTargetField[1]".Data1[6]	The data type of the "SourceTargetField" operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If a positive edge is detected at the "TagReq" operand, the "Write to array data block in load memory" instruction is executed. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written in the third element in the "ArrayDB". After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 1908)

READ_DBL: Read from data block in the load memory (Page 3351)

WRIT_DBL: Write to data block in the load memory (Page 3353)

Using ARRAY data blocks (Page 239)

Read/write access

PEEK: Read memory address

Description

The "Read memory address" instruction is used to read a memory address from a memory area without specifying a data type.

If you specify the 16#84 area for a data block at the AREA parameter, you can only access data blocks with the "Standard" block property.

Note

The instruction can only be used to access "Standard" memory areas.

Syntax

The following syntax is used for the "Read memory address" instruction:

SCL

```
PEEK(AREA:= <Operand>,
     DBNUMBER := <Operand>,
     BYTEOFFSET:= <Operand>)
```

SCL

```
PEEK_<Data type>(AREA := <Operand>,
                 DBNUMBER := <Operand>,
                 BYTEOFFSET:= <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D, L	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#1: Peripheral input (S7-1500 only)
DBNUM-BER	Input	DINT, DB_ANY	D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D, L	Address to read from Only the 16 least significant bits are used.
_<Data type>		Bit strings default: BYTE	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Bit strings	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Note

If you read the memory address from the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := PEEK(AREA := "Tag_Area",
                      DBNUMBER := "Tag_DBNumber",
                      BYTEOFFSET := "Tag_Byte");
```

SCL

```
"Tag_Result2" := PEEK_WORD(AREA := "Tag_Area",
                           DBNUMBER := "Tag_DBNumber",
```

SCL

```
                                BYTEOFFSET := "Tag_Byte");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
Function value	Tag_Result1	Byte value "20"
Function value	Tag_Result2	Word value "20"

The instruction reads the value of address "20" from the "Tag_Byte" operand at data block "5" and returns the result as a function value at the "Tag_Result" operand.

See also

Overview of the valid data types (Page 1908)

PEEK_BOOL: Read memory bit

Description

The "Read memory bit" instruction is used to read a memory bit from a memory area without specifying a data type.

If you specify the 16#84 area for a data block at the AREA parameter, you can only access data blocks with the "Standard" block property.

Note

The instruction can only be used to access "Standard" memory areas.

Syntax

The following syntax is used for the "Read memory bit" instruction:

SCL

```
PEEK_BOOL(AREA := <Operand>,
          DBNUMBER := <Operand>,
          BYTEOFFSET := <Operand>,
          BITOFFSET := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D, L	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#1: Peripheral input (S7-1500 only)
DBNUM-BER	Input	DINT, DB_ANY	D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D, L	Address to read from Only the 16 least significant bits are used.
BITOFFSET	Input	INT	I, Q, M, D, L	Bit to be read from
Function value		BOOL	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Note

If you read the memory bit from the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := PEEK_BOOL(AREA := "Tag_Area",
                          DBNUMBER := "Tag_DBNumber",
                          BYTEOFFSET := "Tag_Byte",
                          BITOFFSET := "Tag_Bit");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
Function value	Tag_Result	3

The instruction reads the value of memory bit "3" from the "Tag_Bit" operand at byte "20" of data block "5" and returns the result at the "Tag_Result" operand as function value.

See also

Overview of the valid data types (Page 1908)

POKE: Write memory address

Description

The "Write memory address" instruction is used to write a memory address to a memory area without specifying a data type.

If you specify the 16#84 area for a data block at the AREA parameter, you can only access data blocks with the "Standard" block property.

Note

The instruction can only be used to access "Standard" memory areas.

Syntax

The following syntax is used for the "Write memory address" instruction:

SCL

```
POKE(AREA := <Operand>,
     DBNUMBER := <Operand>,
     BYTEOFFSET := <Operand>,
     VALUE := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D, L	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#2: Peripheral output (S7-1500 only)
DBNUM-BER	Input	DINT, DB_ANY	D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D, L	Address to be written Only the 16 least significant bits are used.
VALUE	Input	Bit strings	I, Q, M, D, L	Value to be written

For additional information on valid data types, refer to "See also".

Note

If you write the memory address to the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

```
SCL
POKE(AREA := "Tag_Area",
     DBNUMBER := "Tag_DBNumber",
     BYTEOFFSET := "Tag_Byte"),
     VALUE := "Tag_Value";
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
VALUE	Tag_Value	16#11

The instruction overwrites the memory address "20" in the data block "5" with value "16#11".

See also

Overview of the valid data types (Page 1908)

POKE_BOOL: Write memory bit

Description

The "Write memory bit" instruction is used to write a memory bit to a memory area without specifying a data type.

If you specify the 16#84 area for a data block at the AREA parameter, you can only access data blocks with the "Standard" block property.

Note

The instruction can only be used to access "Standard" memory areas.

Syntax

The following syntax is used for the "Write memory bit" instruction:

SCL

```
POKE_BOOL(AREA := <Operand>,
          DBNUMBER := <Operand>,
          BYTEOFFSET := <Operand>,
          BITOFFSET := <Operand>,
          VALUE := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D, L	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#2: Peripheral output (S7-1500 only)
DBNUM-BER	Input	DINT, DB_ANY	D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D, L	Address to be written Only the 16 least significant bits are used.
BITOFFSET	Input	INT	I, Q, M, D, L	Bit to be written
VALUE	Input	BOOL	I, Q, M, D, L	Value to be written

For additional information on valid data types, refer to "See also".

Note

If you write the memory bit to the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

SCL

```
POKE_BOOL(AREA := "Tag_Area",
          DBNUMBER := "Tag_DBNumber",
          BYTEOFFSET := "Tag_Byte",
          BITOFFSET := "Tag_Bit",
          VALUE := "Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
VALUE	Tag_Value	M0.0

The instruction overwrites the memory bit "3" in data block "5" in byte "20" with the value "M0.0".

See also

Overview of the valid data types (Page 1908)

POKE_BLK: Write memory area

Description

The "Write memory area" instruction copies the content a memory area to a different memory area without specifying a data type.

If you specify the 16#84 area for a data block at the AREA parameter, you can only access data blocks with the "Standard" block property.

Note

The instruction can only be used to access "Standard" memory areas.

Syntax

The following syntax is used for the "Write memory area" instruction:

SCL

```
POKE_BLK(AREA_SRC:= <Operand>,
         DBNUMBER_SRC:= <Operand>,
         BYTEOFFSET_SRC:= <Operand>,
         AREA_DEST:= <Operand>,
         DBNUMBER_DEST:= <Operand>,
         BYTEOFFSET_DEST:= <Operand>,
         COUNT := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
AREA_SRC	Input	BYTE	I, Q, M, D, L	The following areas can be selected in the source memory area: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB
DBNUMBER_SRC	Input	DINT, DB_ANY	D	Number of the data block in the source memory area, if AREA = DB, otherwise "0"
BYTEOFF-SET_SRC	Input	DINT	I, Q, M, D, L	Address in the source memory area to be written Only the 16 least significant bits are used.
AREA_DEST	Input	BYTE	I, Q, M, D, L	The following areas can be selected in the destination memory area: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB
DBNUM-BER_DEST	Input	DINT, DB_ANY	D	Number of the data block in the destination memory area, if AREA = DB, otherwise "0"
BYTEOFF-SET_DEST	Input	DINT	I, Q, M, D, L	Address in the destination memory area to be written Only the 16 least significant bits are used.
COUNT	Input	DINT	I, Q, M, D, L	Number of bytes which are copied

For additional information on valid data types, refer to "See also".

Note

If you write the memory address to the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

```
SCL
POKE_BLK(AREA_SRC := "Tag_Source_Area",
         DBNUMBER_SRC := "Tag_Source_DBNumber",
         BYTEOFFSET_SRC := "Tag_Source_Byte"),
         AREA_DEST := "Tag_Destination_Area",
         DBNUMBER_DEST := "Tag_Destination_DBNumber",
         BYTEOFFSET_DEST := "Tag_Destination_Byte",
         COUNT := "Tag_Count");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
AREA_SRC	Tag_Source_Area	16#84
DBNUMBER_SRC	Tag_Source_DBNumber	5
BYTEOFFSET_SRC	Tag_Source_Byte	20
AREA_DEST	Tag_Destination_Area	16#83
DBNUMBER_DEST	Tag_Destination_DBNumber	0
BYTEOFFSET_DEST	Tag_Destination_Byte	30
COUNT	Tag_Count	100

The instruction writes 100 byte from data block "5" starting with address "20" in the memory area of the bit memory starting at address "30".

See also

Overview of the valid data types (Page 1908)

READ_LITTLE: Read data in little endian format

Description

The instruction "Read data in little endian format" is used to read data from a memory area and to write this to a single tag in the little endian byte sequence. With the little endian format, the byte with the least significant bits is saved first, which means at the lowest memory address.

The parameters SRC_ARRAY and DEST_VARIABLE are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at the DEST_VARIABLE parameter must point to an elementary data type. The VARIANT at the SRC_ARRAY parameter points to the memory area to be read from, and this must be an ARRAY of BYTE.

The operand at the POS parameter determines the position in the memory area at which reading starts.

Note

Reading tag of data type VARIANT or BOOL

If you want to read a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to read a tag of the data type BOOL, use a "Slice access".

Syntax

Use the following syntax for the "Read data in little endian format" instruction:

SCL

```
READ_LITTLE(SRC_ARRAY := <Operand>,
            DEST_VARIABLE => <Operand>,
            POS := <Operand>)
```

Parameters

The following table shows the parameters of the "Read data in little endian format" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
SRC_ARRAY	Input	ARRAY of BYTE	ARRAY of BYTE	I, Q, M, D, L	Memory area to be read from
DEST_VARIABLE	Output	Bit strings, integers, floating-point numbers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L	Read value
POS	InOut	DINT	DINT	I, Q, M, D, L	Determines the position at which the reading starts. The POS parameter is calculated zero-based.
Function value (RET_VAL)		INT	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

SCL

```
#TagResult := READ_LITTLE(SRC_ARRAY := #SourceField,
                          DEST_VARIABLE => #DINTVariable,
                          POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC_ARRAY	#SourceField	ARRAY[0..3] of BYTE := 16#1A, 16#2B, 16#3C, 16#4D
DEST_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
POS	#TagPos	0 => 4

The instruction reads the integer 1_295_788_826 from the #SourceField memory area and writes it to the #DINTVariable operand in little endian format. The data type at the DEST_VARIABLE parameter specifies how many bytes are read. The quantity 4 is stored in the #TagPos operand.

See also

Overview of the valid data types (Page 1908)

Deserialize: Deserialize (Page 2819)

Serialize: Serialize (Page 2822)

WRITE_LITTLE: Write data in little endian format

Description

You use the "Write data in little endian format" instruction to write the data of a single tag in the little endian byte sequence to a memory area. With the little endian format, the byte with the least significant bits is saved first, which means at the lowest memory address.

The parameters SRC_VARIABLE and DEST_ARRAY are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at the SRC_VARIABLE parameter must point to an elementary data type. The VARIANT at the DEST_ARRAY parameter points to the memory area to which the data is written, and this must be an ARRAY of BYTE.

The operand at the POS parameter determines the position in the memory area at which writing starts.

Note

Writing tag of data type VARIANT or BOOL

If you want to write a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to write a tag of the data type BOOL, use a "Slice access".

Syntax

Use the following syntax for the "Write data in little endian format" instruction:

SCL

```
WRITE_LITTLE(SRC_VARIABLE := <Operand>,
             DEST_ARRAY := <Operand>,
             POS := <Operand>)
```

Parameters

The following table shows the parameters of the "Write data in little endian format" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
SRC_VARIABLE	Input	Bit strings, integers, floating-point numbers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L	Tag whose data are written
DEST_ARRAY	InOut	ARRAY of BYTE	ARRAY of BYTE	I, Q, M, D, L	Memory area to which the data are written
POS	InOut	DINT	DINT	I, Q, M, D, L	Determines the position at which the writing starts. The POS parameter is calculated zero-based.
Function value (RET_VAL)		INT	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

SCL

```
#TagResult := WRITE_LITTLE(SRC_VARIABLE := #DINTVariable,
                           DEST_ARRAY := #TargetField,
                           POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
DEST_ARRAY	#TargetField	ARRAY[0..10] of BYTE = 16#1A, 16#2B, 16#3C, 16#4D
POS	#TagPos	0 => 4

The instruction writes the integer 1_295_788_826 from the #DINTVariable operand to the #TargetField memory area in little endian format. The data type at the SRC_VARIABLE parameter specifies how many bytes are written. The quantity 4 is stored in the #TagPos operand.

See also

Overview of the valid data types (Page 1908)

Deserialize: Deserialize (Page 2819)

Serialize: Serialize (Page 2822)

READ_BIG: Read data in big endian format

Description

The instruction "Read data in big endian format" is used to read data from a memory area and to write this to a single tag in the big endian byte sequence. With the big endian format, the byte with the most significant bits is saved first, which means at the lowest memory address.

The parameters SRC_ARRAY and DEST_VARIABLE are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at the DEST_VARIABLE parameter must point to an elementary data type. The VARIANT at the SRC_ARRAY parameter points to the memory area to be read from, and this must be an ARRAY of BYTE.

The operand at the POS parameter determines the position in the memory area at which reading starts.

Note

Reading tag of data type VARIANT or BOOL

If you want to read a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to read a tag of the data type BOOL, use a "Slice access".

Syntax

Use the following syntax for the "Read data in big endian format" instruction:

SCL

```
READ_BIG(SRC_ARRAY := <Operand>,
         DEST_VARIABLE => <Operand>,
         POS := <Operand>)
```

Parameters

The following table shows the parameters of the "Read data in big endian format" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
SRC_ARRAY	Input	ARRAY of BYTE	ARRAY of BYTE	I, Q, M, D, L	Memory area to be read from
DEST_VARIABLE	Output	Bit strings, integers, floating-point numbers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L	Read value
POS	InOut	DINT	DINT	I, Q, M, D, L	Determines the position at which the reading starts. The POS parameter is calculated zero-based.
Function value (RET_VAL)		INT	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

SCL

```
#TagResult := READ_BIG(SRC_ARRAY := #SourceField,
                       DEST_VARIABLE => #DINTVariable,
                       POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC_ARRAY	#SourceField	ARRAY[0..10] of BYTE := 16#1A, 16#2B, 16#3C, 16#4D
DEST_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
POS	#TagPos	0 => 4

The instruction reads the integer 439_041_101 from the #SourceField memory area and writes it to the #DINTVariable operand in big endian format. The data type at the DEST_VARIABLE parameter specifies how many bytes are read. The quantity 4 is stored in the #TagPos operand.

See also

Overview of the valid data types (Page 1908)

Deserialize: Deserialize (Page 2819)

Serialize: Serialize (Page 2822)

WRITE_BIG: Write data in big endian format

Description

You use the "Write data in big endian format" instruction to write the data of a single tag in the big endian byte sequence to a memory area. With the big endian format, the byte with the most significant bits is saved first, which means at the lowest memory address.

The parameters SRC_VARIABLE and DEST_ARRAY are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at the SRC_VARIABLE parameter must point to an elementary data type. The VARIANT at the DEST_ARRAY parameter points to the memory area to which the data is written, and this must be an ARRAY of BYTE.

The operand at the POS parameter determines the position in the memory area at which writing starts.

Note

Writing tag of data type VARIANT or BOOL

If you want to write a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to write a tag of the data type BOOL, use a "Slice access".

Syntax

Use the following syntax for the "Write data in big endian format" instruction:

SCL

```
WRITE_BIG(SRC_VARIABLE := <Operand>,
          DEST_ARRAY := <Operand>,
          POS := <Operand>)
```

Parameters

The following table shows the parameters of the "Write data in big endian format" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
SRC_VARIABLE	Input	Bit strings, integers, floating-point numbers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L	Tag whose data are written
DEST_ARRAY	InOut	ARRAY of BYTE	ARRAY of BYTE	I, Q, M, D, L	Memory area to which the data are written
POS	InOut	DINT	DINT	I, Q, M, D, L	Determines the position at which the writing starts. The POS parameter is calculated zero-based.
Function value (RET_VAL)		INT	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

SCL

```
#TagResult := WRITE_BIG(SRC_VARIABLE := #DINTVariable,
                        DEST_ARRAY := #TargetField,
                        POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
DEST_ARRAY	#TargetField	ARRAY[0..10] of BYTE = 16#1A, 16#2B, 16#3C, 16#4D
POS	#TagPos	0 => 4

The instruction writes the integer 439_041_101 from the #DINTVariable operand to the #TargetField memory area in big endian format. The data type at the SRC_VARIABLE parameter specifies how many bytes are written. The quantity 4 is stored in the #TagPos operand.

See also

Overview of the valid data types (Page 1908)

Deserialize: Deserialize (Page 2819)

Serialize: Serialize (Page 2822)

VARIANT

VariantGet: Read out VARIANT tag value

Description

You can use the "Read out VARIANT tag value" instruction to read the value of the tag to which the VARIANT at the SRC parameter points and write it in the tag at the DST parameter.

The SRC parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the DST parameter.

The data type of the tag specified at the DST parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction. For additional information, refer to "See also".

Syntax

The following syntax is used for the "Read out VARIANT tag value" instruction:

SCL

```
VariantGet(SRC := <Operand>,
           DST => <Operand>)
```

Parameters

The following table shows the parameters of the "Read out VARIANT tag value" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC	Input	VARIANT	I, Q, M, L	Tag to be read
DST	Output	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
VariantGet(SRC := #TagIn_Source,
           DST => "TagOut_Dest");
```

The value of the tag to which the VARIANT at the #TagIn_Source operand points is read and written to the "TagOut_Dest" operand.

See also

Overview of the valid data types (Page 1908)

VARIANT (Page 1951)

MOVE_BLK_VARIANT: Move block (Page 2827)

VariantPut: Write VARIANT tag value

Description

You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter to the tag at the DST parameter to which the VARIANT points.

The DST parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction. For additional information, refer to "See also".

Syntax

The following syntax is used for the "Write VARIANT tag value" instruction:

SCL

```
VariantPut(SRC := <Operand>,
           DST := <Operand>)
```

Parameters

The following table shows the parameters of the "Write VARIANT tag value" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	Tag to be read
DST	Input	VARIANT	I, Q, M, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
VariantPut(SRC := "TagIn_Source",
           DST := "#TagIn_Dest");
```

The value of the "TagIn_Source" operand is written to the tag to which the VARIANT at the #TagIn_Dest operand points.

See also

Overview of the valid data types (Page 1908)

VARIANT (Page 1951)

MOVE_BLK_VARIANT: Move block (Page 2827)

CountOfElements: Get number of ARRAY elements

Description

You can use the "Get number of ARRAY elements" instruction to query how many ARRAY elements a tag to which VARIANT is pointing has.

If it is a one-dimensional ARRAY, the difference between the high and low limit +1 is output as a result. If it is a multi-dimensional ARRAY, the product of all dimensions is output as the result.

The <Operand> must have the VARIANT data type.

The result is "0" if the VARIANT tag is not an ARRAY.

If the VARIANT points to an ARRAY of BOOL, the fill elements are included in the count. (For example, 8 is returned for an ARRAY[0..1] of BOOL)

Syntax

The following syntax is used for the "Get number of ARRAY elements" instruction:

SCL

```
CountOfElements(<Operand>)
```

Parameters

The following table shows the parameters of the "Get number of ARRAY elements" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	I, Q, M, L	Tag to be queried
Function value		UDINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF IS_ARRAY(#Tag_VARIANTToArray) THEN
"Tag_Result" := CountOfElements(#Tag_VARIANT);
END_IF;
```

If the tag to which the VARIANT points is an ARRAY, the number of ARRAY elements is output.

See also

Format of array (16-bit limits) (Page 1941)
Format of array (32-bit limits) (Page 1942)
Overview of the valid data types (Page 1908)
VARIANT (Page 1951)

Legacy

BLKMOV: Move block

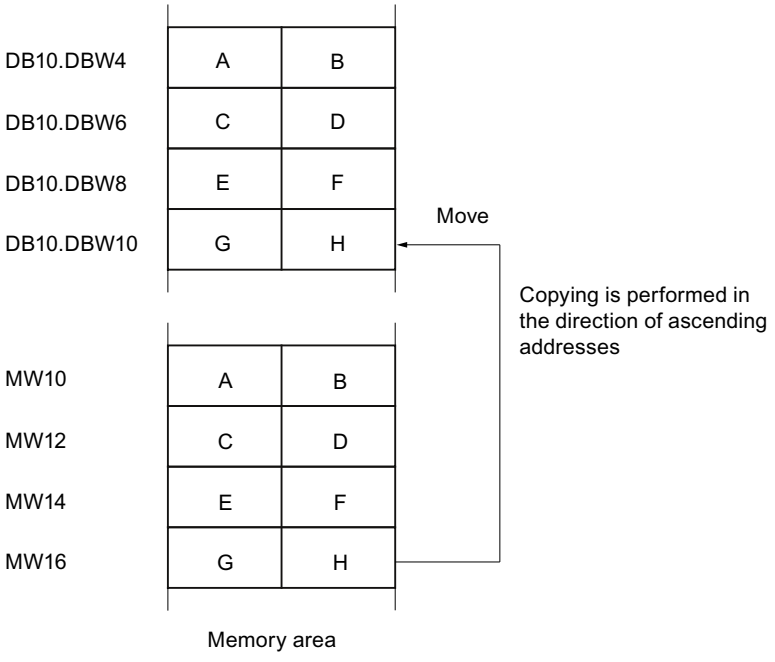
Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

The following figure shows the principle of the move operation:



Consistency of source and destination data

Note that while the instruction "Move block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Syntax

The following syntax is used for the "Move block" instruction:

SCL

```
BLKMOV(SRCBLK := <Operand>,
       DSTBLK => <Operand>)
```

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	I, Q, M, L, P	Specifies the memory area to be moved (source area).
DSTBLK	Output [1)]	VARIANT	I, Q, M, L, P	Specifies the memory area to which the block is to be moved (destination area).
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR and BOOL data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR and BOOL data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_RetVal" := BLKMOV(SRCBLK := P#M100.0 BYTE 10,
                       DSTBLK => P#DB1.DBX0.0 BYTE 10);
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2922)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of data type STRING. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as the area length.

Syntax

The following syntax is used for the "Move block uninterruptible" instruction:

SCL

```
UBLKMOV(SRCBLK := <Operand>,
        DSTBLK => <Operand>)
```

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	I, Q, M, L, P	Specifies the memory area to be moved (source area).
DSTBLK	Output [1]	VARIANT	I, Q, M, L, P	Specifies the memory area to which the block is to be moved (destination area).
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
8091	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR and BOOL data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR and BOOL data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_RetVal" := UBLKMOV(SRCBLK := P#M100.0 BYTE 10,
                        DSTBLK => P#DB1.DBX0.0 BYTE 10);
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2922)

FILL: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

You use VARIANT to define the source and destination areas.

Note

You can also define the source and destination areas using the ANY data type

If you use the ANY data type, you must observe the following in connection with the STRING and WSTRING data types:

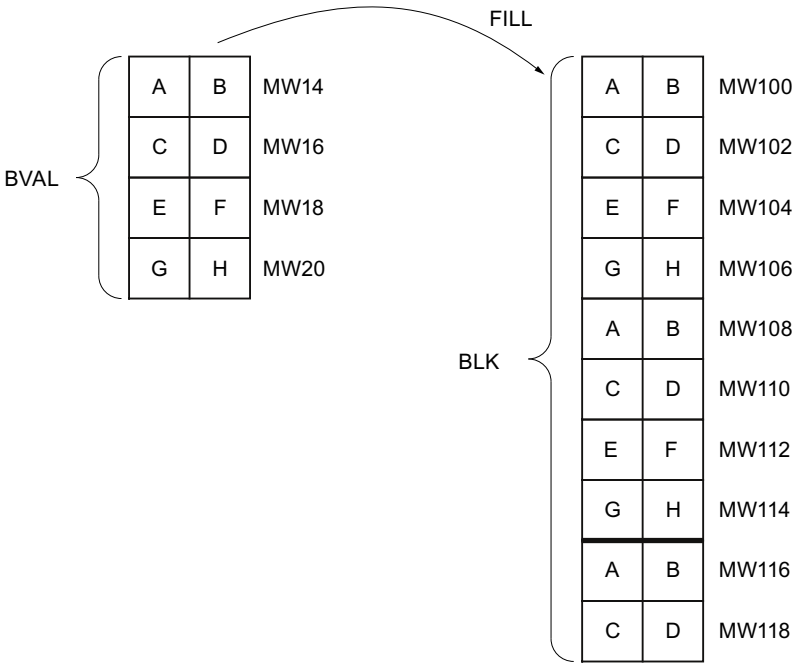
- In the case of an assignment of STRING (source area) using ANY after STRING (destination area), the content of the STRING is copied to the destination area repeatedly until it is full.
  Source area: String#'STEP7-SCL-TIA-Portal'
  Destination area: 'STEP7-SCL-TIA-PortalSTEP7-SCL-TIA-PortalSTEP7-SCL'

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK: Fill block" instruction.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Syntax

The following syntax is used for the "Fill block" instruction:

```
SCL
FILL(BVAL := <Operand>,
     BLK => <Operand>)
```

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
BVAL	Input	VARIANT	I, Q, M, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
BLK	Output 1)	VARIANT	I, Q, M, L, P	Specification of the memory area that will be filled with the content of the source area.
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information
1) The BLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

BVAL parameter

When you transfer a structure as an input parameter, remember that the length of a structure is always based on an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR and BOOL data types are not supported at the BVAL parameter.
8352	The WSTRING, WCHAR and BOOL data types are not supported at the BLK parameter.

Error code (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_RetVal" := FILL(BVAL := P#M14.0 WORD 4,
                     BLK => P#M100.0 WORD 10);
```

The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2922)

Conversion operations

CONVERT: Convert value

Description

Use the "Convert value" instruction to program explicit conversions. When the instruction is inserted, the "CONVERT" dialog opens. You specify the source data type and the destination data type of the conversion in this dialog. The source value is read and converted to the specified destination data type.

For information on possible conversions, refer to section "Explicit conversion" under "See also".

Syntax

The following syntax is used for the "Convert value" instruction:

SCL

```
<DestinationValue> := <???_TO_???(<SourceValue>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Source_value>	Input	Binary numbers, integers, floating-point numbers, times, date and time, character strings, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted.
???_TO_???	-	-		Function that specifies the data type to be converted .
<Target_value>	Output	Binary numbers, integers, floating-point numbers, times, date and time, character strings, BCD16, BCD32	I, Q, M, D, L, P or constant	Result of the conversion

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_INT" := REAL_TO_INT("Tag_REAL");
```

The following table shows how the instruction works using specific operand values:

Operand	Data type	Value
Tag_REAL	REAL	20.56
Tag_INT	INT	21

During the conversion, the value of the "Tag_REAL" operand will be rounded to the nearest integer and saved in the "Tag_INT" operand.

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

ROUND: Round numerical value

Description

The "Round numerical value" instruction is used to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts it into an integer or floating-point number. If the input value is exactly between an even and odd number, the even number is selected.

Syntax

The following syntax is used for the "Round numerical value" instruction:

SCL

```
ROUND(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value to be rounded.
Function value		Integers, floating-point numbers	I, Q, M, D, L, P	Result of the rounding

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ROUND("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	1.50000000	-1.50000000
Tag_Result	2	-2

See also

Overview of the valid data types (Page 1908)

Data type conversion for S7-1200: (Page 2091)

CEIL: Generate next higher integer from floating-point number

Description

Use the "Generate next higher integer from floating-point number" instruction to round the value to the nearest integer. The instruction interprets the input value as floating-point number and converts it to the next higher integer. The function value can be greater than or equal to the input value.

Syntax

The following syntax is used for the "Generate next higher integer from floating-point number" instruction:

SCL

```
CEIL(<Expression>)
CEIL_<Data type>(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
_<Data type>		Integers, floating-point numbers. Default: DINT	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Integers, floating-point numbers	I, Q, M, D, L	Input value rounded up

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := CEIL("Tag_Value");
"Tag_Result2" := CEIL_REAL("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.5	-0.5
Tag_Result1	1	0
Tag_Result2	1.0	0.0

The result of the instruction is returned in the operand "Tag_Resultxy" as a function value.

See also

Overview of the valid data types (Page 1908)

FLOOR: Generate next lower integer from floating-point number

Description

Use the "Generate next lower integer from floating-point number" instruction to round the value of a floating point number to the next lower integer. The instruction interprets the input value as floating-point number and converts it to the next lower integer. The function value can be equal or less than the input value.

Syntax

Use the following syntax for the "Generate next lower integer from floating-point number" instruction:

SCL

```
FLOOR(<Expression>)
FLOOR_<Data type>(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
_<Data type>		Integers, floating-point numbers. Default: DINT	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Integers, floating-point numbers	I, Q, M, D, L	Input value rounded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := FLOOR("Tag_Value");
"Tag_Result2" := FLOOR_REAL("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.5	-0.5
Tag_Result1	0	-1
Tag_Result2	0.0	-1.0

The result of the instruction is returned in the operand "Tag_Resultxy" as a function value.

See also

Overview of the valid data types (Page 1908)

TRUNC: Truncate numerical value

Description

The "Truncate numerical value" instruction is used to generate an integer from the input value without rounding. The instruction selects only the integer part of the input value and returns this part without decimal places as the function value.

Syntax

The following syntax is used for the "Truncate numerical value" instruction:

SCL

```
TRUNC(<Expression>)
TRUNC_<Data type>(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L	Input value
_<Data type>		Integers, floating-point numbers Default: DINT	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Integers, floating-point numbers	I, Q, M, D, L	Integer component of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := TRUNC("Tag_Value1");
```

SCL

```
"Tag_Result2" := TRUNC("Tag_Value2"+"Tag_Value3");
"Tag_Result3" := TRUNC_SINT("Tag_Value4");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value1	-1.5
Tag_Result1	-1
Tag_Value2	2.1
Tag_Value3	3.2
Tag_Result2	5
Tag_Result3	2
Tag_Value4	2.4

See also

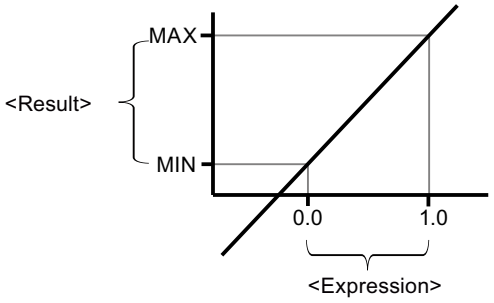
Overview of the valid data types (Page 1908)

SCALE_X: Scale

Description

Use the "Scale" instruction to scale a floating-point number by mapping it to a specific value range. You specify the value range with the MIN and MAX parameters. The result of the scaling is an integer.

The following figure shows an example of how values can be scaled:



The instruction "Scale" works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

Note

For more information on the conversion of analog values, refer to the respective manual.

Syntax

The following syntax is used for the "Scale" instruction:

SCL

```
SCALE_X(MIN := <Operand>,
        VALUE := <Operand>,
        MAX := <Operand>)
SCALE_X_<Data type>(MIN := <Operand>,
                    VALUE := <Operand>,
                    MAX := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L	Value to be scaled. When a constant is specified, you must declare this.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L	High limit of the value range

Parameter	Declaration	Data type	Memory area	Description
_<Data type>		Integers, floating-point numbers Default: INT	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Integers, floating-point numbers	-	Result of scaling

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := SCALE_X(MIN := "Tag_Value1",
                         VALUE := "Tag_Real",
                         MAX := "Tag_Value2");
"Tag_Result2" := SCALE_X_REAL(MIN := "Tag_Value1",
                              VALUE := "Tag_Real",
                              MAX := "Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Real	0.5
Tag_Value1	10
Tag_Value2	30
Tag_Result1	20
Tag_Result2	20.0

See also

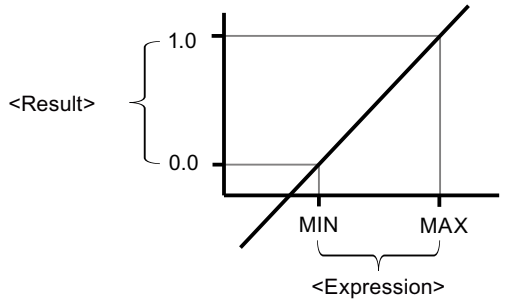
Overview of the valid data types (Page 1908)

NORM_X: Normalize

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. Depending on the location of the normalized value in this value range, the result at output OUT is calculated and stored as a floating-point number. If the value to be normalized equals the value at input MIN, the instruction returns the result "0.0". If the value to be normalized equals the value at input MAX, the instruction returns the result "1.0".

The following figure shows an example of how values can be normalized:



The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

Note

For more information on the conversion of analog values, refer to the respective manual.

Syntax

The following syntax is used for the "Normalize" instruction:

```
SCL
NORM_X(MIN := <Operand>,
       VALUE := <Operand>,
       MAX := <Operand>)
```

SCL

```
NORM_X_<Data type>(MIN := <Operand>,
                   VALUE := <Operand>,
                   MAX := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
MIN [1]	Input	Integers, floating-point numbers	I, Q, M, D, L	Low limit of the value range
VALUE [1]	Input	Integers, floating-point numbers	I, Q, M, D, L	Value to be normalized.
MAX [1]	Input	Integers, floating-point numbers	I, Q, M, D, L	High limit of the value range
_<Data type>		Floating-point numbers Default: REAL	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Floating-point numbers	I, Q, M, D, L	Result of the normalization
[1] If you use constants in these three parameters, you only need to declare one of them.				

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := NORM_X(MIN := "Tag_Value1",
                        VALUE := "Tag_InputValue",
                        MAX := "Tag_Value2");
"Tag_Result2" := NORM_X_LREAL(MIN := "Tag_Value1",
                              VALUE := "Tag_InputValue",
                              MAX := "Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InputValue	20
Tag_Value1	10
Tag_Value2	30
Tag_Result1	0.5
Tag_Result2	0.5

See also

Overview of the valid data types (Page 1908)

VARIANT

VARIANT_TO_DB_ANY: Convert VARIANT to DB_ANY

Description

You use the "Convert VARIANT to DB_ANY" instruction to query the data block number that the operand that is specified at the IN parameter addresses. This can be an instance data block or an ARRAY data block. The operand at the IN parameter has the data type VARIANT, which means you do not need to know the data type of the data block whose number is to be queried when the program is created. The data block number is read during runtime and written to the operand specified at the RET_VAL parameter.

Requirements

If the requirements are met, the instruction is executed. If the requirements are not met, "0" is output as data block number.

The output tag...	homed...	Conversion options
VARIANT	... a data block which is an instance data block of a PLC data type or a system data type (SDT).	The output tag can be converted to a data block number.
VARIANT	... a data block which is an ARRAY DB.	The output tag can be converted to a data block number.
VARIANT	... a tag with an elementary data type.	It is not possible to convert the output tag to a database number because a data block can never comprise only one elementary data type.
VARIANT	... a structure within a data block.	It is not possible to convert the output tags to a database number because it is only a part within the data block.

Syntax

The following syntax is used for the "Convert VARIANT To DB_ANY" instruction:

SCL

```
VARIANT_TO_DB_ANY(IN := <Operand>,
                  ERR => <Operand>)
```

Parameters

The following table shows the parameters of the "Convert VARIANT to DB_ANY" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	I, Q, M, L	Tag to be read
ERR	Output	INT	I, Q, M, D, L	Error information
Function value (RET_VAL)		DB_ANY	I, Q, M, D, L	Result: Number of the DB

For additional information on valid data types, refer to "See also".

ERR parameter

The following table shows the meaning of the values of the ERR parameter:

Error code* (W#16#...)	Explanation
0000	No error
252C	The VARIANT data type at the IN parameter has the value "0" and the CPU changes to STOP mode.
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8131	The data block does not exist, is too short, or is located in load memory.
8132	The data block is too short and not an ARRAY data block.
8150	The data type VARIANT at parameter IN provides the value "0". To receive this error message, the "Handle errors within block" block property must be activated. Otherwise the CPU changes to STOP mode and sends the error code 16#252C.
8153	The VARIANT data type at the IN parameter does not point to the start of an ARRAY data block or the length of the VARIANT does not match that of the data block.
8154	The data block has the incorrect data type.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"OutputDBNumber" := VARIANT_TO_DB_ANY(IN := "InputTag",
                                      ERR := "Tag_Error");
```

The following table shows how the instruction works using specific operand values:

Parameter	Declaration in the block interface	Operand	Value
IN	Input	#InputTag	-
<Function value>	Output	OutputDBNumber	11

The number of a data block that is specified at the #InputTag operand is read. Because the operand has the data type VARIANT, you do not need to know the data type of the tag during program creation. The number is written to the "OutputDBNumber" tag which has the data type DB_ANY.

See also

Overview of the valid data types (Page 1908)

DB_ANY_TO_VARIANT: Convert DB_ANY to VARIANT

Description

The "Convert DB_ANY to VARIANT" instruction is used to generate a VARIANT tag from a data block that meets the requirements listed below. The operand at the IN parameter has the data type DB_ANY, which means that the data block does not have to be known when the program is created. The data block number is read during runtime.

Requirements

If the requirements are met, the instruction is executed. If the requirements are not met or the data block does not exist, the value NULL is output at the RET_VAL parameter. All other accesses with the RET_VAL tag fail.

The input tag of data type ...	homed...	Conversion options
DB_ANY	...a data block which is an instance data block of a PLC data type or a system data type (SDT).	Conversion is possible
DB_ANY	...a data block which is an ARRAY DB.	Conversion is possible
DB_ANY	...a data block which is an instance data block of a function block or a global data block.	Conversion is not possible

Syntax

The following syntax is used for the "Convert DB_ANY to VARIANT" instruction:

SCL

```
DB_ANY_TO_VARIANT(IN := <Operand>,
                  ERR => <Operand>)
```

Parameters

The following table shows the parameters of the "Convert DB_ANY to VARIANT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DB_ANY	I, Q, M, D, L	Tag to be referenced and read
ERR	Output	INT	I, Q, M, D, L	Error information
Function value (RET_VAL) 1)		VARIANT	I, Q, M, L	Number of the data block
1) The RET_VAL parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

ERR parameter

The following table shows the meaning of the values of the ERR parameter:

Error code* (W#16#...)	Explanation
0000	No error
8130	The number of the data block is "0"
8131	The data block does not exist, is too short, or is located in load memory.
8132	The data block is too short and not an ARRAY data block.
8134	The data block is write protected.
8154	The data block has the incorrect data type.
8155	The data block has an unknown data type.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
#tempVARIANT := DB_ANY_TO_VARIANT(IN := "InputDB",
                                  ERR := "Tag_Error");
```

The following table shows how the instruction works using specific operand values:

Parameters	Declaration in the block interface	Operand	Value
IN	Input	InputDB	11
<Function value>	Temp	#tempVARIANT	-

The number of any data block that is specified at the "InputDB" operand is used to generate a tag of data type VARIANT that addresses the data block. Because the operand at the IN parameter has the DB_ANY data type, the data block that will be used during runtime does not have to be known when the program is created (neither the name nor the number of the

data block). Because the operand at the RET_VAL parameter has the data type VARIANT, you do not need to know the data type of the data block when the program is created.

See also

Overview of the valid data types (Page 1908)

Legacy

SCALE: Scale

Description

The instruction "Scale" converts the integer on the parameter IN into a floating-point number, which can be scaled in physical units between a low and a high limit. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output on the OUT parameter.

The instruction "Scale" works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the "K1" and "K2" constants are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case, the constant "K1" has the value -27648.0 and the constant "K2" the value +27648.0.
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case, the constant "K1" has the value 0.0 and the constant "K2" the value +27648.0.

When the value at the IN parameter is greater than the value of the "K2" constant, the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the "K1" constant, the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit is greater than the high limit (LO_LIM >HI_LIM), the result is scaled in reverse proportion to the input value.

Syntax

The following syntax is used for the "Scale" instruction:

```
SCL
SCALE(IN := <Expression>,
      HI_LIM := <Operand>,
      LO_LIM := <Operand>,
      BIPOLAR := <Operand>,
      OUT => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
Function value (RET_VAL)		WORD	I, Q, M, D, L, P	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the parameter IN is greater than 27 648 or is less than 0 (unipolar) or -27 648 (bipolar).
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_ErrorCode" := SCALE(IN := "Tag_InputValue",
                         HI_LIM := "Tag_HighLimit",
                         LO_LIM := "Tag_LowLimit",
                         BIPOLAR := "Tag_Bipolar",
                         OUT => "Tag_Result");
```

The error information is returned in the operand "Tag_ErrorCode" as a function value.

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_Result	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2922)

UNSCALE: Unscale

Description

The instruction "Unscale" unscales the floating-point number on the IN parameter in physical units between a low and a high limit and converts them into an integer. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output on the OUT parameter.

The instruction "Unscale" works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case, the constant "K1" has the value -27648.0 and the constant "K2" the value +27648.0.
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case, the constant "K1" has the value 0.0 and the constant "K2" the value +27648.0.

When the value at the IN parameter is greater than the value of the "HI_LIM" constant, the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Syntax

The following syntax is used for the instruction "Unscale":

```
SCL
UNSCALE(IN := <Expression>,
        HI_LIM := <Operand>,
        LO_LIM := <Operand>,
        BIPOLAR := <Operand>,
        OUT => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
Function value (RET_VAL)		WORD	I, Q, M, D, L, P	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
"Tag_ErrorCode" := UNSCALE(IN := "Tag_InputValue",
                           HI_LIM := "Tag_HighLimit",
```

```
LO_LIM := "Tag_LowLimit",
BIPOLAR := "Tag_Bipolar",
OUT => "Tag_Result");
```

The error information is returned in the operand "Tag_ErrorCode" as a function value.

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_Result	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2922)

Program control operations

IF: Run conditionally

Description

The instruction "Run conditionally" branches the program flow depending on a condition. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Syntax

Depending on the type of branch, you can program the following forms of the instruction:

- Branch through IF:

SCL

```
IF <Condition> THEN <Instructions>
END_IF;
```

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the execution of the program continues with the next instruction after the END_IF.

- Branch through IF and ELSE:

SCL

```
IF <Condition> THEN <Instructions1>
ELSE <Instructions0>;
END_IF;
```

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the instructions programmed after the ELSE are executed. Then the execution of the program continues with the next instruction after the END_IF.

- Branch through IF, ELSIF and ELSE:

SCL

```
IF <Condition1> THEN <Instructions1>
ELSIF <Condition2> THEN <Instruction2>
ELSE <Instructions0>;
END_IF;
```

If the first condition (<Condition1>) is satisfied, the instructions (<Instructions1>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If the first condition is not satisfied, the second condition (<Condition2>) is checked. If the second condition (<Condition2>) is fulfilled, the instructions (<Instructions2>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If none of the conditions are fulfilled, the instructions (<Instructions0>) after ELSE are executed followed by the execution of the program after END_IF.

You can nest as many combinations of ELSIF and THEN as you like within the IF instruction. The programming of an ELSE branch is optional.

The syntax of the IF instruction consists of the following parts:

Parameter	Data type	Memory area	Description
<Condition>	BOOL	I, Q, M, D, L	Expression to be evaluated
<Instructions>	-		Instructions to be executed with satisfied condition. An exception are instructions programmed after the ELSE. These are executed if no condition within the program loop is satisfied.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
IF "Tag_1" = 1
THEN "Tag_Value" := 10;
ELSIF "Tag_2" = 1
THEN "Tag_Value" := 20;
ELSIF "Tag_3" = 1
THEN "Tag_Value" := 30;
ELSE "Tag_Value" := 0;
END_IF;
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_1	1	0	0	0
Tag_2	0	1	0	0
Tag_3	0	0	1	0
Tag_Value	10	20	30	0

See also

Overview of the valid data types (Page 1908)

CASE: Create multiway branch

Description

The instruction "Create multiway branch" executes one of several instruction sequences depending on the value of a numerical expression.

The value of the expression must be an integer. When the instruction is executed, the value of the expression is compared with the values of several constants. If the value of the expression agrees with the value of a constant, the instructions programmed directly after this constant are executed. The constants can assume the following values:

- An integer (for example, 5)
- A range of integers (for example, 15..20)
- An enumeration consisting of integers and ranges (for example, 10, 11, 15..20)

Syntax

The following syntax is used for the "Create multiway branch" instruction:

```
SCL
CASE <Expression> OF
<Constant1>: <Instructions1>
<Constant2>: <Instructions2>
<ConstantX>: <InstructionsX>; // X >=3
ELSE <Instructions0>;
END_CASE;
```

The syntax of the instruction consists of the following parts:

Parameter	Data type	Memory area	Description
<Expression>	Integers	I, Q, M, D, L	Value which is compared to the programmed constant values.
<Constant>	Integers	I, Q, M, D, L	Constant values which form the condition for the execution of an instruction sequence. The constants can assume the following values: • An integer (for example, 5) • A range of integers (for example, 15..20) • An enumeration consisting of integers and ranges (for example, 10, 11, 15..20)
<Instruction>	-		Any instructions which are executed if the value of the expression agrees with the value of a constant. An exception are instructions programmed after the ELSE. These instructions are executed if the values do not agree.

For additional information on valid data types, refer to "See also".

If the value of the expression agrees with the value of the first constant (<Constant1>), the instructions (<Instructions1>) which are programmed directly after the first constant are executed. Program execution subsequently resumes after the END_CASE.

If the value of the expression does not agree with the value of the first constant (<Constant1>), this value is compared to the value of the constant which is programmed next. In this way, the CASE instruction is executed until the values agree. If the value of the expression does not correspond to any of the programmed constant values, the instructions (<Instructions0>) which are programmed after the ELSE are executed. ELSE is an optional part of the syntax and can be omitted.

The CASE instruction can also be nested by replacing an instruction block with CASE. END_CASE represents the end of the CASE instruction.

Example

The following example shows how the instruction works:

```
SCL
CASE "Tag_Value" OF
     0 :
     "Tag_1" := 1;
     1,3,5 :
     "Tag_2" :=1;
     6..10 :
     "Tag_3" := 1;
     16,17,20..25 :
     "Tag_4" := 1;
ELSE "Tag_5" := 1;
END_CASE;
```

The following table shows how the instruction works using specific operand values:

Operand	Values				
Tag_Value	0	1, 3 , 5	6, 7, 8, 9, 10	16,17, 20, 21, 22, 23, 24, 25	2
Tag_1	1	-	-	-	-
Tag_2	-	1	-	-	-
Tag_3	-	-	1	-	-
Tag_4	-	-	-	1	-
Tag_5	-	-	-	-	1
1: The operand is set to the signal state "1". -: The signal state of the operand remains unaltered.					

See also

CONTINUE: Recheck loop condition (Page 2907)

Overview of the valid data types (Page 1908)

EXIT: Exit loop immediately (Page 2908)

FOR: Run in counting loop

Description

The instruction "Run in counting loop" causes repeated execution of a program loop until a loop variable lies within a specified value range.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Limits for FOR statements

To program "safe" FOR statements that do not run endlessly, observe the following rules and limits:

Rule

```
FOR ii := Start TO End BY Step DO
```

If ...	... then	Note
start < end	end < (PMAX step)	Run tag ii runs in a positive direction
start > end AND step < O	end > (NMAX step)	Run tag ii runs in a negative direction

Limits

Different limits apply to the possible data types:

Data type	PMAX	NMAX
ii of type SINT	127	-128
ii of type INT	32767	-32768
ii of type DINT	2147483647	-2147483648
ii of type LINT	9223372036854775807	-9223372036854775808

Syntax

The following syntax is used for the "Run in counting loop" instruction:

SCL

```
FOR<Run_tag> := <Start_value> TO<End_value> BY<Increment> DO<Instructions>
END_FOR;
```

Parameter

The following table shows the parameters of the "Run in counting loop" instruction:

Parameter	Data type		Memory area	Description
	S7-1200	S7-1500		
<Run tag>	SINT, INT, DINT	SINT, INT, DINT, LINT	I, Q, M, D, L	Operand whose value is evaluated with the loop execution. The data type of the loop variable determines the data type of the other parameters.
<Start value>	SINT, INT, DINT	SINT, INT, DINT, LINT	I, Q, M, D, L	Expression whose value is allocated at the start of the loop execution of the loop variables.

Parameter	Data type		Memory area	Description
	S7-1200	S7-1500		
<End value>	SINT, INT, DINT	SINT, INT, DINT, LINT	I, Q, M, D, L	Expression whose value defines the last run of the program loop. The value of the loop variable is checked before each loop: • End value not reached: The instructions according to DO are executed • End value is reached: The FOR loop is executed one last time • End value exceeded: The FOR loop is completed An alteration to the end value is not permitted during execution of the instruction.
<Increment>	SINT, INT, DINT	SINT, INT, DINT, LINT	I, Q, M, D, L	Expression by whose value the loop variable is increased (positive increment) or decreased (negative increment) after each loop. Specification of the increment is optional. If no increment is given, the value of the loop variable is increased by 1 after each loop. An alteration of the increment is not permitted during execution of the instruction.
<Instructions>	-	-		Instructions which are carried out with each loop, as long as the value of the loop variable lies within the value range. The value range is defined by the start and end values.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
FOR i := 2 TO 8 BY 2
    DO "a_array[i] := "Tag_Value"*"b_array[i]";
END_FOR;
```

The operand "Tag_Value" is multiplied with the elements (2, 4, 6, 8) of the ARRAY tag "b_array". The result is read in to the elements (2, 4, 6, 8) of the ARRAY tag "a_array".

See also

CONTINUE: Recheck loop condition (Page 2907)

EXIT: Exit loop immediately (Page 2908)

Overview of the valid data types (Page 1908)

WHILE: Run if condition is met

Description

The instruction "Run if condition is met" causes a program loop to be repeatedly executed until the implementation condition is satisfied. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Run if condition is met" instruction:

```
SCL
WHILE <Condition> DO <Instructions>
END_WHILE;
```

The syntax of the WHILE instruction consists of the following parts:

Parameter	Data type	Memory area	Description
<Condition>	BOOL	I, Q, M, D, L	Expression which is evaluated before each loop.
<Instructions>	-		Instructions to be executed with satisfied condition. If the condition has not been satisfied, program execution continues after END_WHILE.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
WHILE
      "Tag_Value1" <> "Tag_Value2"
      DO "Tag_Result"
      := "Tag_Input";
END_WHILE;
```

As long as the values of the operands "Tag_Value1" and "Tag_Value2" do not match, the value of the operand "Tag_Input" is allocated to the operand "Tag_Result".

See also

EXIT: Exit loop immediately (Page 2908)

CONTINUE: Recheck loop condition (Page 2907)

Overview of the valid data types (Page 1908)

REPEAT: Run if condition is not met

Description

The instruction "Run if condition is not met" causes a program loop to be repeatedly executed until a termination condition is met. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

The instructions are executed once, even if the termination condition is fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Run if condition is not met" instruction:

```
SCL
REPEAT <Instructions>
UNTIL <Condition>END_REPEAT;
```

The syntax of the REPEAT instruction consists of the following parts:

Parameter	Data type	Memory area	Description
<Instructions>	-		Instructions that are executed as long as the programmed condition has the value FALSE. The instructions are executed once, even if the termination condition is fulfilled.
<Condition>	BOOL	I, Q, M, D, L	Expression which is evaluated after each loop. If the expression has the value FALSE, the program loop is executed once again. If the expression has the value TRUE, the program loop continues after END_REPEAT.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
REPEAT "Tag_Result"
        := "Tag_Value";
UNTIL "Tag_Error"
END_REPEAT;
```

As long as the value of the operand "Tag_Error" has the signal state "0", the value of the operand "Tag_Value" is allocated to the operand "Tag_Result".

See also

CONTINUE: Recheck loop condition (Page 2907)

EXIT: Exit loop immediately (Page 2908)

Overview of the valid data types (Page 1908)

CONTINUE: Recheck loop condition

Description

The "Recheck loop condition" instruction ends the current program run of a FOR, WHILE or REPEAT loop.

After execution of the instruction, the conditions for the continuation of the program loop are evaluated again. The instruction affects the program loop which directly contains the instruction.

Syntax

The following syntax is used for the "Recheck loop condition" instruction:

SCL

```
CONTINUE;
```

Example

The following example shows how the instruction works:

SCL

```
FOR i
     := 1 TO 15 BY 2 DO
     IF (i < 5) THEN
     CONTINUE;
END_IF;
      "DB10".Test[i] := 1;
END_FOR;
```

For additional information on valid data types, refer to "See also".

If the condition i < 5 is satisfied, then the subsequent value allocation ("DB10".Test[i] := 1) will not be executed. The run variable (i) is increased by the increment of "2" and checked to see whether its current value lies in the programmed value range. If the run variable lies in the value range, the IF condition is evaluated again.

If the condition i < 5 is not satisfied, then the subsequent value allocation ("DB10".Test[i] := 1) will be executed and a new loop started. In this case, the run variable is also increased by the increment "2" and checked.

See also

EXIT: Exit loop immediately (Page 2908)

Overview of the valid data types (Page 1908)

EXIT: Exit loop immediately

Description

The instruction "Exit loop immediately" cancels the execution of a FOR, WHILE or REPEAT loop at any point regardless of conditions. The execution of the program is continued after the end of the loop (END_FOR, END_WHILE, END_REPEAT).

The instruction affects the program loop which directly contains the instruction.

Syntax

The following syntax is used for the "Exit loop immediately" instruction:

SCL

```
EXIT;
```

Example

The following example shows how the instruction works:

SCL

```
FOR i := 15 TO 1 BY -2 DO
IF (i < 5)
THEN EXIT;
END_IF;
"DB10".Test[i] := 1;
END_FOR;
```

For additional information on valid data types, refer to "See also".

If the condition i < 5 is satisfied, then the execution of the loop will be canceled. Program execution resumes after the END_FOR.

If the condition i < 5 is not satisfied, then the subsequent value allocation ("DB10".Test[i] := 1) will be executed and a new loop started. The run tag (i) is decreased by the increment of "-2" and it is checked whether its current value lies in the programmed value range. If the (i) run variable lies within the value range, the IF condition is evaluated again.

See also

CONTINUE: Recheck loop condition (Page 2907)

Overview of the valid data types (Page 1908)

GOTO: Jump

Description

Use the instruction "Jump" to resume the execution of a program at a given point marked with a jump label.

The jump labels and the instruction "Jump" must be in the same block. The name of a jump label can only be assigned once within a block. Each jump label can be the target of several jump instructions.

A jump from the "outside" into a program loop is not permitted, but a jump from a loop to the "outside" is possible.

Syntax

Use the following syntax for the "Jump" instruction:

SCL

```
GOTO <Jump label>
...
<Jump label>: <Instructions>
```

The syntax of the GOTO instruction consists of the following parts:

Parameter	Data type	Description
<jump label>	-	Jump label to be jumped to
<Instructions>	-	Instructions which are executed after the jump.

Example

The following example shows how the instruction works:

SCL

```
CASE "Tag_Value" OF
1 : GOTO MyLABEL1;
2 : GOTO MyLABEL2;
3 : GOTO MyLABEL3;
ELSE GOTO MyLABEL4;
END_CASE;
MyLABEL1: "Tag_1" := 1;
MyLABEL2: "Tag_2" := 1;
MyLABEL3: "Tag_3" := 1;
MyLABEL4: "Tag_4" := 1;
```

Depending on the value of the "Tag_Value" operand, the execution of the program will resume at the point identified by the corresponding jump label. If the operand "Tag_Value" has the value 2, for example, program execution will resume at the jump label "MyLABEL2". The program line identified by the jump label "MyLABEL1" will be skipped in this case.

See also

Overview of the valid data types (Page 1908)

RETURN: Exit block

Description

The instruction "Exit block" exits the program execution in the currently edited block and continues in the calling block.

The instruction can be omitted at the end of the block.

Syntax

The following syntax is used for the "Exit block" instruction:

SCL

```
RETURN;
```

Example

The following example shows how the instruction works:

SCL

```
IF "Tag_Error" <>0 THEN RETURN;
END_IF;
```

If the signal state of the "Tag_Error" operand is not zero, execution of the program ends in the block currently being processed.

See also

Overview of the valid data types (Page 1908)

(*...*): Insert a comment section

Description

You can use the "Insert a comment section" instruction to add a comment section. The text within the parenthesis "(*...*)" is handled as a comment.

Syntax

The following syntax is used for the "Insert a comment section" instruction:

SCL

```
(*...*)
```

Example

The following example shows how the instruction works:

SCL

```
(*This is a comment section.*)
```

See also

Overview of the valid data types (Page 1908)

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether passwords configured for the CPU are legitimized or not. In this way, you can prevent legitimized connections, even if the correct password is known.

When you call the instruction and the REQ parameter has the signal state "0", the currently set state is displayed at the output parameters. If changes have been made to the input parameters, these are not transferred to the output parameters.

When you call the instruction and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD). FALSE means that legitimation per password is not permitted; TRUE means the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Passwords that are not configured must have the signal state TRUE at the input and return the signal state TRUE at the output. The fail-safe password can only be configured for an F-CPU and must therefore always be interconnected with the signal state TRUE in a standard CPU. If the instruction returns an error, the call has no effect and the previous lock is still in effect.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the operating mode switch to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Plugging an empty memory card (transfer card or program card) into an S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called once again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access from HMI systems if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing accidental lockout of an S7-1500 CPU

The settings can be made on the front panel of the CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode switch to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing accidental lockout of an S7-1200 CPU

Because the S7-1200 CPU does not have a operating mode switch, protection is disabled with POWER OFF - POWER ON. This means that it is possible and advisable to prevent accidental lockout with the help of specific program sequences within your program.

To do so, program a time control either by means of a cyclic interrupt OB or a timer in the Main OB (OB 1). This gives you the option of calling the "Limit and enable password legitimation" instruction in the respective OB (e.g., OB 1 or OB 35) relatively quickly after a transition from POWER OFF to POWER ON and the associated disabling of the protection. Call the instruction in the startup OB (OB 100) to keep the time window in which the instruction is inactive and there are no limitations for password legitimation relatively small. This procedure will give you the best possible protection from unauthorized access.

If there has been an accidental lockout, you can skip the call in the startup OB (by querying an input parameter, for example) and you have the set time (for example, 10 seconds up to 1 minute) to establish a connection to the CPU before the lock becomes active once again.

If there is no timer in your program code and a lockout has occurred, insert an empty transfer card or an empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then download the user program once again from STEP 7 to the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

Warning

Inserting an empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Once you have pulled the transfer card, the content of the transfer card is available in the internal load memory. Make sure that the card does not include a program at this point.

Warning

Inserting an empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. Once you have pulled the previously empty program card, the internal load memory is empty.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200 CPU: The lock is disabled and the instruction must be called once again in the program (e.g., in the startup OB). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Syntax

The following syntax is used for the "Limit and enable password legitimation" instruction:

SCL

```
ENDIS_PW(REQ := <Operand>,
         F_PWD := <Operand>,
         FULL_PWD := <Operand>,
         R_PWD := <Operand>,
         HMI_PWD := <Operand>,
         F_PWD_ON => <Operand>,
         FULL_PWD_ON => <Operand>,
         R_PWD_ON => <Operand>,
         HMI_PWD_ON => <Operand>)
```

Parameters

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password
R_PWD	Input	BOOL	I, Q, M, D, L	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed

Parameter	Declaration	Data type	Memory area	Description
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
Function value (RET_VAL)		WORD	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported
80D0	The password for fail-safe is not configured. The signal state must be TRUE for standard CPUs.
80D1	The read/write access is not configured
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ENDIS_PW(REQ := 0,
                         F_PWD := 0,
                         FULL_PWD := 0,
                         R_PWD := 1,
                         HMI_PWD := 0,
                         F_PWD_ON => "Status_F_PWD",
                         FULL_PWD_ON => "Status_FULL_PWD",
                         R_PWD_ON => "Status_R_PWD",
                         HMI_PWD_ON => "Status_HMI_PWD");
```

The following table shows how the instruction works using specific operand values:

Operand	Data type	Value
REQ	BOOL	0
F_PWD	BOOL	0
FULL_PWD	BOOL	0

Operand	Data type	Value
R_PWD	BOOL	1
HMI_PWD	BOOL	0
Status_F_PWD	BOOL	0
Status_FULL_PWD	BOOL	0
Status_R_PWD	BOOL	1
Status_HMI_PWD	BOOL	0

The instruction is executed because the REQ operand has the signal state "0". The R_PWD operand has the signal state "1", which means that read access is allowed when the assigned password is entered. The R_PWD_ON status operand also has the signal state "1", thereby signaling that the R_PWD operand is activated.

RE_TRIGR: Restart cycle monitoring time

Description

The instruction "Restart cycle monitoring time" restarts the maximum cycle time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

The "Restart cycle monitoring time" can be called, regardless of the priority, in all blocks.

If the instruction is called in a block with a higher priority, for example, a hardware interrupt, diagnostic interrupt, or cyclic interrupt, the instruction will not be executed.

The "Restart cycle monitoring time" instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Syntax

The following syntax is used for the "Restart cycle monitoring time" instruction:

```
SCL
RE_TRIGR()
```

Parameters

The "Restart cycle monitoring time" instruction has no parameters and supplies no error information.

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Syntax

The following syntax is used for the instruction "Exit program":

SCL

```
STP()
```

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports an error during the processing of the block, detailed information is generated for the first error to occur during the execution of the block since the last execution of the instruction.

The error information can only be saved in operands of the "ErrorStruct" system data type. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the block, the error information about the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The <Operand> is only changed if error information is present. To set the operand back to "0" after handling the error, you have the following options:

- Declare the operand in the "Temp" section of the block interface.
- Reset the operand to "0" before calling the instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error locally" instruction enables local error handling within a block. If "Get error locally" is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Syntax

The following syntax is used for the "Get error locally" instruction:

SCL

```
GET_ERROR(<Operand>)
```

Parameter

The following table shows the parameters of the "Get error locally" instruction:

Parameter	Data type	Memory area	Description
<Operand>	ErrorStruct	D, L	Information about errors that have occurred

Data type "ErrorStruct"

The following table shows the structure of the "ErrorStruct" data type:

Structure component		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows if an error occurred during a block call. 16#01: Error during a block call 16#00: No error during a block call
REACTION		BYTE	Default reaction: 0: Ignore (write error), 1: Continue with substitute value "0" (read error), 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of block
	BLOCK_TYPE	BYTE	Type of block where the error occurred: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B Range violations for a directly editable tag of data type DINT: 16#04

Structure component		Data type	Description
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

The following table shows the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:

SCL

```
LABEL: #TagOut := #Field[#index] * REAL#40.5;

GET_ERROR(#Error);

IF #Error.REACTION = 1 THEN
   GOTO LABEL;
   ;
END_IF;
```

An error occurred accessing the #Field[#index] tag. The #TagOut operand returns the signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, we recommend that you program the "Get error locally" instruction after the multiplication to get the error. The error information supplied by the "Get error locally" instruction is evaluated with a comparison. If the #Error.REACTION structure component has the value "1", there is a read/access error and the program execution begins again at the jump label LABEL.

See also

Querying and fixing errors in the program code (Page 228)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the instruction outputs the error ID of the first error that occurred.

The error ID can only be saved in operands of the WORD data type. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The <Operand> is only changed if error information is present. To set the operand back to "0" after handling the error, you have the following options:

- Declare the operand in the "Temp" section of the block interface.
- Reset the operand to "0" before calling the instruction.

The output of the "Get error ID locally" instruction is only set if error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Syntax

The following syntax is used for the "Get error ID locally" instruction:

SCL

```
GET_ERR_ID()
```

Parameter

The following table shows the parameters of the "Get error ID locally" instruction:

Parameter	Data type	Memory area	Description
Function value	WORD	D, L	Error ID

Error ID

The following table shows the values that can be output:

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist

ID* (hexadecimal)	ID* (decimal)	Description
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:

SCL

```
#TagOut := #Field[#index] * REAL#40.5;

#TagID := GET_ERR_ID();

IF #TagID = 16#2522 THEN
   MOVE_BLK(IN := #TagArrayIn[0],
            COUNT := 1,
            OUT => #TagArrayOut[1]);
END_IF;
```

An error occurred accessing the #Field[#index] tag. The #TagOut operand returns the signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, we recommend that you program the "Get error ID locally" instruction after the multiplication to get the error. The error ID supplied by the "Get error ID locally" instruction is evaluated with a comparison. If the #TagID operand returns ID 16#2522, there is a read/access error and the value "100.0" of the #TagArrayIn[0] operand is written to the #TagArrayOut[1] operand.

See also

Querying and fixing errors in the program code (Page 228)

INIT_RD: Initialize all retain data

Description

The "Initialize all retain data" instruction is used to reset the retentive data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution exceeds the program cycle duration.

Syntax

Use the following syntax for the "Initialize all retain data" instruction:

SCL

```
INIT_RD(<Operand>)
```

Parameters

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L	If the input "REQ" has the signal state "1", all retentive data are reset.
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL
```
"Tag_Result" := INIT_RD("Tag_REQ");
```

If the operand "Tag_REQ" has the signal state "1", the instruction is executed. All retentive data of all data blocks, bit memories and SIMATIC timers and counters are reset.

See also

Overview of the valid data types (Page 1908)

GET_ERR_ID: Get error ID locally (Page 2922)

WAIT: Configure time delay

Description

The instruction "Configure time delay" pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to 32767 microseconds (µs). The smallest possible delay time depends on the respective CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction returns no error information.

Syntax

The following syntax is used for the "Configure time delay" instruction:

SCL
```
WAIT(WT:= <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 1908)

RUNTIME: Measure program runtime

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and writes the value to the in/out parameter. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the in/out parameter as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The following applies to S7-1200 CPUs with firmware version earlier than V4.1: The "Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overflows (this can occur up to once per minute), the instruction returns values <= 0.0. Ignore these runtime values.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Syntax

The following syntax is used for the "Measure program runtime" instruction:

```
SCL
RUNTIME(<Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	InOut	LREAL	I, Q, M, D, L	Saves the starting point of the runtime measurement.
Function value		LREAL	I, Q, M, D, L	Returns the measured runtime in seconds

For additional information on valid data types, refer to "See also".

Example

The following example shows the how the instruction works based on the runtime calculation of a program block:

```
SCL
"Tag_Result" := RUNTIME("Tag_Memory");

"Best_before_date_DB" ();

"Tag_Result" := RUNTIME("Tag_Memory");
```

The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called.

When the program block FB1 has been processed, the instruction is executed a second time. The second call of the instruction calculates the runtime of the program block and writes the result to the output "Tag_Result".

See also

Overview of the valid data types (Page 1908)

Word logic operations

DECO: Decode

Description

The instruction "Decode" sets a bit specified by the input value in the output value.

The instruction "Decode" reads the value of the parameter IN and sets the bit in the output value, whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. If the value of the IN parameter is greater than 31, a modulo 32 instruction is executed.

Syntax

The following syntax is used for the instruction "Decode":

```
SCL
DECO(IN := <Expression>)
DECO_WORD(IN := <Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	UINT	I, Q, M, D, L, P	Position of the bit in the output value which is set.
_<Data type>		Bit strings default: DWORD	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Bit strings	I, Q, M, D, L, P	Current output value

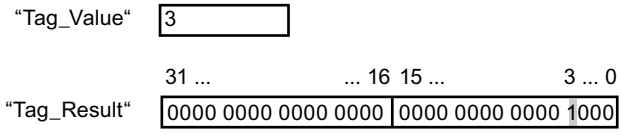
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := DECO(IN := "Tag_Value");
"Tag_Result2" := DECO_BYTE(IN := "Tag_Value2");
```

The following figure shows how the instruction works using specific operand values:



The instruction reads the number "3" from the value of the operand "Tag_Value" and sets the third bit to the value of the operand "Tag_Result".

See also

Overview of the valid data types (Page 1908)

ENCO: Encode

Description

The instruction "Encode" reads the bit number of the lowest-value bit set in the input value and issues this as a result.

Syntax

The following syntax is used for the instruction "Encode":

SCL

```
ENCO(IN := <Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P	Input value
Function value		INT	I, Q, M, D, L, P	Bit number of the bit in the input value that is read out.

For additional information on valid data types, refer to "See also".

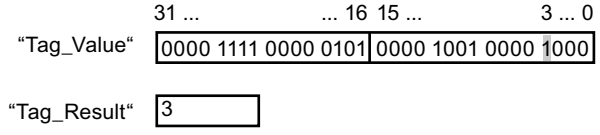
Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ENCO(IN := "Tag_Value");
```

The following figure shows how the instruction works using specific operand values:



The instruction reads the lowest-value set bit of the operand "Tag_Value" and writes the bit position "3" in the operand "Tag_Result".

See also

Overview of the valid data types (Page 1908)

SEL: Select

Description

The instruction "Select" selects one of the parameters IN0 or IN1 depending on a switch (G parameter) and issues its content as a result. When the parameter G has the signal status "0", the value at parameter IN0 is moved. When the parameter G has the signal status "1", the value at parameter IN1 is moved and returned as a function value.

The instruction is only executed if the tags of all parameters have the same data type class.

Syntax

The following syntax is used for the "Select" instruction:

SCL

```
SEL(G := <Expression>,
    IN0 := <Expression>,
    IN1 := <Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
G	Input	BOOL	BOOL	I, Q, M, D, L	Switch
IN0	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, timers, strings, DATE, TOD, LTOD, DT, LDT	I, Q, M, D, L, P	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, timers, strings, DATE, TOD, LTOD, DT, LDT	I, Q, M, D, L, P	Second input value
Function value		Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, timers, strings, DATE, TOD, LTOD, DT, LDT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SEL(G := "Tag_Value",
                    IN0 := "Tag_0",
                    IN1 := "Tag_1");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0	1
Tag_0	W#16#0000	W#16#4C
Tag_1	W#16#FFFF	D#16#5E
Tag_Result	W#16#0000	D#16#5E

See also

Overview of the valid data types (Page 1908)

MUX: Multiplex

Description

The "Multiplex" instruction copies the value of a selected input parameter and issues it. You can use the parameter K to determine the number of the input parameter whose value will be moved. Numbering starts at IN0 and is incremented continuously with each new input. You can declare a maximum of 32 inputs.

Numerical data types and time data types are permitted at the inputs. All tags with assigned parameters must be of the same data type.

The function value is invalid if any of the following conditions are met:

- Errors occurred during execution of the instruction.
- The input at the K parameter is located outside the available inputs. If the INELSE input is not used, the value of the IN0 input is assigned to the function value in this case and the ENO enable output is set to "0".

Syntax

The following syntax is used for the "Multiplex" instruction:

```
SCL
MUX(K := <Expression>,
    IN0 := <Expression>,
    IN1 := <Expression>,
    INELSE := <Expression>)
```

Parameters

The following table shows the parameters of the "Multiplex" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
K	Input	Integers	Integers	I, Q, M, D, L, P	Specifies the parameter whose content is to be transferred. • If K = 0 => Parameter IN0 • If K = 1 => Parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	I, Q, M, D, L, P	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	I, Q, M, D, L, P	Second input value
INn	Input	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	I, Q, M, D, L, P	Optional input values

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
INELSE	Input	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	I, Q, M, D, L, P	Specifies the value to be copied when K <> n.
Function value		Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MUX(K := "Tag_Number",
                    IN0 := "Tag_1",
                    IN1 := "Tag_2",
                    INELSE := "Tag_3");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Number	1	4
Tag_1	DW#16#00000000	DW#16#00000000
Tag_2	DW#16#003E4A7D	DW#16#003E4A7D
Tag_3	DW#16#FFFF0000	DW#16#FFFF0000
Tag_Result	DW#16#003E4A7D	DW#16#FFFF0000

See also

Overview of the valid data types (Page 1908)

DEMUX: Demultiplex

Description

The "Demultiplex" instruction transfers the value of the input parameter IN to a selected output parameter. The selection of the input parameter takes place independent of the parameter value K. The K parameter specifies the output parameter number to which the value of the input parameter IN is transferred. The other output parameters are not changed. Numbering starts at OUT0 and continues consecutively with each new output. You can declare a maximum of 32 output parameters.

If the value of the K parameter is greater than the number of output parameters, the value of the ENO enable output is set to "0" and the value of the input parameter IN is transferred to the output parameter OUTELSE.

The function value is invalid if any of the following conditions are met:

- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Syntax

The following syntax is used for the instruction "Demultiplex":

SCL

```
DEMUX(K := <Expression>,
      IN := <Expression>,
      OUT0 := <Operand>,
      OUT1 := <Operand>,
      OUTELSE := <Operand>)
```

Parameters

The following table shows the parameters of the instruction "Demultiplex":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
K	Input	Integers	Integers	I, Q, M, D, L, P	Specifies the output to which the input value (IN) will be copied. • If K = 0 => Parameter OUT0 • If K = 1 => Parameter OUT1, etc.
IN	Input	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	Input value
OUT0	Output	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	First output
OUT1	Output	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	Second output

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
OUTn	Output	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	Optional outputs
OUTELSE	Output	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	Output to which the value at input IN is copied if K > n.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
DEMUX(K := "Tag_Number",
      IN := "Tag_Value",
      OUT0 := "Tag_1",
      OUT1 := "Tag_2",
      OUTELSE := "Tag_3");
```

The following tables show how the instruction works using specific operand values:

Input values of the "Demultiplex" instruction before the network execution

Parameter	Operand	Values	
K	Tag_Number	2	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

Output values of the "Demultiplex" instruction after the network execution

Parameter	Operand	Values	
OUT0	Tag_1	Unchanged	Unchanged
OUT1	Tag_2	DW#16#FFFFFFFF	Unchanged
OUTELSE	Tag_3	Unchanged	DW#16#003E4A7D

See also

Overview of the valid data types (Page 1908)

Shift and rotate

SHR: Shift right

Description

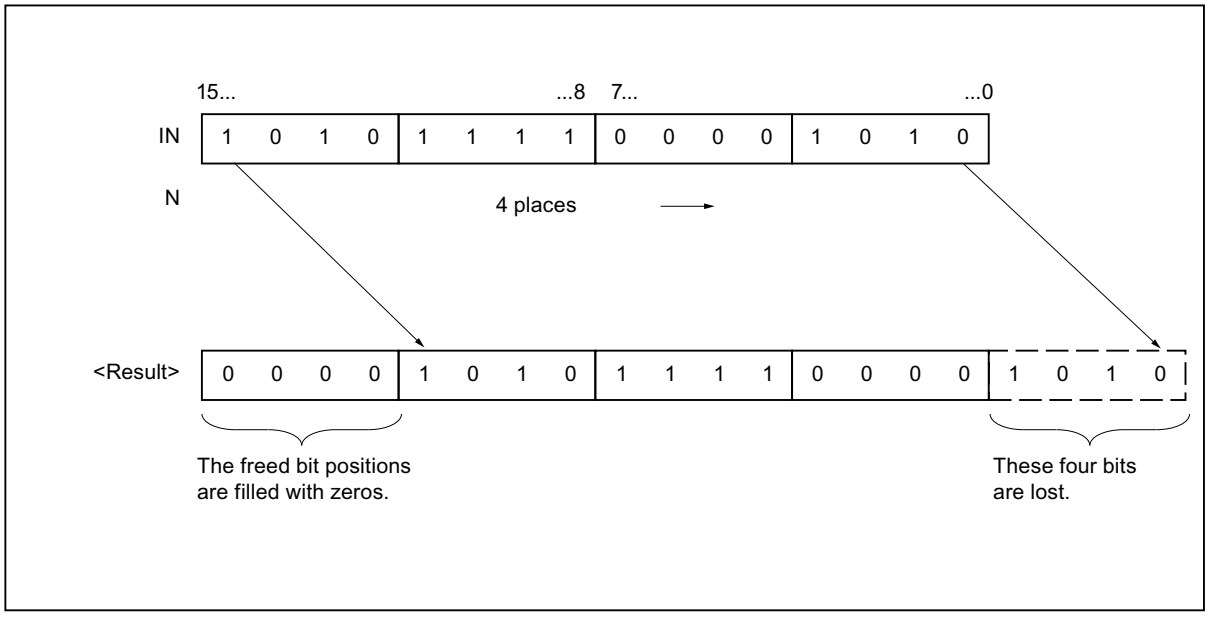
The "Shift right" instruction shifts the contents of the IN parameter bit-by-bit to the right and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of available bit positions, then the value of the IN parameter is shifted to the right by the available number of bit positions.

The bit positions that are freed by shifting in the left operand area are filled with zeros.

The following figure shows how the content of an integer data type operand is shifted by four bit positions to the right:



Syntax

The following syntax is used for the instruction "Shift right":

SCL

```
SHR(IN := <Operand>,
    N := <Operand>)
```

Parameters

The following table shows the parameters of the "Shift right" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Number of bits by which the value (IN) is shifted
Function value		Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SHR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	3
Function value	Tag_Result	0000 0111 1111 010 1

The content of the "Tag_Value" operand is shifted by three bit positions to the right. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 1908)

SHL: Shift left

Description

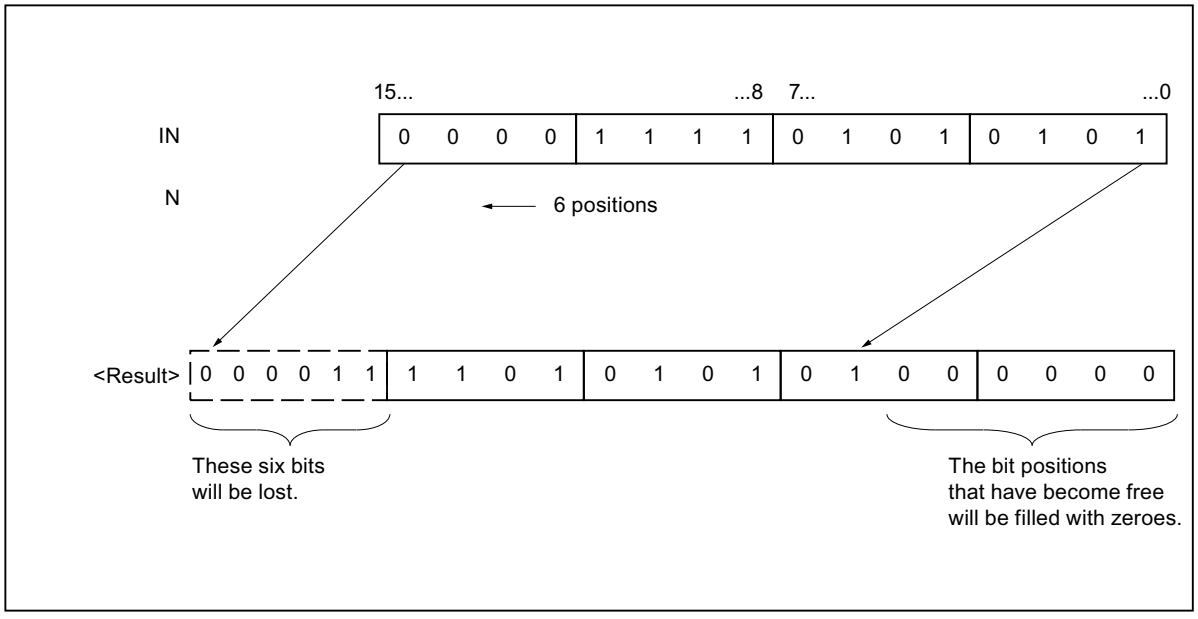
The "Shift left" instruction shifts the contents of the IN parameter bit-by-bit to the left and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of bit positions, the value of the IN parameter is shifted to the left by the available number of bit positions.

The bit positions freed by the shift are filled with zeros in the result value.

The following figure shows how the content of an operand of the WORD data type is shifted six bit positions to the left:



Syntax

The following syntax is used for the instruction "Shift left":

```
SCL
SHL(IN := <Operand>,
    N := <Operand>)
```

Parameters

The following table shows the parameters of the "Shift left" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Number of bits by which the value (IN) is shifted
Function value		Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SHL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	4
Function value	Tag_Result	1111 1010 1111 0000

The value of the "Tag_Value" operand is shifted by four bit positions to the left. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 1908)

ROR: Rotate right

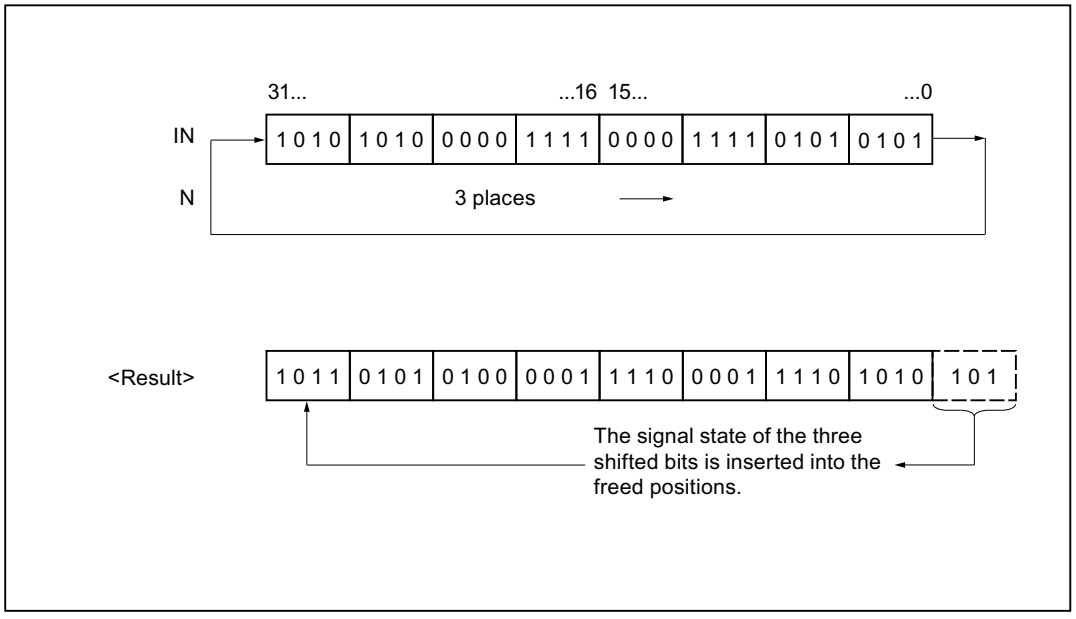
Description

The "Rotate right" instruction rotates the content of the IN parameter bit-by-bit to the right and assigns the result to the specified operand. The parameter N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:



Syntax

The following syntax is used for the instruction "Rotate right":

```
SCL
ROR(IN := <Operand>,
    N := <Operand>)
```

Parameters

The following table shows the parameters of the "Rotate right" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Number of bit positions by which the value (IN) is rotated
Function value		Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ROR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	0000 1111 1001 0101
N	Tag_Number	5
Function value	Tag_Result	1010 1000 0111 1100

The content of the "Tag_Value" operand is rotated by five bit positions to the right. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 1908)

ROL: Rotate left

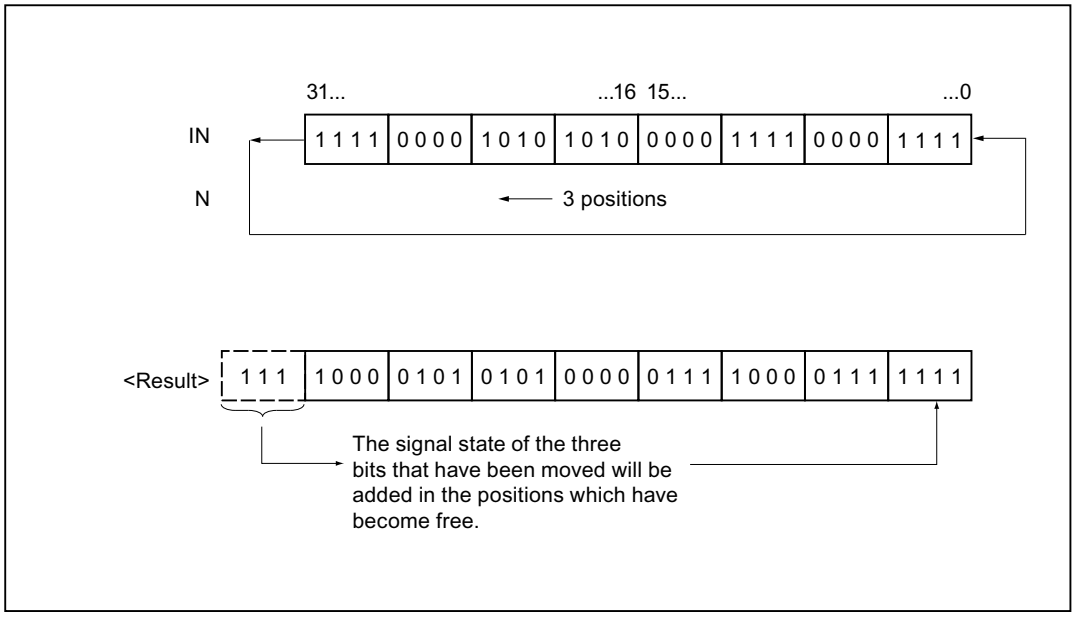
Description

The "Rotate left" instruction rotates the contents of the IN parameter bit-by-bit to the left and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:



Syntax

The following syntax is used for the "Rotate left" instruction:

SCL

```
ROL(IN := <Operand>,
    N := <Operand>)
```

Parameters

The following table shows the parameters of the "Rotate left" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Number of bit positions by which the value (IN) is rotated
Function value		Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
  SCL
"Tag_Result" := ROL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1010 1000 1111 0110
N	Tag_Number	5
Function value	Tag_Result	0001 1110 1101 0101

The content of the operand "Tag_Value" is rotated five bit positions to the left. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 1908)

Legacy

DRUM: Implement sequencer

Description

The "Implement sequencer" instruction is used to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a time value, an event, or both. Steps that have an event bit and the time value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a time value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask has signal state "1", the OUT_VAL value sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Implement sequencer" instruction:

SCL

```
<Instance>(RESET := <Operand>,
           JOG := <Operand>,
           DRUM_EN := <Operand>,
           LST_STEP := <Operand>,
           EVENT1 - 16 := <Operand>,
           OUT1 - 16 => <Operand>,
           Q => <Operand>,
           OUT_WORD => <Operand>,
           ERR_CODE => <Operand>)
```

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L	The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L	Number of the last programmed step.

Parameter	Declaration	Data type	Memory area	Description
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P	Current counter value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L	Previous system time
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L	Preset counter value for each step [1 to 16] where one clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value at the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value at the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

SCL

```
"DRUM_DB"(RESET := "Tag_Reset"
          JOG := "Tag_Input_Jog"
          DRUM_EN := "Tag_Input_DrumEN"
          LST_STEP := "Tag_Number_LastStep"
          EVENT1 := "MyTag_Event_1"
          EVENT2 := "MyTag_Event_2"
          EVENT3 := "MyTag_Event_3"
          EVENT4 := "MyTag_Event_4"
          EVENT5 := "MyTag_Event_5"
          EVENT6 := "MyTag_Event_6"
          EVENT7 := "MyTag_Event_7"
          EVENT8 := "MyTag_Event_8"
          EVENT9 := "MyTag_Event_9"
          EVENT10 := "MyTag_Event_10"
          EVENT11 := "MyTag_Event_11"
          EVENT12 := "MyTag_Event_12"
          EVENT13 := "MyTag_Event_13"
          EVENT14 := "MyTag_Event_14"
          EVENT15 := "MyTag_Event_15"
          EVENT16 := "MyTag_Event_16"
          OUT1 => "MyTag_Output_1"
          OUT2 => "MyTag_Output_2"
          OUT3 => "MyTag_Output_3"
          OUT4 => "MyTag_Output_4"
          OUT5 => "MyTag_Output_5"
          OUT6 => "MyTag_Output_6"
          OUT7 => "MyTag_Output_7"
          OUT8 => "MyTag_Output_8"
          OUT9 => "MyTag_Output_9"
          OUT10 => "MyTag_Output_10"
          OUT11 => "MyTag_Output_11"
          OUT12 => "MyTag_Output_12"
          OUT13 => "MyTag_Output_13"
          OUT14 => "MyTag_Output_14"
          OUT15 => "MyTag_Output_15"
```

SCL

```
OUT16 => "MyTag_Output_16"
Q => "Tag_Output_Q"
OUT_WORD => "Tag_OutputWord"
ERR_CODE => "Tag_ErrorCode");
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameter	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE

Parameter	Address	Value
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE

Parameter	Operand	Address	Value
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 1908)

DCAT: Discrete control-timer alarm

Description

The "Discrete control-timer alarm" instruction is used to accumulate the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state on the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET < PT), OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".

- When the signal state of the parameters CMD and CMD_HIS is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state on the parameter OA is set to "1". If the value of the parameter ET does not exceed the value of the parameter PT, the signal state on the parameter OA is reset to "0". The value at the parameter CMD_HIS is set to the value of the parameter CMD.
- If the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of parameter ET is set to the value of parameter PT. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of parameter CMD_HIS is set to the value of parameter CMD.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state of the parameter CA is set to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of parameter CMD_HIS is set to the value of parameter CMD.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of parameter ET is set to the value of parameter PT. If the signal state of the parameter C_FB changes to "0", the alarm is set the next time the instruction is executed. The value of parameter CMD_HIS is set to the value of parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Syntax

The following syntax is used for the "Discrete control-timer alarm" instruction:

SCL

```
<Instance>(CMD := <Operand>,
           O_FB := <Operand>,
           C_FB := <Operand>,
           Q => <Operand>,
           OA => <Operand>,
           CA => <Operand>)
```

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	D, L	Preset time value, where one count = 1 ms.
PREV_TIME	Static	DWORD	D, L	Previous system time
CMD_HIS	Static	BOOL	D, L	CMD history bit

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.

SCL

```
"DCAT_DB"(CMD := "Tag_Input_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          Q => "Tag_Output_Q",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA");
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 1908)

MCAT: Motor control-timer alarm

Description

The "Motor control-timer alarm" instruction is used to accumulate the time from the point at which one of the command inputs (opening or closing) is switched on. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single

instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	State
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to the signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore not possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

Syntax

The following syntax is used for the "Motor control-timer alarm" instruction:

```
SCL
<Instance>(O_CMD := <Operand>,
           C_CMD := <Operand>,
           S_CMD := <Operand>,
           O_FB := <Operand>,
           C_FB := <Operand>,
           00 => <Operand>,
           C0 => <Operand>,
           OA => <Operand>,
           CA => <Operand>,
           Q => <Operand>)
```

Parameter

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	A signal state of "0" indicates an error condition.
ET	Static	DINT	D, L	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	D, L	Preset time value, where one count = 1 ms
PREV_TIME	Static	DWORD	D, L	Previous system time
O_HIS	Static	BOOL	D, L	"Open" history bit
C_HIS	Static	BOOL	D, L	"Close" history bit

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

SCL

```
"MCAT_DB"(O_CMD := "Tag_Iput_O_CMD",
          C_CMD := "Tag_Input_C_CMD",
          S_CMD := "Tag_Input_S_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          OO => "Tag_OutputOpen",
          CO => "Tag_OutputClosed",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA",
          Q => "Tag_Output_Q");
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 1908)

IMC: Compare input bits with the bits of a mask

Description

The "Compare input bits with the bits of a mask" instruction is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. On the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare input bits with the bits of a mask" instruction:

SCL

```
<Instance>(IN_BIT0 - 15 := <Operand>,
           CMP_STEP := <Operand>,
           OUT => <Operand>,
           ERR_CODE => <Operand>)
```

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P	The step number of the mask used for the comparison.

Parameter	Declaration	Data type	Memory area	Description
OUT	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 1908)

SMC: Compare scan matrix

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare scan matrix" instruction:

```
SCL
<Instance>(IN_BIT0 - 15 := <Operand>,
           OUT => <Operand>,
           OUT_STEP => <Operand>,
           ERR_CODE => <Operand>)
```

Syntax

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 1908)

LEAD_LAG: Lead and lag algorithm

Description

Use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The instruction "Lead and lag algorithm" supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output at the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of output OUT so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Lead and lag algorithm" instruction:

SCL

```
<Instance>(IN := <Operand>,
           SAMPLE_T := <Operand>,
           OUT => <Operand>,
           ERR_CODE => <Operand>)
```

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P	The input value of the current sample time (cycle time) to be processed. Constants can also be specified at the IN parameter.
SAMPLE_T	Input	INT	I, Q, M, D, L, P	Sample time Constants can also be specified at the SAMPLE_T parameter.
OUT	Output	REAL	I, Q, M, D, L	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P	Previous output

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

```
SCL
"LEAD_LAG_DB"(IN := "Tag_Input",
              SAMPLE_T := "Tag_Input_SAMPLE_T",
              OUT => "Tag_Output_Result",
              ERR_CODE => "Tag_ErrorCode");
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

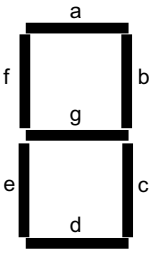
Overview of the valid data types (Page 1908)

SEG: Create bit pattern for seven-segment display

Description

The "Create bit pattern for seven-segment display" instruction is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a 7-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	**Assignment of the segments - g f e d c b a**	**Display** (Hexadecimal)	**Seven-segment display**
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

Syntax

Use the following syntax for the "Create bit pattern for seven-segment display" instruction:

SCL

```
SEG(IN := <Operand>,
    OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display
Function value		VOID		Empty function value

Example

The following example shows how the instruction works:

```
SCL
SEG(IN := "Tag_Input",
    OUT => "Tag_Output");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
		Hexadecimal	Binary
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW16#065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 1908)

BCDCPL: Create tens complement

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the operand. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Syntax

The following syntax is used for the "Create tens complement" instruction:

```
SCL
BCDCPL(<Operand>)
```

Parameter

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Bit strings	I, Q, M, D, L, P	7-digit BCD number
Function value		DWORD	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := BCDCPL("Tag_Input");
```

The following table shows how the instruction functions using specific values:

Operand	Value*
Tag_Input	DW#16#01234567
Tag_Result	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 1908)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that are set to the signal state "1".

Syntax

The following syntax is used for the "Count number of set bits" instruction:

SCL

```
BITSUM(<Operand>)
```

Parameter

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	DWORD	I, Q, M, D, L, P	Operand whose set bits are counted
Function value		INT	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := BITSUM("Tag_Input");
```

The following table shows how the instruction functions using specific values:

Operand	Value*
Tag_Input	DW#16#12345678
Tag_Result	W#16#000D (13 bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 1908)

11.6.3 Extended instructions

11.6.3.1 Date and time-of-day

T_COMP: Compare time tags

Description

This instruction is used to compare the contents of two tags of the "Timers" or "Date and time" data type.

The instruction supports comparisons of the following data types: DATE, TIME, LTIME, TOD (TIME_OF_DAY), LTOD (LTIME_OF_DAY), DT (DATE_AND_TIME), LDT (DATE_AND_LTIME), DTL, S5Time.

The data types must be the same length and format to carry out the comparison.

The comparison result is output at the OUT parameter as a return value. For this purpose, the parameter OUT is set to "1" if the condition used for the comparison is satisfied.

The following comparison options can be used:

Symbol	Description
EQ	The return value has the signal state “1" if the time is the same at the parameter IN1 and IN2.
NE	The return value has the signal state “1" if the time at parameters IN1 and IN2 is not identical.
GE	The return value has the signal state "1" if the time at parameter IN1 is greater (more recent) than or equal to the time at parameter IN2 .
LE	The return value has the signal state "1" if the time at parameter IN1 is less (less recent) than or equal to the time at parameter IN2.
GT	The return value has the signal state “1" if the time at parameter IN1 is greater (more recent) than the time at parameter IN2.
LT	The return value has the signal state "1" if the time at parameter IN1 is less (less recent) than the time at parameter IN2 .

Parameters

The following table shows the parameters of the instruction "T_COMP":

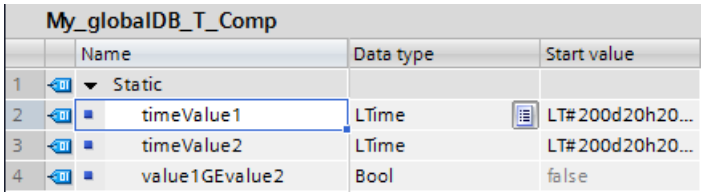
Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DATE, TIME, LTIME, TOD, LTOD, DT, LDT, DTL, S5Time	I, Q, M, D, L, P or constant	First value to be compared.
IN2	Input	DATE, TIME, LTIME, TOD, LTOD, DT, LDT, DTL, S5Time	I, Q, M, D, L, P or constant	Second value to be compared.
OUT	Output	BOOL	I, Q, M, D, L, P	Return value

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

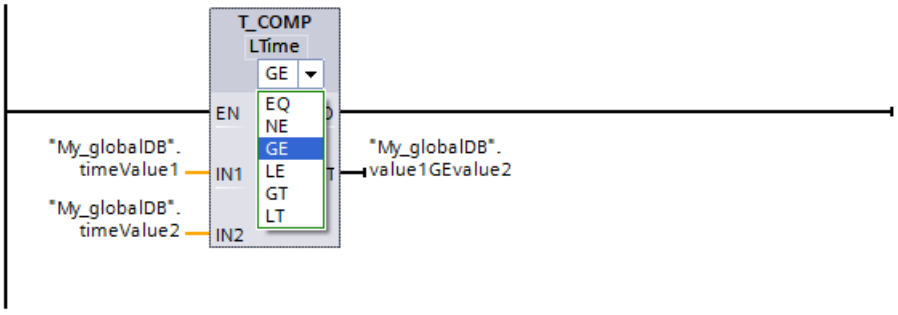
Example

In the following example, you use the "Greater or equal" comparison option to compare two times with the LTIME data type.

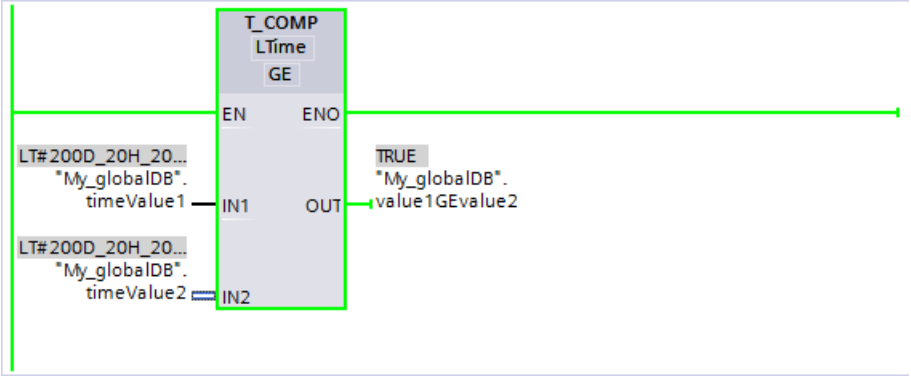
Create three tags in a global data block for storing the data.



Interconnect the parameters of the instruction as follows. Select the "GE" comparison option.



Since the time of the first value to be compared ("timeValue1") is greater than or equal to the second value ("timeValue2"), the return value ("value1GEvalue2") shows the signal state "TRUE".



T_CONV: Convert times and extract

Description

You use the instruction "T_CONV" to convert the data type of the IN input parameter to the data type that is output at the OUT output. You select the data formats for the conversion from the instruction boxes of the input and output.

Parameters

The following table shows the parameters of the instruction "T_CONV". If an input and output parameter of the same data type is used, the instruction copies the corresponding value.

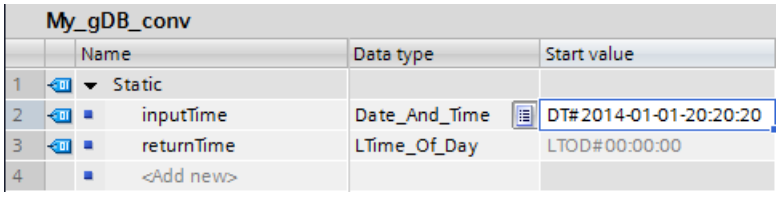
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Integers, TIME, date and time*	WORD, integers, timers, date and time*	I, Q, M, D, L, P or constant	Value to be converted
OUT	Return	Integers, TIME, date and time*	WORD, integers, timers, date and time*	I, Q, M, D, L, P	Result of the conversion
* The range of supported data types depends on the CPU. Please refer to the overviews of valid data types for information on the data types supported by the S7-1200 and S7-1500 modules.					

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

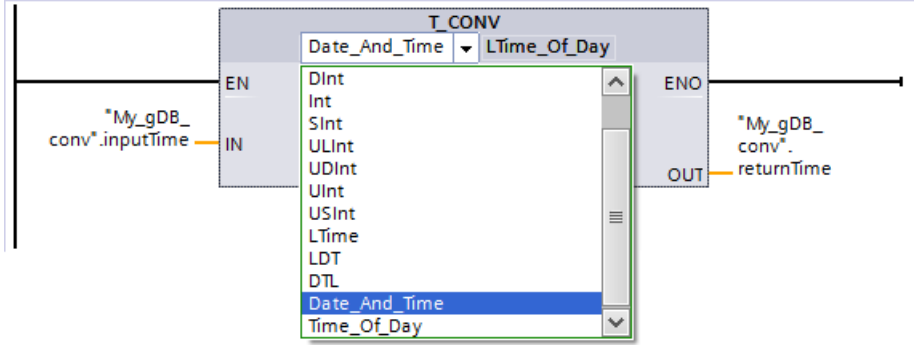
Example

In the following example, you convert a time with the DATE AND TIME data type to a time with the LTIME OF DAY data type.

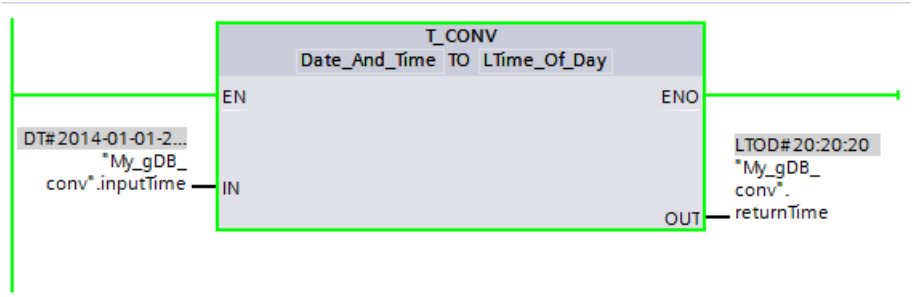
Create two tags in a global data block for storing the data.



Interconnect the parameters of the instruction as follows and select the data types.



The value to be converted ("inputTime") is output at the output parameter as a new value ("returnTime"). The information for the date is lost.



T_ADD: Add times

Description

You use this instruction to add the time information in the IN1 input to the time information in the IN2 input. You can query the result in the OUToutput parameter. You can add the following formats:

- Addition of a time period to another time period.
  Example: Addition of a TIME data type to another TIME data type.
- Addition of a time period to a time.
  Example: Addition of a TIME data type to the DTL data type.

The data type for the value at input parameter IN1 and output parameter OUT is defined by the selection in the instruction boxes of the input and output. You can only specify time information in TIME format in the IN2 input parameter (for S7-1500 modules also LTIME).

Parameters

The following tables show the parameters of the "T_ADD" instruction, according to the possible conversions:

Table 11-34 Addition of a time period to another time period

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Second number to be added
OUT	Return	DINT, DWORD, TIME, TOD	TIME, LTIME,	I, Q, M, D, L, P	Result of addition The data type selection depends on the data types selected for the IN1 and IN2 input parameters.

Table 11-35 Addition of a time period to a time

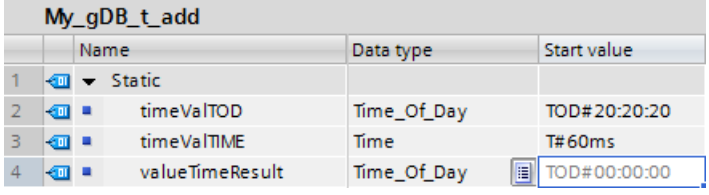
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DTL, TOD	DT, TOD, LTOD, LDT, DTL	I, Q, M, D, L, P or constant	First number to be added For LTIME at parameter IN2, only LTOD, LDT or DTL can be used.
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Second number to be added
OUT	Return	DINT, DWORD, TIME, TOD, UDINT, DTL	DT, DTL, LDT, TOD, LTOD	I, Q, M, D, L, P	Result of addition The data type selection depends on the data types selected for the IN1 and IN2 input parameters.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

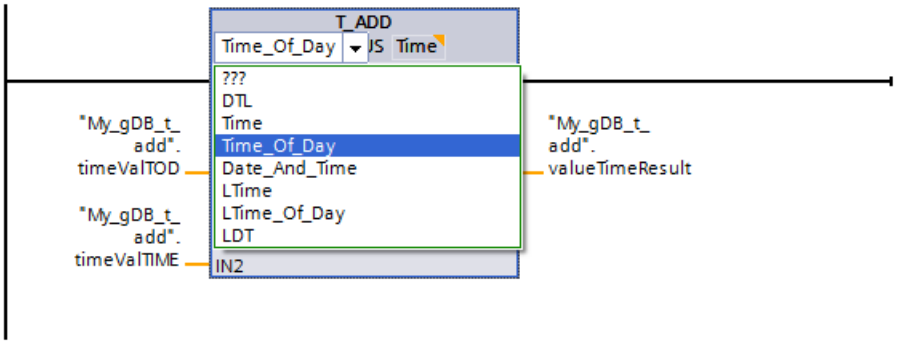
Example

In the following example, you add a time duration with the TIME data type to a time of day with the TOD data type.

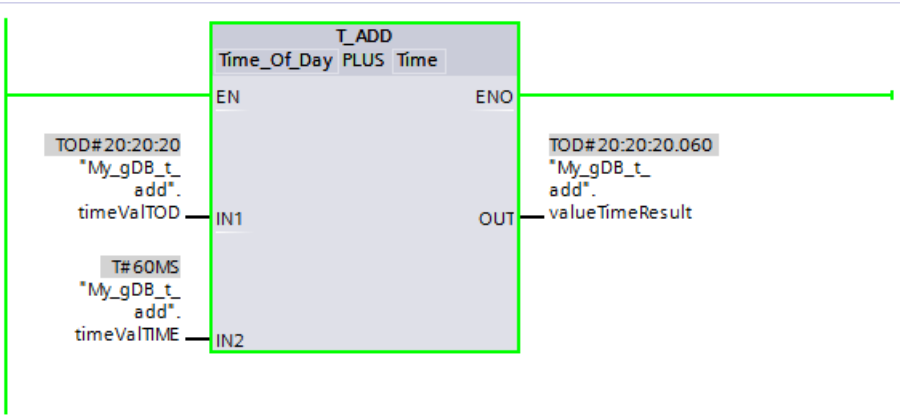
Create three tags in a global data block for storing the data.



Interconnect the parameters of the instruction as follows. You select the data types for time of day ("timeValTOD") and time period ("timeValTIME").



The time of day ("timeValTOD") and the time period ("timeValTIME") are added and the result is displayed as a time of day at output parameter OUT ("valueTimeResult").



T_SUB: Subtract times

Description

You use this instruction to subtract the time information in the IN2 input parameter from the time information in the IN1 input parameter. You can query the difference in the OUT output parameter. You can subtract the following formats:

- Subtraction of a time period from another time period
  Example: Subtraction of a time period of the data type TIME from another time period of the data type TIME. The result can be output to a tag with the TIME format.
- Subtraction of a time period from a time
  Example: Subtraction of a time period of the data type TIME from a time of the data type DTL. The result can be output to a tag with the DTL format.

You decide the formats of the values in the IN1 input parameter and the OUT output parameter by selecting the data types for the input and output parameters of the instruction.

Parameters

The following tables show the parameters of the "T_SUB" instruction, according to the possible conversions:

Table 11-36 Subtraction of a time period from another time period

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Minuend
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Subtrahend
OUT	Return	DINT, DWORD, TIME, TOD, UDINT	TIME, LTIME	I, Q, M, D, L, P	Result of subtraction

Table 11-37 Subtraction of a time period from a time

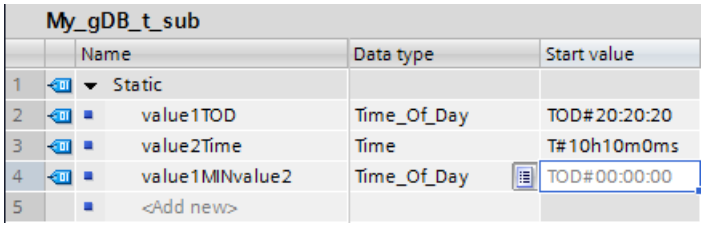
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DTL, TOD	TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Minuend For LTIME at parameter IN2, only LTOD, LDT or DTL can be used.
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Subtrahend
OUT	Return	DTL, DINT, DWORD, TIME, TOD, UDINT	TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P	Result of subtraction

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

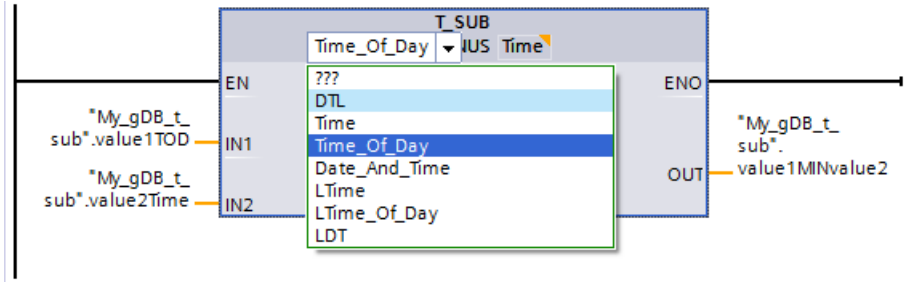
Example

In the following example, you subtract a time period with the TIME data type from a time of day with the TOD data type.

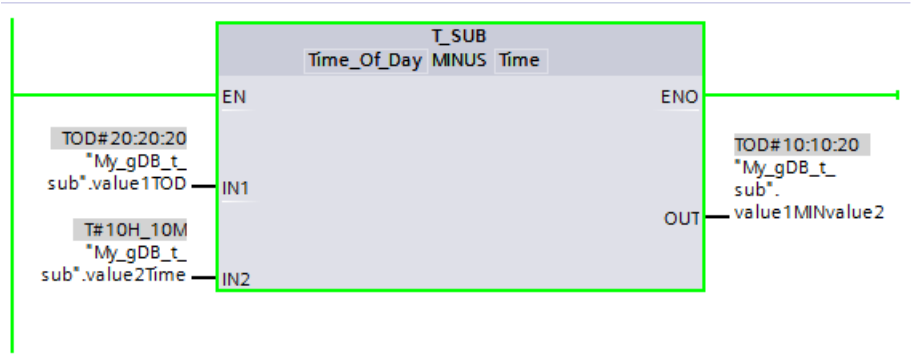
Create three tags in a global data block for storing the data.



Interconnect the parameters of the instruction as follows. You select the data type for the time of day ("value1TOD") and the data type for the time period ("value2Time").



The time of day ("value1TOD") and the time period ("value2Time") are subtracted and the result is displayed as a time of day at output parameter OUT ("value1MINvalue2").



T_DIFF: Time difference

Description

You use this instruction to subtract the time information in the IN2 input parameter from the time information in the IN1 input parameter. The result is sent at output parameter OUT.

- If the time information at the IN2 input parameter is greater than the time information at the IN1 input parameter, the result is output as a negative value at the OUT output parameter.
- If the result of the subtraction is outside the TIME range, the result is set to "0" (0:00) and the enable output ENO = "0".

Parameters

The following table shows the parameters of the "T_DIFF" instruction:

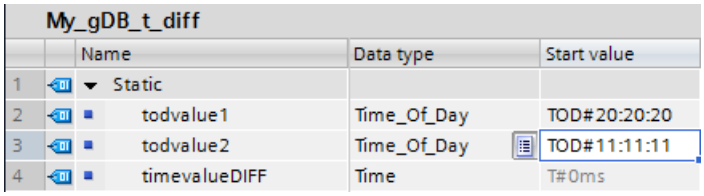
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DTL, DATE, TOD	DTL, DATE, DT, TOD, LTOD, LDT	I, Q, M, D, L, P or constant	Minuend
IN2	Input	DTL, DATE, TOD	DTL, DATE, DT, TOD, LTOD, LDT	I, Q, M, D, L, P or constant	Subtrahend
OUT	Return	TIME, INT	TIME, LTIME, INT	I, Q, M, D, L, P	Difference of input parameters

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

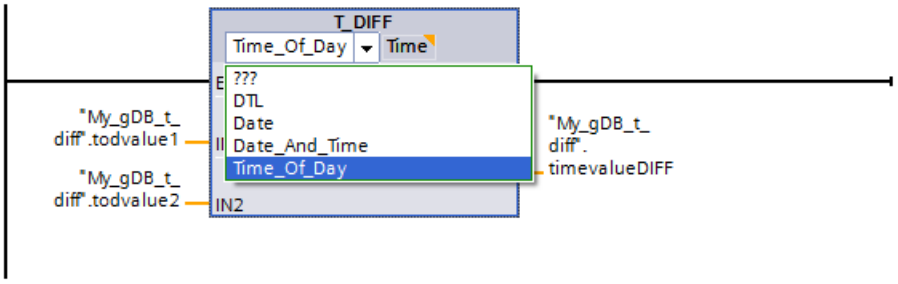
Example

In the following example, you calculate the difference between two times of day with the TOD data type. You specify the difference in the "TIME" data type.

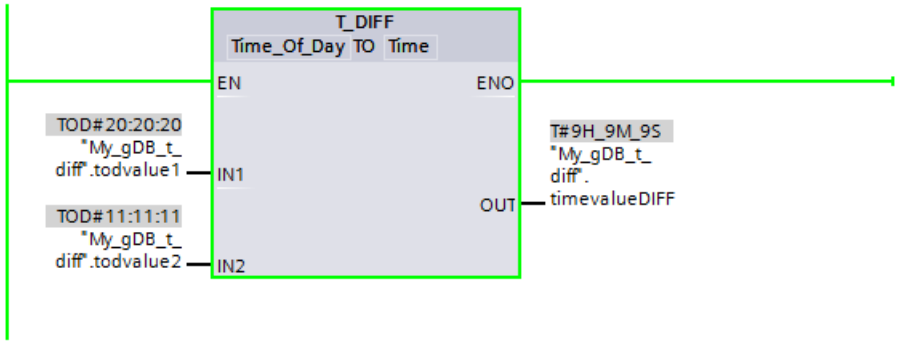
Create three tags in a global data block for storing the data.



Interconnect the parameters of the instruction as follows and select the data types. With the first selection option, you specify the data type of the points in time. With the second selection option, you specify the data type of the difference.



The first time of day ("todvalue1") and the second time of day ("todvalue2") are subtracted and the difference is displayed as a time period at output parameter OUT ("timevalueDIFF").



T_COMBINE: Combine times

Description

The instruction combines the value of a date with the value of a time and converts this into a combined date and time value.

- The date is output at the input parameter IN1. A value of between 1990-01-01 and 2089-12-31 must be used for the data type DATE (this is not checked).
- The time is input at the IN2 input value (TOD/LTOD data type).
- The combined data type for the date and time value is output at the OUT output value.

Parameters

The following table shows the parameters of the instruction "T_COMBINE":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DATE	DATE	I, Q, M, D, L, P or constant	Input tag of the date
IN2	Input	TOD	TOD, LTOD	I, Q, M, D, L, P or constant	Input tag of the time
OUT	Return	DTL	DT, DTL, LDT	I, Q, M, D, L, P	Return value of the date and time

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

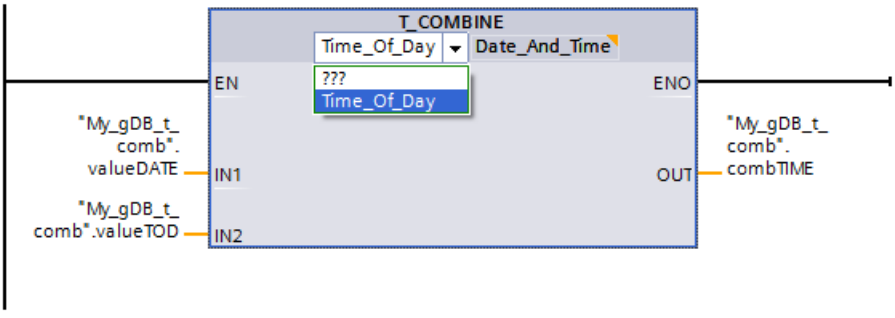
In the following example, you combine a time of day with the TOD data type and a date with the DATE data type. You specify the return value in the DT data type.

Create three tags in a global data block for storing the data.

My_gDB_t_comb

	Name	Data type	Start value
1	Static		
2	valueDATE	Date	D#2014-01-01
3	valueTOD	Time_Of_Day	TOD#20:22:20
4	combTIME	Date_And_Time	DT#1990-01-01-00:00:00

Interconnect the parameters of the instruction as follows and select the data types. With the first selection option, you specify the data type of the time of day ("valueTOD"). With the second selection option, you specify the data type of the return value ("combTIME").



The date ("valueDATE") is expanded by the specification of the time of day ("valueTOD") and the return value is displayed at output parameter OUT ("combTIME").

My_gDB_t_comb

	Name	Data type	Start value	Monitor value
1	Static			
2	valueDATE	Date	D#2014-01-01	D#2014-1-1
3	valueTOD	Time_Of_Day	TOD#20:22:20	TOD#20:22:20
4	combTIME	Date_And_Time	DT#1990-01-01-0	DT#2014-01-01-20:22:20

Time-of-day functions

WR_SYS_T: Set time-of-day

Description

You use this instruction to set the date and time-of-day of the CPU clock. Enter the date and time-of-day at the input parameter IN . The value must be in the following range:

- With DT: min. DT#1990-01-01-0:0:0, max. DT#2089-12-31-23:59:59.999
- With LDT: min. LDT#1970-01-01-0:0:0.000000000, max. LDT##2200-12-31 23:59.999
- With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31 23:59.999

You can query whether errors have occurred during execution of the instruction in the RET_VAL output parameter.

The "WR_SYS_T" instruction cannot be used to pass information about the local time zone or daylight saving time.

Parameters

The following table shows the parameters of the "WR_SYS_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	DTL	DT, DTL, LDT	I, Q, M, D, L, P or constant *	Date and time
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Status of the instruction
* The data types DT and DTL cannot be used for the following memory areas: input, output and bit memory.					

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

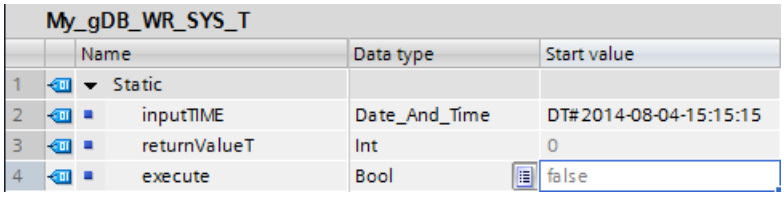
Error code* (W#16#....)	Description
0000	No error
8080	Error in date
8081	Error in time
8082	Month invalid
8083	Day invalid
8084	Hour information invalid
8085	Minute information invalid
8086	Second information invalid
8087	Nanosecond information invalid

Error code* (W#16#....)	Description
80B0	The real-time clock has failed
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

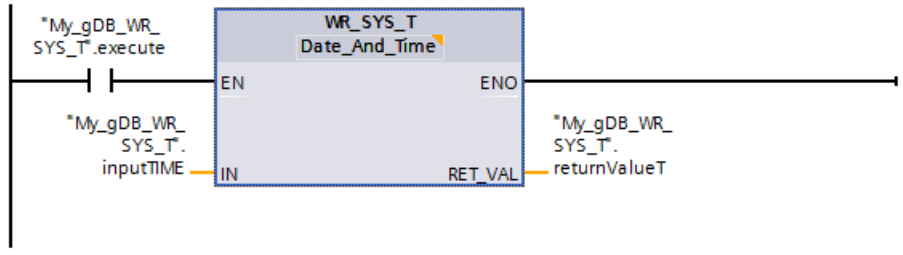
Example

In the following example, you set the date and time of the CPU clock. The data type used is DATE AND TIME.

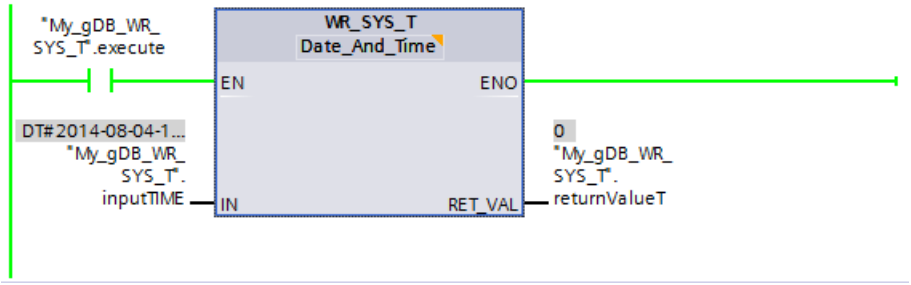
Create three tags in a global data block for storing the data.



Interconnect the parameters of the instruction as follows. Select the DATE AND TIME data type.



If the normally open contact ("execute") supplies the signal state "TRUE", the "WR_SYS_T" instruction is executed. The module time of the CPU clock is overwritten with the time to be set ("inputTIME"). The output parameter RET_VAL ("returnValueT") indicates that processing took place without errors.



You can establish whether the new module time ("inputTIME") has been correctly received by the CPU clock as follows:

- Using the display of an S7-1500 CPU: Navigate on the CPU display to "Settings > Date & Time > General".
- Using TIA Portal: Read out the module time of the CPU clock using the "RD_SYS_T (Page 2983)" instruction.
- Using TIA Portal: Navigate to the "Online & Diagnostics" entry of the CPU, and open the "Functions > Set time of day" tab.
  The Coordinated Universal Time (UTC) is set for the module time of the CPU clock. The Central European Time (local time) is set in the TIA Portal. Accordingly, an hour is added to the time to be set ("inputTIME") in the "Online & Diagnostics" entry in the TIA Portal. An additional hour is added because the settings in the TIA Portal are based on daylight saving time. The calculated local time is output in 12-hour clock format.

RD_SYS_T: Read time-of-day

Description

You use this instruction to read the current date and current time-of-day of the CPU clock.

The read dates are output at the OUT output parameter of the instruction. The provided value does not include information about the local time zone or daylight saving time.

You can query whether errors have occurred during execution of the instruction in the RET_VALoutput.

Parameters

The following table shows the parameters of the "RD_SYS_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Status of the instruction
OUT	Output	DTL	DT, DTL, LDT	I, Q, M, D, L, P	Date and time of CPU

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

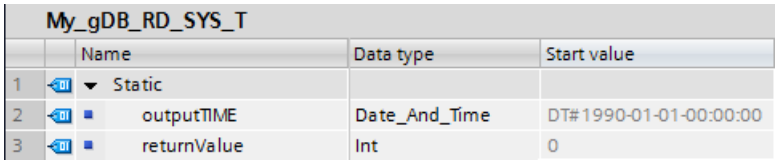
Parameter RET_VAL

Error code* (W#16#....)	Description
0000	No error
8081	Time value specified at the OUT parameter is outside the valid value range: • With DT: min. DT#1990-01-01-0:0:0, max. DT#2089-12-31-23:59:59.999 • With LDT: min. LDT#1970-01-01-0:0:0.000000000, max. LDT#2262-04-11-23:47:16.854775807 • With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2262-04-11-23:47:16.854775807
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

In the following example, you read out the module time of the CPU clock. The data type used is DATE AND TIME.

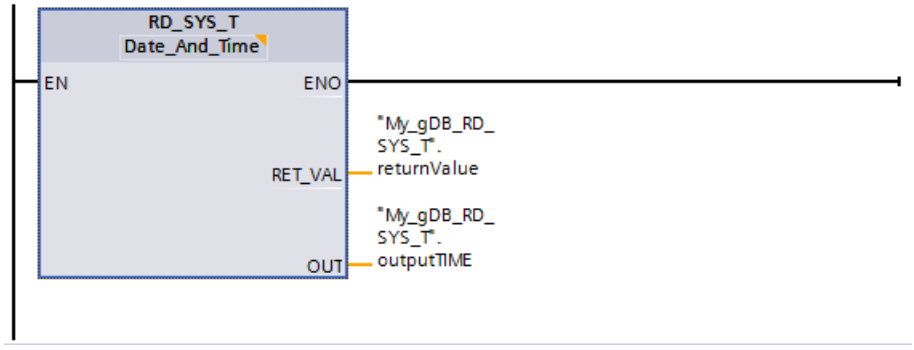
Create two tags in a global data block for storing the data.



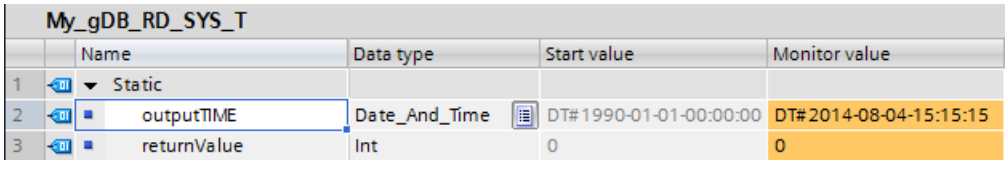
My_gDB_RD_SYS_T

	Name	Data type	Start value
1	Static		
2	outputTIME	Date_And_Time	DT#1990-01-01-00:00:00
3	returnValue	Int	0

Interconnect the parameters of the instruction as follows. Select the DATE AND TIME data type.



The module time of the CPU clock is read out and displayed at output parameter OUT ("outputTIME"). The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.



My_gDB_RD_SYS_T

	Name	Data type	Start value	Monitor value
1	Static			
2	outputTIME	Date_And_Time	DT#1990-01-01-00:00:00	DT#2014-08-04-15:15:15
3	returnValue	Int	0	0

RD_LOC_T: Read local time

Description

You use this instruction to read the current local time from the CPU clock and output this at the OUT output. Information on the time zone and the start of daylight saving time and standard time, which you have set in the configuration of the CPU clock, is used to output the local time.

Parameters

The following table shows the parameters of the "RD_LOC_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Status of the instruction
OUT	Output	DTL	DT, LDT, DTL	I, Q, M, D, L, P	Local time

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

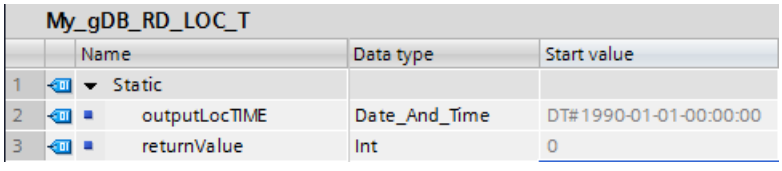
Parameter RET_VAL

Error code* (W#16#....)	Description
0000	No error
0001	No error. Local time is output as daylight saving time.
8080	Local time cannot be read.
8081	Time value specified at the OUT parameter is outside the valid value range: • With DT: min. DT#1990-01-01-0:0:0, max. DT#2089-12-31-23:59:59.999 • With LDT: min. LDT#1970-01-01-0:0:0.000000000, max. LDT#2262-04-11-23:47:16.854775807 • With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2262-04-11-23:47:16.854775807
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

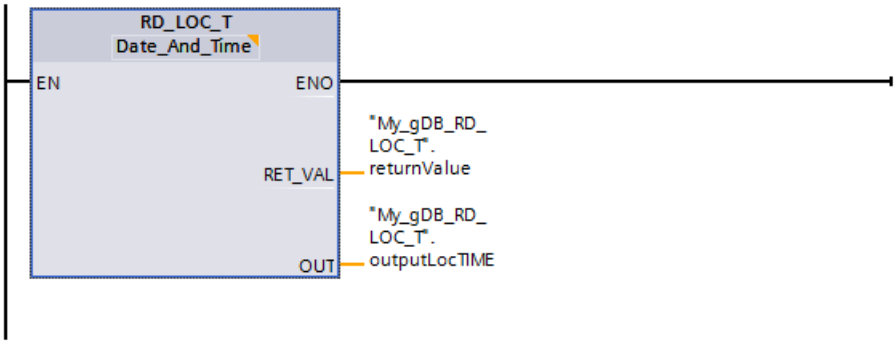
Example

In the following example, you read out the local time of the CPU clock. The data type used is DATE AND TIME.

Create two tags in a global data block for storing the data.



Interconnect the parameters of the instruction as follows. Select the DATE AND TIME data type.



The local time of the CPU clock is read out and displayed at output parameter OUT ("outputLocTIME"). Output parameter RET_VAL ("returnValue") indicates that processing took place without errors and the local time is output as daylight saving time with this call.

My_gDB_RD_LOC_T

	Name	Data type	Start value	Monitor value
1	Static			
2	outputLocTIME	Date_And_Time	DT#1990-01-01-00:00:00	DT#2014-08-04-16:15:15.001
3	returnValue	Int	0	1

WR_LOC_T: Write local time

Description

The instruction "WR_LOC_T" is used to set the date and time of the CPU clock. Enter the date and time-of-day as local time at the input parameter LOCTIME.

The value must be in the following range:

- With DT: min. DT#1990-01-01-00:00:00, max. DT#2089-12-31-23:59:59.999
- With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31 23:59.999
- With LDT: min. LDT#1970-01-01-0:0:0.000000000, max. LDT#2200-12-31 23:59.999

The granularity of the time information for local time and system time is product-specific and is at least one millisecond. Input values at the LOCTIME parameter which are less than those supported by the CPU are rounded up during system time calculation.

You can query whether errors have occurred during execution of the instruction in the RET_VAL output parameter.

Use of WR_LOC_T during the time change to daylight saving or standard time

- Changeover from standard to daylight saving time
  In the following it is assumed that the time of the changeover is 2:00 am, and that the time is moved forward by one hour. This means there is no hour between 02:00:00:000000000 AM and 02:59:59:999999999 AM.
  When you specify a corresponding time for LOCTIME, the error code W#16#8089 is generated.
  DST is irrelevant.
- Changeover from daylight saving to standard time
  In the following it is assumed that the time of the changeover is 3:00 am, and that the time is moved back by one hour. This means there are two hours between 02:00:00:000000000 AM and 02:59:59:999999999 AM.
  For all times for LOCTIME that are between 02:00:00:000000000 AM and 02:59:59:999999999 AM, you therefore need to declare whether the time is before or after the time changeover. The DST parameter is used for this:
  - With DST=TRUE, the time is in the first of the two hours, i.e. still in daylight saving time.
  - With DST=FALSE, the time is in the second of the two hours, i.e. in standard time.

  For all times for LOCTIME that are outside the double hour, DST is irrelevant.

Parameters

The following table shows the parameters of the "WR_LOC_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
LOCTIME	Input	DTL	DT, DTL, LDT	D, L, P or constant	Local time
DST	Input	BOOL	BOOL	I, Q, M, D, L, P, T, C or constant	**D**aylight **S**aving **T**ime Is only evaluated during the "double hour" at changeover to standard time. • TRUE = daylight saving time (first hour) • FALSE = standard time (second hour)
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Error message (see "RET_VAL parameter")

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

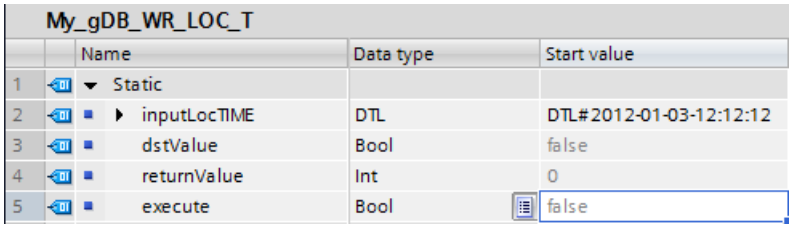
Error code* (W#16#....)	Description
0000	No error.
8080	The LOCTIME parameter has an invalid value.

Error code* (W#16#....)	Description
8081	Time value specified at the LOCTIME parameter is outside the valid value range: • With DT: min. DT#1990-01-01-00:00:00, max. DT#2089-12-31-23:59:59.999 • With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31 23:59.999 • With LDT: min. LDT#1970-1-1-0:0:0.000000000, max. LDT#2200-12-31 23:59.999
8082**	Invalid value specified for the month (byte 2 in DTL format).
8083**	Invalid value specified for the day (byte 3 in DTL format).
8084**	Invalid value specified for the hour (byte 5 in DTL format).
8085**	Invalid value specified for the minute (byte 6 in format DTL).
8086**	Invalid value specified for the second (byte 7 in DTL format).
8087**	Invalid value specified for the nanosecond (byte 8 to 11 in DTL format).
8089	Time value does not exist (hour already passed upon changeover to daylight saving time).
80B0	The real-time clock has failed.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	
** Only for local time information at the LOCTIME parameter in DTL format.	

Example

In the following example, you set the local time of the CPU clock. The data type used is DTL.

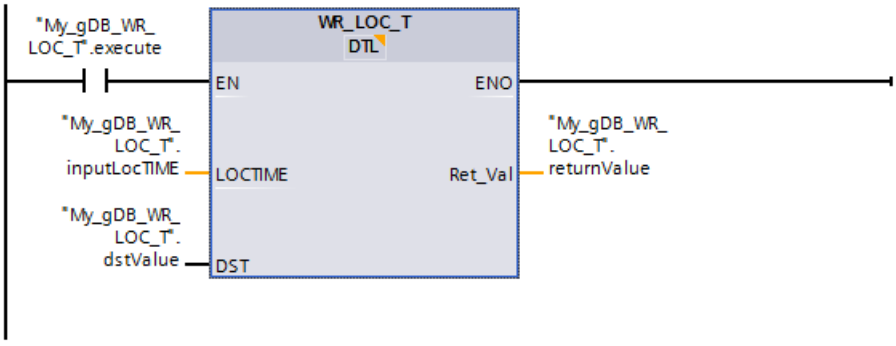
Create four tags in a global data block for storing the data.



My_gDB_WR_LOC_T

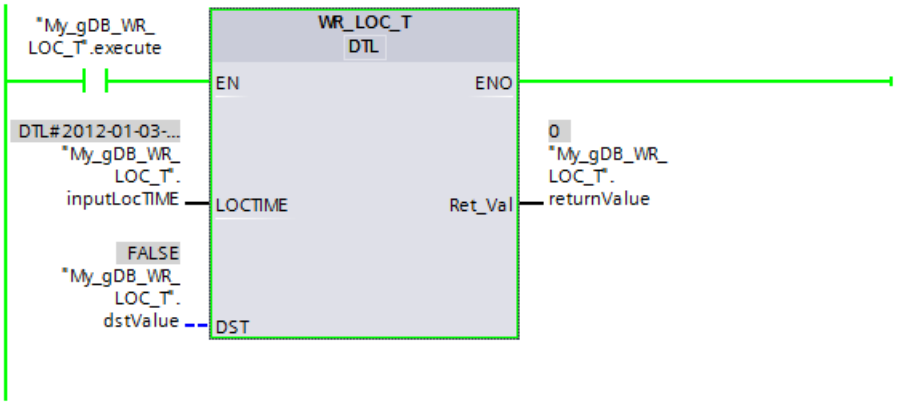
	Name	Data type	Start value
1	Static		
2	inputLocTIME	DTL	DTL#2012-01-03-12:12:12
3	dstValue	Bool	false
4	returnValue	Int	0
5	execute	Bool	false

Interconnect the parameters of the instruction as follows. Select the DTL data type.



If the normally open contact ("execute") supplies the signal state "TRUE", the "WR_LOC_T" instruction is executed. The local time of the CPU clock is overwritten with the time to be set ("inputLocTIME"). The output parameter RET_VAL ("returnValue") indicates that processing

took place without errors. Input parameter DST ("dstValue") specifies that the time information refers to standard time. This parameter is only relevant for the "double hour".



You can establish whether the new local time ("inputLocTIME") has been correctly received by the CPU clock as follows:

- Using the display of an S7-1500 CPU: Navigate on the CPU display to "Settings > Date & Time > General".
- Using TIA Portal: Read out the local time of the CPU clock using the "RD_LOC_T (Page 2985)" instruction.
- Using TIA Portal: Navigate to the "Online & Diagnostics" entry of the CPU, and open the "Functions > Set time of day" tab.
  The local time is output in 12-hour clock format.

SET_TIMEZONE: Set time zone

Description

Use the instruction "SET_TIMEZONE" to set the parameter for the local time zone and the daylight saving / standard time changeover.

The settings which are carried out with the instruction "SET_TIMEZONE" correspond with the settings for the time-of-day in the properties of the CPU. Define the corresponding parameters in the system data type TimeTransformationRule for execution of the instruction "SET_TIMEZONE".

The local time is calculated based on the system time using the settings for the time zone and the daylight saving / standard time changeover. The system time of the CPU is the UTC time. The system time is used exclusively for communication within the system.

Note

Use with CPUs of the S7-1500 series

The "SET_TIMEZONE" instruction can only be used with a CPU of the S7-1500 series starting from firmware version V1.7.

Parameters

The following table shows the parameters of the "SET_TIMEZONE" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, P or constant	Setting of parameters that are stored at parameter TimeZone.
TimeZone	Input	TimeTransformationRule	D, L	You interconnect the TimeTransformationRule system data type (see below) at parameter TimeZone.
DONE	Output	BOOL	I, Q, M, D, L, P	Status parameter: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L, P	Status parameter: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L, P	Status parameter: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L, P	Detailed error and status information is output at the parameter STATUS. The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

TimeZone parameter

You store the parameters for the local time zone and the daylight saving/standard time changeover in the TimeTransformationRule system data type

You create the TimeTransformationRule by entering TimeTransformationRule as the data type in a data block or the local interface of a function block.

The structure of the TimeTransformationRule is as follows:

Name	Data type	Description
TimeTransformationRule	STRUCT	

Name	Data type	Description
Bias	INT	Time difference between the local time and the system time (UTC) in minutes. The value must be between -720 and +780 minutes (-12 to +13 hours). The value (UTC -12 to +13 hours) corresponds to the time zones that you specify in the properties of the CPU.
DaylightBias	INT	Time difference between standard and daylight saving time in minutes. The value must be between 0 and 120 minutes. • The value "0" deactivates the changeover between daylight saving time and standard time. The values for "DaylightStart..." and "StandardStart..." are set to "0". Only the value of Bias is evaluated (time difference local time/ system time). • If the value is not "0", all tags of the TimeTransformationRule structure are evaluated. If an entry is invalid, error code 808F is output at parameter STATUS.
Specification of the time of day for changeover to daylight saving time. The following times always refer to the local time.		
DaylightStartMonth	USINT	Month in which change is made to daylight saving time: 1 = January 2 = February 3 = March ... 12 = December
DaylightStartWeek	USINT	Week in which change is made to daylight saving time. 1 = First occurrence of the weekday in the month ... 5 = Last occurrence of the weekday in the month
DaylightStartWeek-day	USINT	Weekday on which change is made to daylight saving time: 1 = Sunday ... 7 = Saturday
DaylightStartHour	USINT	Hour in which change is made to daylight saving time
DaylightStartMinute	USINT	Minute in which change is made to daylight saving time
Specification of the time of day for changeover to standard time. The following times always refer to the local time.		
StandardStartMonth	USINT	Month in which change is made to standard time: 1 = January 2 = February 3 = March ... 12 = December
StandardStartWeek	USINT	Week in which change is made to standard time: 1 = First occurrence of the weekday in the month ...

Name		Data type	Description
			5 = Last occurrence of the weekday in the month
	StandardStartWeek-day	USINT	Weekday on which change is made to standard time: 1 = Sunday ... 7 = Saturday
	StandardStartHour	USINT	Hour in which change is made to standard time
	StandardStartMinute	USINT	Minute in which change is made to standard time
	TimeZoneName	STRING[80]	Name of the time zone, for example: "(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna"

STATUS parameter

Error code* (W#16#....)	Description
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
808F	The structure, content or data types of the TimeTransformationRule at the TimeZone parameter are invalid or inconsistent.
80C3	Temporary resource error: The CPU is currently processing the maximum possible number of simultaneous block calls. "SET_TIMEZONE" cannot be executed unless at least one of the block calls is finished.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

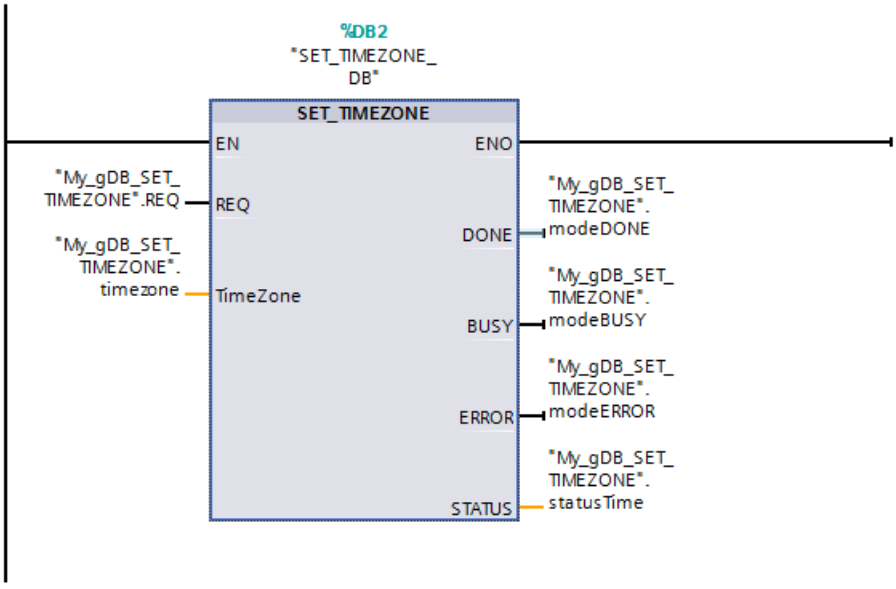
In the following example, you set the parameters for the local time zone and the daylight saving/standard time changeover.

Create the following for storing the data in a global data block: The "timezone" structure (with the TimeTransformationRule data type) and the "REQ", "modeDONE", "modeBUSY", "modeERROR", and "statusTime" tags.

My_gDB_SET_TIMEZONE

	Name	Data type	Start value
1	Static		
2	REQ	Bool	false
3	timezone	TimeTransformationRule	
4	Bias	Int	60
5	DaylightBias	Int	60
6	DaylightStartMonth	USInt	3
7	DaylightStartWeek	USInt	5
8	DaylightStartWeek...	USInt	1
9	DaylightStartHour	USInt	2
10	DaylightStartMinute	USInt	0
11	StandardStartMonth	USInt	10
12	StandardStartWeek	USInt	5
13	StandardStartWeek..	USInt	1
14	StandardStartHour	USInt	3
15	StandardStartMinute	USInt	0
16	TimeZoneName	String[80]	'My_GMT+'
17	modeDONE	Bool	false
18	modeBUSY	Bool	false
19	modeERROR	Bool	false
20	statusTime	Word	16#0

Interconnect the parameters of the instruction as follows.



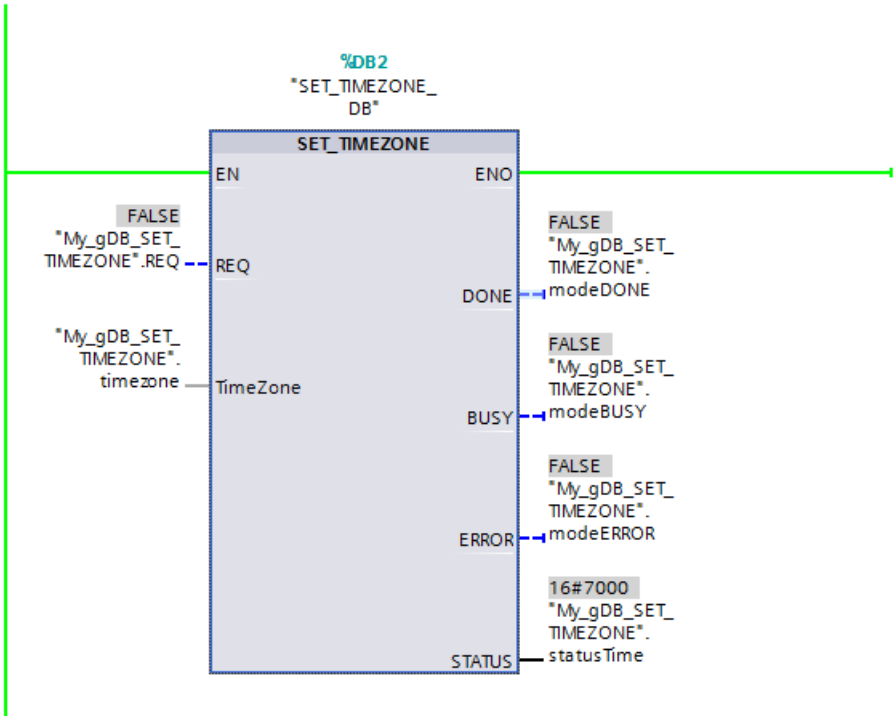
If parameter REQ supplies the signal state "TRUE", the data for the time zone of the CPU clock is overwritten with the data to be set ("timezone"). This also means:

- Output parameter BUSY ("modeBUSY") goes to signal state "TRUE". After processing, output parameter BUSY shows the value "FALSE" and output parameter DONE ("modeDONE") shows the value "TRUE".
- The output parameter STATUS ("statusTime") indicates how processing is running*. (*The job processing begins (value "7001") and the instruction is then shown as already active (value "7002").)
- The output parameter ERROR ("modeERROR") indicates that processing is running without errors (signal state is "FALSE").

My_gDB_SET_TIMEZONE

	Name	Data type	Start value	Monitor value
1	Static			
2	REQ	Bool	false	TRUE
3	timezone	TimeTransformationRule		
4	Bias	Int	60	60
5	DaylightBias	Int	60	60
6	DaylightStartMonth	USInt	3	3
7	DaylightStartWeek	USInt	5	5
8	DaylightStartWeek...	USInt	1	1
9	DaylightStartHour	USInt	2	2
10	DaylightStartMinute	USInt	0	0
11	StandardStartMonth	USInt	10	10
12	StandardStartWeek	USInt	5	5
13	StandardStartWeek..	USInt	1	1
14	StandardStartHour	USInt	3	3
15	StandardStartMinute	USInt	0	0
16	TimeZoneName	String[80]	'My_GMT+'	'My_GMT+'
17	modeDONE	Bool	false	FALSE
18	modeBUSY	Bool	false	TRUE
19	modeERROR	Bool	false	FALSE
20	statusTime	Word	16#0	16#7002

Note: The "SET_TIMEZONE" instruction is level-triggered. Only when parameter REQ supplies the signal state "TRUE" is the instruction executed.



You can establish whether the data to be set ("timezone") has been correctly received by the CPU clock as follows:

- Using the display of an S7-1500 CPU: Navigate on the CPU display to "Settings > Date & Time > Daylight Saving Time".
- Using TIA Portal: Read out the local time of the CPU clock using the "RD_LOC_T (Page 2985)" instruction.
- Using TIA Portal: Read out the module time of the CPU clock using the "RD_SYS_T (Page 2983)" instruction.

SNC_RTCB: Synchronize slave clocks

Definition: Synchronization of slave clocks

The synchronization of clock slaves refers to the transfer of the date and time-of-day from the clock master of a bus segment to all clock slaves of this bus segment.

Description

You use this instruction to synchronize all slave clocks present on a bus segment independent of the assigned synchronization interval. Successful synchronization is only possible if "SNC_RTCB" is called on a CPU whose real-time clock was assigned as the master clock for at least one bus segment.

Parameters

The following table shows the parameters of the "SNC_RTCB" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred during synchronization.
0001	The existing clock was not assigned the master clock function for any of the bus segments.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

TIME_TCK: Read system time

Description

With the "TIME_TCK" instruction, you read the system time of the CPU. The system time is a time counter that counts from 0 to a maximum of 2147483647 ms. In case of an overflow, the system time is counted again starting with "0". The time scale and the accuracy of the system time is 1 ms. The system time is only influenced by the operating modes of the CPU. You can use the system time, for example, to measure the duration of processes by comparing the results of two "TIME_TCK" calls. The instruction does not provide any error information.

The following table provides an overview of how the system time changes depending on the operating modes of the CPU.

Mode	System time ...
Startup	... is constantly updated
RUN	
STOP	... is stopped and retains the current value
Warm restart	... is deleted and restarts with "0"

Parameters

The following table shows the parameters of the instruction "TIME_TCK":

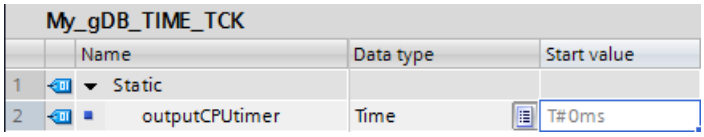
Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	TIME	I, Q, M, D, L	The RET_VAL parameter contains the read system time in the range from 0 to 2^{31} -1 ms.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

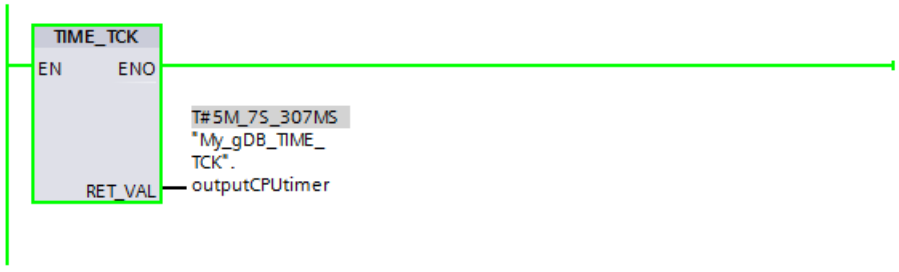
In the following example, you read out the system time of the CPU. You specify the return value in the TIME data type.

Create one tag for storing the data in a global data block.



Interconnect the parameters of the instruction as follows.

The system time of the CPU is read out and displayed at output parameter RET_VAL ("outputCPUtimer").



RTM: Runtime meters

Description

You can use this instruction to set, start, stop, and read out a 32-bit operating hours counter of your CPU.

Ensure that the operating hours counter can also be stopped or restarted during execution of the user program, which may render the saved values incorrect.

Parameters

The following table shows the parameters of the "RTM" instruction:

Parameter	Declaration	Data type	Memory area	Description
NR	Input	RTM	I, Q, M, D, L or constant	Number of the operating hours counter Numbering starts with 0. For information on the number of operating hours counters of your CPU, refer to the technical data.
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID: • 0: Read out (the status is then written to CQ and the current value to CV). After the operating hours counter has reached (2E31) -1 hours, it stops at the highest value that can be displayed and outputs an "Overflow" error message. • 1: start (at the last counter value) • 2: stop • 4: set (to the value specified in PV) • 5: set (to the value specified in PV) and then start • 6: set (to the value specified in PV) and then stop
PV	Input	DINT	I, Q, M, D, L or constant	New value for the operating hours counter
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
CQ	Output	BOOL	I, Q, M, D, L	Status of the operating hours counter (1: running)
CV	Output	DINT	I, Q, M, D, L	Current value of the operating hours counter

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the operating hours counter
8081	A negative value was passed to the PV parameter.
8082	Overflow of the operating hours counter
8091	The MODE input parameter contains an invalid value.
General error information	See also: Evaluating errors with GET_ERR_ID (Page 2922)

Example

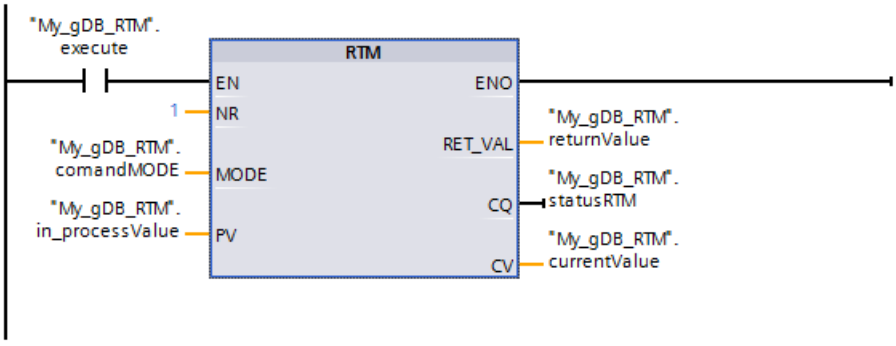
In the following example, you set the operating hours counter of the CPU and read out the value after an hour.

Create six tags in a global data block for storing the data.

My_gDB_RTM

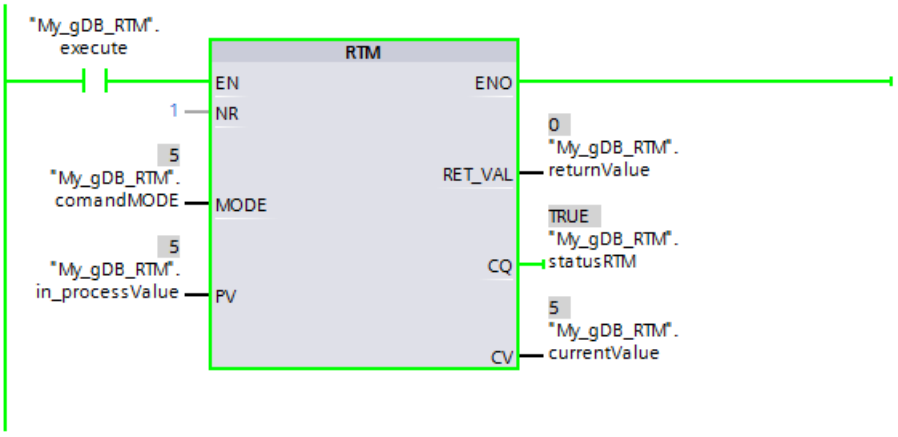
	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	in_processValue	DInt	5
4	returnValue	Int	0
5	statusRTM	Bool	false
6	currentValue	DInt	0
7	comandMODE	Byte	5

Interconnect the parameters of the instruction as follows. At input parameter NR, specify the number of the operating hours counter of the CPU.

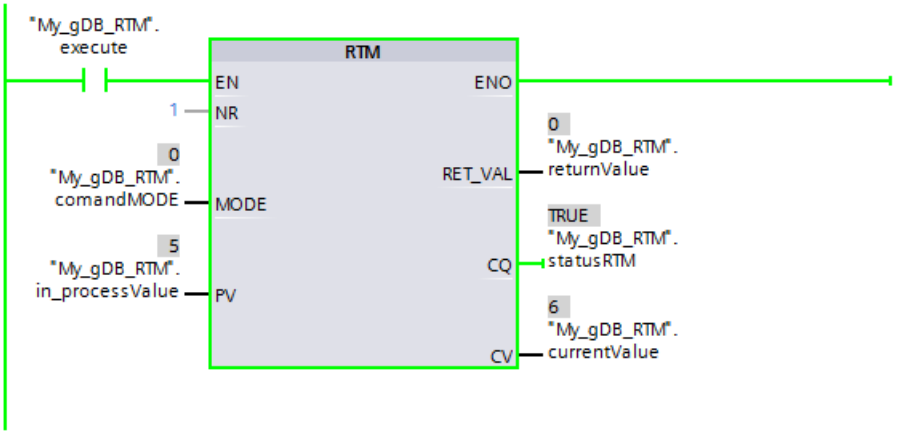


If the normally open contact ("execute") supplies the signal state "TRUE", the "RTM" instruction is executed. The operating hours counter of the CPU is set to the value to be set ("in_processValue") and started. After the start of the operating hours counter, set the value of input parameter MODE ("comandMODE") to "0". (To do so, click the parameter and select "Modify operand > Set to 0".) As a result, the "RTM" instruction only reads the current value ("currentValue") of the operating hours counter without changing it. Output parameter CQ ("statusRTM") indicates after the start of the operating hours counter that the operating hours

counter is running (value is "TRUE"). The output parameter RET_VAL ("returnValue") indicates that processing is running without errors.



After an hour, output parameter CV ("currentValue") indicates the value "6".



11.6.3.2 String + Char

S_MOVE: Move character string

Description

You can use this instruction to write the content of a character string (W)STRING from parameter IN to the data area that you specify at parameter OUT.

You can use the "MOVE_BLK" and "UMOVE_BLK" instructions to copy tags of data type ARRAY.

Parameters

The following table shows the parameters of the "S_MOVE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Source string
OUT	Output	STRING, WSTRING	D, L	Destination string

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

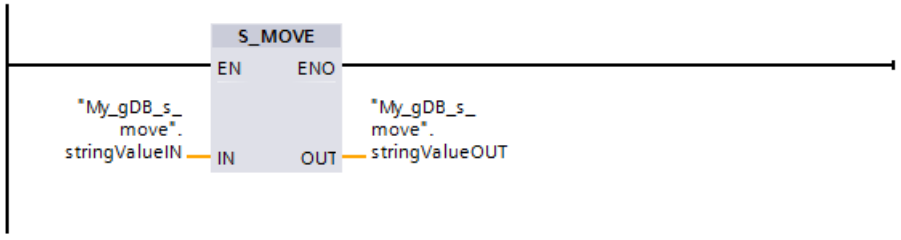
In the following example, you copy the content of a character string from input parameter IN to another character string at output parameter OUT. The data type used is STRING.

Create two tags in a global data block for storing the data.

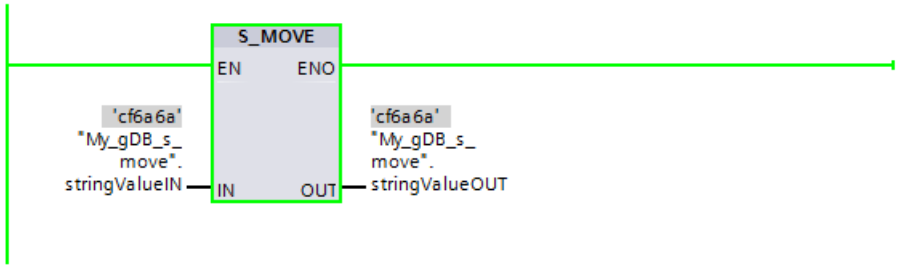
My_gDB_s_move

	Name	Data type	Start value
1	Static		
2	stringValueIN	String	'cf6a6a'
3	stringValueOUT	String	''

Interconnect the parameters of the instruction as follows.



The result of the character string to be copied ("stringValueIN") is output at output parameter OUT ("stringValueOUT").



S_COMP: Compare character strings

Description

The instruction compares the contents of two tags in the (W)STRING format and outputs the result of the comparison as a return value. The tags that are to be compared will be interconnected at the IN1 and IN2 inputs. You can only assign a symbolically defined tag for the input parameters.

Use the instruction box to select the comparison condition. If the comparison condition (for example, greater than or equal to) is satisfied, the signal state is set to "1" at the output parameter OUT .

The following comparison options can be used:

Symbol	Description
EQ	The return value has the signal state "1" if the string at the IN1 parameter is the same as the string at the IN2 parameter.
NE	The return value has the signal state "1" if the string at the IN1 parameter is not equal to the string at the IN2 parameter.
GT (1)	The return value has the signal state “1" if the string at the IN1 parameter is greater than the string at the IN2 parameter.
LT (1)	The return value has the signal state "1" if the string at the IN1 parameter is less than the string at the IN2 parameter.
GE (1)	The return value has the signal state "1" if the string at the IN1 parameter is greater than or equal to the string at the IN2 parameter.
LE (1)	The return value has the signal state "1" if the string at the IN1 parameter is less than or equal to the string at the IN2 parameter.
(1) The characters are compared by their ASCII code (for example, 'a' is greater than 'A'), starting from the left. The first character to be different decides the result of the comparison. If the first characters are the same, the longer string is greater.	

Parameters

The following table shows the parameters of the "S_COMP" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING, WSTRING*	D, L or constant	Input tag in the STRING / WSTRING format.
IN2	Input	STRING, WSTRING*	D, L or constant	Input tag in the STRING / WSTRING format.
OUT	Output	BOOL	I, Q, M, D, L	Result of comparison
* Define the maximum length of the character string if you use the data type STRING / WSTRING in the interface declaration for a temporary tag (you will find further information in the description of the data type).				

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

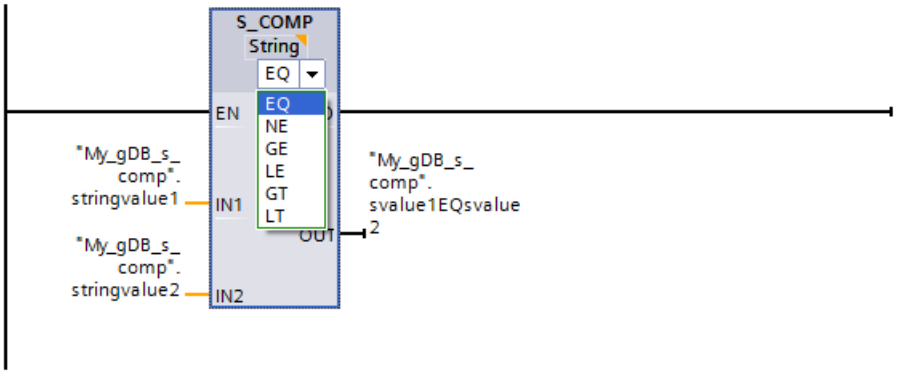
In the following example, you use the "Equal" comparison option to compare two character strings with the "STRING" data type.

Create three tags in a global data block for storing the data.

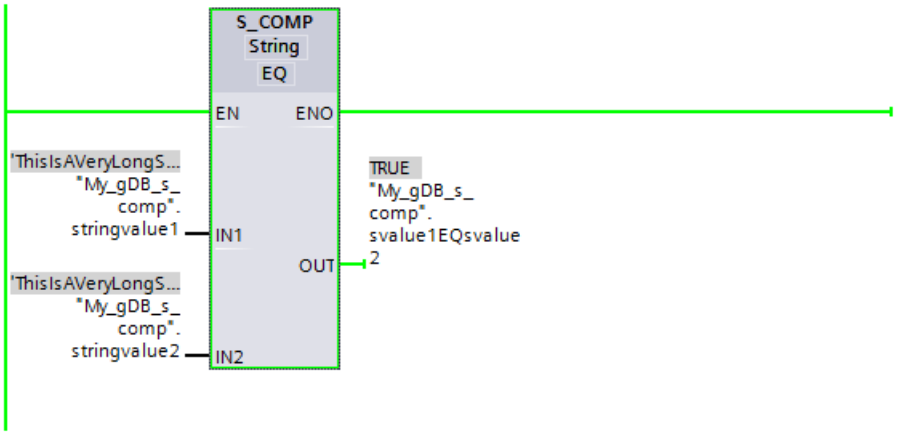
My_gDB_s_comp

	Name	Data type	Start value
1	Static		
2	stringvalue1	String	'ThisIsAVeryLongStringWithSomeNumbers646'
3	stringvalue2	String	'ThisIsAVeryLongStringWithSomeNumbers646'
4	svalue1EQsvalue2	Bool	false

Interconnect the parameters of the instruction as follows. Select the STRING data type and the EQ comparison option.



Since the value of the first character string to be compared ("stringvalue1") is equal to the second character string ("stringvalue2"), the comparison result ("svalue1EQsvalue2") shows the signal state "TRUE".



S_CONV: Convert character string

Description

You use this instruction to convert the value at the IN input to the data format you have specified in the OUT output. You decide the output format of the conversion by selecting a data type for the OUT output parameter.

The following conversions are possible:

- Conversion of a character string to:
  - Numerical value (integer or floating-point number)
    The conversion is performed for all characters of the character string specified in the IN input parameter. Permitted characters are the digits "0" to "9", the decimal point, and the plus and minus signs. The first character of the string may be a valid number or a sign. Leading spaces and exponential notations are ignored.
  - A character
    When a character string is converted to a character, the first character of the character string is transferred to parameter OUT.
  - A character string
- Conversion of a numerical value or character to a character string:
  - You decide the format of the numeric value to be converted by selecting a data type for the IN input. A valid tag of the (W)STRING data type must be specified at the OUT output. The length of the character string after conversion depends on the value at the IN input.
  - The conversion result is saved as a string starting at the third byte. The first byte of the string records the maximum length and the second byte the actual length of the character string. Positive numeric value are output without a sign.
  - If the numeric value 0, which exists as an INT or UINT data type, is converted to a string (e.g. INT_TO_STRING(0)), then the string in the result has a length of 6 characters.
  - When a numeric value is converted to a string, the first characters of the string are padded with spaces. The number of spaces depends on the length of the numerical value.
  - When a (W)CHAR character is converted, the character is written at the first position of the character string.

Note

Exponential notation during conversion from floating-point numbers

Do not use exponential notation ("e" or "E") for the conversion from floating-point numbers with the instruction "S_CONV". Instead, use the instruction "STRG_VAL (Page 3008)" for the conversion of floating-point numbers with exponential notation. You can use the FORMAT parameter of the instruction to select exponential notation as input format.

- Conversion of a character to a character

Parameters

The following tables show the parameters of the "S_CONV" instruction, according to the possible conversions:

Table 11-38 Parameters for converting a character string to a numeric value:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Value to be converted
OUT	Output	CHAR, WCHAR, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL	I, Q, M, D, L	Result of the conversion

Table 11-39 Parameters for converting a character string to a character string:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Value to be converted
OUT	Output	STRING, WSTRING	D, L	Result of the conversion (possible conversions: STRING to WSTRING and vice versa)

Table 11-40 Parameters for converting a numerical value or character to a character string:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	CHAR, WCHAR, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL	I, Q, M, D, L or constant	Value to be converted
OUT	Output	STRING, WSTRING	D, L	Result of the conversion

Table 11-41 Parameters for converting a character to a character:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	CHAR, WCHAR	I, Q, M, D, L or constant	Value to be converted
OUT	Output	CHAR, WCHAR	I, Q, M, D, L	Result of the conversion (possible conversions: CHAR to WCHAR and vice versa)

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

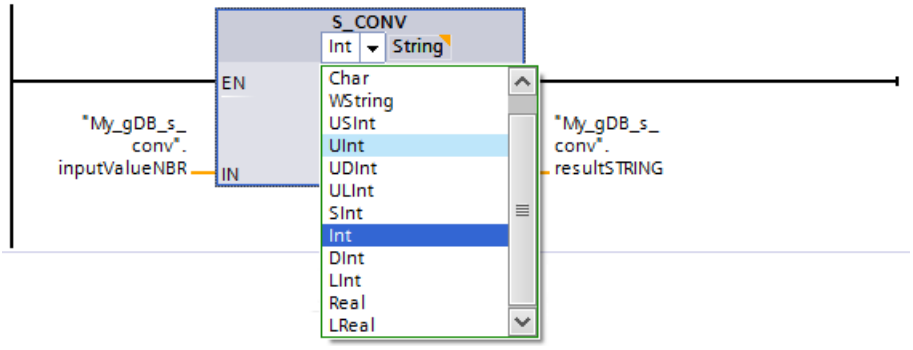
In the following example, you convert a number with the INT data type to a character string with the STRING data type.

Create two tags in a global data block for storing the data.

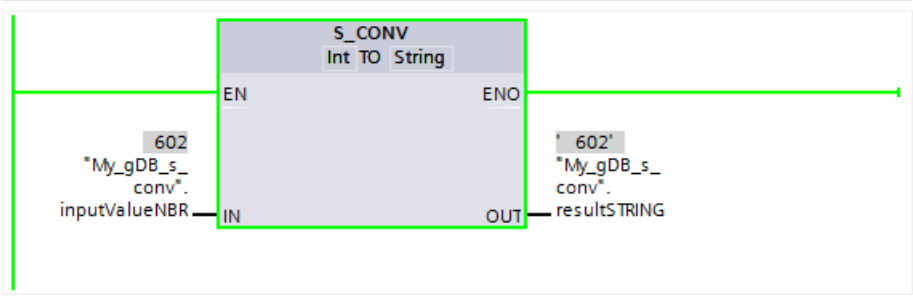
My_gDB_s_conv

	Name	Data type	Start value
1	Static		
2	inputValueNBR	Int	602
3	resultSTRING	String	''
4	<Add new>		

Interconnect the parameters of the instruction as follows. In doing so, select the data types. With the first selection option, you specify the data type of the value to be converted ("inputValueNBR"). With the second selection option, you specify the data type of the character string to be generated ("resultSTRING").



The value to be converted ("inputValueNBR") is converted to the output format. Blank spaces are written to the empty places at the start of the character string. The result of the conversion is output at output parameter OUT ("resultSTRING") as a string.



STRG_VAL: Convert character string to numerical value

Description

The "STRG_VAL" instruction converts a character string to an integer or a floating-point number:

- You specify the character string to be converted in the IN input parameter.
- You define the format of the output value by selecting a data type for the OUT output parameter.

Permitted characters for the conversion are the digits "0" to "9", the decimal point, the decimal comma, notations "E" and "e", and the plus and minus characters. The conversion can be interrupted by invalid characters.

The "STRG_VAL" instruction is not supported by the SCL programming language.

Parameters

The following table shows the parameters of the "STRG_VAL" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	STRING, WSTRING	STRING, WSTRING	D, L or constant	Number string to be converted
FORMAT	Input	WORD	WORD	I, Q, M, D, L, P or constant	Output format of the characters
P	Input	UINT	UINT	I, Q, M, D, L, P or constant	Reference to the first character to be converted (first character = 1, the value "0" or a value > length of the string is invalid)
OUT	Output	USINT, SINT, UINT, INT, UDINT, DINT, REAL, LREAL	USINT, SINT, UINT, INT, UDINT, DINT, ULINT, LINT, REAL, LREAL	I, Q, M, D, L, P	Result of the conversion

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter FORMAT

You use the FORMAT parameter to specify how the characters of a character string are to be interpreted. Exponential values can also be converted and represented with the "STRG_VAL" instruction.

The following table shows the possible values of the FORMAT parameter and their meaning:

Value (W#16#....)	Notation	Decimal representation
0000	Decimal fraction	"."
0001		","
0002	Exponential	"."
0003		","
0004 to FFFF	Invalid values	

Parameter P

The conversion starts at the character whose position you specified in the P parameter. If, for example, the value "1" is specified in the P parameter, the conversion starts at the first character of the specified character string.

Example

The following table shows examples of the conversion of a character string to a numeric value:

IN (STRING)	FORMAT (W#16#....)	OUT (data type)	OUT (value)	ENO status
'123'	0000	INT/DINT	123	1
'-00456'	0000	INT/DINT	-456	1
'123.45'	0000	INT/DINT	123	1
'+2345'	0000	INT/DINT	2345	1
'00123AB'	0000	INT/DINT	123	1
'123'	0000	REAL	123.0	1
'-00456'	0001	REAL	-456.0	1
'+00456'	0001	REAL	456.0	1
'123.45'	0000	REAL	123.45	1
'123.45'	0001	REAL	12345.0	1
'123,45'	0000	REAL	12345.0	1
'123,45'	0001	REAL	123.45	1
'.00123AB'	0001	REAL	123.0	1
'1.23e-4'	0000	REAL	1.23	1
'1.23E-4'	0000	REAL	1.23	1
'1.23E-4'	0002	REAL	1.23E-4	1
'12,345.67'	0000	REAL	12345.67	1
'12,345.67'	0001	REAL	12.345	1
'3.4e39'	0002	REAL	W#16#7F800000	1
'-3.4e39'	0002	REAL	W#16#FF800000	1
'1.1754943e-38'	0002	REAL	0.0	1
'12345'	-/-	SINT	0	0
'A123'	-/-	-/-	0	0

IN (STRING)	FORMAT (W#16#....)	OUT (data type)	OUT (value)	ENO status
''	-/-	-/-	0	0
'++123'	-/-	-/-	0	0
'+-123'	-/-	-/-	0	0

Example

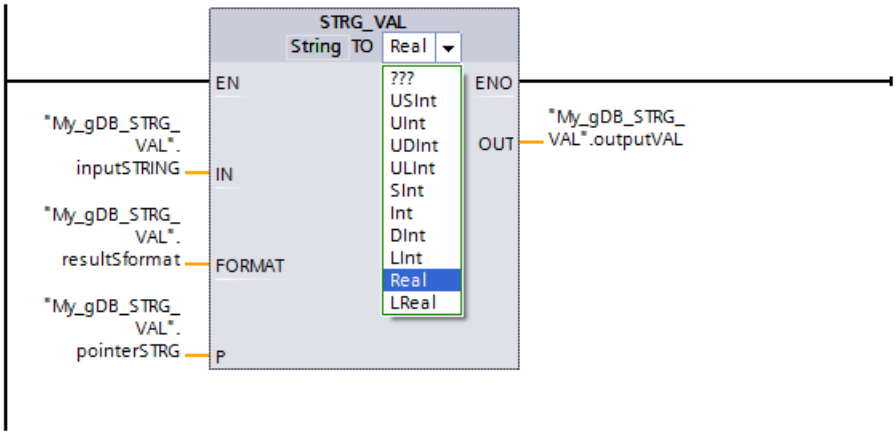
In the following example, you convert a number string with the STRING data type to a floating point number with the REAL data type. The result will have a length of 32 bits and may have a sign due to the REAL data type.

Create four tags in a global data block for storing the data.

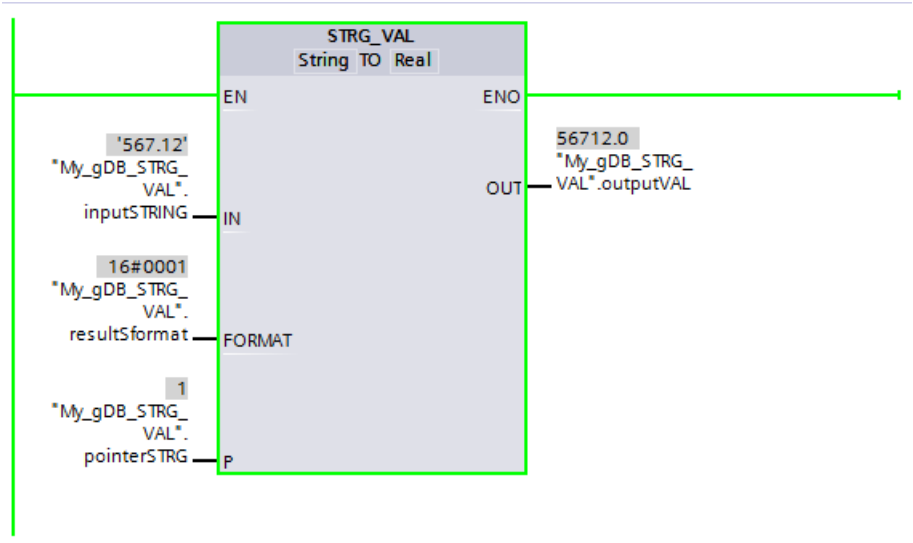
My_gDB_STRG_VAL

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'567.12'
3	resultSformat	Word	16#0001
4	pointerSTRG	UInt	1
5	outputVAL	Real	0.0

Interconnect the parameters of the instruction as follows. Using the left selection option, select the data type for the character string. Using the right selection option, select the data type for the floating-point number.



The number string is converted starting from the first character according to the value "1" of parameter P ("pointerSTRG"). The period in the number string is interpreted as a thousands delimiter due to the value "0001" of parameter FORMAT ("resultSformat"). (The comma is the decimal separator for the value "0001".) The value to be converted ("inputSTRING") is output at output parameter OUT ("outputVAL") as a floating point number.



VAL_STRG: Convert numerical value to character string

Description

The "VAL_STRG" instruction is used to convert a numerical value into a character string.

- You specify the value to be converted in the IN input parameter. You decide the format of the numeric value by selecting a data type.
- You query the result of the conversion in the OUT output parameter.

Permitted characters for the conversion are the digits "0" to "9", the decimal point, the decimal comma, notations "E" and "e", and the plus and minus characters. The conversion can be interrupted by invalid characters.

The "VAL_STRG" instruction is not supported by the SCL programming language.

Parameters

The following table shows the parameters of the "VAL_STRG" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	USINT, SINT, UINT, INT, UDINT, DINT, REAL, LREAL	USINT, SINT, UINT, INT, UDINT, DINT, ULINT, LINT, REAL, LREAL	I, Q, M, D, L, P or constant	Value to be converted
SIZE	Input	USINT	USINT	I, Q, M, D, L, P or constant	Number of character positions
PREC	Input	USINT	USINT	I, Q, M, D, L, P or constant	Number of decimal places

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
FORMAT	Input	WORD	WORD	I, Q, M, D, L, P or constant	Output format of the characters
P	InOut	UINT	UINT	I, Q, M, D, L, P or constant	Character starting at which the result is written.
OUT	Output	STRING, WSTRING	STRING, WSTRING	D, L	Result of the conversion

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter P

With the P parameter, you specify the character in the string starting at which the result is written. If, for example, the value "2" is specified in the P parameter, the converted value is saved starting at the second character of the string.

Parameters SIZE and P

Use the SIZE parameter to specify how many characters of the character string will be written. This is counted starting from the character specified in the P parameter. If the output value is shorter than the specified length, the result is written to the character string right-justified. The empty character positions are filled with blanks.

Parameter FORMAT

Use the FORMAT parameter to specify how the numerical value is interpreted during conversion and written to the character string. You can specify only tags of the FORMAT data type in the USINT parameter.

The following table shows the possible values of the FORMAT parameter and their meaning:

Value (W#16#....)	Notation	Sign	Decimal representation
0000	Decimal fraction	"-"	"."
0001			","
0002	Exponential		"."
0003			","
0004	Decimal fraction	"+" and "-"	"."
0005			","
0006	Exponential		"."
0007			","
0008 to FFFF	Invalid values		

Parameter PREC

Use the PREC parameter to define the number of decimal places when converting floating-point numbers. A maximum precision of seven numbers is supported for numerical values of the REAL data type. If the value to be converted is an integer, you use the PREC parameter to specify the position where the decimal point will be placed.

Example

The following table shows examples of the conversion of numeric values to a character string.

IN (value)	IN (data type)	P	SIZE	FORMAT (W#16#....)	PREC	OUT (STRING)	ENO status
123	UINT	16	10	0000	0	xxxxxxx123 C	1
0	UINT	16	10	0000	2	xxxxxx0.00 C	1
12345678	UDINT	16	10	0000	3	x12345.678 C	1
12345678	UDINT	16	10	0001	3	x12345.678 C	1
123	INT	16	10	0004	0	xxxxxx+123 C	1
-123	INT	16	10	0004	0	xxxxxx-123 C	1
-0.00123	REAL	16	10	0004	4	xxx-0.0012 C	1
-0.00123	REAL	16	10	0006	4	-1.2300E-3 C	1
-Inf [1]	REAL	16	10	-/-	4	xxxxxx-INF C	0
+Inf [2]	REAL	16	10	-/-	4	xxxxxx+INF C	0
NaN [3]	REAL	16	10	-/-	4	xxxxxxxNaN C	0
12345678	UDINT	16	6	-/-	3	xxxxxxxxxx C	0

"x" represents blanks
[1]-Inf: Floating-point number representing a negative infinite value.
[2]+Inf: Floating-point number representing a positive infinite value.
[3]NaN: Value returned as the result of invalid math operations.

Example

In the following example, you convert a floating-point number with the REAL data type to a character string with the STRING data type.

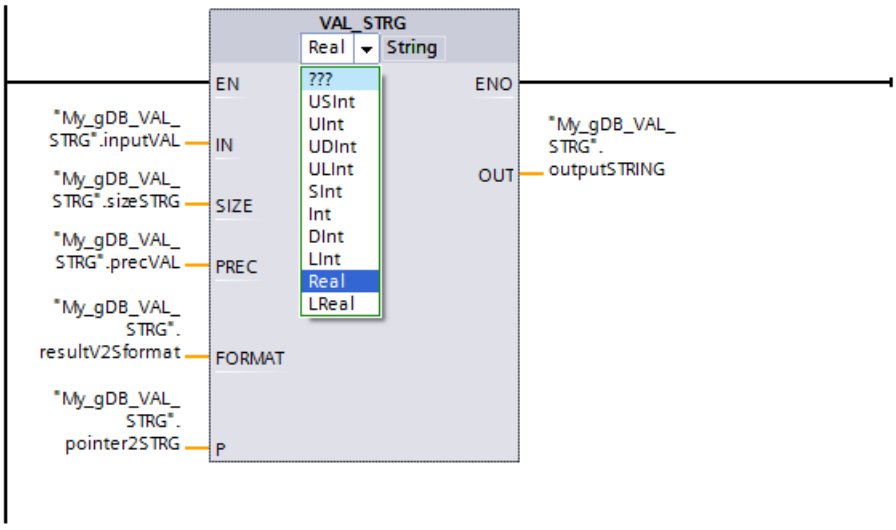
Create six tags in a global data block for storing the data.

My_gDB_VAL_STRG

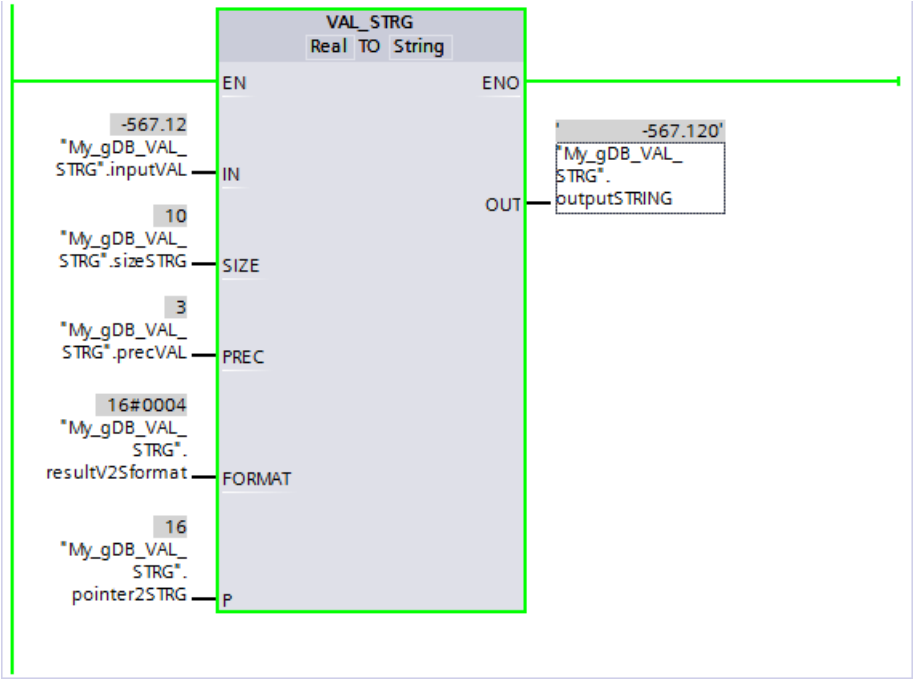
	Name	Data type	Start value
1	Static		
2	inputVAL	Real	-567.12
3	pointer2STRG	UInt	16
4	sizeSTRG	USInt	10
5	precVAL	USInt	3
6	outputSTRING	String	''
7	resultV2Sformat	Word	16#0004

Interconnect the parameters of the instruction as follows. Select the data types:

- The left selection option is for the value to be converted.
- The right selection option is for the character string to be generated.



The character string is written starting from the 16th character according to the value "16" of parameter P" ("pointer2STRG"). Starting from this point, the character string is 10 characters long, according to the value "10" of parameter SIZE ("sizeSTRG"). The point of the value to be converted ("inputVAL") is interpreted as a decimal separator due to the value "0004" of parameter FORMAT ("resultV2Sformat"). Three decimal places are written to the string according to the value "3" of parameter PREC ("precVAL"). The sign of the value to be converted is stored as a character in the string and prefixes the numbers. The remaining 10 characters of the string are written as spaces in front of the sign. The character string is output at output parameter OUT ("outputSTRING").

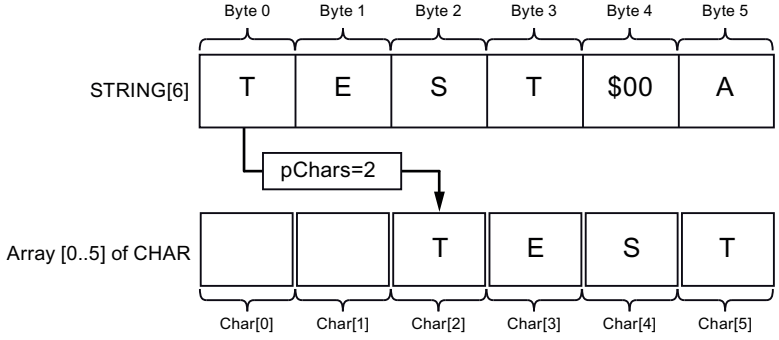


Strg_TO_Chars: Convert character string to Array of CHAR

Description

You use the "Strg_TO_Chars" instruction to copy a character string STRING to an Array of CHAR or an Array of BYTE or a character string WSTRING to an Array of WCHAR. Only ASCII characters are valid for the copy process.

- Specify the character string at the input parameter STRG.
- The characters are written to a data type Array of CHAR / BYTE / WCHAR at the parameter CHARS.
  - The number of characters in the destination field should be at least the same as the number copied from the source string.
  - If the destination field contains fewer characters than the source string, the characters are written up to the maximum length of the destination field.
  - If the character string contains a "$00" or W#16#0000 character, the copy operation is only carried out up to the corresponding position (see graphic).
  - The number of copied characters is output at parameter CNT.
- Use the PCHARS parameter to specify the position starting from which writing should begin in the destination field.
  - Example: If writing should take place starting from the third position, use the value "2" at parameter PCHARS:



  - The default for PCHARS is "0". When PCHAR = 0, the low index limit of the array is used (e.g. CHAR[0] for an Array [0..5] of CHAR). This also applies if the low limit of the array is negative (e.g. CHAR[-5] for an Array [-5..5] of CHAR).

Note

Use of the instruction with S7-1200 V2.0

S7-1200 up to Version 2.0 only supports Array [0 .. n] of CHAR / BYTE. Negative index limits (e.g. Array [-3..2] of CHAR) are not permissible. This restriction is not checked by the software.

Parameters

The following table shows the parameters of the instruction "Strg_TO_Chars":

Parameter	Declaration	Data type	Memory area	Description
STRG	Input	STRING, WSTRING	D, L or constant	Source of the copy operation
PCHARS	Input	DINT	I, Q, M, D, L, P or constant	Position in the structure Array of (W)CHAR / BYTE starting from which the characters of the character string are written.
CHARS	InOut	VARIANT	D, L	Destination of the copy operation The characters can be copied to a structure of the Array of (W)CHAR or Array of BYTE data type.
CNT	Output	UINT	I, Q, M, D, L, P	Number of copied characters.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

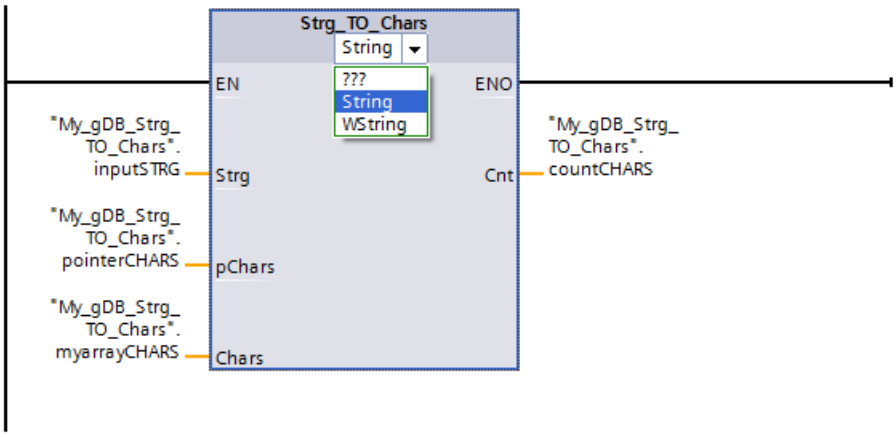
In the following example, you copy characters of a character string with the STRING data type to a structure of the Array of CHAR data type.

Create four tags in a global data block for storing the data.

My_gDB_Strg_TO_Chars

	Name	Data type	Start value
1	Static		
2	inputSTRG	String[10]	'cf7a7a#'
3	pointerCHARS	DInt	2
4	myarrayCHARS	Array[0..9] of Char	
5	myarrayCHARS[0]	Char	' '
6	myarrayCHARS[1]	Char	' '
7	myarrayCHARS[2]	Char	' '
8	myarrayCHARS[3]	Char	' '
9	myarrayCHARS[4]	Char	' '
10	myarrayCHARS[5]	Char	' '
11	myarrayCHARS[6]	Char	' '
12	myarrayCHARS[7]	Char	' '
13	myarrayCHARS[8]	Char	' '
14	myarrayCHARS[9]	Char	' '
15	countCHARS	UInt	0

Interconnect the parameters of the instruction as follows and select the data type of the character string.



A structure made up of individual characters is created due to the Array of CHAR data type. The CHARS structure ("myarrayCHARS") is ten characters long (Array ... [0..9]). According to the value "2" of parameter PCHARS ("pointerCHARS"), writing begins starting from the third character in the structure ("0" and "1" are empty, "2" contains the first character of the character string ("inputSTRG")). Once the characters of the string ("inputSTRG") are written to the structure ("myarrayCHARS"), the last character of the structure to be created is written empty. The number of characters of the character string that are moved is output at output parameter CNT ("countCHARS").

My_gDB_Strg_TO_Chars

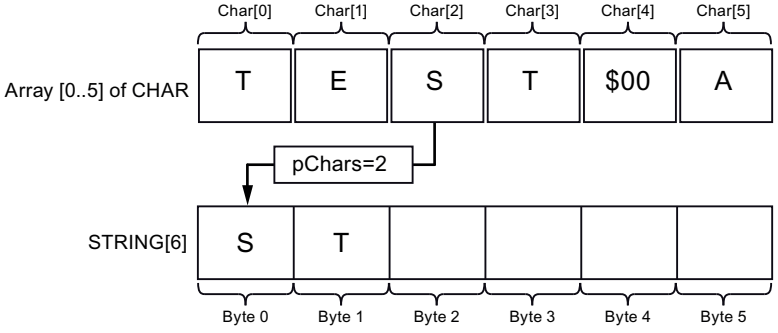
	Name	Data type	Start value	Monitor value
1	Static			
2	inputSTRG	String[10]	'cf7a7a#'	'cf7a7a#'
3	pointerCHARS	DInt	2	2
4	myarrayCHARS	Array[0..9] of Char		
5	myarrayCHARS[0]	Char	' '	' '
6	myarrayCHARS[1]	Char	' '	' '
7	myarrayCHARS[2]	Char	' '	'c'
8	myarrayCHARS[3]	Char	' '	'f'
9	myarrayCHARS[4]	Char	' '	'7'
10	myarrayCHARS[5]	Char	' '	'a'
11	myarrayCHARS[6]	Char	' '	'7'
12	myarrayCHARS[7]	Char	' '	'a'
13	myarrayCHARS[8]	Char	' '	'#'
14	myarrayCHARS[9]	Char	' '	' '
15	countCHARS	UInt	0	7

Chars_TO_Strg: Convert Array of CHAR to character string

Description

You can use the "Chars_TO_Strg" instruction to copy characters from an Array of CHAR or Array of BYTE to a STRING or from an ARRAY of WCHAR to a WSTRING.. Only ASCII characters are valid for copying.

- Specify the characters of the Array of (W)CHAR / BYTE to be copied to a character string at the input parameter CHARS.
- The characters are written at the parameter STRG to a (W)STRING data type.
  - The number of characters in the string should be at least the same number as were copied from the source field.
  - If the character string is shorter than the number of characters in the source field, the characters are written up to the maximum length of the string.
  - If the Array of (W)CHAR / BYTE contains a "$00" character or if the Array of (W)CHAR contains a W#16#0000 character, the copy operation is only carried out up to the corresponding position (see graphic).
- Use the PCHARS parameter to specify the position of the source field starting from which the characters are to be copied. PCHARS = 0 is the default value and always specifies the low index limit of the array, even if this is negative.
  - Example: if copying is to start with the third character of the source field, use the value "2" at parameter PCHARS:



  - If an index is specified at parameter PCHARS and the index is not contained in the copy source (e.g. "7" at Array [0..5] of CHAR) then the instruction is not executed.

Note

Use of the instruction with S7-1200 V2.0

S7-1200 up to Version 2.0 only supports Array [0 .. n] of CHAR / BYTE. Negative index limits (e.g. Array [-3..2] of CHAR) are not permissible. This restriction is not checked by the software.

Parameters

The following table shows the parameters of the instruction "Chars_TO_Strg":

Parameter	Declaration	Data type	Memory area	Description
CHARS	Input	VARIANT	D, L	Source of the copy operation Array of (W)CHAR / BYTE from which the characters are copied.
PCHARS	Input	DINT	I, Q, M, D, L, P or constant	Position in the Array of (W)CHAR / Array of BYTE from which the characters are copied.
CNT	Input	UINT	I, Q, M, D, L, P	Number of characters to be copied. Use "0" to copy all characters.
STRG	Output	STRING, WSTRING	D, L	Destination of the copy operation Character string with the (W)STRING data type. Observe the maximum length of the data types: • STRING: 254 characters • WSTRING: 254 characters (default) / 16382 characters (maximum) When using WSTRING, note that you must define a length > 254 characters explicitly with square brackets (e.g., WSTRING[16382]).

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

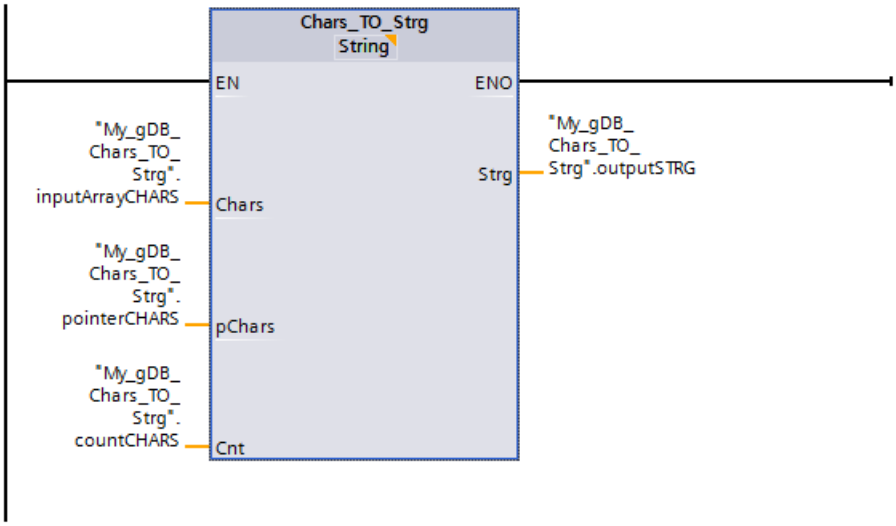
In the following example, you copy characters of a structure of the Array of CHAR data type to a character string with the STRING data type.

Create four tags in a global data block for storing the data.

My_gDB_Chars_TO_Strg

	Name	Data type	Start value
1	Static		
2	inputArrayCHARS	Array[0..9] of Char	
3	inputArrayCHARS[0]	Char	'M'
4	inputArrayCHARS[1]	Char	'y'
5	inputArrayCHARS[2]	Char	'S'
6	inputArrayCHARS[3]	Char	'7'
7	inputArrayCHARS[4]	Char	'P'
8	inputArrayCHARS[5]	Char	'L'
9	inputArrayCHARS[6]	Char	'C'
10	inputArrayCHARS[7]	Char	' '
11	inputArrayCHARS[8]	Char	' '
12	inputArrayCHARS[9]	Char	' '
13	pointerCHARS	DInt	2
14	outputSTRG	String	''
15	countCHARS	UInt	0

Interconnect the parameters of the instruction as follows and select the data type of the character string.



The CHARS structure ("inputArrayCHARS") is ten characters long (Array ... [0..9]). According to the value "2" of parameter PCHARS ("pointerCHARS"), characters are copied starting from the third position of the structure to the character string ("outputSTRG"). Starting from position "2", all characters of the structure ("inputArrayCHARS") are copied to the character string ("outputSTRG"), since parameter CNT ("countCHARS") has the value "0".

My_gDB_Chars_TO_Strg

	Name	Data type	Start value	Monitor value
1	Static			
2	inputArrayCHARS	Array[0..9] of Char		
3	inputArrayCHARS[0]	Char	'M'	'M'
4	inputArrayCHARS[1]	Char	'y'	'y'
5	inputArrayCHARS[2]	Char	'S'	'S'
6	inputArrayCHARS[3]	Char	'7'	'7'
7	inputArrayCHARS[4]	Char	'P'	'P'
8	inputArrayCHARS[5]	Char	'L'	'L'
9	inputArrayCHARS[6]	Char	'C'	'C'
10	inputArrayCHARS[7]	Char	' '	' '
11	inputArrayCHARS[8]	Char	' '	' '
12	inputArrayCHARS[9]	Char	' '	' '
13	pointerCHARS	DInt	2	2
14	outputSTRG	String	''	'S7PLC '
15	countCHARS	UInt	0	0

MAX_LEN: Determine the length of a character string

Description

A tag of the (W)STRING data type contains two lengths: the maximum length and the current length (this is the number of currently valid characters).

- The maximum length of the character string is specified for each tag in the STRING keyword in square brackets. The number of bytes occupied by a string is 2 greater than the maximum length.
- The maximum length of the character string is specified for each tag in the WSTRING keyword in square brackets. The number of words occupied by a character string is 2 greater than the maximum length.
- The current length represents the number of the character places actually used. The current length must be less than or equal to the maximum length.

You use the "MAX_LEN" instruction to query the maximum length of the character string specified at the IN input parameter and output this information as a numerical value at the OUT output parameter.

If errors occur during processing of the instruction, then an empty string will be output.

Note

Reading out the current length

You can also use the LEN (Page 3039) instruction to read out the current length of a string.

Parameters

The following table shows the parameters of the "MAX_LEN" instruction:

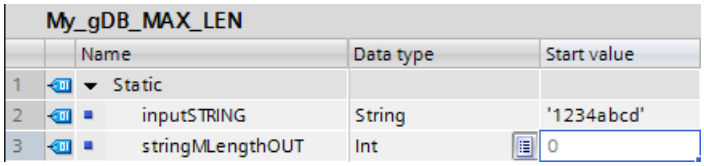
Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
OUT	Return	INT	I, Q, M, D, L, P	Maximum number of characters

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

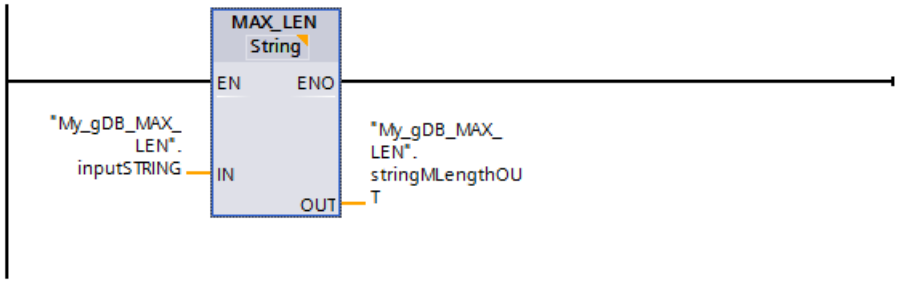
Example

In the following example, you determine the maximum length of a character string with the STRING data type.

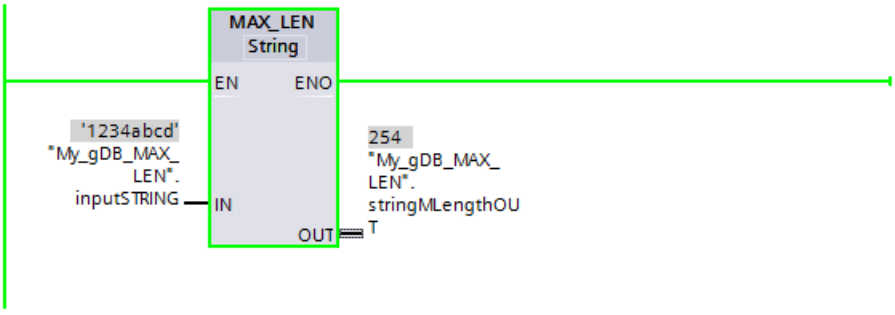
Create two tags in a global data block for storing the data.



Interconnect the parameters of the instruction as follows.



The maximum length of the specified character string ("inputSTRING") is determined and returned at output parameter OUT ("stringMLengthOUT") as a numerical value.



JOIN: Join multiple strings

Description

The "JOIN" instruction connects multiple strings into an array.

To convert multiple strings into a single one, the instruction provides the following functions:

- **Format selection**
  You can use the first bit at the Mode parameter to specify whether the sequence of the source strings for an array should be in the format CSV or FSR.
  In the following example, two source strings are specified by the two columns of the table. The maximum number of characters amounts to 4 characters for the first source string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth.

1963	1974
Miller	Jackson
John	Peter
Roadname	VeryLongRoadna

  – With CSV (Comma Separated Values), the contents of the source strings are written consecutively into the destination array and separated with a delimiter. (See examples below.)

  – With FSR (Fixed Size Records), a certain number of characters is defined in the destination array for each source string. If the characters of a source string do not need the space reserved for them in the destination array, the corresponding array elements are filled with delimiters. However, if the number of characters in a source string is greater than the space reserved for them, the corresponding array elements are filled from the front, and the excess characters of the source string are truncated (see example below).

- **Selection of the delimiter for the source strings**
  You can use the RecSeparator parameter to select the delimiter to be used for each string. You should select a character based on the content of the input strings at the SrcStruct parameter. If the input strings contain a comma within a string, for example, the comma should not used as a delimiter. The data type that you use for the delimiter must correspond to the destination array at the DstArray parameter in order for the delimiter to be included in what is written in the array.

- **Selection of the delimiter for the end of all source strings**
  You can use the third bit of the Mode parameter to specify whether an additional character should be written as a delimiter at the end of the copied characters in the destination array (DstArray parameter). Specify the character to be used as a delimiter at the EndSeparator parameter. Make sure to use a character other than the one set for the RecSeparator parameter (delimiter for individual strings). If the two delimiters are indistinguishable, you will obtain unexpected results if you reverse the conversion using the "SPLIT" instruction.

- **Selection of the source strings**
  You specify the source strings at the SrcStruct parameter. For the data type, you can use Array of STRING or Array of WSTRING or structures which only contain the STRING or WSTRING data type. This also applies to user data types or nested structures. You can use these as long as they exclusively contain the data type STRING or WSTRING respectively.

- **Specification of the number of joined strings**
  If you use an Array of STRING or Array of WSTRING at the SrcStruct parameter (source string) (no nested structures), you can use the Count parameter to specify the number of source strings that have been joined to form a single string. If you use a data type other than Array of (W)STRING at the SrcStruct parameter, the Count parameter is ignored. This makes it possible to connect only a part of a large array.

- **Selection of a destination area for writing the array**
  You use the Array of (W)CHAR data type at the DstArray parameter. It is not possible to use the STRING or WSTRING data types here, because the length would be limited to 254 characters or 256 bytes for STRING.
- **Index of the position in the array (DestArray target parameter)**
  The conversion starts at this position and the instruction reads the Position parameter to determine the position at which the conversion is completed. This enables follow-up calls of the instruction for filling the array.

Parameters

The following table shows the parameters of the instruction "JOIN":

Parameter	Declaration	Data type	Memory area	Description
Mode	Input	DWORD	I, Q, M, D, L or constant	Specifies how the merger to string is performed (see Mode" parameter).
RecSeparator	Input	VARIANT	I, Q, M, D, L	Delimiter for the source strings • With CSV: Character that is used as a delimiter for the individual strings. • With FSR: Character that is used as a fill character for the individual strings.
EndSeparator	Input	VARIANT	I, Q, M, D, L or constant	Delimiter for the end of the conversion Delimiter that is written at the end of the characters if bit 3 = 1 has been set for the Mode parameter.
SrcStruct	Input	VARIANT	I, Q, M, D, L	Pointer to the source strings.
Count	Input	UDINT	I, Q, M, D, L	Number of strings that were joined. The Count parameter can only be used when an Array of (W)STRING has been used at the SrcStruct parameter.
DestArray	InOut	VARIANT	I, Q, M, D, L	Area in which the characters are written after the conversion. Use the Array [0 .. x] of CHAR/WCHAR data type at the DestArray parameter. You can set the length (x) of the array based on the length of the source strings at the SrcStruct parameter.
Position	InOut	UDINT	I, Q, M, D, L	Index the position in the total string
Ret_Val	Return	INT	I, Q, M, D, L	Status of instruction (see table "RET_VAL parameter").

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter Mode

Bit	Bit value "0"	Bit value "1"	Description
0	CSV format (Comma Separated Values)	FSR format (Fixed Size Record)	Selection of the format: • With CSV, the source strings are separated by a delimiter in the destination array. • With FSR, the source strings are written into the destination array with the fill characters defined at the RecSeparator parameter.
1	-	-	Irrelevant for the "JOIN" instruction.
2	-	-	Reserved (irrelevant for bit value)
3	Write no additional delimiters.	Write character defined by the EndSeparator parameter at the end of the read characters.	Select whether an additional character is to be written as a delimiter at the end of the characters in the array (DestArray parameter).
4	-	-	Irrelevant for the "JOIN" instruction.

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error.
0001	Additional characters are ignored.
8190	Selection at the Mode parameter is not supported.
8x20	Source strings invalid.
8x53	VARIANT points to data structure that is too short.
8x54	Invalid data type
8082	The value at parameter Count is greater than the number of source character strings in SrcStruct.
8x84	Additional characters found
8xB4	Different data types at parameters SrcStruct (source) and DestArray (destination) or at separators (parameters RecSeparator and EndSeparator).
80B5	Buffer overflow with the instruction. The characters are not completely output at parameter DestArray, or the value at parameter Position is outside of DestArray.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)

* Note the following for the error codes:

- The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".
- The "x" in the second position of the listed error codes stands for the parameter that caused the error.
  Example: Error code 8352 hex = error occurred at 3rd parameter (EndSeparator), see Parameter table.
- If the error cannot be clearly assigned to a specific parameter, "0" is output.
  Example: The CHAR data type is used for the delimiter (RecSeparator parameter). WCHAR is used as data type for the array at the DestArray parameter. Error code 80B4 is output in this case.

Two examples for the JOIN instruction when the destination ARRAY should have the format CSV

- First example:
  There are the following source strings:
  - `1963`
  - `Miller`
  - `John`
  - `Roadname`

  If you select "," as the delimiter, calling JOIN provides the following destination array:
  `1963,Miller,John,Roadname`
- Second example:
  There are the following source strings:
  - `1974`
  - `Jackson`
  - `Peter`
  - `VeryLongRoadname`

  If you select "," as the delimiter, calling JOIN provides the following destination array:
  `1974,Jackson,Peter,VeryLongRoadname`

Two examples for the JOIN instruction when the destination ARRAY should have the format FSR

- First example:
  There are the following source strings:
  - `1963`
  - `Miller`
  - `John`
  - `Roadname`

  The number of reserved characters in the destination array amounts to 4 characters for the first source string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth.
  If you select "," as the fill characters, calling JOIN provides the following destination array:
  `1963Miller,,,,,,,John,,,,,,Roadname,,,,,,`
- Second example:
  There are the following source strings:
  - `1974`
  - `Jackson`
  - `Peter`
  - `VeryLongRoadname`

  The number of reserved characters in the destination array amounts to 4 characters for the first source string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth.
  If you select "," as the fill characters, calling JOIN provides the following destination array:
  `1974Jackson,,,,,,Peter,,,,,VeryLongRoadna`

SPLIT: Splitting an array of characters into multiple strings

Description

The "SPLIT" instruction converts an array (Array of CHAR / WCHAR) into several separate strings (Array of STRING / WSTRING or structure).

To convert the array into several strings, specify the following information:

- **Selection of array to be read**
  You specify the array to be read at the SrcArray parameter. Make sure that the data types used for the input and output parameters match any other parameters used. If you use an array with the CHAR data type at the SrcArray parameter, for example, you must use the CHAR data type for the delimiter (Rec/EndSeparator) and the structure must only contain strings with the STRING data type at the DestStruct parameter.
- **Format selection for the source array**
  You use the first bit of the Mode parameter to specify whether the array to be read has the CSV or FSR format.
  - With CSV (Comma Separated Values), characters in the source array that belong together are separated by a delimiter from the next characters that belong together.
    Two examples of a source array:
    `1963,Miller,John,CitynameA,Roadname`
    `1974,Jackson,Peter,CitynameB,VeryLongRoadname`
  - With FSR (Fixed Size Records), a certain number of characters is defined for each item of logical information of the source array. Each item of information should fit into the space defined for it. When information does not need the space defined for it, it is filled with delimiters.
    Two examples of source arrays, whereby the length of the information is 4 characters for the first information item (year of birth), 13 characters for the second (last name), 10 characters for the third (first name), 9 characters for the fourth (city) and 16 characters for the fifth (street):
    `1963Miller,,,,,,,John,,,,,,CitynameARoadname,,,,,,,,`
    `1974Jackson,,,,,,Peter,,,,,CitynameBVeryLongRoadname`
- **Delimiter used for the array to be read**
  - If the array to be read has CSV format, specify which delimiter was used at the RecSeparator parameter.
  - If the array to be read has FSR format, specify which fill character was used at the RecSeparator parameter.
- **Selection of the delimiter for the end of the total string**
  Specify the delimiter after which the reading of the array should stop at the EndSeparator parameter. The "SPLIT" instruction stops at this position and outputs the string that has been found. It should be noted, that the EndSeparator delimiter is evaluated with higher priority than the RecSeparator delimiter. If the delimiter at the EndSeparator parameter is used in a string to be read (between two RecSeparator delimiters), all content after the EndSeparator delimiter is ignored.
- **Specification of the position starting at which the array is read**
  The conversion starts at this position in the array and the instruction reads the Position parameter to determine the position at which the conversion is completed. This enables follow-up calls of the instruction for filling the various strings at the DestStruct parameter.
- **Outputting the number of read strings**
  If you use an Array of STRING at the DestStruct parameter, you can use the Count parameter to output the number of read strings. Only strings with content are counted. If you use a data type other than Array of STRING at the DestStruct parameter, "0" is output at the Count parameter.

Parameters

The following table shows the parameters of the "SPLIT" instruction:

Parameter	Declaration	Data type	Memory area	Description
Mode	Input	DWord	I, Q, M, D, L or constant	Specifies how the split into multiple strings is performed (see "Mode" parameter).
RecSeparator	Input	Variant	I, Q, M, D, L	Delimiter or fill character • With CSV: Character that was used in the array to be read to identify the individual strings. • With FSR: Character that has been used as a fill character in the array to be read.
EndSeparator	Input	Variant	I, Q, M, D, L	Delimiter used to define the end of the total string in the array to be read.
SrcArray	Input	Variant	I, Q, M, D, L	Pointer to the array to be read (Array of CHAR/WCHAR)
DestStruct	InOut	Variant	I, Q, M, D, L	Structure which includes the converted strings (Array of STRING / WSTRING).
Position	InOut	UDInt	I, Q, M, D, L	Position from which the array is to be read at the SrcArray parameter.
Ret_Val	Return	Int	I, Q, M, D, L	Result of instruction execution / error code (see table "Ret_Val parameter")
Count	Output	UDInt	I, Q, M, D, L	Number of strings that were found.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter "Mode"

Bit	Bit value "0"	Bit value "1"	Description
0	CSV format (Comma Separated Values)	FSR format (Fixed Size Record)	Basic mode: Selection of CSV or FSR
1	• With CSV: Additional characters cause an error. • With FSR: Additional fill characters remain in string.	• With CSV: Additional characters are ignored. • With FSR: Additional fill characters are removed.	Use bit 1 to specify how you want to deal with additional characters: • With CSV: – If the bit is set, additional characters that do not fit in the destination string are ignored. Example: The instruction writes to a string with a length of 16 characters (STRING[16] data type). No delimiter is contained in the source after the first 16 characters. If bit 1 is set, the extra characters are ignored and the instruction continues to read the array. – If the bit is not set, the instruction stops in such a case and generates an error message at the Ret_Val parameter. • With FSR: – If the bit is set, the fill characters located to the right of characters containing information are not written to the destination character string when the source array is transferred to the destination character strings (see example). – If the bit is not set, the fill characters located to the right of characters containing information are written to the destination character string when the source array is transferred to the destination character strings (see example).
2	-	-	Reserved for use in future versions.
3	Separator is retained.	Separator is removed.	Specification of whether the delimiter is removed at the end of the total string.
4	Leave unwritten strings (STRING) at length.	Set unwritten strings (STRING) to length "0".	Specification of whether unused strings (STRING) at the DestStruct parameter are to be set to the length "0".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error.
0001	Additional characters are ignored.
8190	Selection at the Mode parameter is not supported.
8x20	Source strings invalid.
8x53	VARIANT points to data structure that is too short.
8x54	Invalid data type
8x84	Additional characters found.
8xB4	Different data types at parameters SrcArray (source) and DestStruct (destination) or at separators (parameters RecSeparator and EndSeparator).
80B5	Buffer overflow with the instruction. The character strings are not completely output at parameter DestStruct.

Error code* (W#16#...)	Explanation
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)

* Note the following for the error codes:

- The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".
- The "x" in the second position of the listed error codes stands for the parameter that caused the error.
  Example: Error code 8352 hex = error occurred at 3rd parameter (EndSeparator), see Parameter table.
- If the error cannot be clearly assigned to a specific parameter, "0" is output.
  Example: The CHAR data type is used for the delimiter (RecSeparator parameter). WCHAR is used as data type for the array at the DestArray parameter. Error code 80B4 is output in this case.

Two examples for the SPLIT instruction when the ARRAY to be read has the CSV format

In the first step, the source array is split into strings as specified by the delimiter (RecSeparator parameter, e.g. ",").

In the second step, the resulting strings after the split are stored in destination strings. You can use the DestStruct parameter to specify their length.

If the strings produced by the split have more characters than can fit in the destination string, the behavior of SPLIT after this point depends on the Mode parameter. With Mode=2#00, this results in an error: The associated error code is returned and no other destination strings are

filled. With Mode=2#10, the surplus characters are ignored and the next destination string is filled with the characters after the next delimiter.

- First example:
  In this example, there are these two source arrays:
  - `1963,Miller,John,Roadname`
  - `1974,Jackson,Peter,VeryLongRoadname`

  The length of the destination string amounts to 4 characters for the first destination string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth. The first call for SPLIT produces the following destination strings with Mode=2#10:
  - `1963`
  - `Miller`
  - `John`
  - `Roadname`

  The second call for SPLIT produces the following destination strings with Mode=2#10:
  - `1974`
  - `Jackson`
  - `Peter`
  - `VeryLongRoadna`
- Second example:
  In this example, the logical information in the source array is not in the order expected. There are these two source arrays:
  - `1963,Miller,Roadname,John`
  - `1974,Jackson,VeryLongRoadname,Peter`

  As in the first example, the length of the destination string amounts to 4 characters for the first destination string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth.
  The first call for SPLIT produces the following destination strings with Mode=2#10:
  - `1963`
  - `Miller`
  - `Roadname`
  - `John`

  The second call for SPLIT produces the following destination strings with Mode=2#10:
  - `1974`
  - `Jackson`
  - `VeryLongRo`
  - `Peter`

Two examples for the SPLIT instruction when the ARRAY to be read has format FSR

The source array is split into strings as you specified in the DestStruct parameter for the length of the destination string.

With regard to the fill characters (RecSeparator parameter e.g. ","), the behavior of SPLIT depends on the Mode parameter: With Mode=2#01, the fill characters are entered in the destination strings; with Mode=2#11, they are not entered.

- First example:
  In this example, there are these two source arrays:
  - `1963Miller,,,,,,,John,,,,,,Roadname,,,,,,`
  - `1974Jackson,,,,,,Peter,,,,,VeryLongRoadname`

  The length of the destination string amounts to 4 characters for the first destination string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth. The first call for SPLIT produces the following destination strings with Mode=2#01:
  - `1963`
  - `Miller,,,,,,,`
  - `John,,,,,,`
  - `Roadname,,,,,,`

  The second call for SPLIT produces the following destination strings with Mode=2#01:
  - `1974`
  - `Jackson,,,,,,`
  - `Peter,,,,,`
  - `VeryLongRoadna`
- Second example:
  In this example, the logical information in the source array is not in the order expected. There are these two source arrays:
  - `1963Miller,,,,,,,Roadname,,,,,,John,,,,,,`
  - `1974Jackson,,,,,,VeryLongRoadnamePeter,,,,,`

  As in the first example, the length of the destination string amounts to 4 characters for the first destination string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth.
  The first call for SPLIT produces the following destination strings with Mode=2#11:
  - `1963`
  - `Miller`
  - `Roadname`
  - `,,,,John`

  Explanation: The SPLIT instruction lays templates over the source array, so to speak. The length of these templates is determined by the length of the destination strings. During transfer to the destination strings, only those commas to the right of characters with information content are interpreted as fill characters. The fill characters are not entered with Mode=2#11. In the first call, the commas are to the right of the "John" fill characters. The commas to the left of "John", however, are not fill characters, because the relevant "John" characters follow before the end of the corresponding destination string is reached. Therefore, the commas to the left of "John" appear in the destination string.
  The second call for SPLIT produces the following destination strings with Mode=2#11:

- `1974`
- `Jackson`
- `VeryLongRo`
- `adnamePeter`

See also

JOIN: Join multiple strings (Page 3022)

ATH: Convert ASCII string to hexadecimal number

Description

You use the instruction "ATH" to convert the ASCII character string specified at the IN input parameter into a hexadecimal number. The result of the conversion is output to the OUT output parameter.

- You can reference the following data types using the pointer in the IN parameter (ASCII): STRING, Array of CHAR, Array of BYTE, WSTRING, Array of WCHAR.
- You can reference the following data types using the pointer in the OUT parameter (hexadecimal): Bit strings, integers, STRING, Array of CHAR, Array of BYTE, WSTRING, Array of WCHAR.

With the N parameter, you specify the number of ASCII characters to be converted. A maximum of 32767 valid ASCII characters can be converted. Only digits "0" to "9", upper case letters "A" to "F", and lower case letters "a" to "f" can be interpreted. All other characters are converted to zeros.

Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output word length is only half of the input word length. The ASCII characters are converted and positioned in the output in the same order as they are read in. If there is an odd number of ASCII characters, the hexadecimal number is padded with zeros in the nibble to the right of the last converted hexadecimal number.

Parameters

The following table shows the parameters of the "ATH" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	I, Q, D, L	Pointer to ASCII character string
N	Input	INT	I, Q, M, D, L or constant	Number of ASCII characters to be converted
RET_VAL	Return	WORD	I, Q, M, D, L	Status of the instruction
OUT	Output	VARIANT	I, Q, M, D, L	Hexadecimal number

You can find additional information on valid data types in "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code (W#16#....)*	Description
0000	No error
0007	Invalid character. Only the following ASCII characters may be used: Digits "0" to "9", upper case letters "A" to "F", lower case letters "a" to "f".
8101	Invalid pointer in the IN parameter, e.g., because a non-existing data block is referenced.
8182	Input buffer is too small for data in the N parameter.
8120	Invalid format in the IN parameter.
8151	Non-supported data type in the IN parameter.
8401	Invalid pointer in the OUT parameter, e.g., because a non-existing data block is referenced.
8482	Output buffer is too small for data in the N parameter.
8420	Invalid format in the OUT parameter.
8451	Non-supported data type in the OUT parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

ASCII characters and hexadecimal values

The following table shows the ASCII characters and the corresponding hexadecimal values:

ASCII character	ASCII-coded hexadecimal value	Hexadecimal digit
"0"	30	0
"1"	31	1
"2"	32	2
"3"	33	3
"4"	34	4
"5"	35	5
"6"	36	6
"7"	37	7
"8"	38	8
"9"	39	9
"A"	41	A
"B"	42	B
"C"	43	C
"D"	44	D
"E"	45	E
"F"	46	F

Example

The following table shows examples of the conversion of ASCII character strings to hexadecimal numbers:

IN	N	OUT	ENO status
'0123'	4	16#0123	1
'123AFx1a23'	10	16#123AF01a23	0

HTA: Convert hexadecimal number to ASCII string

Description

You use the instruction "HTA" to convert the hexadecimal number specified at the IN input into an ASCII character string. The result of the conversion is stored at the address specified in the OUT parameter.

- You can reference the following data types using the pointer in the IN parameter (hexadecimal): Bit strings, integers, STRING, Array of CHAR, Array of BYTE, WSTRING, Array of WCHAR.
- You can reference the following data types using the pointer in the OUT parameter (ASCII): STRING, Array of CHAR, Array of BYTE , WSTRING, Array of WCHAR.

With the N parameter, you specify the number of hexadecimal bytes to be converted. Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output value is twice as long as the input value. Each nibble of the hexadecimal number is converted to a character while maintaining the original order.

A maximum of 32767 characters can be written to the ASCII character string. The result of the conversion is represented by the digits "0" to "9" and upper-case letters "A" to "F".

If the complete result of the conversion cannot be displayed in the OUT parameter, the result is will only be partially written to the parameter.

Parameters

The following table shows the parameters of the "HTA" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	I, Q, M, D, L	Start address of the hexadecimal digits
N	Input	UINT	I, Q, M, D, L or constant	Number of hexadecimal bytes to be converted
RET_VAL	Return	WORD	I, Q, M, D, L	Error message
OUT	Output	VARIANT	I, Q, D, L	Address at which the result is stored.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0000	No error
8101	Invalid pointer at the IN parameter, e.g., because a non-existing data block is referenced.
8182	Input buffer is too small for data in the N parameter.
8120	Invalid format in the IN parameter.
8151	Non-supported data type in the IN parameter.
8401	Invalid pointer in the OUT parameter, e.g., because a non-existing data block is referenced.
8482	Output buffer is too small for data in the N parameter.
8420	Invalid format in the OUT parameter.
8451	Non-supported data type in the OUT parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

ASCII characters and hexadecimal values

The following table shows the ASCII characters and the corresponding hexadecimal values:

Hexadecimal digit	ASCII-coded hexadecimal value	ASCII character
0	30	"0"
1	31	"1"
2	32	"2"
3	33	"3"
4	34	"4"
5	35	"5"
6	36	"6"
7	37	"7"
8	38	"8"
9	39	"9"
A	41	"A"
B	42	"B"
C	43	"C"
D	44	"D"
E	45	"E"
F	46	"F"

Example

The following table shows examples of the conversion of hexadecimal numbers to ASCII character strings:

IN	N	OUT	ENO status
W#16#0123	2	'0123'	1
16#123AF01023	4	'123AF010'	0

LEN: Determine the length of a character string

Description

A tag of the (W)STRING data type contains two lengths: the maximum length and the current length (this is the number of currently valid characters).

- The maximum length of the character string is specified for each tag in the STRING keyword in square brackets. The number of bytes occupied by a string is 2 greater than the maximum length.
- The maximum length of the character string is specified for each tag in the WSTRING keyword in square brackets. The number of words occupied by a character string is 2 greater than the maximum length.
- The current length represents the number of the character places actually used. The current length must be less than or equal to the maximum length.

You use the instruction "LEN" to query the current length of the character string specified at the IN input parameter and output this information as a numerical value at the OUT output parameter. An empty string ('') has the length zero.

If errors occur during processing of the instruction, then an empty string will be output.

Note

Reading out the maximum length

You can use the "MAX_LEN (Page 3021)" instruction to read the maximum length of a string.

Parameters

The following table shows the parameters of the "LEN" instruction:

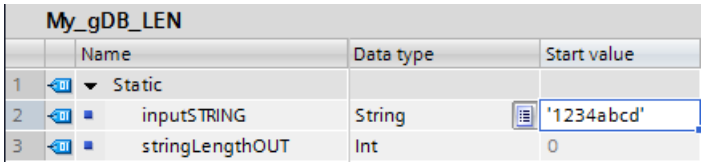
Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
OUT	Return	INT, DINT, REAL, LREAL	I, Q, M, D, L	Number of valid characters

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

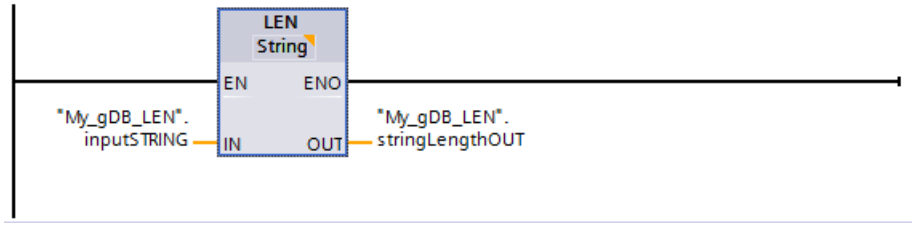
Example

In the following example, you determine the length of a character string with the STRING data type.

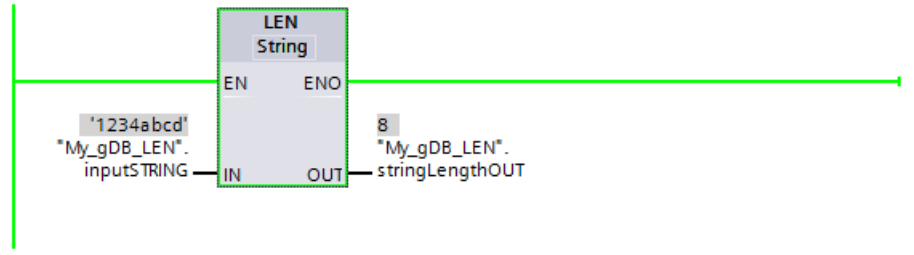
Create two tags in a global data block for storing the data.



Interconnect the parameters of the instruction as follows.



The number of actually occupied characters of the character string ("inputSTRING") is determined and returned as a numerical value at output parameter "OUT" ("stringLengthOUT").



CONCAT: Combine character strings

Description

You use the instruction "CONCAT" to combine the character string at the IN1 input parameter with the character string at the IN2 input parameter. The result is output at the OUT output parameter in (W)STRING format. If the resulting character string is longer than the tag specified in the OUT output parameter, then the resulting character string will be limited to the available length.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "CONCAT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING, WSTRING	D, L or constant	Character string
IN2	Input	STRING, WSTRING	D, L or constant	Character string
OUT	Return	STRING, WSTRING	D, L	Resulting character string

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

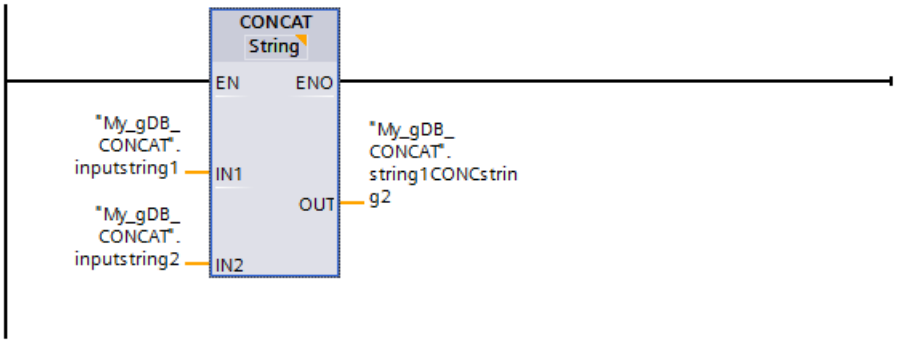
In the following example, you connect two character strings with the STRING data type.

Create three tags in a global data block for storing the data.

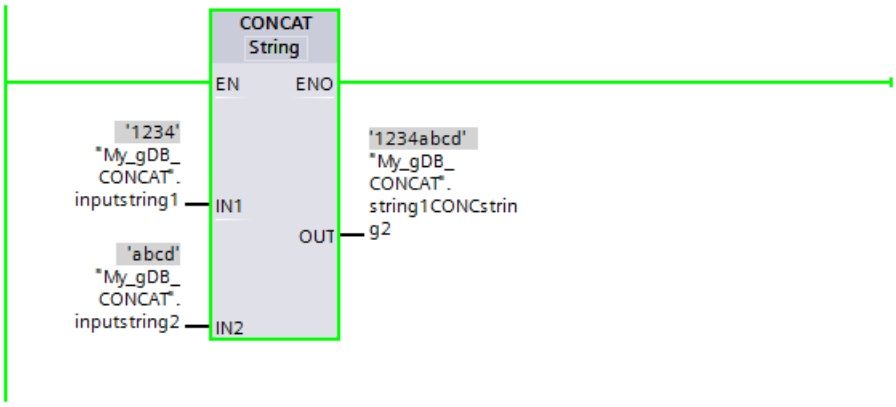
My_gDB_CONCAT

	Name	Data type	Start value
1	Static		
2	inputstring1	String	'1234'
3	inputstring2	String	'abcd'
4	string1CONCstring2	String	''

Interconnect the parameters of the instruction as follows.



The characters of the second character string ("inputstring2") are appended to the first character string ("inputstring1") and the result is output at output parameter OUT ("string1CONCstring2").



LEFT: Read the left character of a character string

Description

You use the instruction "LEFT" to extract a partial string beginning with the first character of the string at the IN input parameter. You specify the number of characters to be extracted in the L parameter. The extracted characters are output at the OUT output parameter in (W)STRING format.

If the number of characters to be extracted is greater than the current length of the character string, the OUT output parameter returns the input character string as a result. If the L parameter contains the value "0" or the input value is an empty string, an empty string will be returned. If the value in the L parameter is negative, an empty string will be output.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "LEFT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be extracted
OUT	Return	STRING, WSTRING	D, L	Extracted partial string

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

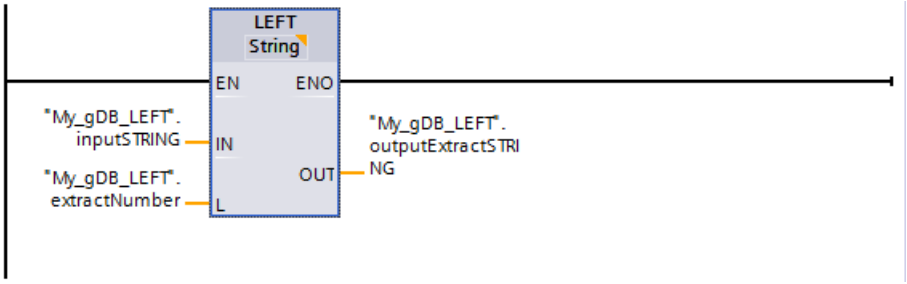
Example

In the following example, you extract a partial character string with the STRING data type from a character string starting from the first character.

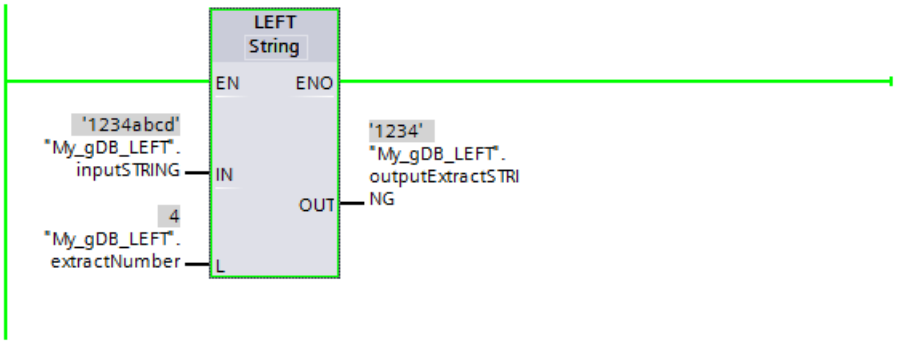
Create three tags in a global data block for storing the data.

My_gDB_LEFT			
	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	extractNumber	Int	4
4	outputExtractSTRING	String	''

Interconnect the parameters of the instruction as follows.



According to the value "4" of parameter L ("extractNumber"), you extract a four-character partial character string from the character string ("inputSTRING") starting from the first character from the left. The extracted partial character string is output at output parameter OUT ("outputExtractSTRING").



RIGHT: Read the right characters of a character string

Description

You use this instruction to extract the last L character in a character string in the input parameter IN. You specify the number of characters to be extracted in the L parameter. The extracted characters are output at the OUT output parameter in (W)STRING format.

If the number of characters to be extracted is greater than the current length of the character string, the OUT output parameter returns the input character string as a result. If the L parameter contains the value "0" or the input value is an empty string, an empty string will be returned. If the value in the L parameter is negative, an empty string will be output.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "RIGHT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be extracted
OUT	Return	STRING, WSTRING	D, L	Extracted partial string

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

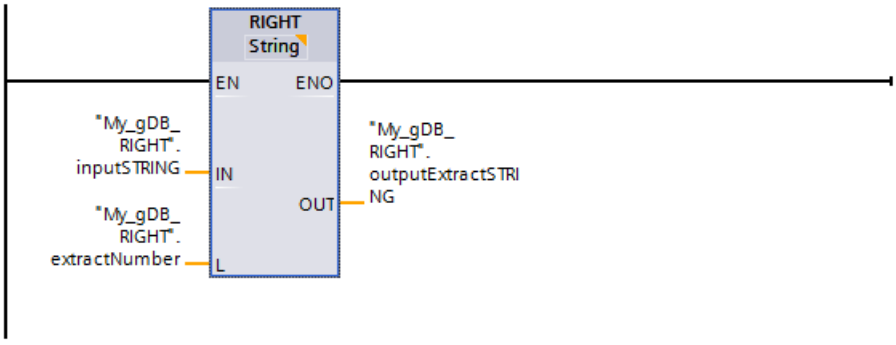
In the following example, you extract a partial character string with the STRING data type from a character string starting from the last character.

Create three tags in a global data block for storing the data.

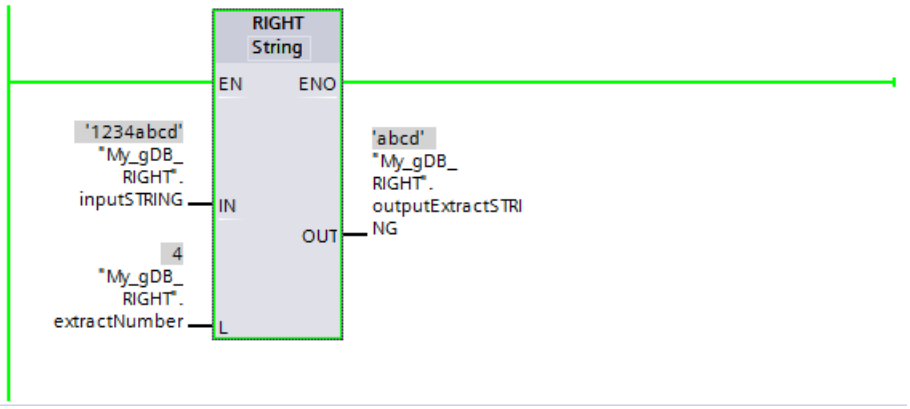
My_gDB_RIGHT

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	extractNumber	Int	4
4	outputExtractSTRING	String	''

Interconnect the parameters of the instruction as follows.



According to the value "4" of input parameter L ("extractNumber"), you extract a four-character partial character string from the character string ("inputSTRING") starting from the first character from the right. The extracted partial character string is output at output parameter OUT ("outputExtractSTRING").



MID: Read middle characters of a character string

Description

You use this instruction to extract a portion of the character string in the IN input parameter. With the P parameter, you specify the position of the first character to be extracted. With the L parameter, you define the length of the character string to be extracted. The extracted partial character string is output to the OUT output parameter.

The following rules must be observed when executing the instruction:

- If the number of characters to be extracted exceeds the current length of the character string in the IN input parameter, a partial character string will be output, starting from character position P and continuing to the end of the character string.
- If the character position specified in the P parameter falls outside the current character string length in the IN input parameter, an empty character string will be output in the OUT output parameter.
- If the value of the P or L parameter equals zero or is negative, an empty character string will be output in the OUT output parameter.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "MID" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Length of the string to be extracted

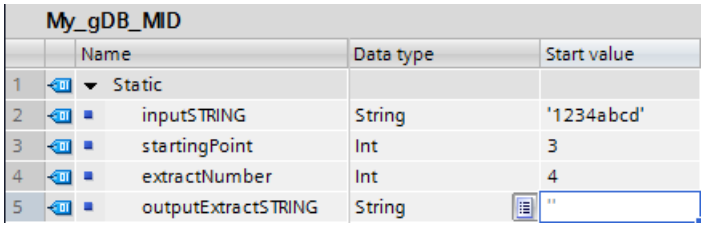
Parameter	Declaration	Data type	Memory area	Description
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of the first character to be extracted (first character = 1)
OUT	Return	STRING, WSTRING	D, L	Extracted partial string

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

In the following example, you extract a partial character string with the STRING data type from the middle of a character string.

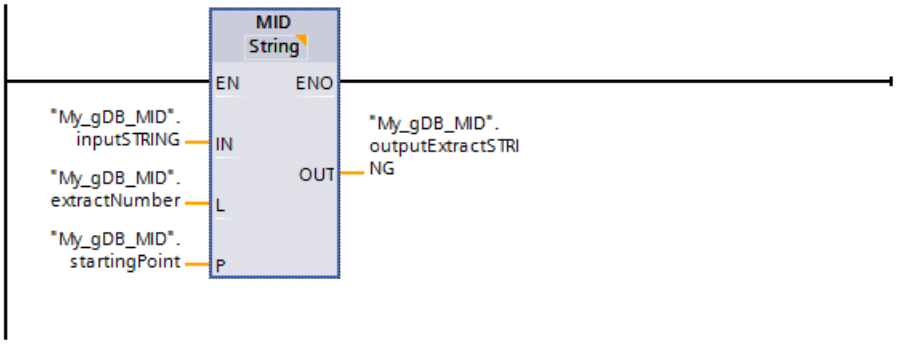
Create four tags in a global data block for storing the data.



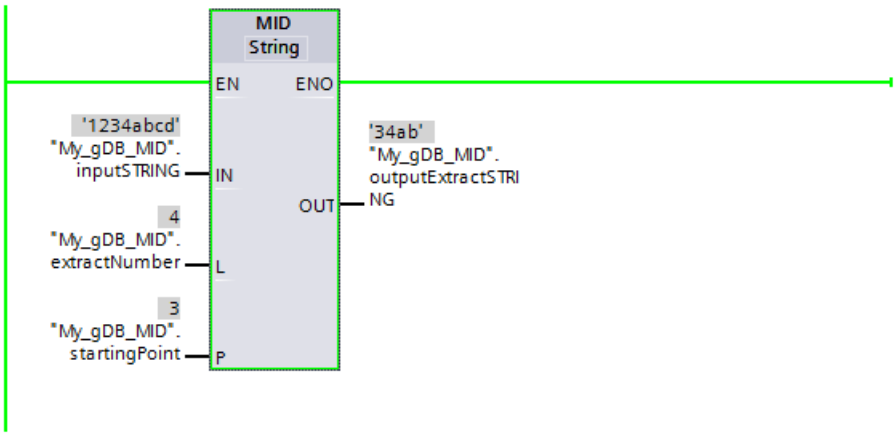
My_gDB_MID

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	startingPoint	Int	3
4	extractNumber	Int	4
5	outputExtractSTRING	String	''

Interconnect the parameters of the instruction as follows.



According to the value "4" of input parameter L ("extractNumber"), you extract a four-character partial character string from the character string ("inputSTRING"). The extraction starts from the third character ("startingPoint" has the value "3") of the character string ("inputSTRING"). The extracted partial character string is output at output parameter OUT ("outputExtractSTRING").



DELETE: Delete characters in a character string

Description

You use this instruction to delete a portion of the character string in the IN input parameter. With the P parameter, you specify the position of the first character to be deleted. You specify the number of characters to be deleted in the L parameter. The remaining partial character string is output at the OUT output parameter in (W)STRING format.

The following rules must be observed when executing the instruction:

- If the value in the P parameter is less than or equals zero, an empty character string will be output in the OUT output parameter.
- If the value in the P parameter is greater than the current length of the character string in the IN input, the input character string will be returned in the OUT output parameter.
- If the value in the L parameter equals zero, the input character string will be returned in the OUT output parameter.
- If the number of characters to be deleted at the L parameter is greater than the length of the character string at the IN input parameter, the characters starting at the position specified by the P parameter are deleted. The string resulting from this is output.
- If the value in the L parameter is negative, an empty character string will be output.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "DELETE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be deleted

Parameter	Declaration	Data type	Memory area	Description
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of first character to be deleted
OUT	Return	STRING, WSTRING	D, L	Resulting character string

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

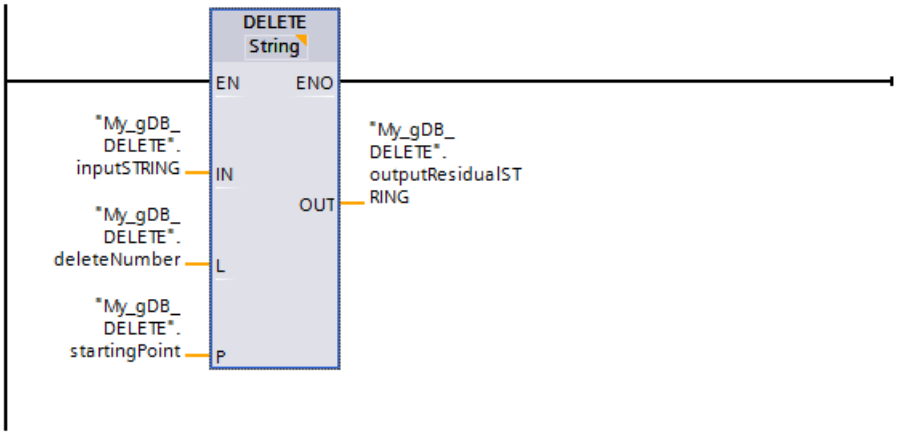
In the following example, you delete characters from a character string with the STRING data type.

Create four tags in a global data block for storing the data.

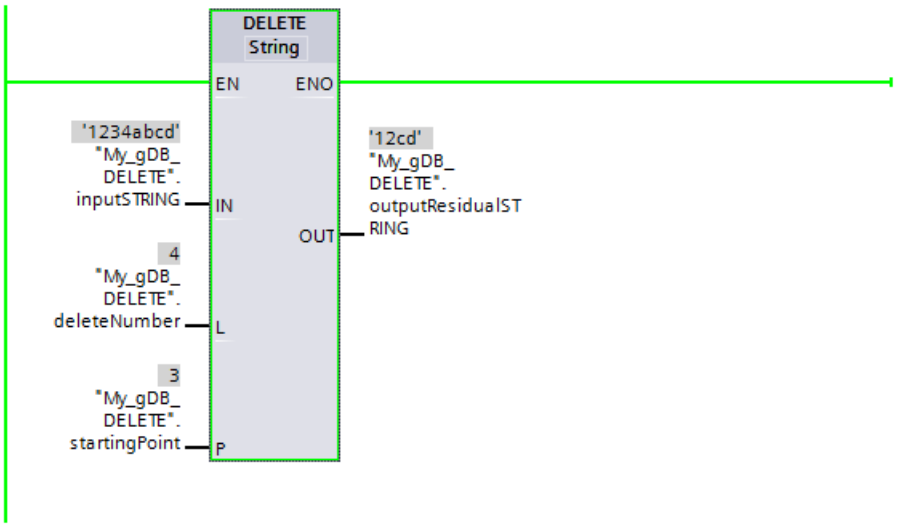
My_gDB_DELETE

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	startingPoint	Int	3
4	deleteNumber	Int	4
5	outputResidualSTRING	String	''

Interconnect the parameters of the instruction as follows.



According to the value "4" of input parameter L ("deleteNumber"), you delete four characters from the character string ("inputSTRING") starting from the third character ("startingPoint" has the value "3"). The remaining character string is output at output parameter OUT ("outputResidualSTRING").



INSERT: Insert characters in a character string

Description

You use this instruction to insert the character string in the IN2 input parameter to the character string in the IN1 input parameter. With the P parameter, you specify the position of the character starting at which the characters are inserted. The result is output at the OUT output parameter in (W)STRING format.

The following rules must be observed when executing the instruction:

- If the value in the P parameter exceeds the current length of the character string in the IN1 input parameter, the character string of the IN2 input parameter will be appended to the character string of the IN1 input parameter.
- If the value at the P parameter is zero, the character string at the IN2 parameter followed by the character string at the IN1 parameter will be output in the OUT output parameter.
- If the value in the P parameter is negative, an empty character string will be output in the OUT output parameter.
- If the resulting character string is longer than the tag specified in the OUT output parameter, the resulting character string will be limited to the available length.

Parameters

The following table shows the parameters of the "INSERT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING, WSTRING	D, L or constant	Character string
IN2	Input	STRING, WSTRING	D, L or constant	String to insert

Parameter	Declaration	Data type	Memory area	Description
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Insert position
OUT	Return	STRING, WSTRING	D, L	Resulting character string

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

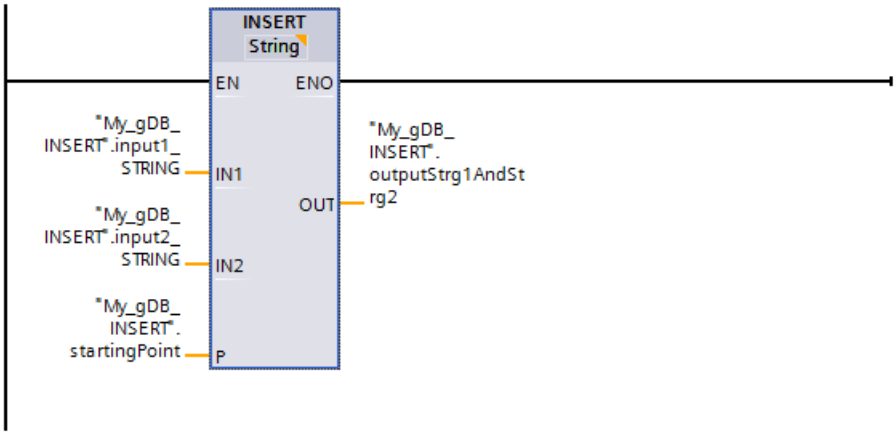
In the following example, you insert a character string in another character string. The data type used is STRING.

Create four tags in a global data block for storing the data.

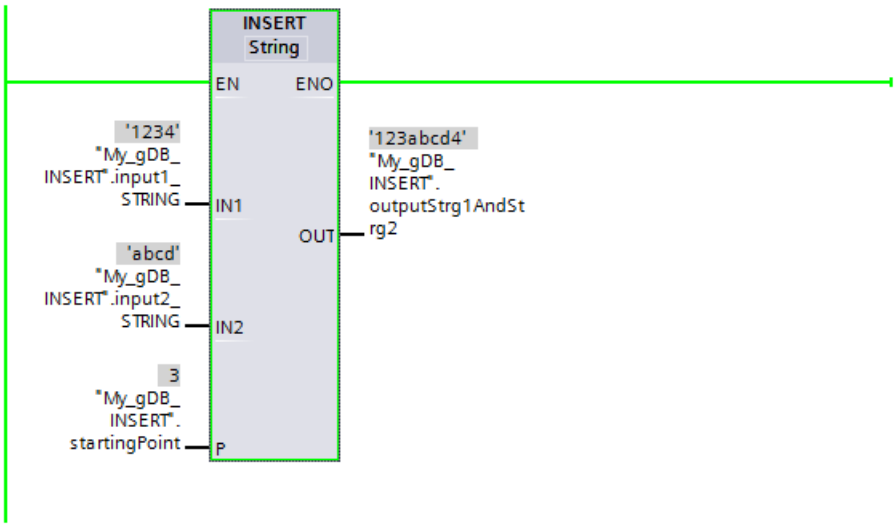
My_gDB_INSERT

	Name	Data type	Start value
1	Static		
2	input1_STRING	String	'1234'
3	input2_STRING	String	'abcd'
4	startingPoint	Int	3
5	outputStrg1AndStrg2	String	''

Interconnect the parameters of the instruction as follows.



The characters of the second character string ("input2_STRING") are inserted in the first character string ("input1_STRING"). According to the value "3" of parameter P ("startingPoint"), characters are inserted after the third character of the first character string ("input1_STRING"). The result is output at output parameter OUT ("outputStrg1AndStrg2").



REPLACE: Replace characters in a character string

Description

You use this instruction to replace a portion of the character string in the IN1 input with the character string in the IN2 input. You specify the position of the first character to be replaced in the P parameter. You specify the number of characters to be replaced in the L parameter. The result is output at the OUT output parameter in (W)STRING format.

The following rules must be observed when executing the instruction:

- If the value in the P parameter is less than or equals zero, an empty character string will be output in the OUT output parameter.
- If the value in the L parameter is less than zero, an empty character string will be output in the OUT output parameter.
- If P equals one, the character string in the IN1 input will be replaced beginning with (and including) the first character.
- If the value in the P parameter exceeds the current length of the character string in the IN1 input parameter, the character string of the IN2 input parameter will be appended to the character string of the IN1 input parameter.
- If the resulting character string is longer than the tag specified in the OUT output parameter, the resulting character string will be limited to the available length.
- If the value at parameter L is zero, characters are not replaced but inserted. Similar conditions apply as with the instruction INSERT. See also: INSERT: Insert characters in a character string (Page 3049)

Parameters

The following table shows the parameters of the "REPLACE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING, WSTRING	D, L or constant	String with characters to be replaced.
IN2	Input	STRING, WSTRING	D, L or constant	String with characters to be inserted.
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be replaced
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of first character to be replaced
OUT	Return	STRING, WSTRING	D, L	Resulting character string

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

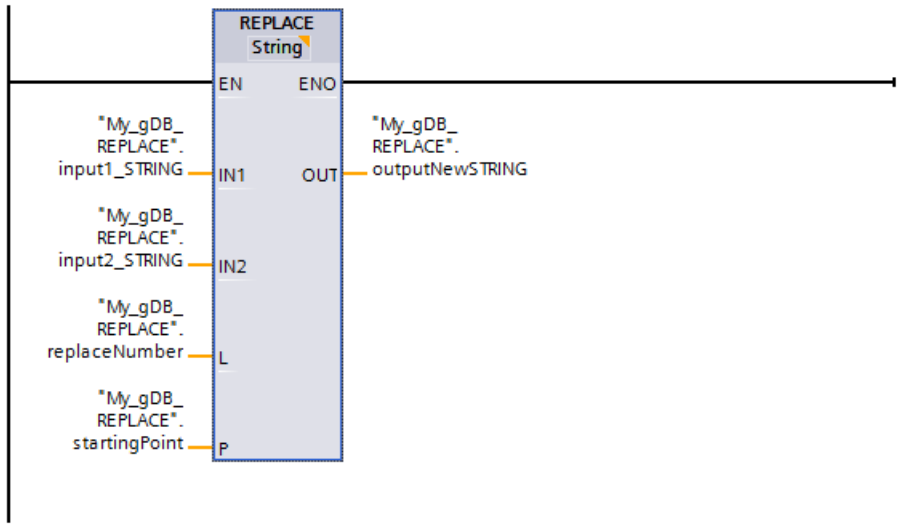
In the following example, you replace part of a character string with another character string. The data type used is STRING.

Create five tags in a global data block for storing the data.

My_gDB_REPLACE

	Name	Data type	Start value
1	Static		
2	input1_STRING	String	'1234'
3	input2_STRING	String	'abcd'
4	replaceNumber	Int	2
5	startingPoint	Int	3
6	outputNewSTRING	String	''

Interconnect the parameters of the instruction as follows.



The second character string ("input2_STRING") is added to the first character string ("input1_STRING") starting from the third character ("startingPoint" has the value "3"). According to the value "2" of parameter L ("replaceNumber"), the third and fourth characters of the first character string ("input1_STRING") are replaced in the process. The result is output at output parameter OUT ("outputNewSTRING").

My_gDB_REPLACE

	Name	Data type	Start value	Monitor value
1	Static			
2	input1_STRING	String	'1234'	'1234'
3	input2_STRING	String	'abcd'	'abcd'
4	replaceNumber	Int	2	2
5	startingPoint	Int	3	3
6	outputNewSTRING	String	''	'12abcd'

FIND: Find characters in a character string

Description

You can use this instruction to search through the character string in the IN1 input parameter for a specific character or a specific sequence of characters.

- You specify the value to be searched for in the IN2 input parameter. The search is made from left to right.
- The position of the first occurrence is output in the OUT output parameter. If the search returns no match, the value "0" will be output in the OUT output parameter.

If an invalid character is specified at the IN2 parameter or if an error occurs during processing, the value "0" is output at the OUT parameter.

Parameters

The following table shows the parameters of the "FIND" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING, WSTRING	D, L or constant	String searched through
IN2	Input	STRING, WSTRING, CHAR, WCHAR	D, L or constant (For (W)CHAR also I, Q, M)	Characters to search for
OUT	Return	INT	I, Q, M, D, L	Character position

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

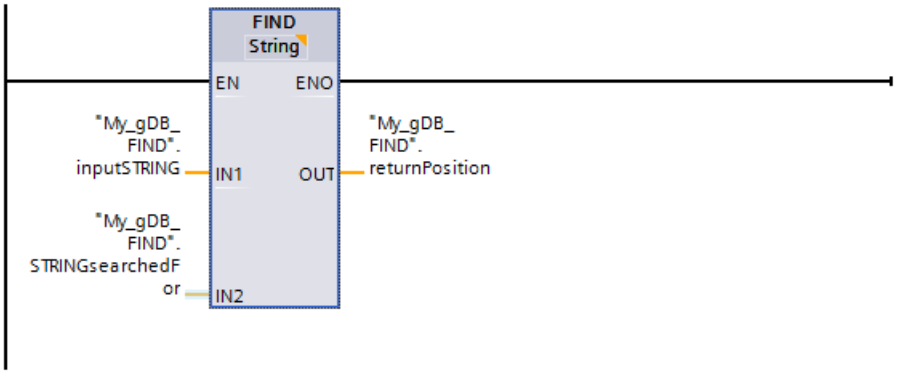
In the following example, you search for a character string in another character string. The data type used is STRING.

Create three tags in a global data block for storing the data.

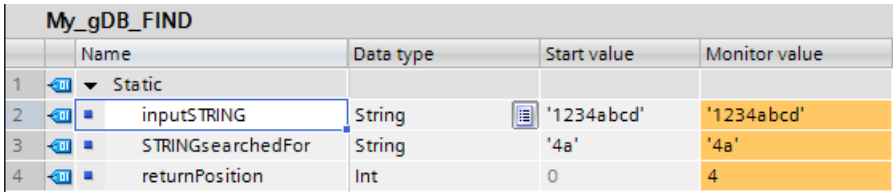
My_gDB_FIND

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	STRINGsearchedFor	String	'4a'
4	returnPosition	Int	0

Interconnect the parameters of the instruction as follows.



The first character string ("inputSTRING") searches for the value "4a" of the second character string ("STRINGsearchedFor"). The character position ("4") at which the searched character string begins is output at output parameter OUT ("returnPosition").



My_gDB_FIND

		Name	Data type	Start value	Monitor value
1		Static			
2		inputSTRING	String	'1234abcd'	'1234abcd'
3		STRINGsearchedFor	String	'4a'	'4a'
4		returnPosition	Int	0	4

Runtime information

GetSymbolName: Read out a tag on the input parameter

Description

With the "GetSymbolName" instruction, you use the input parameter of a block to read out the name of a tag interconnected at the parameter.

If a block is used multiple times in the project and called each time by a different tag, you can use the "GetSymbolName" instruction to evaluate the name of the calling tag. The process value of the tag is irrelevant in this case.

- You specify the input parameters of the block interface at the VARIABLE parameter. Use only an interface parameter for this parameter and no PLC or data block tags.
- You can limit the length of the read tag name using the SIZE parameter. If the name has been truncated, this is indicated by the characters "..." (Unicode character 16#2026) appears at the end of the name. Note that the "..." character itself has a length of 1. If you enter 6 as the maximum length at the SIZE parameter, for example, and read the tag name "MyPLCTag" via the block interface, the following is output: "MyPLC...".
- The name read is output at the OUT parameter.

Parameters

The following table shows the parameters of the "GetSymbolName" instruction:

Parameter	Declaration	Data type	Memory area	Description
VARIABLE	Input	PARAMETER	L	Input parameters of the block interface
SIZE	Input	DINT	I, Q, M, D, L	Length limit for the read names
OUT	Return	WSTRING	I, Q, M, D, L	Name of the tag that was read via the block interface

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

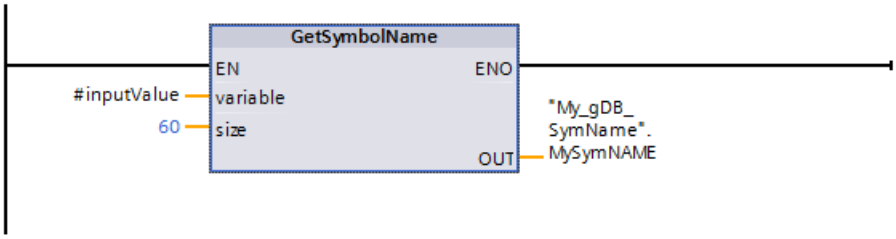
In the following example, you read out the name of a tag that is interconnected via the input parameter of a block.

Create two tags in a global data block for storing the data.

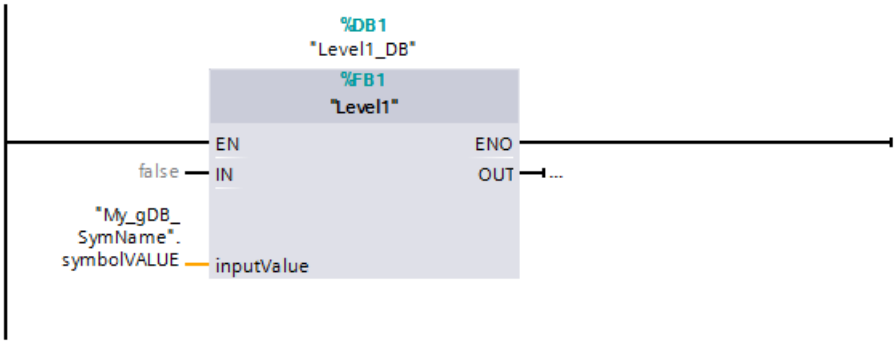
My_gDB_SymName

		Name	Data type	Start value
1		Static		
2		MySymNAME	WString	WSTRING#''
3		symbolVALUE	Byte	16#42

Create an input parameter inputValue with the BYTE data type in the "Level1" block. Call the "GetSymbolName" instruction in the "Level1" block. Interconnect the parameters of the instruction as follows.



Interconnect the inputValue parameter of the "Level1" block as follows.



The "GetSymbolName" instruction is executed in the "Level1" block. Input parameter inputValue of the "Level1" block is examined for its interconnection using input parameter VARIABLE of the instruction. In doing so, the "symbolVALUE" tag is read out and output as a character string at output parameter OUT ("MySymNAME"). According to the value of input parameter SIZE, the length of the character string is limited to 60 characters.

My_gDB_SymName

		Name	Data type	Start value	Monitor value
1		Static			
2		MySymNAME	WString	WSTRING#''	WSTRING#'"My_gDB_SymName".symbolVALUE'
3		symbolVALUE	Byte	16#42	16#42

GetSymbolPath: Query composite global name of the input parameter assignment

Description

You can use the "GetSymbolPath" instruction to read the composite global name of an input parameter at the local interface. The name consists of the storage path and the tag name.

- Specify the block interface through which the name of the input tag is to be read at the VARIABLE parameter of the instruction:
  - If the tag of a data block is used to supply the input parameters, the name of the DB, contained structures and the name of the tag are output as the path.
  - If a PLC tag is used to supply the input parameters, the name of the PLC tag is output.
  - If a constant is used to supply the input parameters, the content of the constant is output.
- You can limit the length of the read tag name using the SIZE parameter. If the name has been truncated, this is indicated by the characters "..." (Unicode character 16#2026) at the end of the name. Note that even the "..." character itself has a length of 1.
  If you enter 6 as the maximum length at the SIZE parameter, for example, and read the tag name "MyPLCTag" via the block interface, the following is output: "MyPLC...".

Parameters

The following table shows the parameters of the "GetSymbolPath" instruction:

Parameters	Declaration	Data type	Memory area	Description
VARIABLE	Input	VARIANT	L	Selection of the local interface to which you want to read the global name of the input parameter supply.
SIZE	Input	DINT	I, Q, M, D, L or constant	Limits the number of characters output at the OUT parameter. If the output of characters should not be limited, enter "0" for the SIZE parameter.
OUT	Output	WSTRING	I, Q, M, D, L	Output of the tag name of the input parameters supply.

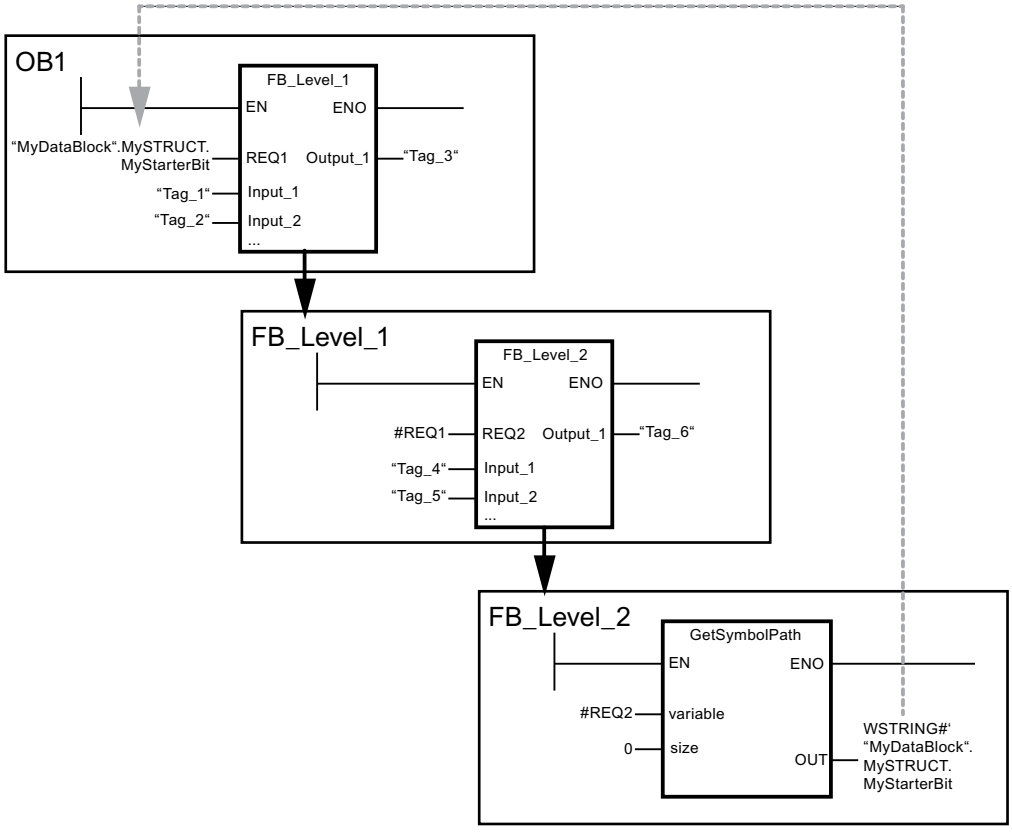
You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

The following example shows the use of GetSymbolPath over several call levels:

- The FB_Level_1 block is called from the organization block OB1, and from there the FB_Level_2 block is called in turn.
- GetSymbolPath is executed in the FB_Level_2 block to read the path of the parameter at the REQ2 interface.

- Since REQ2 is supplied by the REQ1 interface in turn, the instruction determines the path of the input parameter of REQ1.
- MyStarterBit is used as a parameter at the REQ1 parameter. The bit is located in the MySTRUCT structure in the MyDatablock data block.
  This information is read by GetSymbolPath and output at the OUT parameter.



GetInstanceName: Read out name of the block instance

Description

You can use the "GetInstanceName" instruction to read the name of the instance data block within a function block.

- You can specify how many characters of the instance name should be read out using the SIZE parameter. If you use the "0" value for SIZE, the entire name is read.
- The name of the instance data block is written to the OUT parameter. If the name of the instance data block is longer than the maximum length of WSTRING, the name is truncated.

Parameters

The following table shows the parameters of the "GetInstanceName" instruction:

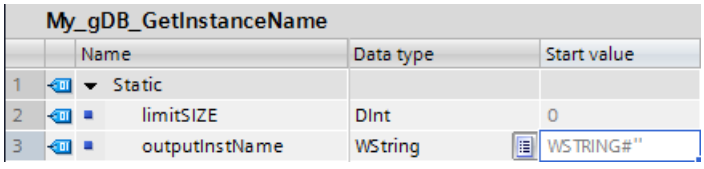
Parameter	Declaration	Data type	Memory area	Description
SIZE	Input	DINT	I, Q, M, D, L or constant	Number of characters up to which the name of the instance data block is to be read
OUT	Output	WSTRING	I, Q, M, D, L	Read name of the instance data block

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

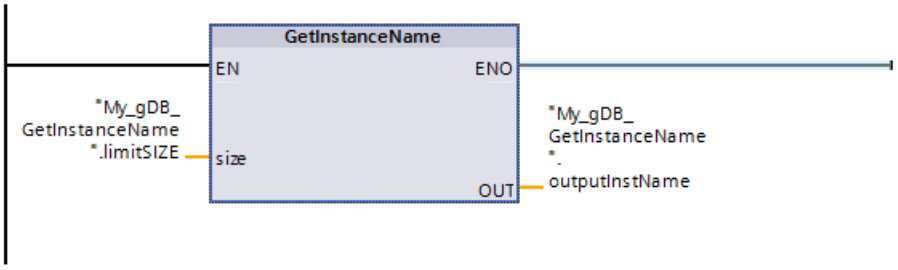
Example

In the following example, you read out the name of an instance data block.

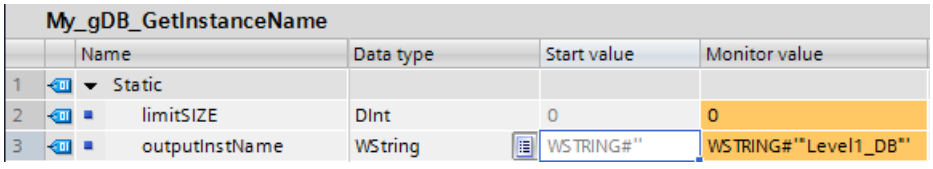
Create two tags in a global data block for storing the data.



Interconnect the parameters of the instruction as follows.



The "GetInstanceName" instruction is executed in the "Level1_gin" block. By means of the "GetInstanceName" instruction, the associated instance data block of the "Level1_gin" block is determined and output as a character string at output parameter OUT ("outputInstName"). According to the value "0" of parameter SIZE ("limitSIZE"), the length of the character string is unlimited.



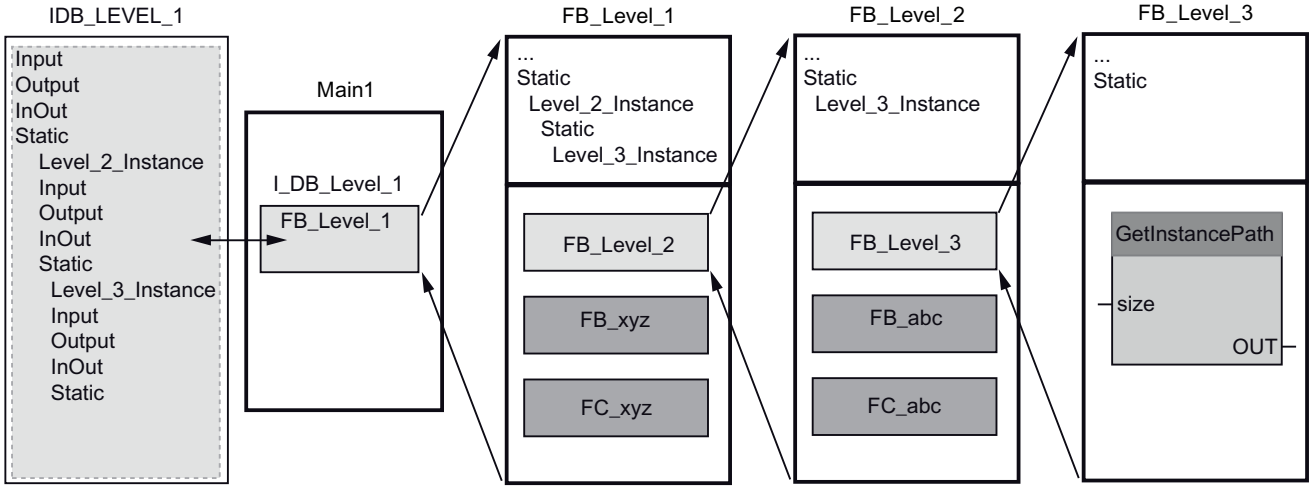
GetInstancePath: Query composite global name of the block instance

Description

You use the "GetInstancePath" instruction to read the composed global name of the block instance within a function block. The composed global name of the block instance is the path of the complete call hierarchy when multi-instances are used.

In the following example, the "GetInstancePath" instruction is called in the FB_Level_3 function block.

- The FB_Level_3 function block stores its data in the calling FB_Level_2 function block.
- The FB_Level_2 function block in turn stores its data in the calling FB_Level_1 function block.
- The FB_Level_1 function block in turn stores its data in its instance data block IDB_LEVEL_1. Through the use of multi-instances, the instance data block of FB_Level_1 contains all data of the three function blocks.



The "GetInstancePath" instruction returns the following path for this example:
'"IDB_LEVEL_1".Level_2_Instance.Level_3_Instance'

Note

Use of "GetInstancePath" in function blocks with single instance

If the function block in which you call "GetInstancePath" saves data in its own instance data block, the name of the single instance is output as the global name. The result at parameter OUT corresponds in this case to the ""GetInstanceName" instruction.

Parameters

The following table shows the parameters of the "GetInstancePath" instruction:

Parameter	Declaration	Data type	Memory area	Description
SIZE	Input	DINT	I, Q, M, D, L or constant	Number of characters up to which the global name of the block instance is to be read out. When SIZE = 0, the complete global name is output.
OUT	Output	WSTRING	I, Q, M, D, L	Read global name of the block instance. If the global name of the block instance is longer than the maximum length of WSTRING (254 characters), the name is truncated.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

See also

GetInstanceName: Read out name of the block instance (Page 3058)

GetBlockName: Read out name of the block

Description

You can use the "GetBlockName" instruction to read the name of the block in which the instruction is called.

- You can specify how many characters of the block name should be read using the SIZE parameter. If you use the "0" value for SIZE, the entire name is read.
- The name of the block is written at the OUT parameter. If the name of the block is longer than the maximum length of WSTRING, the name is truncated.

Parameters

The following table shows the parameters of the "GetBlockName" instruction:

Parameter	Declaration	Data type	Memory area	Description
SIZE	Input	UINT	I, Q, M, D, L or constant	Number of characters up to which the name of the instance data block is to be read.
RET_VAL	Output	WSTRING	I, Q, M, D, L	Read name of the instance data block

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

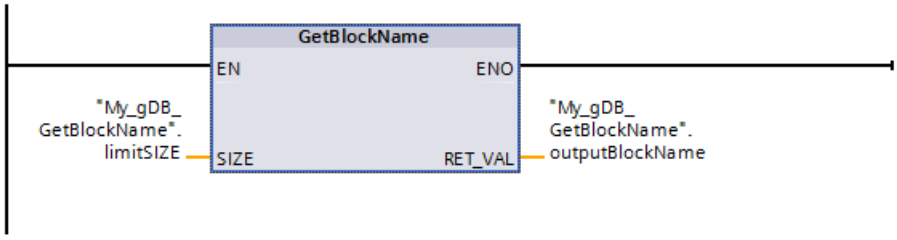
In the following example, you read out the block name.

Create two tags in a global data block for storing the data.

My_gDB_GetBlockName

		Name	Data type	Start value
1		Static		
2		limitSIZE	DInt	0
3		outputBlockName	WString	WSTRING#''

Interconnect the parameters of the instruction as follows.



The "GetBlockName" instruction is executed in the "Level1_gbn" block. By means of the "GetBlockName" instruction, the name of the "Level1_gbn" block is read out and output as a character string at output parameter OUT ("outputBlockName"). According to the value "0" of parameter SIZE ("limitSIZE"), the length of the character string is unlimited.

My_gDB_GetBlockName

		Name	Data type	Start value	Monitor value
1		Static			
2		limitSIZE	DInt	0	0
3		outputBlockName	WString	WSTRING#''	WSTRING#'"Level1_gbn"'

11.6.3.3 Process image

UPDAT_PI: Update the process image inputs

Description

With the instruction, you update the OB 1 process image (=process image partition 0) of the inputs or a process image partition of the inputs of the inputs defined by configuration.

If you have configured repeated signaling of I/O access errors for the system-side process image update, then the selected process image will be updated constantly.

Otherwise, this update will only be performed if the selected process image partition is not updated by the system, in other words:

- If you have not assigned this process image partition to an interrupt OB
  or
- If you selected process image partition 0 and have disabled updating of the OB 1 process image partition during configuration.

Note

Each logical address that you have assigned to a process image partition of the inputs per configuration no longer belongs to the OB 1 process image of the inputs.

When you update a process image partition with "UPDAT_PI", you must not perform a simultaneous update with the "SYNC_PI (Page 3066)" instruction.

System-side updating of the OB 1 process image of the inputs and the process image partitions of inputs that you have assigned to an interrupt OB takes place independently of "UPDAT_PI" calls.

Parameters

The following table shows the parameters of the "UPDAT_PI"instruction:

Parameter	Declaration	Data type	Memory area	Description
PART	Input	PIP	I, Q, M, D, L or constant	Number of the process image partition of the inputs to be updated. Maximum value range (depending on the CPU): 0 to 31 (0 means OB 1 process image, n where 1 <= n <= 31 means process image partition n).
RET_VAL	Return	INT	I, Q, M, D, L, P	Error information
FLADDR	Output	WORD	I, Q, M, D, L, P	Address of the first byte to cause an error if an access error has occurred.

You can find additional information on valid data types in "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for PART parameter
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU.
8092	The process image partition is being updated by the system and cannot be used for the PART parameter.
8093	The process image partition update is being processed in another OB.

Error code* (W#16#...)	Explanation
80A0	An access error was detected when accessing the I/O.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

If you use the instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPRD_DAT (Page 3124)" instruction are also possible.

UPDAT_PO: Update the process image outputs

Description

You use the instruction to transfer the signal states of the OB 1 process image (= process image partition 0) of the outputs or a process image partition of the outputs defined per configuration to the output modules.

If you have specified a consistency range for the selected process image partition, corresponding data is transferred as consistent data to the respective I/O module.

Note

Each logical address you have assigned to a process image partition of the outputs during configuration no longer belongs to the OB 1 process image of the outputs.

Outputs that you update with "UPDAT_PO" must not be updated simultaneously with the "SYNC_PO (Page 3067)" instruction.

The OB 1 process image of the outputs and the process image partitions of the outputs that you have assigned to an interrupt OB are transferred by the system to the output modules independently of "UPDAT_PO" calls.

Parameters

The following table shows the parameters of the "UPDAT_PO" instruction:

Parameter	Declaration	Data type	Memory area	Description
PART	Input	PIP	I, Q, M, D, L or constant	Number of the process image partition of the outputs to be transferred. Maximum value range (depending on the CPU): 0 to 31. (0 means OB 1 process image, n where 1 <= n <= 31 means process image partition n)
RET_VAL	Return	INT	I, Q, M, D, L, P	Error information
FLADDR	Output	WORD	I, Q, M, D, L, P	Address of the first byte to cause an error if an access error has occurred.

You can find additional information on valid data types in "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for PART parameter
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU.
8092	The process image partition is being updated by the system and cannot be used for the PART parameter.
8093	The process image partition update is being processed in another OB.
80A0	An access error was detected when accessing the I/O.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

If you use the instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPWR_DAT (Page 3126)" instruction are also possible.

SYNC_PI: Synchronize the process image inputs

Description

The "SYNC_PI" is used to update the process image partition of inputs in isochronous mode. A user program linked to a DP cycle or PN send cycle can use this instruction to update input data acquired in a process image partition of the inputs isochronously and consistently.

Call

"SYNC_PI" is interruptible and can only be called in OBs 61, 62, 63 and 64.

Note

A call of the "SYNC_PI" instruction in OBs 61 to 64 is only permitted if you have assigned the affected process image partition to the associated OB in the HW configuration.

A process image partition that you update with "SYNC_PI" must not be updated simultaneously with the "UPDAT_PI (Page 3062)" instruction.

Parameters

The following table shows the parameters of the "SYNC_PI" instruction:

Parameter	Declaration	Data type	Memory area	Range of values	Description
PART	Input	PIP	I, Q, M, D, L or constant	1 to 31	Number of the process image partition of the inputs to be updated isochronously.
RET_VAL	Return	INT	I, Q, M, D, L, P	-	Error information
FLADDR	Output	WORD	I, Q, M, D, L, P		Address of the first byte to cause an error if an access error has occurred.

You can find additional information on valid data types in "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code (W#16#...)*	Explanation
0000	No error occurred.
0001	Consistency warning. The update of the process image partition was distributed over two DP or PN cycles. However, the data in one slave or IO device were consistently transferred.
8090	Illegal value on PART parameter or updating the specified process image partition of the inputs is not permitted in this OB. The process image partition of the inputs was not updated.
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU. The process image partition of the inputs was not updated.
8092	The process image partition is being updated by the system and cannot be used for the PART parameter.
8093	The process image partition update is being processed in another OB.
80A0	During updating an access error was detected. The affected inputs were set to "0".

Error code (W#16#...)*	Explanation
80A1	The update time is after the permitted access window. The process image partition of the inputs was not updated. The DP or PN cycle is too short to ensure enough time for processing the instruction. You must therefore increase the TDP (also known as T_DC), Ti and To timers.
80A2	Access error with consistency warning In the case of update of the specified process image partition an access error with simultaneous consistency warning was detected. • The data of the incorrect inputs was not read by the I/O. In the process image partition of the inputs the involved inputs are set to zero. • The update of the process image partition of the not involved by an access error input data was distributed over two DP and PN cycles.
80C1	The update time is before the permitted access window. The process image partition of the inputs was not updated.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

If you use the "SYNC_PI" instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPRD_DAT (Page 3124)" instruction are also possible.

SYNC_PO: Synchronize the process image outputs

Description

The instruction "SYNC_PO" is used to update a process image partition of outputs in isochronous mode. An application program linked to a DP cycle or a PN send cycle can use this instruction for the consistent isochronous transfer of the computed output data of a process image partition from outputs to the I/O devices.

Call

"SYNC_PO" is interruptible and can only be called in OBs 61, 62, 63 and 64.

Note

A call of the "SYNC_PO" instruction in OBs 61 to 64 is only permitted if you have assigned the affected process image partition to the associated OB in the HW configuration.

A process image partition that you update with "SYNC_PO" must not be updated simultaneously with the "UPDAT_PO (Page 3064)" instruction.

Parameters

The following table shows the parameters of the "SYNC_PO" instruction:

Parameter	Declaration	Data type	Memory area	Range of values	Description
PART	Input	PIP	I, Q, M, D, L or constant	1 to 31	Number of the process image partition of the outputs to be updated isochronously.
RET_VAL	Return	INT	I, Q, M, D, P	-	If an error occurs while the instruction is being executed, the return value contains an error code.
FLADDR	Output	WORD	I, Q, M, D, L, P	-	Address of the first byte that has caused the error.

You can find additional information on valid data types in "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code (W#16#...)*	Explanation
0000	No error occurred.
0001	Consistency warning. The update of the process image partition was distributed over two DP or PN cycles. However, the data in one slave or IO device were consistently transferred.
8090	Illegal value in PART parameter, or updating the specified process image partition of the outputs is not permitted in this OB. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
8092	The process image partition is being updated by the system and cannot be used for the PART parameter.
8093	The process image partition update is being processed in another OB.
80A0	In the case of update of the specified process image partition of the outputs an access error was detected. Incorrect outputs were not transferred to the I/O. During process image partition of the outputs, these outputs remain unchanged.
80A1	Access error with consistency warning In the case of update of the specified process image partition of the outputs an access error with simultaneous consistency warning was detected. • The data of the incorrect outputs were not transferred to the I/O. During process image partition of the outputs, the involved outputs remain unchanged. • The update of the process image partition of the not involved by an access error input data was distributed over two DP and PN cycles.
80A2	The update time is after the permitted access window. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
80C1	The update time is before the permitted access window. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

If you use the "SYNC_PO" instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPWR_DAT (Page 3126)" instruction are also possible.

11.6.3.4 Distributed I/O

RDREC: Read data record

Description

You use the instruction "RDREC" to read the data record with the number INDEX from the module addressed using the ID. This may be a module in a central rack or a distributed module (PROFIBUS DP or PROFINET IO).

- Use the parameter ID to select from which module (DP/PROFINET IO) a data record is to be read. Use only the hardware identifier of the module for the "ID" parameter.
- Use the INDEX parameter to select the information to be read from the module. Which data records can be read with which data record number depends on the module. For additional information, see the manual to the specific module.
- The data records readable via "RDREC" have a different length. Use MLEN to specify the maximum number of data record bytes you want to read. If length "0" is selected at the parameter MLEN, the complete data record is written at the parameter RECORD.
- The destination area RECORD should have at least the length of MLEN bytes. If you read out the full data record with MLEN=0, use the maximum length of the data record for RECORD. The structure (configuration, data types and length) that you use at the parameter RECORD, also depends on which data record is read out by which module.
- The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the destination area RECORD. In this case, the LEN output parameter contains the length of the read data in bytes.
- If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR. In this case, the output parameter STATUS contains the error information.

Note

The interface of the "RDREC" instruction is identical to the "RDREC" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Functional description

"RDREC" is an instruction that works asynchronously, which means its execution extends over multiple calls. You start the data record transfer by calling "RDREC" with REQ= 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS. The two central bytes of STATUS correspond to the RET_VAL output parameter of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

The transfer of the data record is complete when the output parameter BUSY has the value FALSE .

Parameters

The following table shows the parameters of the "RDREC" instruction:

Parameter	Declaration	Data type*	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T**, C** or constant	REQ = 1: Transfer data record
ID	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the hardware module (DP/PROFINET IO) The number is assigned automatically and is stored in the properties of the module or of the interface in the hardware configuration.
INDEX	Input	BYTE, DINT, INT, SINT, UINT, USINT, WORD	I, Q, M, D, L or constant	Data record number
MLEN	Input	BYTE, UINT, USINT	I, Q, M, D, L or constant	Maximum length in bytes of the data record information to be read
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information
LEN	Output	UINT	I, Q, M, D, L	Length of the read data record information
RECORD	InOut	VARIANT	I, Q, M, D, L	Destination area for the data record read.
* There is no implicit conversion in STL, which is why the range of valid data types may be limited. During programming in STL, note the permissible data types in each case in the tooltip of the parameter ** For S7-1500 only.				

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Note

If you use "RDREC" to read a data record for PROFINET IO, then negative values in the INDEX, MLEN, and LEN parameters will be interpreted as an unsigned 16-bit integer.

Parameter STATUS

For interpretation of the STATUS parameter, see Parameter STATUS (Page 3078).

WRREC: Write data record

Description

The instruction "WRREC" is used to transfer the RECORD data record to the component addressed using ID. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

- Use LEN to specify the length of the data record to be transmitted in bytes. The selected length of the source range RECORD should have at least the length of LEN bytes.
- The value TRUE at output parameter DONE indicates that the data record has been successfully transferred.
- If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR. In this case, the output parameter STATUS contains the error information.

Note

The interface of the "WRREC" instruction is identical to the "WRREC" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Functional description

"The WRREC" instruction works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling "WRREC" with REQ = 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS. The two central bytes of STATUS correspond to the RET_VAL output parameter of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

Note that you must assign the same value to the actual parameter of RECORD for all "WRREC" calls that belong to one and the same job. The same applies to the actual parameters of LEN.

The transfer of the data record is complete when the output parameter BUSY has the value FALSE.

Parameters

The following table shows the parameters of the instruction "WRREC":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T*, C* or constant	REQ= 1: Transfer data record
ID	Input	HW_IO	I, Q, M, D, L or constant	ID number of the hardware component (DP/ PROFINET IO) The number is assigned automatically and is stored in the properties of the component or of the interface in the hardware configuration.
INDEX	Input	DINT	I, Q, M, D, L or constant	Data record number
LEN	Input	BYTE, UINT, USINT	I, Q, M, D, L or constant	(hidden) Maximum length of the data record to be transferred in bytes
DONE	Output	BOOL	I, Q, M, D, L	Data record was transferred
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the writing process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information For interpretation of the STATUS parameter, see Parameter STATUS (Page 3078).
RECORD	InOut	VARIANT	I, Q, M, D, L	Data record
* For S7-1500 only.				

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Note

If you use "WRREC" to write a data record for PROFINET IO, negative values in the INDEX and LEN parameters will be interpreted as an unsigned 16-bit integer.

Parameter STATUS

For interpretation of the STATUS parameter, see Parameter STATUS (Page 3078).

GETIO: Read process image

Description

You use the instruction "GETIO" to consistently read out all inputs of a DP standard slave/ PROFINET IO device. The instruction "GETIO" calls the instruction "DPRD_DAT (Page 3124)". If there was no error during the data transmission, the data that have been read are entered in the target range indicated by INPUTS .

The target range must have the same length that you configured for the selected component.

If you read from a DP standard slave with a modular configuration or with several DP identifiers, you can only access the data of one component/DP identifier at the configured start address with a "GETIO" call.

Parameter

The following table shows the parameters of the instruction "GETIO":

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUB-MODULE	I, Q, M, D, L or constant	Hardware ID of the DP standard slave / PROFINET IO device.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPRD_DAT (Page 3124)" in the form DW#16#40xxxx00.
LEN	Output	INT	I, Q, M, D, L	Amount of data read in bytes
INPUTS	InOut	VARIANT	I, Q, M, D	Target range for the read data. It must have the same length as the range that you configured for the selected DP standard slave / PROFINET IO device. Only the BYTE data type is permitted.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

STATUS parameter

See also: DPRD_DAT: Read consistent data of a DP standard slave (Page 3124).

SETIO: Transfer process image

Description

You use the instruction "SETIO" to consistently transfer data from the source range defined by the parameter OUTPUTS to the addressed DP standard slave / PROFINET IO device, and, if necessary, to the process image (if you have configured the relevant address area of the DP standard slave / PROFINET IO device as a consistent range in a process image). "SETIO" calls the instruction "DPWR_DAT (Page 3126)".

The source range must have the same length that you configured for the selected component.

In the case of a DP standard slave / PROFINET IO device with modular configuration or with several DP identifiers, you can only access one DP identifier / component per "SETIO" call.

Parameters

The following table shows the parameters of the "SETIO" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUBMOD-ULE	I, Q, M, D, L or constant	Hardware ID of the DP standard slave / PROFINET IO device.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPWR_DAT (Page 3126)" in the form DW#16#40xxxx00.
OUTPUTS	InOut	VARIANT	I, Q, M, D	Source range for the data to be written. It must have the same length as the range that you configured for the selected DP standard slave / PROFINET IO device. Only the BYTE data type is permitted.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter STATUS

See also: DPWR_DAT: Write consistent data of a DP standard slave (Page 3126).

GETIO_PART: Read process image area

Description

You use the instruction "GETIO_PART" to consistently read out a related part of the inputs of an IO module. "GETIO_PART" calls the instruction "DPWR_DAT (Page 3124)".

Use the ID input parameter to select the IO module by means of the hardware ID.

You use the OFFSET and LEN parameters to specify the portion of the process image area to be read. If the input area spanned by OFFSET and LEN is not completely covered by the module, the block returns the error code DW#16#40**80B7**00.

The length of the target range must be larger than or equal to the amount of bytes to be read:

- If there was no error during the data transmission, ERROR receives the value FALSE. The data that are read are written to the target range defined at the INPUTS parameter.
- If there was no error during the data transmission, ERROR receives the value TRUE. The STATUS parameter receives the error information from "DPRD_DAT".
- If the target range is greater than LEN, then the first LEN bytes of the target range will be written. ERROR receives the value FALSE.

Parameters

The following table shows the parameters of the "GETIO_PART" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUBMOD-ULE	I, Q, M, D, L or constant	Hardware identifier of the module
OFFSET	Input	INT	I, Q, M, D, L or constant	Number of the first byte to be read in the process image for the component (smallest possible value: 0).
LEN	Input	INT	I, Q, M, D, L or constant	Number of bytes to be read.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPRD_DAT" in the form DW#16#40xxxx00, if ERROR = TRUE.
ERROR	Output	BOOL	I, Q, M, D, L	Error display: ERROR = TRUE if an error occurs when "DPRD_DAT" is called.
INPUTS	InOut	VARIANT	I, Q, M, D	Target range for read data: If the target range is greater than LEN, the first LEN bytes of the target range are written.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter STATUS

See "RET_VAL" parameter of the "DPRD_DAT (Page 3124)" instruction.

SETIO_PART: Transfer process image area

Description

You can use the "SETIO_PART" instruction to consistently write data from the source area spanned by OUTPUTS to the outputs of an IO module. "SETIO_PART" calls the instruction "DPWR_DAT (Page 3126)".

Use the ID input parameter to select the IO module by means of the hardware ID.
You use the OFFSET and LEN parameters to specify the portion of the process image area to be written for the components addressed by means of ID. If the output area spanned by OFFSET and LEN is not completely covered by the module, the block returns the error code DW#16#40**80B7**00.

The length of the source range must be greater than or equal to the number of bytes to be written:

- If there was no error during the data transmission, ERROR receives the value FALSE.
- If there was an error during the data transmission, ERROR receives the value TRUE, and STATUS receives the error information of "DPWR_DAT".
- If the source range is greater than LEN, the first LEN bytes from OUTPUTS are transferred. ERROR receives the value FALSE.

Parameter

The following table shows the parameters of the instruction "SETIO_PART":

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUB-MODULE	I, Q, M, D, L or constant	Hardware identifier of the IO module.
OFFSET	Input	INT	I, Q, M, D, L or constant	Number of the first byte to be written in the process image for the component (smallest possible value: 0).
LEN	Input	INT	I, Q, M, D, L or constant	Number of bytes to be written.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPWR_DAT" in the form DW#16#40xxxx00, if ERROR = TRUE.
ERROR	Output	BOOL	I, Q, M, D, L	Error display: ERROR = TRUE if an error occurs when "DPWR_DAT" is called.
OUTPUTS	InOut	VARIANT	I, Q, M, D	Source range for the data to be written: If the source range is greater than LEN, the first LEN bytes are transferred from OUT-PUTS.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameters STATUS and ERROR

See instruction "DPWR_DAT (Page 3126)".

RALRM: Receive interrupt

Description RALRM

Description

The instruction receives an interrupt with all corresponding information from an I/O module (centralized configuration) or from a module of a DP slave or PROFINET IO device and supplies this information to its output parameters.

The information in the output parameters contains the start information of the called OB as well as information of the interrupt source.

For a central configuration, the configuration of the data structure of the destination area AINFO corresponds to the data structure of PROFINET IO.

Call "RALRM" only within the interrupt OB started by the CPU operating system as a result of the I/O interrupt that is to be examined.

Note

If you call "RALRM" in an OB whose start event is not an I/O interrupt, the instruction will provide correspondingly reduced information in its outputs.
Make sure to use different instance DBs when you call "RALRM" in different OBs. If you evaluate data resulting from an "RALRM" call outside of the associated interrupt OB, you should moreover use a separate instance DB per OB start event.

Note

The interface of the "RALRM" instruction is identical to the "RALRM" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Calling RALRM

"RALRM" can be called in three operating modes (MODE parameter). These are explained in the table below.

MODE	RALRM ...
0	... shows the component that triggered the interrupt in the ID output parameter and writes TRUE in the NEW output parameter.
1	... writes all output parameters, independent of the interrupt triggering component.
2	... checks whether the component specified in the F_ID input parameter has triggered the interrupt. • If not, NEW = FALSE • If yes, NEW = TRUE and all other output parameters are written.

Parameters

The following table shows the parameters of the "RALRM" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module. The number is assigned automatically and is stored in the hardware configuration properties of the component or interface.
MLEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the interrupt information to be received in bytes. With MLEN = 0, all data specified at the AINFO parameter are read.
NEW	Output	BOOL	I, Q, M, D, L	A new interrupt was received.
STATUS (Page 3078)	Output	DWORD	I, Q, M, D, L	Error code
ID	Output	HW_IO	I, Q, M, D, L	Hardware identifier of the module from which the interrupt was received.

Parameter	Declaration	Data type	Memory area	Description
LEN	Output	UINT	I, Q, M, D, L	Length of the received interrupt information
TINFO (Page 3082)	InOut	VARIANT	I, Q, M, D, L	Destination area for OB start and management information
AINFO (Page 3100)	InOut	VARIANT	I, Q, M, D, L	Destination area for header information and additional interrupt information For AINFO , you should provide a length of at least MLEN bytes.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Note

If you select a destination area (Page 3107)TINFO or AINFO that is too short, RALRM cannot enter the full information.

Parameter STATUS

Description

The STATUS output parameter contains error information. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	• B#16#00, if no error • Function ID from DPV1-PDU: In the event of an error, B#16#80 is output (in the event of an error reading a data record B#16#DE and writing a data record B#16#DF). If no DPV1 protocol element is used, then B#16#C0 will be output.
STATUS[2]	Error_Decode	Location of the error ID
STATUS[3]	Error_Code_1	Error ID
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Field element STATUS[2]

STATUS[2] can have the following values:

Error_Decode (B#16#...)	Source	Meaning
00 to 7F	CPU	No error or no warning
80	DPV1	Error according to IEC 61158-6
81 to 8F	CPU	B#16#8x shows an error in the xth call parameter of the instruction.
FE, FF	DP profile	Profile-specific error

Field element STATUS[3]

STATUS[3] can have the following values:

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
00	00		No error, no warning
70	00	reserved, reject	Initial call; no active data record transfer
	01	reserved, reject	Initial call; data record transfer has started
	02	reserved, reject	Intermediate call; data record transfer already active
80	81		The system data type at parameter TINFO does not match the call environment of the instruction. The used system data type must match the organization block in the user program (example: for a time-delay interrupt OB you need the system data type TI_Delay).
	90	reserved, pass	Invalid logical start address
	92	reserved, pass	Illegal type for VARIANT pointer
	93	reserved, pass	The DP component addressed via ID or F_ID is not configured.
	96		The "RALRM (Page 3076)" cannot supply the OB start information, management information, header information, or additional interrupt information. For OBs 4x, 55, 56, 57, 82 and 83, you can use the "DPNRM_DG (Page 3139)" instruction to read the current diagnostic frame of the relevant DP slave asynchronously (address information from OB start information).
	A0	read error	Negative acknowledgment while reading the module
	A1	write error	Negative acknowledgment when writing to the module
	A2	module failure	DP protocol error at layer 2 (e.g., slave failure or bus problems)
	A3	reserved, pass	• PROFIBUS DP: DP protocol error with Direct-Data-Link-Mapper or User-Interface/User • PROFINET IO: General CM error
	A4	reserved, pass	Communication on PBUS+ disrupted
	A5	reserved, pass	–
	A7	reserved, pass	DP slave or module is occupied (temporary error
	A8	version conflict	DP slave or module reports non-compatible versions
	A9	feature not supported	Function is not supported by DP slave or module
	AA to AF	user specific	DP slave or module reports a manufacturer-specific error in its application. Please check the documentation from the manufacturer of the DP slave or module.
	B0	invalid index	Data record not known in module Illegal data record number ≥ 256

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	B1	write length error	Error at length specification: • With "RALRM (Page 3076)": length error in AINFO (Page 3100) • With "RDREC (Page 3069)": length error in MLEN • With "WRREC (Page 3071)": length error in LEN
	B2	invalid slot	The configured slot is not assigned.
	B3	type conflict	Actual module type does not match specified module type
	B4	invalid area	DP slave or module reports access to an invalid area
	B5	state conflict	DP slave or module not ready
	B6	access denied	DP slave or module denies access
	B7	invalid range	DP slave or module reports an invalid range for a parameter or value
	B8	invalid parameter	DP slave or module reports an invalid parameter
	B9	invalid type	DP slave or module reports an invalid type With "RDREC (Page 3069)": buffer too small (subsets cannot be read) With "WRREC (Page 3071)": buffer too small (subsets cannot be written)
	BA to BF	user specific	DP slave or module reports a manufacturer-specific error when accessing. Please check the documentation from the manufacturer of the DP slave or module.
	C0	read constrain conflict	With "WRREC (Page 3071)": the data can only be written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You can only write the data online with PG/PC. With "RDREC (Page 3069)": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. In this case, you can only read the data online with PG/PC.
	C1	write constrain conflict	The data of the previous write job on the module for the same data record have not yet been processed by the module.
	C2	resource busy	The module is currently processing the maximum possible number of jobs for a CPU.
	C3	resource unavailable	The required operating resources are currently occupied.
	C4		Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.
	C5		DP slave or module not available.
	C6		Data record transfer was canceled due to priority class cancellation
	C7		Job aborted due to warm or cold restart on the DP master

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	C8 to CF		DP slave or module reports a manufacturer-specific resource error. Please check the documentation from the manufacturer of the DP slave or module.
	Dx	user specific	DP slave specific. Refer to the description of the DP slave.
81	00 to FF		Error in the initial call parameter (with "RALRM (Page 3076)": MODE)
	00		Illegal operating mode
82	00 to FF		Error in the second call parameter
:	:		:
88	00 to FF		Error in the eighth call parameter (with "RALRM (Page 3076)": TINFO (Page 3082))
	01		Wrong syntax ID
	23		Quantity structure exceeded or target range too small
	24		Wrong range ID
	32		DB/DI no. out of user range
	3A		DB/DI no. is NULL for area ID DB/DI or specified DB/DI does not exist
89	00 to FF		Error in the ninth call parameter (with "RALRM (Page 3076)": AINFO (Page 3100))
	01		Wrong syntax ID
	23		Quantity structure exceeded or target range too small
	24		Wrong range ID
	32		DB/DI no. out of user range
	3A		DB/DI no. is NULL for area ID DB/DI or specified DB/DI does not exist
8A	00 to FF		Error in the 10th call parameter
:	:		:
8F	00 to FF		Error in the 15th call parameter
FE, FF	00 to FF		Profile-specific error

Field element STATUS[4]

With DPV1 errors, the DP master passes on STATUS[4] to the CPU and the instruction. Without DPV1 error, this value is set to 0, with the following exceptions for "RDREC":

- STATUS[4] contains the target range length from RECORD, if MLEN > the target range length from RECORD
- STATUS[4]=MLEN, if the actual data record length < MLEN < the target range length from RECORD
- STATUS[4]=0, if STATUS[4]> 255 would have to be set

In PROFINET IO, STATUS[4] has the value "0".

Parameter TINFO

Data structure of the destination area TINFO

The data structure of the destination area TINFO contains the start information of the organization block in which "RALRM" is currently being called.

The TINFO destination area can contain the start information with standard access or optimized access. The format of the start information in the TINFO destination area must always match the start information of the corresponding organization block.

- The start information of an OB with standard access is always stored in the first 20 bytes of the "Temp" section in the block interface. Use the "TI_Classic" data structure for this.
- The start information of an OB with optimized access is always written to the "Input" section. Use a data structure for the specific OB type for these OBs.

Changing the block access (standard/optimized) also changes the block interface.

The following table provides an overview of the data structures that are used depending on the organization block at the TINFO parameter.

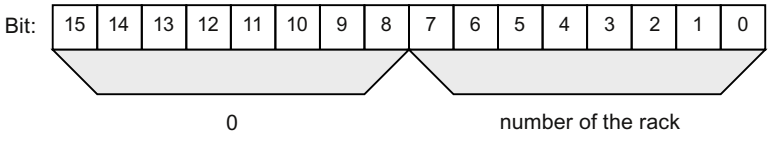
Name of the data structure		S7-1200 CPUs as of version	S7-1500 CPUs as of version	Data structure used for:
Data structure for organization blocks with standard access				
	TI_Classic	-	V1	Organization blocks without optimized block access.
Data structure for organization blocks with optimized block access				

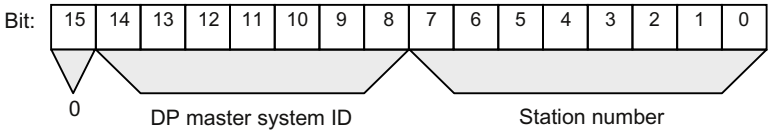
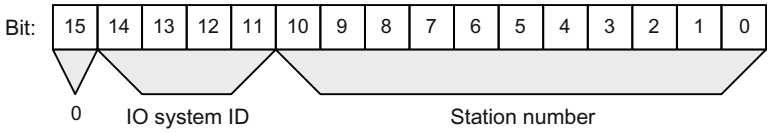
Name of the data structure		S7-1200 CPUs as of version	S7-1500 CPUs as of version	Data structure used for:
	TI_ProgramCycle	V2	V1	Cycle OB (program cycle)
	TI_Startup	V2	V1	Startup OB (startup)
	TI_Delay	V2	V1	Time delay interrupt
	TI_Cyclic	V2	V1	Cyclic interrupt OB
	TI_HWInterrupt	V2	V1	Hardware interrupt OB
	TI_TimeError	V2	V1	Time error OB
	TI_DiagnosticInterrupt	V2	V1	Diagnostic error interrupt OB
	TI_PlugPullModule	V2	V1	Pull/plug of modules OB
	TI_StationFailure	V2	V1	Rack or station failure OB
	TI_ProgIOAccessError	V2	V1	• Programming error OB • I/O access error OB
	TI_TimeOfDay	V2	V1	Time-of-day interrupt OB
	TI_SynchCycle	-	V1	Synchronous cycle interrupt OB
	TI_Submodule	V2	V1	• Status interrupt OB • Update interrupt OB • OB for manufacturer or profile-specific interrupt

Data structure for organization blocks with standard access

The following shows the TI_Classic data structure:

Parameter	Data type	Byte	Description
TI_Classic			Data structure for organization blocks with optimized block access.

Parameter	Data type	Byte	Description
Bytes 0 to 19: Start information of the OB in which "RALRM" is currently being called.*			
EV_CLASS	BYTE	0	Event class Example OB1: • Bits 0-3: ID of the event (1 = incoming event) • Bits 4-7: Event class (1 = Event class 1)
EV_NUM	BYTE	1	Event number (depending on OB type) Example OB1 (SCAN_1): • SCAN_1 = 1 for first call • SCAN_1 = 3 for all additional calls
PRIORITY	BYTE	2	Priority class
NUM	BYTE	3	OB number
TYP2_3	BYTE	4	Additional information
TYP1	BYTE	5	Different information is stored in the BYTES "TYP2_3" and "TYPE 1" depending on the OB type used. What these are is described in the documentation for the organization blocks. Example (OB1): • TYP2_3: OB1_RESERVED_1 (reserved) • TYP1: OB1_RESERVED_2 (reserved)
ZI1	WORD	6 to 7	Additional information Different information is stored in "ZI1" depending on the OB type used. What these are is described in the documentation for the organization blocks. Example (OB1): • ZI1: OB1_PREV_CYCLE (runtime of previous cycle in ms)
ZI2_3	DWORD	8 to 11	Additional information Different information is stored in "ZI2_3" depending on the OB type used. What these are is described in the documentation for the organization blocks. Example (OB1): • ZI2: OB1_MIN_CYCLE (minimum cycle time (ms) since the last startup) • ZI3: OB1_MAX_CYCLE (maximum cycle time (ms) since the last startup)
OB_DATE_TIME	DATE_AND_TIME (DT)	12 to 19	Date and time-of-day when the OB was called.
Bytes 20 and 21: Address information			
address	WORD	20 and 21	Address information like S7-300/400 CPUs: • In a central configuration, the rack number (0-31):  • For a distributed configuration with PROFIBUS DP: – DP master system ID (1-31) – Station number (0-127).

Parameter	Data type	Byte	Description
			 • In a distributed configuration with PROFINET IO: – The last two positions in the PROFINET IO system ID (0-15). To obtain the complete PROFINET IO system ID, you must add 100 (decimal) to it. – Station number (0-2047). 
Bytes 22 to 31: Management information			
slv_prfl	BYTE	22	Slave profile like S7-300/400 CPUs: • In a central configuration: 0 (data record 0 or data record 1) • In a distributed configuration: – Bit 0 to 3: Slave type - 0000: DP (structure data record 0) - 0001: DPS7 (structure data record 0 or data record 1) - 0010: DPS7 V1 (structure data record 0 or data record 1) - 0011: DPV1 (structure according to PROFIBUS DP standard) - 0100 – 0111: Reserved - 1000: PROFINET IO (structure according to PROFINET IO standard) - from 1001: Reserved – Bit 4 to 7: Profile type (reserved)
intr_type	BYTE	23	Interrupt info type like S7-300/400 CPUs: • In a central configuration: 0 • In a distributed configuration: – Bit 0 to 3: Interrupt info type - 0000: Interrupt comes from a configured remote module - 0001: Interrupt comes from a non-DPV1 slav / non IO device or a slot that is not configured - 0010: Interrupt generated in the CPU - from 0011: Reserved – Bit 4 to 7: Structure version - 0000: Initial - from 0001: Reserved
flags1	BYTE	24	Flags of the PROFIBUS DP master interface module / PROFINET IO controller interface module like S7-300/400 CPUs: • In a central configuration: 0 • In a distributed configuration: – Bit 0 = 0: Interrupt originating from an integrated interface module (PROFINET IO or PROFIBUS DP)

Parameter		Data type	Byte	Description
				– Bit 0 = 1: Interrupt originating from an external interface module (PROFINET IO or PROFIBUS DP) – Bit 1 to 7: Reserved
	flags2	BYTE	25	Flags of the PROFIBUS DP master interface module / PROFINET IO controller interface module like S7-300/400 CPUs: • In a central configuration: 0 • For a distributed configuration with PROFIBUS DP: – Bit 0: EXT_DIAG_FLAG from the diagnostics message frame, or 0 if this bit does not exist in the interrupt The bit is 1 if the DP slave is faulty. – Bit 1 to 7: Reserved • In a distributed configuration with PROFINET IO: – Bit 0: ARDiagnosisState or 0 if there is no information in the interrupt. The bit is 1 if the IO device is faulty. – Bit 1 to 7: Reserved
	id	UINT	26 and 27	Management information • In a central configuration: 0 • For a distributed configuration with PROFIBUS DP: PROFIBUS ID number as unique identifier of the PROFIBUS DP slave • In a distributed configuration with PROFINET IO: PROFINET IO device ID number as unique identifier of the PROFINET IO device
	manufacturer	UINT	28 and 29	Manufacturer ID (only in a distributed configuration with PROFINET IO).
	instance	UINT	30 and 31	Ident number of the instance (only in a distributed configuration with PROFINET IO).
*The start information depends on the OB used. You can find the start information of each OB type at the interface or in the documentation of the OB.				

Data structures for organization blocks with optimized block access

The data structures for organization blocks with optimized block access have the following format:

- Bytes 0 to 3: Format of the start information, class and number of the called OB (same structure for all data structures).
- Bytes 4 to 19: Optimized start information (structure depends on the OB type). The data in bytes 4 to 19 correspond to the structure and content of the corresponding OB interface.
- Bytes 20 to 31: In addition, the address and management information for certain OBs. The data in bytes 20 to 31 correspond to the data of 20 to 31 bytes of the TI_Classic data structure.

The format of the data structures is described in the following tables.

Table 11-42 **Cycle OB: Data structure TI_ProgramCycle**

Parameter		Data type	Byte	Description
TI_ProgramCycle				Data structure for cycle OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=1)
	OB_Nr	UINT	2	OB number (1...32767).
	Initial_Call	BOOL	4	= TRUE in the first call of this OB: • Transition from STOP or HOLD to RUN • After reloading
	Remanence	BOOL	5	= TRUE, if retentive data are available.

Table 11-43 **Startup OB: Data structure TI_Startup**

Parameter		Data type	Byte	Description
TI_Startup				Data structure for startup OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=100)
	OB_Nr	UINT	2	OB number (1...32767).
	LostRetentive	BOOL	4	=TRUE, if the contents of retentive data areas have been lost.
	LostRTC	BOOL	5	=TRUE, if the time-of-day of the real-time clock has been lost.

Table 11-44 **Time delay interrupt OB: Data structure TI_Delay**

Parameter		Data type	Byte	Description
TI_Delay				Data structure for time delay interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=20)
	OB_Nr	UINT	2	OB number (1...32767).
	Sign	WORD	4	User ID: Input parameter SIGN from the call of the "SRT_DINT (Page 3230)" instruction.

Table 11-45 **Cyclic interrupt OB: Data structure TI_Cyclic**

Parameter	Data type	Byte	Description
TI_Cyclic			Data structure for cyclic interrupt OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=30)
OB_Nr	UINT	2	OB number (1...32767).
Initial_Call	BOOL	4	= TRUE in the first call of this OB • At the transition from STOP or HOLD to RUN • After reloading
Event_Count	INT	6	Number of discarded start events since the last start of this OB.

Table 11-46 **Hardware interrupt OB: Data structure TI_HWInterrupt**

Parameter	Data type	Byte	Description
TI_HWInterrupt			Data structure for hardware interrupt OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=40)
OB_Nr	UINT	2	OB number (1...32767).
LADDR	HW_IO	4	Hardware identifier of the module that triggers the hardware interrupt.
USI	WORD	6	Identifier for future extensions (not user-relevant).
IChannel	USINT	8	Number of the channel that triggered the hardware interrupt.
EventType	BYTE	9	Identifier for the event type associated with the event triggering the interrupt (e.g., rising edge). You can find this ID in the description of the respective module.
address	WORD	20	See "address" parameter for the TI_Cassic. data structure
slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
id	UNIT	26	See "id" parameter for the TI_Cassic data structure.
manufacturer	UNIT	28	See "manufacturer" parameter for the TI_Cassic data structure.
instance	UNIT	30	See "instance" parameter for the TI_Cassic data structure.

Table 11-47 **Time error OB: Data structure TI_TimeError**

Parameter	Data type	Byte	Description
TI_TimeError			Data structure for time error OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=80)
OB_Nr	UINT	2	OB number (1...32767).
Csg_OBnr	OB_ANY	4	Number of the OB that was being executed when the time error occurred.
Fault_ID	BYTE	7	Error code. Possible values: • B#16#01: Cycle time exceeded • B#16#02: The requested OB is still being processed. • B#16#05: Expired time-of-day interrupt due to time jump. • B#16#06: Expired time-of-day interrupt upon re-entry into RUN after HOLD. • B#16#07: Overflow of the OB request buffer for the current priority class. • B#16#08: Isochronous mode interrupt - time error. • B#16#09: Interrupt loss due to high interrupt load • B#16#0B: Technology synchronization interrupt - time error
Csg_Prio	UNIT	8	Priority of the OB that was being executed when the time error occurred.

Table 11-48 **Diagnostic interrupt OB: Data structure TI_DiagnosticInterrupt**

Parameter	Data type	Byte	Description
TI_DiagnosticInterrupt			Data structure for diagnostic interrupt OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=82)
OB_Nr	UINT	2	OB number (1...32767).
LADDR	HW_ANY	4	Hardware identifier of the hardware object that triggered the diagnostic interrupt.
IO_State	WORD	6	Status of the hardware object: • Bit 0: Good • Bit 1: Disabled • Bit 2: Maintenance required • Bit 3: Maintenance demanded • Bit 4: Error • Bit 5: Not accessible • Bit 6: Qualified • Bit 7: Not available
Channel	UINT	8	Channel number
MultiError	BOOL	10	=TRUE, if there is more than one error.
address	WORD	20	See "address" parameter for the TI_Cassic. data structure
slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
id	UINT	26	See "id" parameter for the TI_Cassic data structure.
manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Table 11-49 **Pull/plug interrupt OB: Data structure** TI_PlugPullModule

Parameter	Data type	Byte	Description
TI_PlugPullModule			Data structure for pull/plug interrupt OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=83)
OB_Nr	UINT	2	OB number (1...32767).
LADDR	HW_IO	4	Hardware identifier of the affected module or submodule
Event_Class	BYTE	6	• B#16#38: (Sub)module plugged • B#16#39: (Sub)module pulled or not responding
Fault_ID	BYTE	7	Error code The following table shows the events that cause the start of the pull/plug interrupt OB. • With Event_Class = B#16#38 - module/submodule plugged: – B#16#54: Submodule plugged and matches configured submodule – B#16#55: Submodule plugged, but does not match configured submodule – B#16#56: Submodule plugged, but error in module parameter assignment – B#16#57: Submodule or module plugged, but with a fault or maintenance – B#16#58: Submodule access error corrected – B#16#61: Module plugged, module type OK – B#16#63 Module plugged but module type is incorrect – B#16#64: Module plugged but faulty (module ID cannot be read) – B#16#65: Module plugged, but error in module parameter assignment – B#16#66: Module responding again, load voltage error corrected • With Event_Class = B#16#39 - module/submodule pulled or not responding: – B#16#51: Module pulled – B#16#54: Submodule pulled – B#16#61: Module pulled or not responding – B#16#66: Module not responding, load voltage error
address	WORD	20	See "address" parameter for the TI_Cassic. data structure
slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
id	UINT	26	See "id" parameter for the TI_Cassic data structure.
manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Table 11-50 **Rack error OB: Data structure TI_StationFailure**

Parameter	Data type	Byte	Description
TI_StationFailure			Data structure for rack error OB

Parameter	Data type	Byte	Description
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=86)
OB_Nr	UINT	2	OB number (1...32767).
LADDR	HW_Device	4	Hardware identifier of the defective hardware object.
Event_Class	BYTE	6	• B#16#32: Activation of a station with the "D_ACT_DP" instruction • B#16#33: Deactivation of a station with the "D_ACT_DP" instruction • B#16#38: Outgoing event • B#16#39: Incoming event
Fault_ID	BYTE	7	Error code The error code is used to output the event that caused the start of the rack error OB. • Event_Class = B#16#39, FAULT_ID = C1: Expansion unit failure • Event_Class = B#16#38, FAULT_ID = C1: Expansion unit return • Event_Class = B#16#38, FAULT_ID = C2: Expansion unit return (outgoing expansion unit failure with discrepancy between expected and actual configuration) • Event_Class = B#16#39, FAULT_ID = C3: Distributed I/O: Failure of a DP master system • Event_Class = B#16#38, FAULT_ID = C4: Failure of a DP station • Event_Class = B#16#38, FAULT_ID = C5: Return of a DP station, but station faulty • Event_Class = B#16#38, FAULT_ID = C6: Expansion unit return, but error in module parameter assignment • Event_Class = B#16#38, FAULT_ID = C7: Return of a DP station, but error in module parameter assignment • Event_Class = B#16#38, FAULT_ID = C8: Return of a DP station, but discrepancy between expected and actual configuration • Event_Class = B#16#32/33, FAULT_ID = C9: Activation/deactivation of a DP slave with "D_ACT_DP" • Event_Class = B#16#39, FAULT_ID = CA: PROFINET IO system failure • Event_Class = B#16#39/38, FAULT_ID = CB: PROFINET IO station failure/ station return • Event_Class = B#16#38, FAULT_ID = CC: PROFINET IO station return with problem or maintenance • Event_Class = B#16#38, FAULT_ID = CD: PROFINET IO station return, expected configuration different from actual configuration • Event_Class = B#16#38, FAULT_ID = CE: PROFINET IO station return, error in module parameter assignment • Event_Class = B#16#32/33, FAULT_ID = CF: Activation/deactivation of a PROFINET IO device with the "D_ACT_DP" instruction • Event_Class = B#16#39/38, FAULT_ID = F8: Failure/return of some of the submodules of a PROFINET I-device

Parameter		Data type	Byte	Description
				• Event_Class = B#16#38, FAULT_ID = F9: Return of some of the submodules of a PROFINET I-device with a device configuration difference
	address	WORD	20	See "address" parameter for the TI_Cassic. data structure
	slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
	intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
	flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
	flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
	id	UINT	26	See "id" parameter for the TI_Cassic data structure.
	manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
	instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Table 11-51 **Programming error OB / I/O access error OB: Data structure TI_ProgIOAccessError**

Parameter	Data type	Byte	Description*
TI_ProgIOAccessError			Data structure for programming error OB and I/O access error OB

Parameter	Data type	Byte	Description*
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class: • =121 for programming error OB • =122 for I/O access error OB
OB_Nr	UINT	2	OB number (1...32767).
BlockNr	UINT	4	Number of the block in which the programming error occurred.
Reaction	USINT	6	• 0: Ignore error • 1: Substitute bad value • 2: Skip command
Fault_ID	BYTE	7	Error code • B#16#21: BCD conversion error • B#16#22: Area length error when reading • B#16#23: Area length error when writing • B#16#24: Range error when reading • B#16#25: Range error when writing • B#16#26: Error in timer no. • B#16#27: Error in counter no. • B#16#28: Read access to a byte, word or double word with a pointer whose bit address is not zero • B#16#29: Write access to a byte, word or double word with a pointer whose bit address is not zero • B#16#:30: Write access to a write-protected global DB • B#16#31: Write access to a write-protected instance DB • B#16#32: DB number error when accessing a global DB • B#16#33: DB number error when accessing an instance DB • B#16#34: Number error during FC call • B#16#35: Number error during FB call • B#16#42: I/O access error, reading • B#16#43: I/O access error, writing • B#16#3A: Access to a DB that is not loaded; the DB number is located in the permitted area • B#16#3C: Access to an FC that is not loaded; the FC number is within the permissible range • B#16#3D: Access to an instruction (SFC) that is not loaded; the SFC number is within the permissible range • B#16#3E: Access to an FB that is not loaded; the FB number is within the permissible range • B#16#3F: Access to an SFB that is not available; the SFB number is within the permissible range
BlockType	USINT	8	Type of block in which the error has occurred:

Parameter		Data type	Byte	Description*
				• OB: B#16#88 • FC: B#16#8C • FB: B#16#8E
	Area	USINT	9	Memory area in which the incorrect access occurred: • Local data: B#16#40 to 4E, 86, 87, 8E, 8F, C0 to CE • Process image input: B#16#01 • Process image output: B#16#02 • Technology DB: B#16#04 • I: B#16#81 • Q: B#16#82 • M: B#16#83 • DB: B#16#84, 85, 8A, 8B
	DBNr	DB_ANY	10	DB no. if AREA = DB (global DB or instance DB) Only relevant for programming error OB.
	Csg_OBNr	OB_ANY	12	OB number: • 121: Programming error OB • 122: I/O access error OB
	Csg_Prio	USINT	14	OB priority
	Width	USINT	15	Type of access during which the error occurred: • Bit: B#16#00 • Byte: B#16#01 • Word: B#16#02 • DWord: B#16#03 • LWord: B#16#04
* Only certain output values are possible depending on the OB (I/O access error OB or programming error OB) from which information is read.				

Table 11-52 **Time-of-day interrupt OB: Data structure TI_TimeOfDay**

Parameter		Data type	Byte	Description
TI_TimeOfDay				Data structure for time-of-day interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=10)
	OB_Nr	UINT	2	OB number (1...32767).
	CaughtUp	BOOL	4	=TRUE if an OB call was executed subsequently because time was adjusted ahead
	SecondTime	BOOL	5	=TRUE if an OB was called a second time because time was adjusted back. Note: SecondTime is set only once in situations in which the time is set back.

Table 11-53 **Isochronous mode interrupt OB: Data structure TI_SynchCycle**

Parameter	Data type	Byte	Description
TI_SynchCycle			Data structure for isochronous mode interrupt OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=61)
OB_Nr	UINT	2	OB number (1...32767).
Initial_Call	BOOL	4	= TRUE in the first call of this OB: • Transition from STOP or HOLD to RUN • After reloading
IO _System	USINT	5	Number of the distributed I/O system triggering the interrupt
Event_Count	INT	6	• = n: Number of lost cycles • = -1: An unknown number of cycles has been lost (e.g., because cycle has changed).
PIP_Input	BOOL	10	=TRUE: The associated process image of the inputs is up-to-date
PIP_Output	BOOL	11	=TRUE: The associated process image of the outputs was transferred to the outputs in good time after the last cycle
SyncCycle-Time	LTIME	16	Calculated cycle time

Table 11-54 **Status interrupt OB / update interrupt OB for manufacturer or profile-specific interrupt: Data structure TI_Submodule**

Parameter		Data type	Byte	Description
TI_Submodule				Data structure for status interrupt OB / update interrupt OB for manufacturer or profile-specific interrupt
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class • =55 for status interrupt OB • =56 for update interrupt OB • =57 for manufacturer or profile-specific interrupt OB
	OB_Nr	UINT	2	OB number (1...32767).
	LADDR	HW_IO	4	Hardware address of the component triggering the interrupt
	Slot	UINT	6	Slot number of the component triggering the interrupt
	Specifier	WORD	8	Interrupt specifier from the interrupt frame
	address	WORD	20	See "address" parameter for the TI_Cassic. data structure
	slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
	intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
	flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
	flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
	id	UINT	26	See "id" parameter for the TI_Cassic data structure.
	manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
	instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Parameter AINFO

Data structure of the destination area AINFO with interrupts from PROFIBUS DP

Byte	Meaning	
0 to 3	Header information, for exact description, see below	
4 to 63	Additional interrupt information: data for the respective interrupt:	
	distributed:	ARRAY[0] to ARRAY[59]

Structure of the header information with interrupts from PROFIBUS DP

Byte	Data type	Meaning	
0	BYTE	Length of the received interrupt information in bytes	
		central:	4 to 224
		distributed:	4 to 63

Byte	Data type	Meaning		
1	BYTE	central:	Reserved	
		distributed:	ID for the interrupt type	
			1: 2: 3: 4: 5: 6: 31	Diagnostics interrupt Process interrupt Pull interrupt Plug interrupt Status interrupt Update interrupt Failure of an expansion device, DP master system or DP station
			32 to 126:	Manufacturer-specific interrupt
2	BYTE	Slot number of the component that triggered the interrupt		
3	BYTE	central:	Reserved	
		distributed:	Specifier	
			Bits 0 and 1:	0: No further information; 1: Incoming event, fault at slot 2: Outgoing event, fault at slot cleared 3: Outgoing event, slot still faulty
			Bit 2:	Add_Ack
			Bits 3 to 7:	Sequence number

Data structure of the destination area AINFO with interrupts from PROFINET IO or central I/O devices

Byte	Meaning
0 to 25	Header information, for exact description, see below
26 to 1431	Additional interrupt information: Standardized diagnostic data for the respective interrupt: ARRAY[0] to ARRAY[1405] Note: The additional interrupt information may also be omitted.

Structure of the header information with interrupts from PROFINET IO or central I/O devices

Byte	Data type	Meaning
0 and 1	WORD	• Bits 0 to 7: Block type • Bits 8 to 15: Reserved
2 and 3	WORD	Block length
4 and 5	WORD	Version: • Bits 0 to 7: low byte • Bits 8 to 15: high byte

Byte	Data type	Meaning
6 and 7	WORD	ID for interrupt type: • 1: Diagnostics interrupt (incoming) • 2: Process interrupt • 3: Remove module interrupt • 4: Insert module interrupt • 5: Status interrupt • 6: Update interrupt • 7: Redundancy interrupt • 8: Controlled by supervisor • 9: Released by supervisor • 10: Configured module not inserted • 11: Return of the sub-module • 12: Diagnostics interrupt (outgoing) • 13: Slave-to-slave connection alarm • 14: Neighborhood change alarm • 15: Clock synchronization message (bus end) • 16: Clock synchronization alarm (device end) • 17: Network component alarm • 18: Time synchronization alarm (bus end) • 19 to 31: Reserved • 32 to 127: Manufacturer-specific interrupt • 128 to 65535: Reserved
8 to 11	DWORD	API (Application Process Identifier)
12 to 13	WORD	Slot number of the component triggering the interrupt (value range 0 to 65535)
14 to 15	WORD	Submodule slot number of the component triggering the interrupt (value range 0 to 65535)
16 to 19	DWORD	Module identification; specific information on the source of the interrupt

Byte	Data type	Meaning
20 to 23	DWORD	Submodule identification; specific information on the source of the interrupt
24 to 25	WORD	Interrupt specifier: • Bits 0 to 10: Sequence number (value range 0 to 2047) • Bit 11: Channel diagnostics: 0: No channel diagnostics available 1: Channel diagnostic information exists • Bit 12: Status of manufacturer-specific diagnostics: 0: No manufacturer-specific status information available 1: Manufacturer-specific status information available • Bit 13: Status of diagnostics for sub-module: 0: No status information available, all errors have been corrected 1: At least one item of channel diagnostics and/or status information is available • Bit 14: Reserved • Bit 15: Application relationship diagnosis state: – 0: None of the modules configured within this application relationship reports diagnostic information – 1: At least one of the modules configured in this AR is reporting diagnostic information

Structure of the additional interrupt information for interrupts from PROFINET IO or central I/O

The additional interrupt information for PROFINET IO depends on the format identifier. It can comprise multiple data blocks with the same or different format identifier. The following format identifiers are available:

- W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics
2 to n	BYTE	See manufacturer's manual.

- W#16#8000: Channel diagnostics
  Channel diagnostics is output in blocks of 6 bytes each. The additional interrupt information (without format identifier) is only output for disrupted channels.

Byte	Data type	Meaning	
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8000: Channel diagnostics	
2 to 3	WORD	Channel number of the component triggering the interrupt (value range: 0 to 65535): • W#16#0000 to W#16#7FFF: Channel number of the interface module/sub-module • W#16#8000: The generic substitute for the entire sub-module • W#16#8001 to W#16#FFFF: Reserved	
4	BYTE	Bits 0 to 2:	Reserved
		Bits 3 to 4:	Type of error: • 0: Reserved • 1: Incoming error • 2: Outgoing error • 3: Outgoing error, other errors present
		Bits 5 to 7:	Type of channel: • 0: Reserved • 1: Input channel • 2: Output channel • 3: Input/output channel

Byte	Data type	Meaning
5	BYTE	Data format: • B#16#00: Free data format • B#16#01: Bit • B#16#02: 2 bits • B#16#03: 4 bits • B#16#04: Byte • B#16#05: Word • B#16#06: Double word • B#16#07: 2 double words • B#16#08 to B#16#FF: Reserved
6 to 7	WORD	Type of error: • W#16#0000: Reserved • W#16#0001: Short circuit • W#16#0002: Undervoltage • W#16#0003: Overvoltage • W#16#0004: Overload • W#16#0005: Overtemperature • W#16#0006: Wire break • W#16#0007: High limit exceeded • W#16#0008: Low limit exceeded • W#16#0009: Error • W#16#000A to W#16#000F: Reserved • W#16#0010 to W#16#001F: Manufacturer-specific • W#16#0020 to W#16#00FF: Reserved • W#16#0100 to w#16#7FFF: Manufacturer-specific • W#16#8000: Device diagnostics available • W#16#8001 to W#16#FFFF: Reserved Not all channels support every error type. For detailed information, refer to the description of the diagnostic data for the specific device.

Note

The section from "channel number" to "type of error" can occur from 0 to n times.

- W#16#8001: MULTIPLE (different types of diagnostic information are transmitted). In this case, the additional interrupt information is transmitted as blocks of variable length.

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8001: Manufacturer-specific diagnostics and/or channel diagnostics
2 to 3	WORD	Block type
4 to 5	WORD	Block length
6	BYTE	Version: high byte
7	BYTE	Version: low byte
8 to 11	DWORD	API (only if low byte of version = 1)
12 to 13	WORD	Slot number
14 to 15	WORD	Subslot number
16 to 17	WORD	Channel number
18 to 19	WORD	Channel properties
20 to 21	WORD	Format identifier: • W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics • W#16#8000: Channel diagnostics • W#16#8002: Extended channel diagnostics • W#16#8003: Stepped extended channel diagnostics • W#16#8004 to W#16#80FF: Reserved
22 to n	BYTE	Data depend on the format identifier

Note

The section starting from "block type" can occur from 1 to n times.

- W#16#8002: Extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8002
2 to 3	Channel number
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information

- W#16#8003: Stepped extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8003
2 to 3	Channel number
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information
14 to 17	Qualified channel qualifier

- W#16#8100: Maintenance information

Byte	Meaning
0 to 1	Format identifier W#16#8100
2 to 3	Block type
4 to 5	Block length
6 to 7	Block version
8 to 9	Reserved
10 to 13	Maintenance status

Note

You can find more detailed information about the structure of the additional alarm information in the *Programming Manual SIMATIC PROFINET IO from PROFIBUS DP to PROFINET IO* and the current version of IEC 61158-6-10-1.

Destination area TINFO and AINFO

Destination area TINFO and AINFO

Depending on the respective OB in which "RALRM (Page 3076)" is called, the destination areas TINFO and AINFO are only partially written. Refer to the table below to find out which information is entered respectively.

Interrupt type	OB class	TINFO OB start information	TINFO Management information	AINFO Header information	AINFO Additional interrupt information	
Process interrupt	4x	Yes	Yes	Yes	central:	No
					distributed:	as supplied by PROFIBUS DP slave/PROFINET IO device
Status interrupt	55	Yes	Yes	Yes	Yes	Yes
Update interrupt	56	Yes	Yes	Yes	Yes	Yes
Manufacturer-specific interrupt	57	Yes	Yes	Yes	Yes	Yes

Interrupt type	OB class	TINFO OB start information	TINFO Management information	AINFO Header information	AINFO Additional interrupt information	
I/O redundancy error	70	Yes	Yes	No	No	No
Diagnostics interrupt	82	Yes	Yes	Yes	central:	Structure according to PROFINET IO standard
					distributed:	as supplied by PROFIBUS DP slave/PROFINET IO device
Insert/remove interrupt	83	Yes	Yes	Yes	central:	No
					distributed:	as supplied by PROFIBUS DP slave/PROFINET IO device
Special form of the remove module interrupt: Controlled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Special form of the insert module interrupt: Enabled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Unconfigured module inserted	83	Yes	Yes	Yes	PROFINET IO only	
Rack failure/ station failure	86	Yes	Yes	No	No	
... all other OBs		Yes	No	No	No	

D_ACT_DP: Enable/disable DP slaves

Description

Use the "D_ACT_DP" instruction to specifically deactivate and reactivate configured DP slaves/ PROFINET IO devices. In addition, you can determine whether each assigned DP slave or PROFINET IO device is currently activated or deactivated.

If you use the instruction to deactivate an IE/PB Link PN IO type of gateway, then all connected PROFIBUS DP slaves will also cease to function. These failures will be reported.

The instruction cannot be used on PROFIBUS PA field devices that are connected by a DP/PA link to a DP master system.

Note

As long as one or more "D_ACT_DP" jobs are active, you cannot load a changed configuration from the programming device to the CPU (within the scope of CiR).
If a changed configuration is being loaded from the programming device to the CPU during ongoing operation (CiR), the CPU will reject activation of a "D_ACT_DP" job.

Several runs through the cycle control point are required to perform the disabling or enabling job. Therefore, you cannot wait for the end of such a job in a programmed loop.

Functional description

"D_ACT_DP" is an instruction that works asynchronously, which means its execution extends over multiple calls. You start the job by calling D_ACT_DP with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

Application

If you configure DP slaves/PROFINET IO devices in a CPU which are not actually present or not currently required, the CPU will nevertheless continue to access these DP slaves/PROFINET IO devices at regular intervals. After the slaves are deactivated, further CPU accessing will stop. With PROFIBUS DP, the fastest possible DP bus cycle can be achieved in this manner and the corresponding error events no longer occur.

Examples

From a machine OEM's point of view, there are numerous device options possible in series production of machines. However, each delivered machine includes only one combination of selected options.

Every one of these possible machine options is configured as a DP slave/PROFINET IO device by the manufacturer in order to create and maintain a common user program having all possible options. Use "D_ACT_DP" to deactivate all DP slaves/PROFINET IO devices not present at machine startup.

A similar situation exists for machine tools having numerous tooling options available but actually using only a few of them at any given time. These tools are implemented as DP slaves/PROFINET IO devices. With "D_ACT_DP", the user program activates the tools currently needed and deactivates those required later.

Identification of a job

If you have started a deactivation or activation job and you call "D_ACT_DP" again before the job is complete, the behavior of the instruction depends on whether or not the new call involves the same job. If the input parameter LADDR matches, then the call will be interpreted as a follow-on call.

Deactivating DP slaves/PROFINET IO devices

When you deactivate a DP slave/PROFINET IO device with "D_ACT_DP", its process outputs are set to the configured substitute values or to 0 (safe state). The assigned DP master/ PROFINET IO controller does not continue to address this component. Deactivated DP slaves/ PROFINET IO devices are not identified as faulty or missing by the error LEDs on the DP master/PROFINET IO controller or CPU.

The process image of the inputs of deactivated DP slaves/PROFINET IO devices is updated with 0, that is, it is handled just as it is for failed DP slaves/PROFINET IO devices.

If you are using your program to directly access the user data of a previously deactivated DP slave/PROFINET IO device, the I/O access error OB is called and the corresponding start event is entered in the diagnostics buffer. If you attempt to access a deactivated DP slave/ PROFINET IO device via an instruction (such as "RD_REC (Page 3118)"), you will receive the same error information in RET_VAL as for an unavailable DP slave/PROFINET IO device.

Deactivating a DP slave/PROFINET IO device does not start the program error OB, even if its inputs or outputs belong to the system-side process image to be updated. Also there is no entry in the diagnostics buffer.

If a DP station/PNIO station fails after you have deactivated it with "D_ACT_DP", the operating system does not detect the failure.

Applies to PROFIBUS DP: If you wish to deactivate DP slaves functioning as transmitters in slave-to-slave communication, we recommend that you first deactivate the receivers (listeners) that detect which input data the transmitter is transferring to its DP master. Deactivate the transmitter only after you have performed this step.

Activating DP slaves/PROFINET IO devices

When you reactivate a DP slave/PROFINET IO device with "D_ACT_DP", this component is configured and assigned parameters by the associated DP master/PROFINET IO controller (as with the return of a failed DP station/PROFINET IO station). This activation is complete when the component is able to transfer user data.

Activating a DP slave/PROFINET IO device does not start the program error OB, even if its inputs or outputs belong to the system-side process image to be updated. Also there is no entry in the diagnostics buffer.

If you try to activate a DP slave/PROFINET IO device with "D_ACT_DP" that cannot be accessed (for example, because it was physically separated from the bus), the instruction returns the error code W#16#80A7 after expiration of the configured parameter assignment time for distributed I/O. The DP slave/PROFINET IO device is activated and the fact that the activated DP slave/PROFINET IO device cannot be accessed results in a corresponding display in the system diagnostics.

If the DP slave/PROFINET IO device is accessible again afterwards, this results in standard system behavior (for example, a call of the OB configured for this purpose).

Note

Activating a DP slave/PROFINET IO device may be time-consuming. If you wish to cancel a current activation job, start "D_ACT_DP" again with the same value for LADDR and MODE = 2. Repeat the call of "D_ACT_DP" with MODE = 2 until successful cancellation of the activation job is indicated by RET_VAL = 0.

If you wish to activate DP slaves which take part in the slave-to-slave communication, we recommend that you first activate the transmitters and then the receivers (listeners).

Parameters

The following table shows the parameters of the "D_ACT_DP" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Level-triggered control parameter REQ=1: Execute activation or deactivation
MODE	Input	USINT	I, Q, M, D, L or constant	Job identifier Possible values: • 0: Request information on whether the addressed component is activated or deactivated (output via RET_VAL parameter) • 1: Activate the DP slave/PROFINET IO device • 2: Deactivate the DP slave/PROFINET IO device
LADDR	Input	HW_DEVICE	I, Q, M, D, L or constant	Hardware identifier of the DP slave (HW_DPSlave)/PROFINET IO device (HW_Device) The number can be taken from the properties of the DP slave / PROFINET IO device in the Network view or from the "System constants" tab of the standard tag table. If both the identifier for the device diagnostics as well as the ID for operating state transitions are both specified there, you must use the code for the device diagnostics.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	Active code: • BUSY = 1: The job is still active. • BUSY = 0: The job was terminated.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	Job completed without error.
0001	The DP slave/PROFINET IO device is active (this error code is possible only with MODE = 0.)
0002	The DP slave/PROFINET IO device is deactivated (this error code is possible only with MODE = 0.)
7000	First call with REQ=0. The job specified with LADDR is not active; BUSY has the value "0".
7001	First call with REQ=1. The job specified with LADDR was initiated. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant). The activated job is still active; BUSY has the value "1".
8090	• You have not configured a module with the address specified in LADDR. • You operate your CPU as I-Slave and you have specified in LADDR an address of this I-Slave.
8092	The deactivation of the currently addressed DP slave/PROFINET IO device (MODE=2) cannot be canceled by being activated (MODE=1). Activate the component at a later time.
8093	No DP slave / PROFINET IO device that can be activated or deactivated is assigned to the address specified in LADDR.
8094	You have attempted to activate a device which is potential partner for a tool change port. However, another device is already activated on this tool change port at this time. The activated device will remain activated.
80A0	Error during the communication between the CPU and the IO controller.
80A1	Parameters could not be assigned for the addressed component. (This error code is only available when MODE = 1.) Note: "D_ACT_DP" returns this error information only if the activated slaves/devices of this component fails again during parameter assignment. If the parameter assignment of a single module was unsuccessful, "D_ACT_DP" returns the error information W#16#0000.
80A2	The addressed DP slave does not return an acknowledgment. (This error information is not available with PROFINET IO devices. The process is not time-monitored by PROFINET.)
80A3	The DP master/PROFINET IO controller concerned does not support this function.
80A4	The CPU does not support this function for external DP masters/PROFINET IO controller.
80A6	Slot error in the DP slave/PROFINET IO device; not all user data can be accessed (this error code is only available when MODE=1). Note: "D_ACT_DP" returns this error information only if the activated component fails again after parameter assignment and before the end of "D_ACT_DP". If only a single module is unavailable, "D_ACT_DP" returns the error information W#16#0000.
80A7	Activation of a non-accessible device.
80AA	Activation with errors in the DP slave/PROFINET IO device: Differences in the configuration
80AB	Activation with errors in the DP slave/PROFINET IO device: Parameter assignment error
80AC	Activation with errors in the DP slave/PROFINET IO device: Maintenance required
80C1	"D_ACT_DP" has been started and is being continued with another address (this error code is possible when MODE=1 and MODE=2).
80C3	• Temporary resource error: The CPU is currently processing the maximum possible activation and deactivation jobs (8). (This error code is available only when MODE = 1 and MODE = 2.) • The CPU is busy receiving a modified configuration. Currently you can not enable/disable DP slaves/ PROFINET IO devices.
80C5	DP: Jobs not collected by the user are discarded at restart.
80C6	PROFINET: Jobs not collected by the user are discarded at restart.

Error code* (W#16#...)	Explanation
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

ReconfigIOSystem: Reconfigure IO system

Description

You can use the "ReconfigIOSystem" instruction to transfer a data record to the PROFINET interface of the CPU. This data record ("CTRLREC" parameter) contains the following information for controlling the configuration:

- A list of optional IO devices to be activated
- A list of the partner ports to be specified, if you have set the "Partner is set in the user program" option in the port properties of IO devices

Note

The "ReconfigIOSystem" instruction uses the "D_ACT_DP" instruction internally in MODE 1 and MODE 3 to enable/disable IO devices. Therefore, observe the rules and notes in the description of this instruction.

See also "D_ACT_DP: Disable/enable distributed I/O devices (Page 1908)".

Functional description

"The "ReconfigIOSystem" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "ReconfigIOSystem" with REQ=1.

The output parameters STATUS and BUSY indicate the status of the job.

Parameters

The following table shows the parameters of the "ReconfigIOSystem" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Edge-triggered control parameter REQ=1: Perform data transfer
MODE	Input	UINT	I, Q, M, D, L or constant	You control how the instruction works with the MODE parameter. You can find a detailed functional description in the table below. Possible values: • 1: Disable all IO devices of the IO system for the conversion phase • 2: Reconfigure the IO system according to the data record settings (CTRLREC) • 3: Enable all IO devices of the IO system again after reconfiguration
LADDR	Input	HW_INTERFACE	I, Q, M, D, L or constant	Hardware identifier of the PROFINET interface (IO controller)
CTRLREC	Input	VARIANT	I, Q, M, D, L	Data record for controlling the actual configuration of the IO system
DONE	Output	BOOL	I, Q, M, D, L	0: Instruction not yet completed 1: Instruction completed
BUSY	Output	BOOL	I, Q, M, D, L	Active code: 0: Instruction active 1: Instruction completed
ERROR	Output	BOOL	I, Q, M, D, L	0: Instruction completed without error 1: Instruction completed with error
STATUS	Output	DWORD	I, Q, M, D, L	Result/error code
ERR_INFO	Output	WORD	I, Q, M, D, L	HW ID of the most recently determined IO device that caused an error.

You can find additional information on valid data types under "Overview of valid data types (Page 1908)".

Parameter MODE

The following values are possible for the MODE input parameter:

MODE	Description
1	All IO devices of the IO system can be disabled by calling the instruction with mode 1. The "ReconfigIOSystem" instruction uses the "D_ACT_DP" instruction internally. "ReconfigIOSystem" returns errors that are detected by D_ACT_DP in the following output parameters: • STATUS (error code) • ERR_INFO (hardware identifier of the IO device causing the error). In STATUS and ERR_INFO, the CPU enters the last determined error/HW identifier and in so doing overwrites an existing error code. For this reason, additional errors can be present besides the entered error.
2	For controlling the actual configuration of the IO system, the instruction transfers the data record to the PROFINET interface, which is addressed with LADDR. See the following section for information on the structure of the data record. See the following section about the STATUS parameter for error analysis.
3	All non-optional IO devices in the IO system and optional IO devices that are listed in the control data record CTRLREC are enabled. The optional IO devices that are **not** listed in the CTRLREC data record remain disabled. If IO devices that are part of docking units are listed in the CTRLREC control data record, the PN IO system reacts as follows: • IO devices of the docking units remain disabled when ReconfigIOSystem is called with Mode 3. This reaction corresponds to the reaction of a configuration without configuration-controlled IO devices. IO devices of docking units are disabled by default and must be enabled in the user program.

Structure of the control data record

You can use the control data record ("CTRLREC" parameter) to inform the PROFINET interface of the CPU which optionally configured IO devices are available in the real IO system configuration and which port interconnections are to be set.

This requires a configuration that allows IO system configuration to be adapted:

- The IO devices listed in the control data record must be enabled as "Optional IO device" (IO device properties: PROFINET interface [X1] > Advanced options > Interface options).
- The port interconnections listed in the control data record are only possible when the partner port is set to "Partner is set by the user program" for the appropriate ports.

The data type of parameter CTRLREC is "VARIANT". The "CTRLREC" control data record must have the following structure:

- Array from elements of the Word data type.

Below, the basic structural composition of "CTRLREC" is described with the Word element data type.

You read the required hardware identifiers directly in the "System constants" tab in the network view or in the device view. To do this, select the object in the network view (IO device) or in the device view (PROFINET interface).

Recommendation: Use the symbolic addressing of the HW identifiers by using the names of the HW identifiers.

Name	Data type	Comment
Version_High, Version_Low	Word	Version of the control data record(High Byte: 01 Low Byte: 00
Number_of_opt_Devices_used	Word	Number of optional IO devices that are used in the real IO system configuration. Optional IO devices that are not listed below remain disabled.
Activate_opt_Device_1	Word / Hw_Device	The optional IO devices that are present in the real configuration must be listed with their hardware identifiers. Use the system constant of the IO device object. Example: The IO device object has the name "IO-Device_4~IO-Device" and its type is "Hw_Device".
Activate_opt_Device_2	Word / Hw_Device	2nd optional IO device (e.g. with value 262)
...	...	...
Activate_opt_Device_n	Word / Hw_Device	nth optional IO device (e.g. with value 282)
Number_of_Port_Interconnections_used	Word	Number of port interconnections listed below. If you do not specify port interconnections, enter "0". For all the ports for which you have configured "Partner is set by the user program" and that are not listed below, the CPU uses the "Any partner" setting.
Port_Interconnection_1_Local	Word / Hw_Interface	1st port interconnection, HW identifier of local port Use the system constant of the port object. Example: The port object has the name "IO-Device_2~PROFINET_interface~Port_2" and its type is "Hw_Interface".
Port_Interconnection_1_Remote	Word / Hw_Interface	1st port interconnection, HW identifier of partner port
Port_Interconnection_2_Local	Word / Hw_Interface	2nd port interconnection, HW identifier of local port
Port_Interconnection_2_Remote	Word / Hw_Interface	2nd port interconnection, HW identifier of partner port
...	...	...
Port_Interconnection_n_Local	Word / Hw_Interface	nth port interconnection, HW identifier of local port
Port_Interconnectio_n_Remote	Word / Hw_Interface	nth port interconnection, HW identifier of local port

STATUS parameter

The STATUS output parameter contains error information. You can find a detailed list of possible error codes in the next section. If STATUS is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	B#16#DF: Error writing data record, otherwise B#16#00.
STATUS[2]	Error_Decode	B#16#80 is output for an error (corresponding to IEC 61158-6 in the context of reading and writing data records).

Field element	Name	Meaning
STATUS[3]	Error_Code_1	B#16#AA in the case of an error in the structure of the data record
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension: • B#16#00 Error in the data record (e.g. incorrect padding byte values (not equal to 0) • B#16#01 Reserved • B#16#02 At least one station number for an IO device in the data record is invalid (not configured, points to a non-optional IO device or has the value 0 for an IO controller) • B#16#03 At least one partner port specified in the data record is invalid: Examples: – Subslot address of a partner port is not available – Partner port is configured incorrectly (the correct setting is: "Partner is set by the user program") – Partner port belongs to an IO device that is disabled • B#16#10 Version of the CTRLREC control data record invalid (specified version is not supported) • B#16#11 The number of optional IO devices to be activated in the CTRLREC control data record is not supported • B#16#12 The number of specified interconnections ("Partner set in the user program") in the CTRLREC control data record is not supported • B#16#13 Internal conversion of the hardware identifier to device number failed. The ERR_INFO output parameter contains the hardware identifier of the device causing the error. • B#16#14 Consistency error: The length of the CTRLREC control data record does not match the information in the control data record. Example: 20 optional IO devices are specified, but the control data record only has a length of 10 bytes.

Error codes (STATUS parameter)

Error code	Description
16#0000_0000	Job completed without error
16#0070_0000	No job active
16#0070_0100	First call of the instruction
16#0070_0200	Following call of the instruction (instruction is still running, BUSY = 1)
16#0080_8000	MODE is not supported
16#0084_5100	Incorrect data type of CRTLREC data record. Use an Array of Word.
16#0080_9100	The LADDR parameter is not addressing any PROFINET interface (does not exist or wrong type, e.g. PROFIBUS interface). The PROFINET interface does not support configuration control of IO systems.
16#0080_Cx00	Temporary error, e.g., due to temporary lack of resources.

Error code	Description
16#DF80_AAxy	Error in the structure of the data record (MODE 2). For the meaning of "xy", see the definition of STATUS[4] Error_Code_2 above.
16#DF80_B600	Configuration control not possible because no optional IO device was configured and no port was assigned with "Partner is set in the user program". This configuration is a prerequisite for calling the instruction.
16#0080_9400 16#0080_A000 16#0080_A700 16#0080_AA00 16#0080_AB00 16#0080_AC00	Forwarded error codes of the internally called D_ACT_DP instruction. For the meaning of these error codes, see "D_ACT_DP (Page 3108)". The hardware identifiers of IO devices causing errors are entered in ERR_INFO (entry is consecutively overwritten by subsequent errors). If more than one IO device is involved, online diagnostics with STEP 7 is recommended. With IRT configuration: The device numbers of the IO devices should follow from the IO controller in ascending order of their topological interconnection, see here.

Others

RD_REC: Read data record from I/O

Description

Use the instruction to read the data record with the number RECNUM from the addressed module. You start the read process by assigning the value "1" to the input parameter REQ during the call. If the read process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY . If BUSY has the value "1", reading is not yet complete.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193). The data record read is entered in the destination area spanned by the RECORD parameter, providing the data transfer was free of errors.

Note

When you read out a data record with a number greater than one from an FM or a CP you have purchased prior to February 1997 (below referred to as "old modules"), "RD_REC" responds differently than it does in new modules. This special situation is covered in the section "Using old S7-300 FMs and CPs with data record numbers >1" (see below).

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not read any data records from the I/O modules in the user program with "RD_REC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: Set the interface of the DP master to "DPV1".

Parameters

The following table shows the parameters of the "RD_REC" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. In addition: the length of the data record actually transferred in bytes (possible values: +1 to +240), if the destination area is greater than the transferred data record and if no error occurred during the transfer.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
RECORD	Output	ANY	I, Q, M, D, L	destination area for the data record read. With asynchronous execution of "RD_REC", make sure that the actual parameters of RECORD have the same length information for all calls. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RECORD

Note

If you want to ensure that the entire data record is always read, select a destination area with a length of 241 bytes. If the data transfer is without error, RET_VAL contains the actual data record length.

Using old S7-300 FMs and CPs with data record numbers > 1

If you want to use the instruction "RD_REC" to read out a data record with a number greater than one from an old S7-300 FM or old S7-300 CP, remember the following points:

- If the destination area is greater than the actual length of the required data record, no data is entered in RECORD . RET_VAL is written with W#16#80B1.
- If the destination area is smaller than the actual length of the required data record, the CPU reads as many bytes beginning at the start of the record as are specified in the length information of RECORD and enters this number of bytes inRECORD . RET_VAL has the value "0".
- If the length specified in RECORD is the same as the actual length of the required data record, the CPU reads the data record and enters it in RECORD . RET_VAL is written with "0".

Parameter RET_VAL

- If an error has occurred while the function was being executed, the return value contains an error code.
- If no error occurred during the transfer, RET_VAL contains the following:
  - 0, if the entire destination area was filled with data from the selected data record (the data record can also be incomplete).
  - The length of the data record actually transferred in bytes (possible values: +1 to +240) if the destination area is greater than the transferred data record.

Note

If the general error W#16#8745 occurs, this only indicates that access to at least one byte of the process image was blocked. The data record was read by the module correctly and written to the I/O memory area.

When looking at the "real" error information (error codes W#16#8xyz) in the following table, we distinguish between two cases:

- Temporary errors (error codes W#16#80A2 to 80A3, 80Cx):
  This type of error can possibly be eliminated without user action, meaning it is helpful to call the instruction again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#809x, 80A0, 80A1, 80Bx):
  This type of error does not correct itself. A new call of the instruction will not be successful until you have eliminated the error. Example of a permanent error: Wrong length specification in RECORD (W#16#80B1).

Note

If you transfer data records to a DPV1 slave with "WR_REC (Page 3123)" or if you read data records from a DPV1 slave with RD_REC and if this DPV1 slave operates in DPV1 mode, the DP master evaluates the error information it received from the slave as follows:

If the error information lies within the range from W#16#8000 to W#16#80FF or W#16#F000 to W#16#FFFF, the DP master passes the error information to the instruction. If the error information is outside this range, the DP master passes the value W#16#80A2 to the instruction and suspends the slave.

For a description of the error information originating from DPV1 slaves, refer to STATUS[3] Parameter STATUS (Page 3078).

Parameter RET_VAL for WR_REC and RD_REC

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer started; BUSY has the value 1.	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value 1.	Distributed I/O
8090	Address specified at parameter ADDR is invalid.	-
8092	A type other than BYTE is specified in the ANY reference.	-
8093	This instruction is not permitted for the module selected by means of LADDR and IOID .	-
80A0	Negative acknowledgment when reading from the module: Module was removed during the reading process or is defective	With "RD_REC" only
80A1	Negative acknowledgment when writing to the module: Module was removed during the writing process or is defective	With "WR_REC (Page 3123)" only
80A2	• DP protocol error at layer 2 (for example, slave failure or bus problems) • For ET 200S, data record cannot be read in DPV0 mode.	Distributed I/O
80A3	DP protocol error with user interface/user	Distributed I/O
80A4	Communication on PROFIBUS disrupted.	-

Error code* (W#16#...)	Explanation	Restriction
80B0	• Instruction not possible for module type. • The module does not recognize the data record. • Data record number 241 not permitted. • With "WR_REC (Page 3123)", data records 0 and 1 are not permitted.	-
80B1	The length specified in parameter RECORD is incorrect.	• With "WR_REC (Page 3123)": Length incorrect • With "RD_REC" (only when using old S7-300 FMs and S7-300 CPs): specified length > data record length • With DPNRM_DG: specified length < data record length
80B2	The configured slot is not assigned.	-
80B3	Actual module type does not match specified module type	-
80B5	DP slave or module is not ready.	-
80B7	DP slave or module reports an invalid range for a parameter or value.	With "RD_REC" only
80C0	With "WR_REC (Page 3123)": the data can only be written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You can only write the data online with PG/PC. With "RD_REC": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. In this case, you can only read the data online with PG/PC. With "DPNRM_DG (Page 3139)": There are no diagnostic data available.	With "WR_REC (Page 3123)" or "RD_REC" or "DPNRM_DG (Page 3139)"
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	-
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available.	Distributed I/O
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O

Error code* (W#16#...)	Explanation	Restriction
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

WR_REC: Write data record to I/O

Description

Use the instruction "WR_REC" to transfer the data record RECORD to the addressed module.

The data to be transferred are read from the RECORD parameter during the first call. If the transfer of the data record takes longer than the duration of one call, the contents of the RECORD parameter are no longer relevant for the subsequent instruction calls (for the same job).

You start the writing process by assigning the value "1" to the input parameter REQ during the call. If the writing process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY. If BUSY has the value "1", writing is not yet complete.

Parameters

The following table shows the parameters of the instruction "WR_REC":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	REQ = 1: Write request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number
RECORD	Input	VARIANT	I, Q, M, D, L	Data record
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet completed.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

See also: RD_REC: Read data record from I/O (Page 3118)

Note

If the general error W#16#8544 occurs, it only indicates that access to at least one byte of the I/O memory area containing the data record was denied. The data transfer was continued.

DPRD_DAT: Read consistent data of a DP standard slave

Description

Use the instruction "DPRD_DAT" to read out consistent data from an I/O module.

The instruction can be used for the central modules as well as for the DP standard slaves and PROFINET IO devices.

You require "DPRD_DAT" because you can only read out a maximum of four continuous bytes using the load commands that access the I/O or the process image input table. If required, you can also read consistent data via the process image of the inputs. Refer to the related documentation to find out if your CPU supports this functionality. For additional information on consistent data of a DP standard slave/PROFINET IO device, refer to Section "Data consistency (Page 3608)".

If necessary, the instruction "DPRD_DAT" can also be used for a data area of 1 byte or larger. For information on the maximum length of the data, refer to the documentation of your CPU (e.g. 64 bytes for an S7-1214).

- Use the parameter LADDR to select the module of the DP normslave/PROFINET IO device. If an access error occurs, the error code W#16#8090 is output.
- Use the RECORD parameter to define the target range of the read data:
  - The target range has to be at least as long as the inputs of the selected module. Only the inputs are transferred, the other bytes are not considered. If you read from a DP standard slave with a modular configuration or with several DP identifiers, you can only access the data of a module of the configured hardware identifier per "DPRD_DAT" call. If you select a target range that is too small, the error code 80B1 is output at the RET_VAL parameter.
  - All bit strings and all integers can be used as data type. It is also possible to use these data types in a data structure of the ARRAY type . The data type STRING is not supported.
- If there was no error during the data transmission, the data that have been read are entered in the target range defined at the RECORD parameter.

Parameters

The following table shows the parameters of the instruction "DPRD_DAT":

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Hardware ID of the module from which the data is to be read. The hardware ID can be found in the properties of the module in the device view or system constants.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	VARIANT	I, Q, M, D, L	Destination area for the user data that were read.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	• You have not configured a module for the specified hardware identifier or • You have ignored the restriction concerning the length of consistent data, or • you have not specified a hardware identifier as an address at parameter LADDR .
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.
8093	No DP module/PROFINET IO device from which you can read consistent data exists for the hardware identifier specified in LADDR . This error code also occurs if the module addressed by means of LADDR does not have inputs.
80A0	An access error was detected when accessing the I/O.
80B1	The length of the specified target range at parameter RECORD is shorter than the configured user data length.
80C0	The data have not been read yet.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Parameter STATUS (Page 3078)

DPWR_DAT: Write consistent data of a DP standard slave

Description

Use the instruction "DPWR_DAT" to transfer the data at the parameters RECORD consistently to the addressed module of the central module or of the DP standard slave/PROFINET IO device and if necessary to the process image (i.e. if the affected address range of the DP standard slaves is projected as consistency range in a process image).

You require "DPWR_DAT" because you can only write a maximum of four continuous bytes using the transfer commands that access the I/O or process image output. If required, you can also write consistent data via the process image outputs. Refer to the related documentation to find out if your CPU supports this functionality. Do not use both possibilities concurrently when writing consistent data: Either use "DPWR_DAT" or write via the process image output table. For additional information on consistent data of a DP standard slave/PROFINET IO device, refer to Section "Data consistency (Page 3608)". If the DP standard slave has a modular design, you can only access one module of the DP slave.

Caution

I/O access

When using "DPWR_DAT", avoid accessing I/O areas that have process image partitions with OB6x connections (isochronous mode interrupts) assigned to them.

If necessary, the instruction "DPRD_DAT" can also be used for a data area of 1 byte or larger. For information on the maximum length of the data, refer to the documentation of your CPU (e.g. 64 bytes for an S7-1214).

- Use the LADDR parameter to select the DP standard slave / PROFINET IO device. If an access error occurs on the addressed module, the error code 8090 is output.
- Use the RECORD parameter to define the source range of the data to be written:
  - The source range has to be at least as long as the outputs of the selected module. Only the outputs are transferred, the other bytes are not considered. If the source range at parameter RECORD is longer than the outputs of the configured module, only the data up to the maximum length of the outputs is transferred. If the source range at parameter RECORD is shorter than the outputs of the configured module, the error code 80B1 is output.
  - The following data types can be used: Byte, Char, Word, LWord, DWord, Int, UInt, USInt, SInt, LInt, ULInt, DInt, UDInt. The use of these data types in a data structure of the type ARRAY or STRUCT is also permissible.
  - The data type STRING is not supported.

The data is transferred synchronously, that is, the write process is completed when the instruction is completed.

Parameters

The following table shows the parameters of the instruction "DPWR_DAT":

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Hardware ID of the module to which the data is to be written. The hardware ID can be found in the properties of the module in the device view or system constants.
RECORD	Input	VARIANT	I, Q, M, D, L	Source area for the user data to be written.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	• You have not configured a module for the specified hardware identifier or • You have ignored the restriction concerning the length of consistent data, or • you have not specified a hardware identifier at parameter LADDR .
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.
8093	No DP module /PROFINET IO device to which you can write consistent data exists at the HW ID specified in LADDR . This error code also occurs if the DP standard slave / PROFINET IO device addressed via LADDR does not have outputs.
80A1	Access error detected while I/O devices were being accessed.
80B1	The length of the specified source range at the RECORD parameter is shorter than the outputs of the configured DP standard slave / PROFINET IO device.
80C1	The data of the previous write job have not yet been processed by the DP standard slave / PROFINET IO device.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Parameter STATUS (Page 3078)

iDevice / iSlave

RCVREC: Receive data record

Description

An I-device can receive a data record from a superordinate controller. The receipt takes place in the user program with the instruction "RCVREC" (receive record).

The instruction has the following operating modes:

- Check whether the I-device has a request for a data record receipt.
- Make the data record available to the output parameters.
- Send an answer to the superordinate controller.

You can determine the operating mode executed by the instruction using the input parameter MODE (see below).

The I-device must be in the RUN or STARTUP mode.

With MLEN, you specify the maximum number of bytes you want to receive. The selected length of the target range RECORD should have at least the length of MLEN bytes.

If a data record was received (MODE=1 or MODE=2), the output parameter NEW will indicate that the data record was stored in RECORD. Note that RECORD has a sufficient length. The output parameter LEN contains the actual length of the data record received in bytes.

Set CODE1 and CODE2 to zero for the positive answer to the superordinate controller. If the received data record is to be rejected, enter the negative answer to the superordinate controller in Error Code 1 of the CODE1 and in Error Code 2 of the CODE2.

Note

If the I-device has received a request for a data record receipt, you must recognize the delivery of this request within a certain duration. After recognition, you must send an answer to the superordinate controller within this time period. Otherwise the I-device will experience a timeout error which causes the operating system of the I-device to send a negative answer to the superordinate controller. For information on the value for the time period, refer to the specifications of your CPU.

The output parameter STATUS receives the error information after the occurrence of an error.

Operating modes

You can determine the operating mode of the instruction "RCVREC" with the input parameter MODE. This step will be explained in the following table.

MODE	Meaning
0	Check whether a request for a data record receipt exists If a data record from a higher-level controller exists on the I-device, the instruction will only write the output parameters NEW, SLOT, SUBSLOT, INDEX and LEN. If you call the instruction several times with MODE=0, then the output parameter will only refer to one and the same request.
1	Receiving a data record for any subslot of the I-device If a data record from a superordinate controller exists on the I-device for any subslot of the I-device, the instruction will write the output parameter and transfer the data record to the parameter RECORD.
2	Receiving a data record for a specific subslot of the I-device If a data record from a superordinate controller exists on the I-device for a specific subslot of the I-device, the instruction will write the output parameter and transfer the data record to the parameter RECORD.
3	Sending a positive answer to the superordinate controller The instruction checks the request of the superordinate controller to receive a data record, accepts the existing data record and sends a positive acknowledgment to the superordinate controller.
4	Sending a negative answer to the superordinate controller The instruction checks the request of the superordinate controller to receive a data record, rejects the existing data record and sends a negative acknowledgment to the superordinate controller. Enter the reason for the rejection in the input parameters CODE1 and CODE2.

Note

After the receipt of a data record (NEW=1) you must call "RCVREC" twice to ensure complete processing. You must do this in the following order:

- First call with MODE=1 or MODE=2
- Second call with MODE=3 or MODE=4

Parameters

The following table shows the parameters of the "RCVREC" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Subslot in the transfer area of the I-device for the data record to be received (only relevant for MODE=2). The high word is always set to zero.
MLEN	Input	INT*	I, Q, M, D, L or constant	Maximum length of the data record to be received in bytes
CODE1	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 1 (for-MODE=4)
CODE2	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 2 (for-MODE=4)

Parameter	Declaration	Data type	Memory area	Description
NEW	Output	BOOL	I, Q, M, D, L	• MODE = 0: New data record was received • MODE=1 or 2: Data record was transferred to RECORD
STATUS	Output	DWORD	I, Q, M, D, L	Error information
SLOT	Output	HW_SUB-MODULE	I, Q, M, D, L	Identical to F_ID
SUBSLOT	Output	HW_SUB-MODULE	I, Q, M, D, L	Identical to F_ID
INDEX	Output	UINT	I, Q, M, D, L	Number of the data record received
LEN	Output	UINT	I, Q, M, D, L	Length of the data record received
RECORD	InOut	VARIANT	I, Q, M, D, L	Target range for the data record received. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 Byte 100). The omission of an explicit DB number is not permitted for S7-300 CPUs and results in an error message in the user program.
* Use the data type UINT in the STL programming language.				

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter STATUS

For interpretation of the parameter STATUS refer to section: Parameter STATUS (Page 3078)

PRVREC: Make data record available

Description

An I-device can receive a request from a superordinate controller to make a data record available. The I-device makes the data record available in the user program with the instruction "PRVREC" (provide record).

The instruction has the following operating modes:

- Check whether the I-device has a request for making a data record available.
- Transfer the requested data record to the superordinate controller.
- Sending an answer to the superordinate controller.

You can determine the operating mode executed by the instruction using the input parameter MODE (see below).

The I-device must be in the RUN or STARTUP mode.

Enter the maximum number of bytes the data record to be sent should have with LEN. The selected length of the target range RECORD should have at least the length of LEN bytes.

If a request to make a data record available exists, (MODE=0), the output parameter NEW will be set to TRUE.

If the request for making a data record available is accepted, write RECORD for the positive answer to the superordinate controller with the requested data record and write zero for CODE1 and CODE2. If the request for making a data record available is to be rejected, enter the negative answer to the superordinate controller in Error Code 1 of the CODE1 and in Error Code 2 of theCODE2.

Note

If the I-device has received a request for making a data record available, you must recognize the delivery of this request within a certain time period. After recognition, you must send an answer to the superordinate controller within this time period. Otherwise the I-device will experience a timeout error which causes the operating system of the I-device to send a negative answer to the superordinate controller. For information on the value for the time period, refer to the specifications of your CPU.

The output parameter STATUS receives the error information after the occurrence of an error.

Operating modes

You can determine the operating mode of the instruction "PRVREC" with the input parameter MODE. This step will be explained in the following table.

MODE	Meaning
0	Check whether a request for making a data record available exists If a request from a higher-level controller for making a data record available exists on the I-device, the instruction will only write the output parameters NEW, SLOT, SUBSLOT, INDEX and RLEN. If you call the instruction several times with MODE=0, then the output parameter will only refer to one and the same request.
1	Receiving a request for making a data record available for any subslot of the I-device If such a request from a superordinate controller for any subslot of the I-device exists on the I-device, the instruction will write the output parameter.
2	Receiving a request for making a data record available for a specific subslot of the I-device If such a request from a superordinate controller for a specific subslot of the I-device exists on the I-device, the instruction will write the output parameter.
3	Make the data record available and send a positive answer to the superordinate controller The instruction checks the request of the superordinate controller to make a data record available, makes the request data record available to RECORD and sends a positive acknowledgement to the superordinate controller.
4	Sending a negative answer to the superordinate controller The instruction checks the request of the superordinate controller to make a data record available, rejects this request and sends a negative acknowledgement to the superordinate controller. Enter the reason for the rejection in the input parameters CODE1 and CODE2.

Note

After the receipt of a request (NEW=1) you must call the instruction twice to ensure complete processing. You must do this in the following order:

- First call with MODE=1 or MODE=2
- Second call with MODE=3 or MODE=4

Parameters

The following table shows the parameters of the "PRVREC" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Subslot in the transfer area of the I-device for the data record to be sent (only relevant for MODE=2). The high word is always set to zero.
CODE1	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 1 (for-MODE=4)
CODE2	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 2 (for-MODE=4)
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the data record to be sent in bytes
NEW	Output	BOOL	I, Q, M, D, L	The new data record was requested by the superordinate controller.
STATUS	Output	DWORD	I, Q, M, D, L	Error information
SLOT	Output	HW_SUBMODULE	I, Q, M, D, L	Identical to F_ID
SUBSLOT	Output	HW_SUBMODULE	I, Q, M, D, L	Identical to F_ID
INDEX	Output	UINT	I, Q, M, D, L	Number of the data record to be sent
RLEN	Output	UINT	I, Q, M, D, L	Length of the data record to be sent
RECORD	InOut	VARIANT	I, Q, M, D, L	Data record made available Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 Byte 100). The omission of an explicit DB number is not permitted for S7-300 CPUs and results in an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter STATUS

For interpretation of the parameter STATUS refer to section: Parameter STATUS (Page 3078)

PROFIBUS

DPSYC_FR: Synchronize DP slaves / Freeze inputs

Description

You use the instruction to synchronize one or more groups of DP slaves.

The function involves sending one of the control commands below, or a combination of them, to the relevant groups:

- SYNC (simultaneous output and freezing of output states on the DP slaves)
- UNSYNC (cancels the SYNC control command)
- FREEZE (freeze the input states on the DP slaves and read in the frozen inputs)
- UNFREEZE (cancels the FREEZE control command)

Before you send the control commands listed above, you must assign the DP slaves to groups per configuration. You must know which DP slave is assigned to which group, with which number and know the reactions of the various groups to SYNC/FREEZE.

Note

Note that the control commands SYNC and FREEZE also remain valid when you perform a warm/cold restart.

Please note also that you may initiate only one SYNC/UNSYNC job or only one FREEZE/ UNFREEZE job at a time.

Functional description

"DPSYC_FR" works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "DPSYC_FR" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

Identification of a job

If you have triggered a SYNC/FREEZE job and called "DPSYC_FR" again before the first job was completed, the response of the instruction depends on whether the new call is for the same job. If the input parameters LADDR, GROUP and MODE match, the call is interpreted as a follow-on call.

Writing outputs of DP modules

The writing of outputs of DP modules is triggered as follows:

- By transfer commands to the DP I/O
- By writing the process image of the outputs to the modules (by the operating system at the end of OB 1 or by calling the instruction "UPDAT_PO (Page 3064)")
- Calling the "DPWR_DAT (Page 3126)" instruction.

In normal operation, the DP master transfers the output bytes cyclically (within the cycle of the PROFIBUS DP bus) to the outputs of the DP slaves.

If you want to have certain output data (possibly distributed on several slaves) applied to the outputs to the process at exactly the same time, you can send the SYNC control command to the relevant DP master using the "DPSYC_FR" instruction.

What are the effects of SYNC?

With the SYNC control command, the DP slaves of the selected groups are switched to Sync mode. In other words, the DP master transfers the current output data and instructs the DP slaves involved to freeze their outputs. With the following output message frames, the DP slaves enter the output data in an internal buffer and the state of the outputs remains unchanged.

Following each SYNC control command, the DP slaves of the selected groups apply the output data of their internal buffer to the outputs on the process.

The outputs are only updated cyclically again when you send the UNSYNC control command using the "DPSYC_FR" instruction.

Note

If the DP slaves of the selected group(s) are not currently connected to the network or have failed when the control command was sent, they will not be switched to SYNC mode. This information will not be communicated in the return value of the instruction.

Reading input data of DP modules

The input data of the DP modules are read as follows:

- Using load commands to the DP I/O
- When the process image of the inputs is updated (by the operating system at the start of OB 1 or by calling the "UPDAT_PI (Page 3062)" instruction)
- By calling the "DPRD_DAT (Page 3124)" instruction.

In normal operation, the DP master receives this input data cyclically (within the cycle of the PROFIBUS DP bus) from its DP slaves and makes them available to the CPU.

If you want to have certain input data (possibly distributed on several slaves) to be read from the process at exactly the same time, send the FREEZE control command to the relevant DP master using the "DPSYC_FR" instruction.

What are the effects of FREEZE?

With the FREEZE control command, the DP slaves involved are switched to Freeze mode, in other words the DP master instructs the DP slaves to freeze the current state of the inputs. It then transfers the frozen data to the input area of the CPU.

Following each FREEZE control command, the DP slaves freeze the state of their inputs again.

The DP master only receives the current state of the inputs cyclically again after you have sent the UNFREEZE control command with the "DPSYC_FR" instruction.

Note

If the DP slaves of the selected group(s) are not currently connected to the network or have failed when the control command has been sent, they will not be switched to FREEZE mode. This information will not be communicated in the return value of the instruction.

Data consistency

Because the DPSYC_FR functions are asynchronous and can be interrupted by higher priority classes, you should make sure that the process images are consistent with the actual inputs and outputs of the I/O when using the "DPSYC_FR" instruction.

This is guaranteed if you comply with the following consistency rules:

- Define suitable process image partitions for the "SYNC outputs" and the "FREEZE inputs" (only possible on the S7-400). Call the "UPDAT_PO (Page 3064)" instruction immediately before each first call of a SYNC job. Call the "UPDAT_PI (Page 3062)" instruction immediately after the respective last call of a FREEZE job.
- As an alternative: Use only direct I/O access for outputs involved in a SYNC job and for inputs involved in a FREEZE job. Do not write to these outputs when a SYNC job is active and do not read in these inputs when a FREEZE job is active.

Use of DPWR_DAT and DPRD_DAT

If you use the "DPWR_DAT (Page 3126)" instruction, it must be complete before you send a SYNC job for the outputs involved.

If you use the "DPRD_DAT (Page 3124)" instruction, it must be complete before you send a FREEZE job for the inputs involved.

Startup and "DPSYC_FR"

The user alone must take responsibility for sending the SYNC and FREEZE control commands in the startup OBs.

If you want the outputs of one or more groups to be in SYNC mode when the user program starts, you must initialize these outputs during startup and completely execute the "DPSYC_FR" instruction with the SYNC control command.

If you want the inputs of one or more groups to be in FREEZE mode when the user program starts, you must execute the "DPSYC_FR" instruction with the FREEZE control command completely for these inputs during startup.

Parameters

The following table shows the parameters of the "DPSYC_FR" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Level-triggered control parameter REQ = 1: Triggering of the SYNC/FREEZE job
LADDR	Input	HW_DPMASTER	I, Q, M, D, L or constant	Hardware identifier of the DP master interface The number can be taken from the properties of the DP master interface in the Network view or from the "System constants" tab of the standard tag table.
GROUP	Input	BYTE	I, Q, M, D, L or constant	Group selection Bit 0 = 1: group 1 selected Bit 1 = 1: group 2 selected : Bit 7 = 1: group 8 selected You can select several groups per job. The value B#16#0 is invalid.

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID (coding complies with EN 50 170 Volume 2, PROFIBUS) Bit 0: reserved (value 0) Bit 1: reserved (value 0) Bit 2: • = 1: UNFREEZE will be executed • = 0: no meaning Bit 3: • = 1: FREEZE will be executed • = 0: no meaning Bit 4: • = 1: UNSYNC will be executed • = 0: no meaning Bit 5: • = 1: SYNC will be executed • = 0: no meaning Bit 6: reserved (value 0) Bit 7: reserved (value 0) Possible values: • with exactly one ID per job: – B#16#04 (UNFREEZE) – B#16#08 (FREEZE) – B#16#10 (UNSYNC) – B#16#20 (SYNC) • with more than one ID per job: – B#16#14 (UNSYNC, UNFREEZE) – B#16#18 (UNSYNC, FREEZE) – B#16#24 (SYNC, UNFREEZE) – B#16#28 (SYNC, FREEZE)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. Make sure that you evaluate RET_VAL each time the block has been executed.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The SYNC/FREEZE job is not yet complete.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. For a description of this error information, refer to STATUS[3], STATUS (Page 3078) parameter.

Error code* (W#16#...)	Explanation
0000	Job completed without error.
7000	First call with REQ=0. The job specified with LADDR, GROUP and MODE is not active; BUSY has the value 0.
7001	First call with REQ=1. The job specified with LADDR, GROUP and MODE was triggered; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). The activated SYNC/FREEZE job is still running; BUSY has the value 1.
8090	The module selected with LADDR is not a DP master.
8093	This instruction is not permitted for the module selected with LADDR (configuration or version of the DP master).
8094	GROUP parameter is incorrect
8095	MODE parameter is incorrect
80A4	Communication on PROFIBUS disrupted.
80B0	The group selected with GROUP is not configured.
80B1	The group selected with GROUP is not assigned to this CPU.
80B2	The SYNC job specified with MODE is not permitted on the group selected with GROUP.
80B3	The FREEZE job specified with MODE is not permitted on the group selected with GROUP .
80C2	Temporary shortage of resources on the DP master: The DP master is currently processing the maximum number of jobs for a CPU.
80C3	This SYNC/UNSYNC job cannot be activated at present because only one SYNC/UNSYNC job can be triggered at a time. Check your user program.
80C4	This FREEZE/UNFREEZE job cannot be activated at present because only one FREEZE/UNFREEZE job can be triggered at a time. Check your user program.
80C5	Short circuit directly at DP interface
80C6	Job aborted due to I/O disconnection by CPU
80C7	Job aborted due to warm or cold restart on the DP master
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

DPNRM_DG: Read diagnostics data from a DP slave

Description

You use the "DPNRM_DG" instruction to read the current diagnostic data of a DP slave in the form specified in EN 50 170 Volume 2, PROFIBUS.

Refer to the following table for the basic structure of the slave diagnostic data and to the manuals of the DP slaves for further information.

Byte	Meaning
0	Station status 1
1	Station status 2
2	Station status 3
3	Master station number
4	Vendor ID (high byte)
5	Vendor ID (low byte)
6 ...	Additional slave-specific diagnostic information

The data that has been read is entered in the destination area indicated by RECORD following without error data transfer. You start the read process by assigning the value "1" to the REQ input parameter when the "DPNRM_DG" instruction is called.

Functional description

The reading process is executed asynchronously, in other words, it can extend over several calls. The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

Parameters

The following table shows the parameters of the "DPNRM_DG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_DPSLAVE	I, Q, M, D, L or constant	Configured diagnostic address of the DP slave Note: Address must be specified in hexadecimal form, for example, diagnostic address 1022 means: LADDR:=W#16#3FE.
RET_VAL	Return	DINT, INT, LREAL, REAL	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error has occurred, the length of the data actually transferred is entered in RET_VAL .

Parameter	Declaration	Data type	Memory area	Description
RECORD	Output	VARIANT	I, Q, M, D, L	Destination area for the diagnostic data that were read. Only the BYTE data type is permitted. The minimum length of the data record to be read or the destination area is 6. The maximum length of the data record to be sent is 240. Standard slaves can provide more than 240 bytes of diagnostic data up to a maximum of 244 bytes. In this case, the first 240 bytes are transferred to the destination area and the overflow bit is set in the data.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

You can find information on data type conversion in the different programming languages under "Overview of data type conversion (Page 2157)".

Parameter RECORD

The CPU evaluates the actual length of the read diagnostic data:

If the length specified for RECORD

- is less than the number of data bytes supplied, the data is discarded and a corresponding error code is entered in RET_VAL.
- is greater than or equal to the number of data bytes supplied, the data is accepted in the destination area and the actual length is entered in RET_VAL as a positive value.

Note

You must ensure that the actual parameters of RECORD correspond in all calls belonging to a job.

A job is uniquely identified by the LADDR input parameter.

Standard slaves with more than 240 bytes of diagnostic data

With standard slaves on which the number of standard diagnostic data is between 241 and 244 bytes, note the following points:

If the length specified for RECORD

- is less than 240 bytes, the data is discarded and a corresponding error code is entered in RET_VAL.
- If the length specified for RECORD is greater than or equal to 240 bytes, the first 240 bytes of the standard diagnostic data is transferred to the destination area and the overflow bit is set in the data.

Parameter RET_VAL

- If an error occurs while the function is being executed, the return value contains an error code.
- If no errors have occurred during data transfer, RET_VAL contains the length of the data read in bytes as a positive number.

Note

The amount of data read in a DP slave depends on its diagnostics status.

For the evaluation of the error information of the RET_VAL parameter, refer to the following table.

Error code (W#16#...)	Explanation	Restriction
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".	-
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Address specified at parameter LADDR is invalid.	-
8093	This instruction is not permitted for the module selected by means of LADDR and IOID.	-
80A2	• DP protocol error at layer 2 (for example, slave failure or bus problems) • For ET 200S, data record cannot be read in DPV0 mode.	Distributed I/O
80A3	DP protocol error with user interface/user	Distributed I/O
80A4	Communication on PROFIBUS disrupted.	Distributed I/O
80B0	• Instruction not possible for module type. • The module does not recognize the data record. • Data record number 241 not permitted. • With "WR_REC (Page 3071)", data records 0 and 1 are not permitted.	-
80B1	The length specified in parameter RECORD is incorrect.	specified length < data record length
80B2	The configured slot is not assigned.	-
80B3	Actual module type does not match specified module type	-
80C0	There are no diagnostic data available.	-
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	-

Error code (W#16#...)	Explanation	Restriction
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available.	Distributed I/O
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

See also

Overview of data type conversion (Page 2157)

DP_TOPOL: Determine topology for DP master system

Description

You use the instruction to trigger the topology determination for a selected DP master system. Calling the instruction will address all diagnostics repeaters on a DP master system.

Note

The topology can only be determined for one DP master system at a time.

Topology determination is the prerequisite for the detailed display of the error location if line errors occur. After configuration and after every change to the physical configuration of a DP master system, you must repeat the topology determination with "DP_TOPOL".

Changes to the physical configuration are:

- Changes in line lengths
- Adding or removing stations or components with repeater function
- Changing station addresses

If an error is reported by a diagnostics repeater, "DP_TOPOL" writes the DPR and DPRI outputs for the duration of one "DP_TOPOL pass. If errors are reported by multiple diagnostics repeaters of the selected DP master system, "DP_TOPOL" writes DPR and DPRI with information regarding the first diagnostics repeater reporting the error. You can read out the entire diagnostic information on the programming device or using the "DPNRM_DG (Page 3139)" instruction. If no diagnostics repeater reports an error, the DPR and DPRI outputs have the value NULL.

If you want to repeat a topology determination after an error occurs, you must first reset "DP_TOPOL". This is done by calling "DP_TOPOL" with REQ=0 and R=1.

Functional description

"DP_TOPOL" works asynchronously, that is, its execution extends over multiple calls. You start determining the bus topology by calling "DP_TOPOL" with REQ=1 . If you want to cancel the process, call "DP_TOPOL" with R=1 .

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

Note

Determining the topology may take several minutes.

Identification of a job

The input parameter DP_ID uniquely specifies a job.

If you have called "DP_TOPOL" and you call this instruction again before the topology is re-determined, the manner in which the instruction reacts depends on whether the new call involves the same job: If the DP_ID parameter matches a job that has not yet been completed, then the call is interpreted as a follow-up call and the value W#16#7002 will be entered in RET_VAL. On the other hand, if another job is involved, the CPU rejects it.

Parameters

The following table shows the parameters of the "DP_TOPOL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ=1: Trigger topology determination
R	Input	BOOL	I, Q, M, D, L or constant	R=1: Cancel topology determination
DP_ID	Input	HW_IOSYSTEM	I, Q, M, D, L or constant	DP master system ID of those master systems for which the topology will be determined
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: Topology determination is not yet complete.
DPR	Output	BYTE	I, Q, M, D, L	PROFIBUS address of the diagnostics repeater reporting the error.
DPRI	Output	BYTE	I, Q, M, D, L	Measuring segment diagnostics repeater reporting the error: • Bit 0 = 1: Temporary faults on segment DP2 • Bit 1 = 1: Permanent faults on segment DP2 • Bit 4 = 1: Temporary faults on segment DP3 • Bit 5 = 1: Permanent faults on segment DP3

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

When looking at the "real" error information (error codes W#16#8xyz) in the following table, we distinguish between two cases:

- Temporary errors (error codes W#16#80A2 to 80A4, 80C3, 80C5):
  It is possible to eliminate this type of error without user action; in other words, it is advisable to call "DP_TOPOL" again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#8082, 80B0, 80B2):
  This type of error does not correct itself. A new call of "DP_TOPOL" is only advisable after you have eliminated the error. Example of a permanent error: The DP master/CPU does not support this service. (W#16#80B0).

Error code* (W#16#...)	Explanation
0000	Job completed without error.
7000	First call with REQ=0. Topology determination is not triggered. BUSY has the value "0".
7001	First call with REQ=1. The request to execute topology determination was sent. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Topology determination is not yet complete. BUSY has the value "1".
7010	You have attempted to cancel topology identification. But there is no running job with the specified DP_ID. BUSY has the value "0".
7011	First call with R=1. The cancellation of the topology determination was triggered. BUSY has the value "1".
7012	Intermediate call: The cancellation of the topology determination is not yet complete. BUSY has the value "1".
7013	Final call: The topology determination was cancelled. BUSY has the value "0".
8082	No DP master system is configured with the specified DP_ID.
80A2	Error during topology determination; for more detailed information, refer to output parameters DPR and DPRI.
80A3	Error during topology determination: Monitoring time has elapsed (timeout).
80A4	Communication on PROFIBUS disrupted.
80B0	The DP master/CPU does not support this service.
80B2	Error during topology determination: No diagnostics repeater was found at the selected DP master system.
80C3	Resources required are currently in use. Possible cause: You have initiated a second topology determination (only one topology determination is permitted at one time).
80C5	The DP master system is currently not available.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

ASi

ASI_CTRL

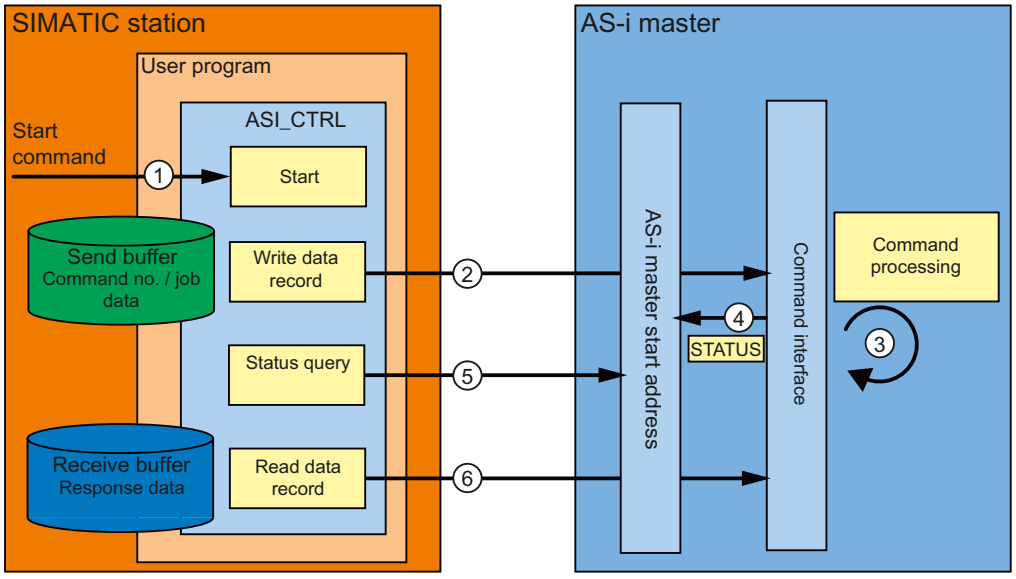
Description of ASI_CTRL

Description

Using the instruction "ASI_CTRL", you can control the behavior of the AS-i master from the user program of the PLC. The instruction processes the command protocol automatically. It also enables parameter assignment on SIMATIC AS-i masters and reading out information data. The functions and operation of the command interface are detailed in the manual for the AS-i master.

Both centrally inserted AS-i masters and distributed AS-i masters via PROFIBUS DP are supported. Combinations with PROFINET IO (e.g. IE/PB Link PN IO) are also possible.

The schematic diagram below shows the functions of the "ASI_CTRL" instruction:



① Start of processing at the REQ parameter.

② The program sends the required command to the AS-i master via the instruction "RDREC".

③ The AS-i master processes the command.

④ The current status of the AS-i master is stored in the input area of the binary data (logical start address).

⑤ The instruction "ASI_CTRL" cyclically queries and evaluates the 4 status bits.

⑥ Once command processing is complete, the command job is completed with "RDREC". Depending on the command, the data field of "RDREC" may contain the response data for the command or additional status information.

Differences in the command call between IE/ AS-i Link and DP/AS-i Links

There are significant differences in how a controller exchanges commands with an AS-i master.

- **IE/AS-i Link** (PROFINET) used the data record interface. The different commands are called with either "Write data record" ("WRREC" instruction) or "Read data record" ("RDREC" instruction) by various data record numbers.
- **DP/AS-i Links** (PROFIBUS) use the command interface. All commands are called by data record number 2 with "Write data record" ("WRREC" instruction) plus "Read data record" ("RDREC" instruction). The type of command is defined by the data content in the write job.

Changes compared to the instruction "ASi_3422".

The "ASI_CTRL" instruction is a revised version of the "ASi_3422" instruction (for S7-300/400 CPUs only) and provides improved functionality and compatibility. The specific changes are as follows:

- For writing and reading diagnostic data records, the instructions "WR_REC (Page 3123)" and "RD_REC (Page 3118)" have been replaced by the instructions "RDREC (Page 3069)" and "WRREC (Page 3071)". Their function is identical; however, they support data transfer via PROFINET IO.
- The block type of the instruction has been changed from a function (FC) to a function block (FB). "ASI_CTRL" has an instance data block and is multi-instance-capable.
- The designation of the formal parameters of "ASI_CTRL" complies with the SIMATIC system blocks. There is no STARTUP input parameter. The definition of the STATUS parameter is based on the instructions "RDREC (Page 3069)" and "WRREC (Page 3071)". The status identifiers for the parameter DONE and the new parameter BUSY have also been adjusted.

Operation of the "ASI_CTRL" instruction

The instruction "ASI_CTRL" is an asynchronous function block; in other words, its processing extends over multiple calls.

- A job is started with REQ = TRUE.
- The job status is displayed via the BUSY output parameters and the two central bytes of the output parameter STATUS.
- The BUSY parameter is set during job processing. Upon the first call, STATUS is assigned the value 00700100_H. Upon all subsequent calls for this job, it is assigned the value 00700200_H. When the job is complete, the result is output at the parameters DONE or ERROR.
  - If no errors have occurred, DONE is set. For jobs with response data from the AS-i master, this data is provided in the specified receive buffer. In such cases, the STATUS parameter also displays the volume of data supplied in bytes. For jobs without response data, the value 00000000_H is entered in STATUS.
  - If an error occurs during job processing, ERROR is set. The content of the receive buffer is invalid in such a case. An error code is entered in the STATUS parameter for a more detailed description of the error which has occurred.

Number of command calls

If you use the instruction "ASI_CTRL" to send commands, you may not simultaneously send other commands to the same AS-i master with "RDREC (Page 3069)" or "WRREC (Page 3071)". This also applies to multiple calls of the instruction for the same AS-i master.

The instruction "ASI_CTRL" cannot be run with interruptions. Calls therefore cannot be programmed in program priority classes which interrupt each other (for example with calls in OB 1 and in OB 35).

Parameters

The following table shows the parameters of the instruction "ASI_CTRL":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, constant	REQ = TRUE starts a new job unless a job is already in progress. No edge evaluation takes place.
LADDR	Input	WORD	I, Q, M, D, L, constant	Hardware identifier of the AS-i master. You can obtain the address from the module properties.
SD	Input	VARIANT	I, Q, M, D, L	Send buffer The parameter refers to a memory area in which the command is to be specified (see " ASi commands (Page 3148)").
RD	Input	VARIANT	I, Q, M, D, L	Receive buffer This buffer is only relevant for commands which deliver answer data. The parameter refers to a memory area in which a command response is stored (see "ASi commands (Page 3148)").
DONE	Output	BOOL	I, Q, M, D, L	DONE = TRUE: Job completed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = TRUE: Job in progress.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = TRUE: Job aborted with error.
STATUS	Output	DWORD	I, Q, M, D, L	Job status / error code See the "STATUS parameter" description.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Note

Parameters LADDR, SD, and RD

The parameters LADDR, SD, and RD must not be changed in any block cycle when a job is in progress: they must remain constant.

Parameter STATUS

The following table shows the possible STATUS displays dependent upon DONE and ERROR.

DONE	ERROR	STATUS	Meaning
0	0	00700000_H	First call with REQ = FALSE; no active job.
0	0	00700100_H	First call with REQ = TRUE; job has started.
0	0	00700200_H	Subsequent call (REQ irrelevant); job is still in progress.
1	0	00000000_H	Job completed without errors. No response data.
1	0	$0000xx00_H$	Job completed without errors. Number of xx bytes of response data.
0	1	$C0818400_H$	Data type of formal operand RD invalid.
0	1	$C0818500_H$	Error in communication with AS-i master (incorrect address configured at the LADDR parameter).
0	1	$C0838100_H$	Incorrect AS-i slave address.
0	1	$C0838200_H$	AS-i slave is not enabled (not in LAS).
0	1	$C0838300_H$	Error at the AS interface (the SD parameter may have been set too small).
0	1	$C0838400_H$	The command is not valid with the current AS-i master status.
0	1	$C0838500_H$	There is no AS-i slave with the address "0".
0	1	$C0838600_H$	The AS-i slave has invalid configuration data (I/O or ID codes).
0	1	$C083A100_H$	The AS-i slave addressed has not been found at the AS interface.
0	1	$C083A200_H$	There is no AS-i slave with the address "0".
0	1	$C083A300_H$	There is already an AS-i slave with the new address at the AS interface.
0	1	$C083A400_H$	The AS-i slave address cannot be deleted.
0	1	$C083A500_H$	The AS-i slave address cannot be set.
0	1	$C083A600_H$	The AS-i slave address cannot be permanently saved.
0	1	$C083A700_H$	Error during reading of the extended ID1 code.
0	1	$C083A800_H$	The target address is not plausible (e.g. a B slave address has been used for a standard slave).
0	1	$C083B100_H$	A length error has occurred during string transfer.
0	1	$C083B200_H$	A protocol error has occurred during string transfer.
0	1	$C083F800_H$	Unknown job number or job parameter.
0	1	$C083F900_H$	The AS-i master has detected an EEPROM error.

See also

GET_ERR_ID: Get error ID locally (Page 2417)

ASi commands

Description

The command interface allows the controller and AS-i master to exchange parameter assignment and information data.

These commands

- provide the complete functionality of the M4 master profile of the AS-i master specifications.
- enable the AS-i master to be completely configured from the controller.

Note

AS-i commands supported

Please see the manual of the relevant AS-i master for the AS-i commands supported and a detailed description.

General structure of the send buffer

The general structure of the send buffer for commands and job data is set out in the table below. The area for the command number must always be filled. The number of bytes for the job data can be found in the description of the command (see AS-i master documentation). "q" is the start address of the send buffer.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Meaning							
q + 0	Command number							
q + 1	Job data							
q + 2	Job data							
q + ...	Job data							

General structure of the receive buffer

The general structure of the receive buffer for the command response data is set out in the table below. The number of bytes for the response data depends on the command. Some commands do not return response data, and therefore only require the definition of a virtual receive buffer which is not filled with data. "n" is the start address of the receive buffer.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Meaning							
n + 0	Command number (echo)							
n + 1	Response data							
n + 2	Response data							
n + ...	Response data							

Caution

Memory areas can be overwritten

If the receive buffer of the "ASI_CTRL" instruction is too short, neighboring memory areas may be overwritten. The length specified in the ANY pointer of the RD parameter when the instruction "ASI_CTRL" is called is irrelevant. The required length for the receive buffer can be found in the description of the command.

The following applies for command numbers 39_H, 41_H, 42_H, 43_H and 44_H:

The receive buffer must be 221 bytes long (bytes 0 to 220), even if the command returns less data. Depending on the command, the highest bytes in the receive buffer may be overwritten with zero values by the AS-i master.

AS-i commands

A selection of possible AS-i commands is set out in the table below.

Name	Parameter	Return	Coding
Set_permanent_parameter (Set_Permanent_Parameter)	Slave address, parameter		00_H
Get_permanent_parameter (Get_Permanent_Parameter)	Slave address	Parameter	01_H
Write_parameter (Write_Parameter)	Slave address, parameter	Parameter echo	02_H
Read_parameter (Read_Parameter)	Slave address	Parameter value	03_H
Store_actual_parameters (Store_Actual_Parameters)			04_H
Set_configuration_data	Slave address, configuration		25_H
Get_configuration_data	Slave address	set configuration data	26_H
Store_actual_configuration (Store_Actual_Configuration)			07_H
Get_actual_configuration	Slave address	Actual configuration data	28_H
Configure_LPS	LPS		29_H
Set_offline_mode	Mode		$0A_H$
Select_auto-program	Mode		$0B_H$
Set_mode	Mode		$0C_H$
Change_AS–iSlave_address (Change_AS–iSlave_Address)	Address1, Address2		$0D_H$
Get_AS-iSlave_status	Slave address	Error record of the AS-i slave	$0F_H$
Read_lists_and_flags		LDS, LAS, LPS, flags	30_H
Get_overall_configuration		Actual configuration data, current parameters, LAS, flags	39_H

Name	Parameter	Return	Coding
Set_overall_configuration	Overall configuration		3A $_H$
Write_parameter_list	Parameter list		3C $_H$
Read_parameter_echo_list		Parameter echo list	33 $_H$
Write_CTT2_request	Slave address CTT2 string	CTT2 string	44 $_H$
Read_version_identifier		Version string	14 $_H$
Read_AS-i_slave	Slave address	ID code	17 $_H$
Read_AS-i_slave_extended_ID1	Slave address	Extended ID1 code	37 $_H$
Write_AS-iSlave_extended_ID1	Extended ID1 code		3F $_H$
Read_AS-iSlave_extended_ID2	Slave address	Extended ID2 code	38 $_H$
Read_AS-iSlave_IO	Slave address	I/O configuration	18 $_H$
Read_I/O_error_list		LPF	3E $_H$
Write_AS–i-slave_parameter_string	Slave address, parameter string		40 $_H$
Read_AS–iSlave_parameter_string	Slave address	Parameter string	41 $_H$
Read_AS-iSlave_ID_string	Slave address	ID string	42 $_H$
Read_AS-iSlave_diagnostic_string	Slave address	Diagnostic string	43 $_H$
Read_AS-i_line_error_counter			4A $_H$
Read_and_clear_AS-i_line_error_counter			4B $_H$
Read_AS-iSlave_error_counter	Slave address		4C $_H$
Read_and_clear_AS-iSlave_error_counter	Slave address		4D $_H$
Additional command for DP/ AS-i F-Link:			
AS–i_status/diag_of_F_slaves		Status/diagnostics of all AS-i safe slaves	51 $_H$

Note

Re-initializing the AS-i master command interface

Another command which is not mentioned in the table is command 77 $_H$. This call re-initializes the command interface of the AS-i master. Any command which the AS-i master specified is currently processing will be terminated.

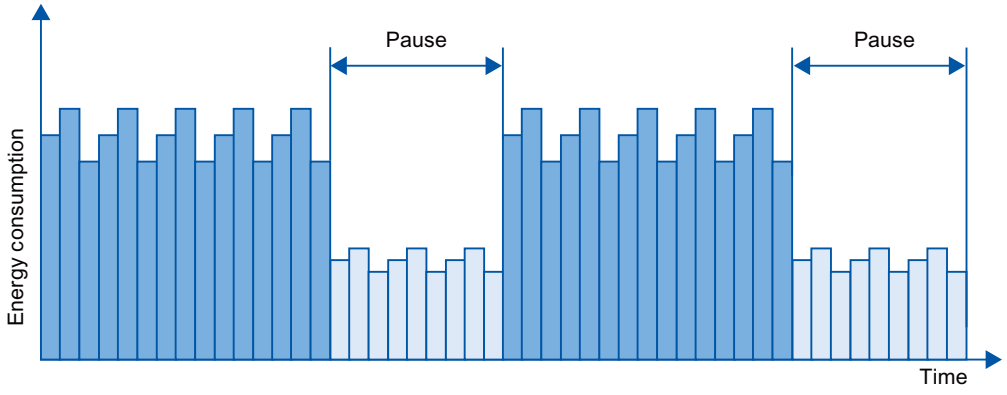
As of version V2.1.20 of DP/AS-i LINK Advanced, command 0E $_H$ is also available. This call enables you to release and block the ground fault monitoring function for a line.

11.6.3.5 PROFIenergy

Description of PROFIenergy

PROFIenergy

PROFIenergy is a manufacturer- and device-neutral profile for energy management with PROFINET. PROFIenergy enables central, coordinated device shutdown to reduce electricity consumption during breaks in production and unplanned interruptions.



The PROFINET devices/power modules are switched off via special commands in the user program of the PROFINET IO controller. No additional hardware is required. The PROFINET devices interpret the PROFIenergy commands directly.

PROFIenergy controller (PE controller)

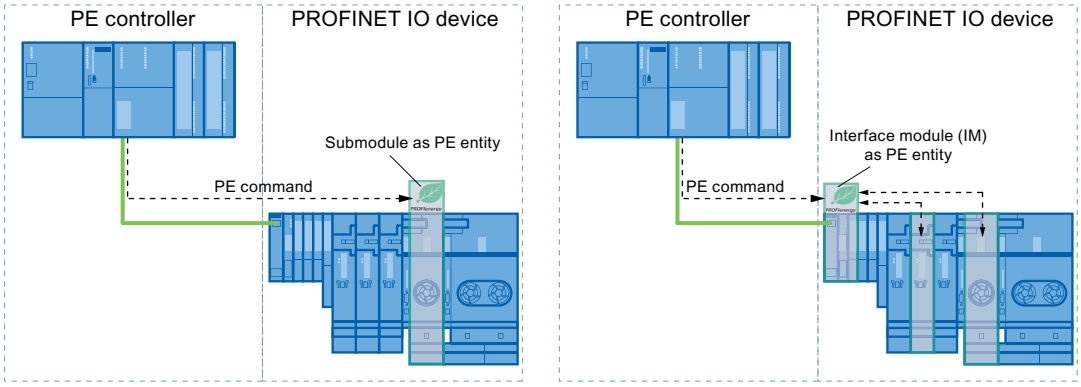
The PE controller is a PLC which enables/disables the idle state in lower-level devices. Individual production components and entire production lines are switched off and reactivated from the user program. Commands (e.g. "Start_Pause" and "End_Pause") are sent to the lower-level device using corresponding instructions (function blocks). The commands are sent via the PROFINET communication protocol.

PROFIenergy entity (PE entity)

The PE entity receives the PROFIenergy commands of the PE controller and executes these accordingly (for example, by returning a measured value or activating an energy saving mode). Implementation of the PE entity in a PROFIenergy-capable device is device- and manufacturer-specific.

The PE entity can, for example, be executed:

- Within the proxy of a submodule: The PE commands are valid for the addressed submodule and any existing lower-level submodules.
- Within the proxy of a module: The PE commands are valid for different submodules within the module.



- For networked submodules without a proxy function: The PE commands in this case are only valid for the respective submodule.

PROFIenergy instructions

- Instructions for IO controller
  - The "PE_START_END (Page 3154)" instruction represents the simplest way to activate or deactivate the idle state of the PROFINET devices (PROFIenergy commands "Start_Pause" and "End_Pause"). This is done using a positive and negative signal edge in the instruction.
  - The "PE_CMD (Page 3159)" instruction allows you to transmit all PROFIenergy commands, including "Start_Pause" and "End_Pause". With other commands, you can query the current status of the PROFINET devices or the behavior during breaks, for example.
  - The instruction "PE_DS3_Write_ET200S (Page 3164)" is used to define the switching characteristic of up to 8 slots of ET 200S. The instruction is not a PROFIenergy instruction; however, it supplements the PROFIenergy functions for an ET 200S.
- Instruction for iDevices
  The "PE_I_DEV (Page 3189)" instruction allows you to implement PROFIenergy on iDevices as well. The instruction receives PROFIenergy commands on the iDevice and forwards these to the user program for processing. After processing the command, the user program calls the "PE_I_DEV (Page 3189)" instruction again in order to send the acknowledgement to the IO controller. For these replies, each command offers you a corresponding auxiliary block that supplies the response data to the instruction "PE_I_DEV (Page 3189)".

PROFIenergy commands (PE commands)

The PE controller transmits PE commands to the PE entity. The PE command can be either a control command to switch a PE entity to the energy-saving mode, or a command to read a status or measured value:

- PI commands for control
  PROFIenergy supports two control commands that can be executed using either the "PE_Start_End (Page 3154)" instruction or the "PE_CMD (Page 3159)" instruction:
  - Start_Pause: Starting a suitable energy-saving mode (PE Energy-saving mode)
  - End_Pause: Exiting the energy-saving mode (switch to PE_ready_to_operate mode)
- PI commands for reading a status or measured value
  Certain condition information can be read by the control system using the following status commands with the instruction "PE_CMD (Page 3159)":
  - PE_Identify: Read out which PE commands are supported by the PE entity.
  - PEM_Status: Read out the currently active mode of a PE entity (e. g. PE_ready_to_operate).
  - Query_Modes: Output an overview of all supported energy-saving modes, including the time and energy information
  - Query_Measurement: Output the measured values of a PE entity

Example applications

Examples for use of PROFIenergy instructions can be found in the Industry Online Support in the entry "PROFIenergy - Saving Energy with SIMATIC S7 (http://support.automation.siemens.com/WW/view/en/41986454)".

See also

Service and Support (http://support.automation.siemens.com/)

IO controller

PE_START_END: Start and exit energy-saving mode

Description

The instruction "PE_START_END" is used to start and exit the energy-saving mode of a specified PE entity (e.g. ET 200S).

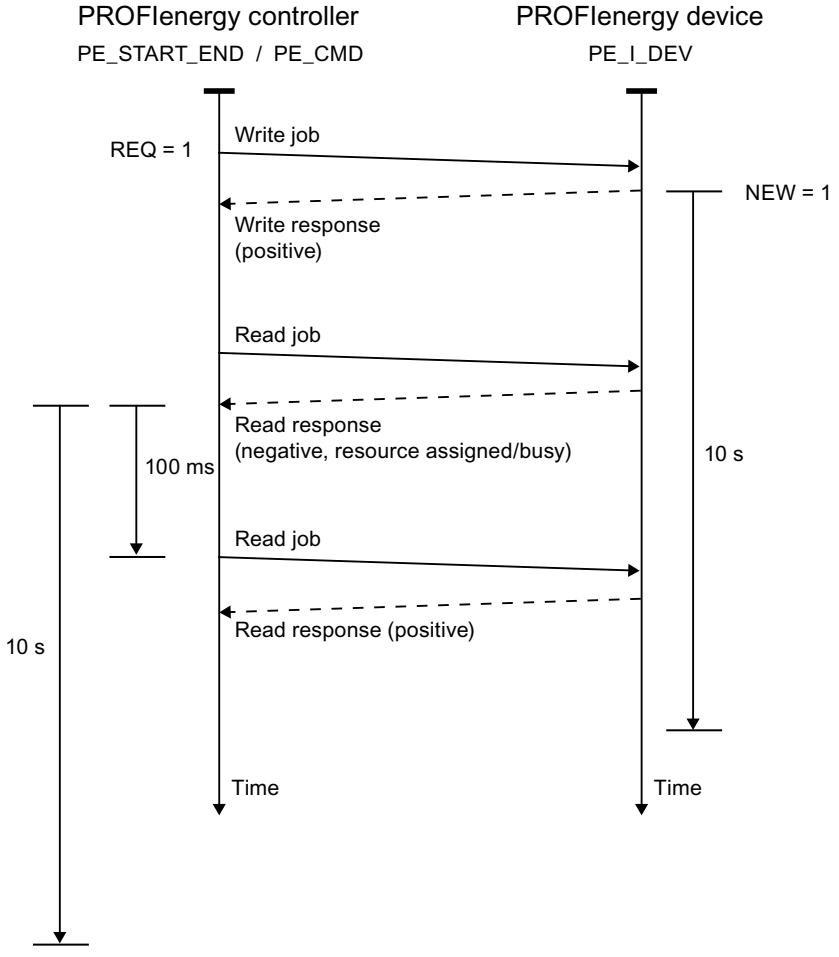
The instruction "PE_START_END" is used in the PE controller, preferably when only field devices from which energy data does not need to be read out are connected to the corresponding PE devices. Alternatively, the instruction "PE_CMD (Page 3159)" can be used to read energy data.

The energy-saving modes are configured in the user program of the PE controller. On completion of the "PE_START_END" instruction, the PE entity reports its current energy-saving mode and outputs this data at parameter PE_MODE_ID.

Write and read jobs of the "PE_START_END" instruction.

The instruction "PE_START_END" sends a PROFIenergy command internally as a write job to the PE entity using "WRREC (Page 3071)". "PE_START_END" then waits for acknowledgment from the PE entity. The acknowledgment data record is read with the instruction "RDREC (Page 3069)" every 100 milliseconds. Until acknowledgment is received from the PE entity, the function repeats the read job for 10 seconds at intervals of 100 milliseconds. The response data of the PE entity is also read with the instruction "RDREC (Page 3069)".

Below is a flowchart of the write and read jobs:



Parameter

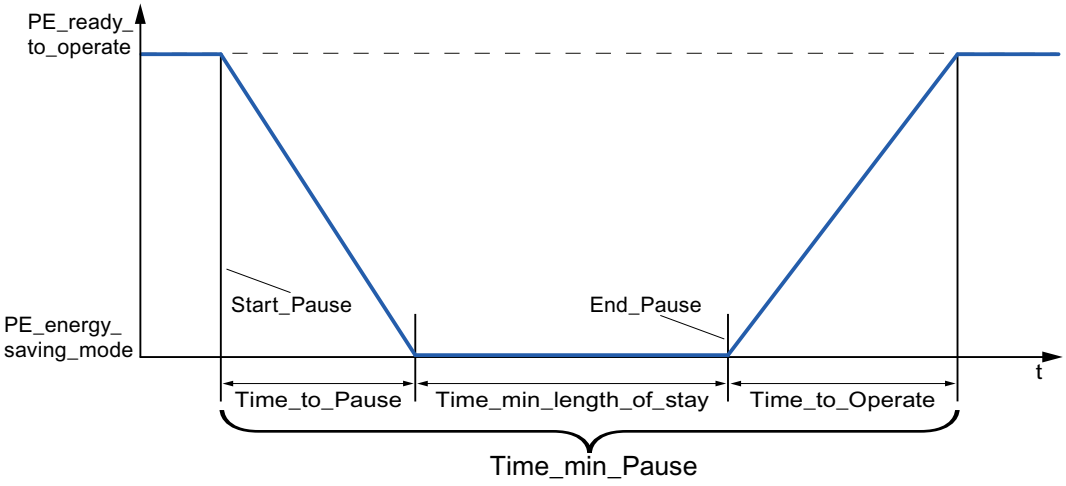
The following table shows the parameters of the instruction "PE_START_END":

Parameter	Declaration	Data type	Memory area	Description
START	Input	BOOL	I, Q, M, D, L or constant	Transmitting the PE command "Start_Pause" to the PE entity with the address set at parameter ID.
END	Input	BOOL	I, Q, M, D, L or constant	Transmitting the PE command "End_Pause" to the PE entity with the address set at parameter ID.
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Address of the PE entity Use the hardware ID of the head module for a PROFINET IO device. Refer to the system constants of the assigned IO controller for the hardware ID. The name of the head module is composed of the name of the IO device and the suffix [Head] (example: "IO_Device_1[Head]"). If the PE entity is an I-device, use the hardware identifier of a transfer area instead.
PAUSE_TIME	Input	TIME	I, Q, M, D, L, P or constant	Planned pause duration. • Range: T#1MS to T#24D20H31M23S647MS • Start value: T#0MS
VALID	Output	BOOL	I, Q, M, D, L	PE command successfully sent.
BUSY	Output	BOOL	I, Q, M, D, L	PE command still being processed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred during processing. The error message is output at the STATUS parameter.
STATUS	Output	DWORD	I, Q, M, D, L, P	Block status / error number (see "STATUS parameter")
PE_MODE_ID	Output	BYTE	I, Q, M, D, L, P	Identification number of energy-saving mode (energy-saving level for the duration of the pause).

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

PAUSE_TIME parameter

The parameter PAUSE_TIME is used to set the default duration of the energy-saving period at the PE entity. The PE entity checks whether the pause period is of sufficient length and whether it can be implemented. The minimum pause duration (Time_min_Pause) must be greater than the sum of the times the device needs to switch to energy-saving mode (Time_to_Pause) and to switch back to operating mode (Time_to_Operate).



ET 200S checks whether the scheduled pause duration is greater than or equal to the minimum pause duration (PM-E_Pause_Min) saved on the ET 200S. This is fixed at 10 seconds. If a shorter pause is used, the power modules (PM-E) of the ET 200S will remain on.

The module does not switch back on automatically at the end of the pause; it remains in OFF mode until the "END" command is sent. This prevents uncoordinated switch-on, which can cause undesired peak loads.

STATUS parameter

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error during reading of data record • B#16#**DF**: Error during writing of data record • B#16#**C0**: Error message from the instruction or from the communication instructions "RDREC (Page 3069)" and "WRREC (Page 3071)" used internally.
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific • **FE**: DP/PNIO profiles - PROFIenergy-specific error

Field element	Name	Meaning
STATUS[3]	Error_Code_1	Error ID • When Error_Decode = **80**: – **80**: Simultaneous positive edge at the input parameters START and END. – **81**: Length conflict at the CMD_PARAM and CMD_PARAM_LEN parameters. – **82-8F**: Other error messages (reserved) • When Error_Decode = **FE**: – **01**: "Service Request ID" invalid – **02**: Incorrect "Request_Reference" – **03**: "Modifier" invalid – **04**: "Data Structure Identifier RQ" invalid – **05**: "Data Structure Identifier RS" invalid – **06**: "PE energy-saving modes" are not supported – **07**: "Response" is too long (maximum transmissible length exceeded) – **08**: "Count" invalid – **50**: No suitable "energy mode" is available. – **51**: Time value specified is not supported. – **52**: "PE_Mode_ID" invalid – **53**: Cannot switch to "PE energy-saving mode" because the device is in operation – **54**: Function cannot be used at this time. Wrong device configured or incorrect setup. – **55** to **FF**: Reserved
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Note

Error messages of the instructions RDREC and WRREC

The instruction "PE_START_END" uses the instructions "WRREC (Page 3071)" and "RDREC (Page 3069)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 3078) parameter for an explanation of the error codes of the "WRREC (Page 3071)" and "RDREC (Page 3069)" instructions.

See also

Description of PROFIenergy (Page 3152)

PE_CMD: Start and exit energy-saving mode / Read out status information

Description

The instruction "PE_CMD" is used in the PE controller to start or terminate a pause in the energy-saving mode in the PE entity. Additional information and energy measurements can also be read out from a PE entity using "PE_CMD".

The instruction is best used with PE controllers to which assigned PE devices are connected from which energy measurements are to be read out. If this is not the case, the instruction "PE_START_END (Page 3154)" can also be used to start and end the pauses.

Transfer of the PROFIenergy commands (PE commands)

The instruction "PE_CMD" transfers a PROFIenergy command to a PE entity.

The instruction can also be used if additional commands are to be added to the PROFIenergy profile in future. The commands which can be used with the current PROFIenergy profile are listed in the description of the CMD and CMD_MODIFIER parameters (see the "CMD and CMD_MODIFIER parameters" table).

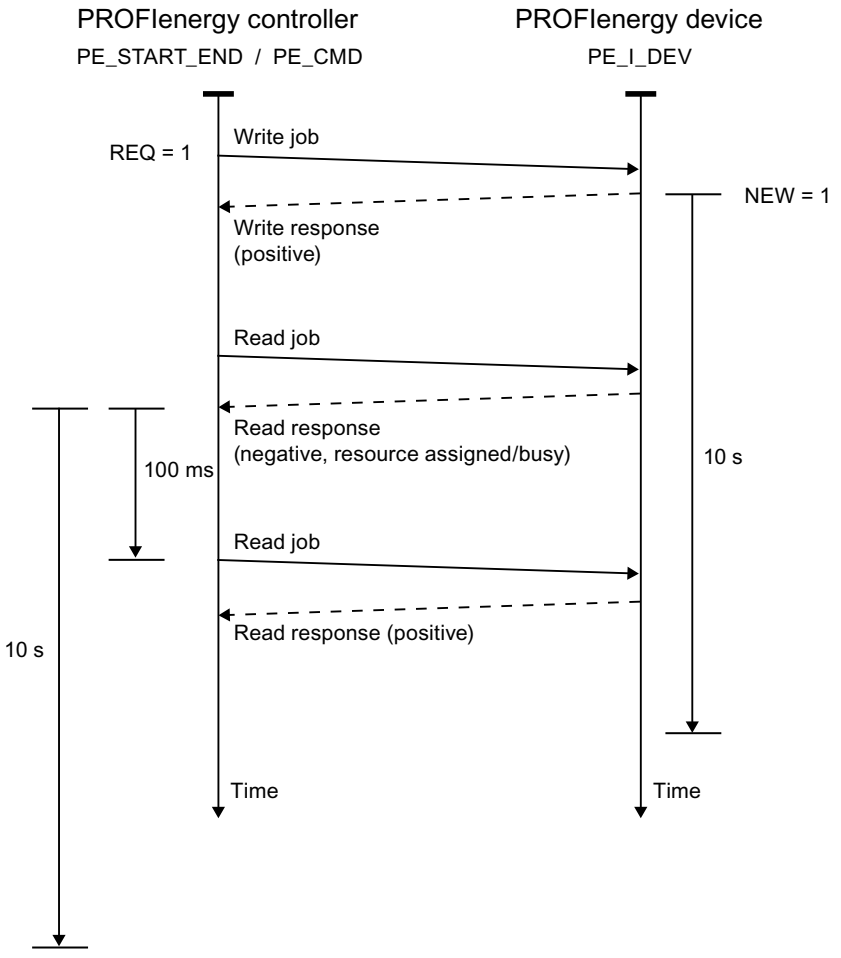
- The individual PE commands which are transferred to the PE entity using the instruction are assigned defined "Service_Request_IDs". The Service_Request_IDs 01 to 05 and 16 are assigned at the CMD parameter.
- The two PE commands 04 (Query_Modes) and 16 (Query_Measurement) are defined in more detail with the CMD_MODIFIER parameter.
- Additional values are transferred at the CMD_PARA parameter for individual PE commands (see the description of the individual PE commands). The parameter CMD_PARA_LEN defines the length of the data at the parameter CMD_PARA.

The commands are transferred without plausibility test. The response data of the PE entity is saved to the RESPONSE_DATA data area that is addressed by VARIANT pointer (for information about the response frame contents, refer to the description of the respective PE command).

Write and read jobs of the "PE_CMD" instruction.

The instruction "PE_CMD" uses "WRREC (Page 3071)" internally to transmit a PROFIenergy command as a write job to the PE entity. "PE_CMD" then waits for the acknowledgment from the PE entity. The acknowledgment data record is read with the instruction "RDREC (Page 3069)" every 100 milliseconds. Until acknowledgment is received from the PE entity, the function repeats the read job for 10 seconds at intervals of 100 milliseconds. The response data of the PE entity is also read with the instruction "RDREC (Page 3069)".

Below is a flowchart of the write and read jobs:



Parameter

The following table shows the parameters of the instruction "PE_CMD":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts transferring the PE command upon a positive edge.
ID	Input	HW_ SUBMODULE	I, Q, M, D, L or constant	Address of the PE entity Use the hardware ID of the head module for a PROFINET IO device. Refer to the system constants of the assigned IO controller for the hardware ID. The name of the head module is composed of the name of the IO device and the suffix [Head] (example: "IO_Device_1[Head]"). If the PE entity is an I-device, use the hardware identifier of a transfer area instead.

Parameter	Declaration	Data type	Memory area	Description
CMD	Input	BYTE	I, Q, M, D, L, P or constant	Service-Request-ID of the PROFIenergy command in accordance with the PROFIenergy profile (see "CMD and CMD_MODIFIER parameters"). Further service request IDs are possible following extensions of the PROFIenergy profile.
CMD_MODIFIER	Input	BYTE	I, Q, M, D, L, P or constant	PROFIenergy sub-command (only when CMD=3 or CMD=16, see "CMD and CMD_MODIFIER parameters") Further sub-commands are possible following extensions of the PROFIenergy profile.
CMD_PARA	Input	VARIANT	I, Q, M, D, L	Parameters for the PE commands: • Get mode: PE_mode_ID • Get measurement values: List of Measurement_Ids The complete Service Data Request is entered.
CMD_PARA_LEN	Input	INT	I, Q, M, D, L, P or constant	The actual length of the command parameters (<= length defined in CMD_PARA; is verified by the instruction).
RESPONSE_DATA	InOut	VARIANT	I, Q, M, D, L	PROFIenergy information May be complete response frame including block header, depending on the command. Note: If the buffer is too small, only the number of bytes which is specified in the VARIANT pointer will be entered.
VALID	Output	BOOL	I, Q, M, D, L	Command successfully sent.
BUSY	Output	BOOL	I, Q, M, D, L	Command still being processed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred during processing.
STATUS	Output	DWORD	I, Q, M, D, L	Block status / error number (see "STATUS parameter"):

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameters CMD and CMD_MODIFIER

CMD	CMD_ MODIFIER	PROFIenergy command	Description
01	0	Start_Pause (Page 3179)	Start energy-saving mode or switch to a different energy-saving mode.
02	0	End_Pause (Page 3179)	End energy-saving mode.

CMD	CMD_ MODIFIER	PROFIenergy command	Description
03	1	Query_Modes - List of energy-saving modes (Page 3180)	Output the energy-saving modes supported.
	2	Query_Modes - Get mode (Page 3181)	Output attributes of the energy-saving mode currently enabled.
04	0	PEM_Status (Page 3183)	Query status of energy-saving mode.
05	0	PE_Identify (Page 3185)	Read out the number and description of PE commands supported.
16	1	Query_Measurement - Get_Measurement_List (Page 3185)	List of measured values supported by the PE entity.
	2	Query_Measurement - Get_Measurement_Values (Page 3188)	Output the measured values of the PE entity.

STATUS parameter

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error during reading of data record • B#16#**DF**: Error during writing of data record • B#16#**C0**: Error message from the internal communication instructions "RDREC (Page 3069)" and "WRREC (Page 3071)".
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific • **FE**: DP/PNIO profiles - PROFIenergy-specific error

Field element	Name	Meaning
STATUS[3]	Error_Code_1	Error ID • When Error_Decode = **80**: – **81**: Length conflict at the CMD_PARA and CMD_PARA_LEN parameters, or maximum data record length (4095 bytes) exceeded. – **82-8F**: Other error messages (reserved) • When Error_Decode = **FE**: – **01**: "Service Request ID" invalid – **02**: Incorrect "Request_Reference" – **03**: "Modifier" invalid – **04**: "Data Structure Identifier RQ" invalid – **05**: "Data Structure Identifier RS" invalid – **06**: "PE energy-saving modes" are not supported – **07**: "Response" is too long (maximum transmissible length exceeded) – **08**: "Count" invalid – **50**: No suitable energy-saving mode (energy mode) is available. – **51**: Time value specified is not supported. – **52**: "PE_Mode_ID" invalid
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Note

Error messages of the instructions RDREC and WRREC

The instruction "PE_CMD" uses the instructions "WRREC (Page 3071)" and "RDREC (Page 3069)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 3078) parameter for an explanation of the error codes of the "WRREC (Page 3071)" and "RDREC (Page 3069)" instructions.

See also

Description of PROFIenergy (Page 3152)

PE_DS3_Write_ET200S: Set power module switching behavior

Description

The instruction "PE_DS3_Write_ET200S" sends basic settings for power module switching behavior to the ET 200S. The switching behavior of up to 8 slots in the ET 200S (e.g. for power modules) can be defined with the instruction "PE_DS3_Write_ET200S".

Note

This instruction is not part of the PROIenergy profile but instead supplements SIMATIC-specific functions.

Parameter

The following table shows the parameters of the instruction "PE_DS3_Write_ET200S":

Parameter	Declaration	Data type	Memory area	Description
ENABLE	Input	BOOL	I, Q, M, D, L or constant	A positive edge triggers the transfer of the data record. The data record must be transferred again after voltage OFF/ON.
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Address of the ET 200S The address may be taken from the hardware configuration.
SLOT_NO_X	Input	INT	I, Q, M, D, L, P or constant	Slot number of switchable power module X.
FUNC_X	Input	INT	I, Q, M, D, L, P or constant	Function of the module in this slot. Use the parameter FUNC_X to define the switching behavior of the PM-E (power module of the ET 200S): • FALSE: – With "PAUSE_START": -No effect on PM-E -PM-E remains on – With "PAUSE_STOP": -Switches PM_E back on • TRUE: – With "PAUSE_START": -Switches PM_E off – With "PAUSE_STOP": -Switches PM-E back on
BUSY	Output	BOOL	I, Q, M, D, L	Transfer not yet complete.
DONE	Output	BOOL	I, Q, M, D, L	Transfer completed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Transfer completed with error.
STATUS	Output	DWORD	I, Q, M, D, L, P	Error number (see STATUS parameter of the instruction "PE_Start_End (Page 3154)")

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

See also

Description of PROFIenergy (Page 3152)

PE_WOL: Starting and stopping energy-saving mode via WakeOnLan

Description PE_WOL

Description

The "PE_WOL" instruction sends the PROFIenergy commands "Start_Pause" and "End_Pause" to multiple PROFIenergy-enabled devices in the PROFINET I/O systems.

Multiple PE devices can be coordinated via the instruction provided that the PE devices support the "Wake on LAN" function over a UDP connection.

The "PE_WOL" instruction can only be executed on a CPU with and integrated Ethernet interface. This CPU should be able to load blocks with a size of about 400 KB. You cannot use the block with PROFINET I/O systems that are connected via an Ethernet CP.

The instruction "PE_WOL" is executed asynchronously.

Definition: Wake on LAN

The Wake on LAN function allows data processing equipment to resume working from a state near total shutdown when a special Ethernet packet is received.

For this procedure to work, the data processing equipment must have a network controller that is capable of receiving such a packet.

This packet (Magic Packet™) has a special format. It contains the MAC address of the network adapter 15 times.

Selection of devices

The devices are selected using the user data block at the parameter PENERGY (Type: "PE_PLUS"). The user DB represents the database for processing multiple devices.

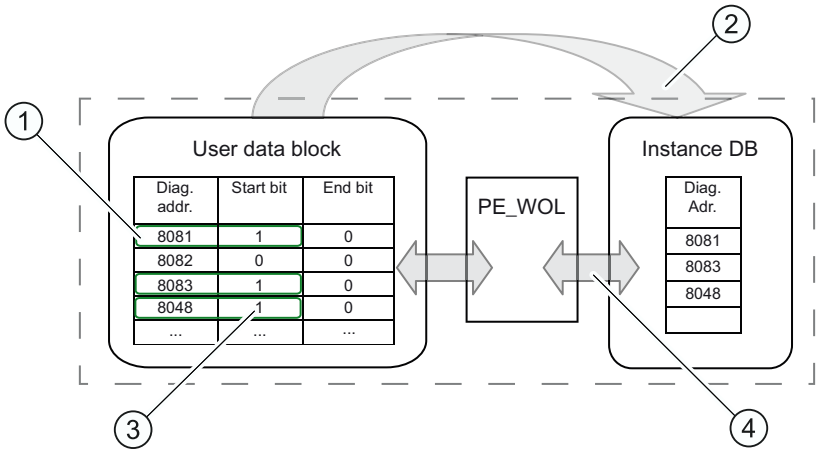
Prior to initialization of "PE_WOL", you must store at least the following information in the user DB:

- ID of the PROFINET I/O system
- Data of the connection which is used for "Wake On LAN".

- Port number which is used for "Wake On LAN".
- For each device
  - Pause time (PauseTime)
  - Switching of the device to the PE_SLEEP_MODE (EnableSleep)

You initialize the "PE_WOL" instruction with the COM_RST parameter. The jobs stored in the user DB are processed one after the other after initialization.

The graphic below illustrates how the PE command "Start_Pause" is transmitted to several devices:



(1) Step 1: The bit "CmdStartPause" of the devices to be shut down is set to "1" by the user.

(2) Step 2: The diagnostic addresses of the devices to be shut down (CmdStartPause = "1") are linked to the queue.

(3) Step 3: The bit "CmdStartPause" is reset automatically once the jobs have been linked.

(4) Step 4: The instruction "PE_WOL" processes the jobs as soon as they are linked.

A PROFIenergy command "CmdStartPause" or "CmdEndPause" can be sent for all recognized devices in the PROFINET IO system via the START and END parameters.

The status of the job processing as well as possible errors during processing are output by the STATUS parameter.

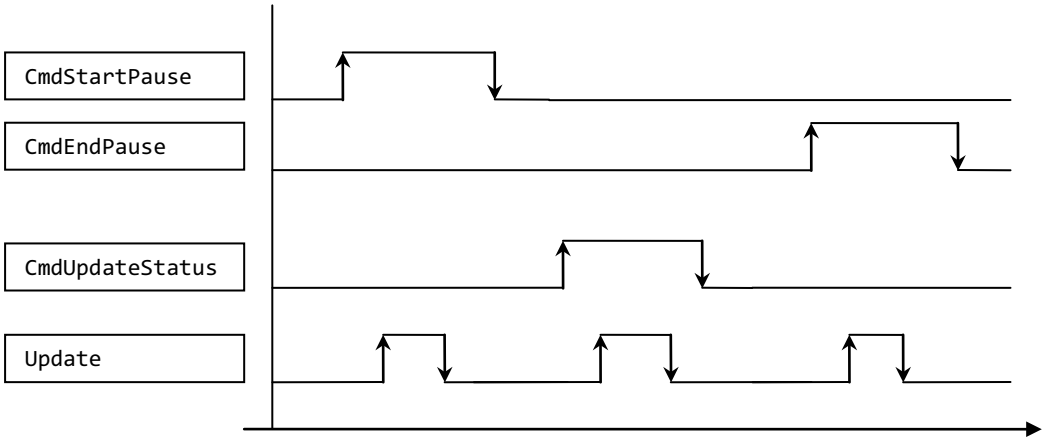
Operation of the instruction using the user DB

The operation of the instruction "PE_WOL" can only take place via the user DB. A basic procedure applies in this case:

1. Selection of the command to be executed for a device:
   - START_PAUSE ("CmdStartPause" in user DB)
   - ENDE_PAUSE ("CmdEndPause" in user DB)
   - UPDATE_STATUS ("CmdUpdateStatus" in user DB)
2. Setting the bit for updating ("Update" in the header of the user DB)
   At least one CPU cycle should pass with an "Update" = False between two updates; otherwise, edge detection cannot be guaranteed.

Prioritizing the PE commands

The graphic below shows the chronological sequence of three possible commands.



These are processed one after the other, regardless of whether the previous command call was successful or completed with an error.

Only one command is executed if you set two commands, such as "CmdEndPause" and "CmdUpdateStatus" simultaneously. A prioritization is present within the block:

- The command "CmdStartPause" has the highest priority and is therefore always executed if selected.
- The command "CmdEndPause" has the second highest priority.
- The command "CmdUpdateStatus" has the lowest priority.

If all three commands are set at the same time, the commands that are not executed remain preselected. In this case, the next rising edge executes the next command.

Parameters

The following table shows the parameters of the "PE_WOL" instruction:

Parameter	Declaration	Data type	Memory area	Description
COM_RST (Page 3168)	Input	BOOL	I, Q, M, D, L	Resets the block and performs a re-initialization. As long as True is set here, the initialization is started but not yet fully completed. Only the falling edge continues the initialization and switches to the normal operating mode after initialization.
START (Page 3169)	Input	BOOL	I, Q, M, D, L	A rising edge performs a "CmdStartPause" PROFIenergy command for all detected devices that support this function.
END (Page 3170)	Input	BOOL	I, Q, M, D, L	A rising edge performs a "CmdEndPause" PROFIenergy command for all detected devices that support this function.

Parameter	Declaration	Data type	Memory area	Description
PENERGY (Page 3170)	InOut	PE_PLUS	D	Pointer to the user DB that contains the database for processing multiple devices.
STATUS (Page 3175)	Output	DWORD	I, Q, M, D, L	Status/error number for the current status of the instruction (see "STATUS parameter").

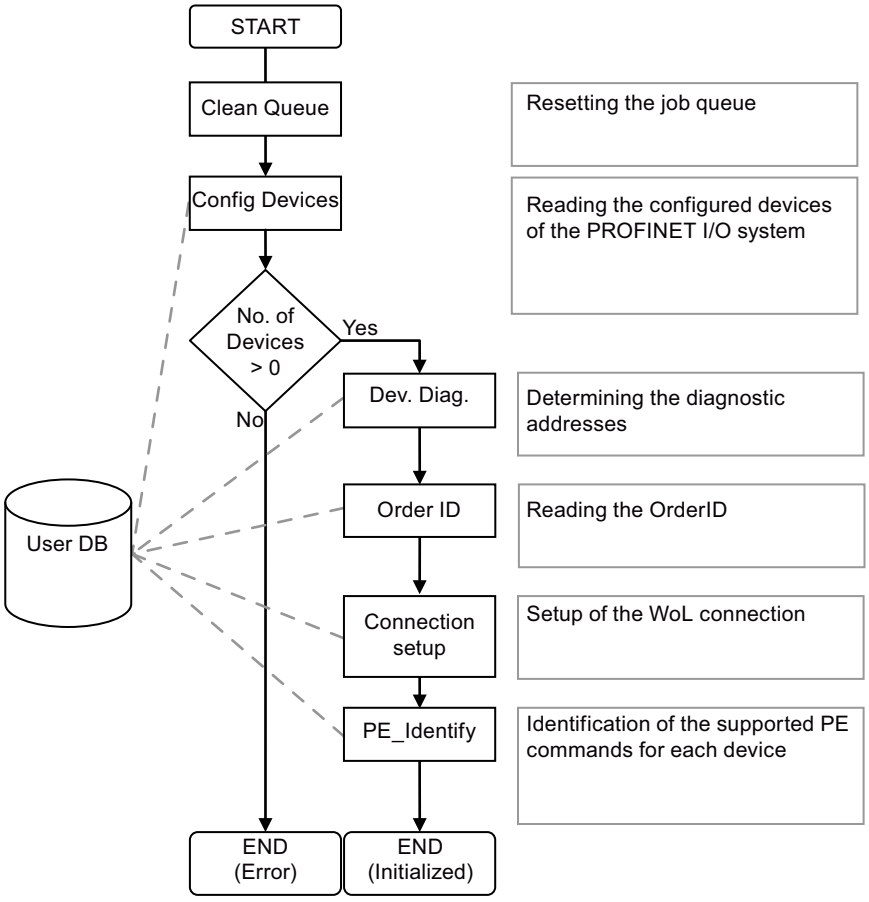
For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter COM_RST

Sequence of the initialization routine

You start the initialization of the "PE_WOL" instruction with the COM_RST parameter.

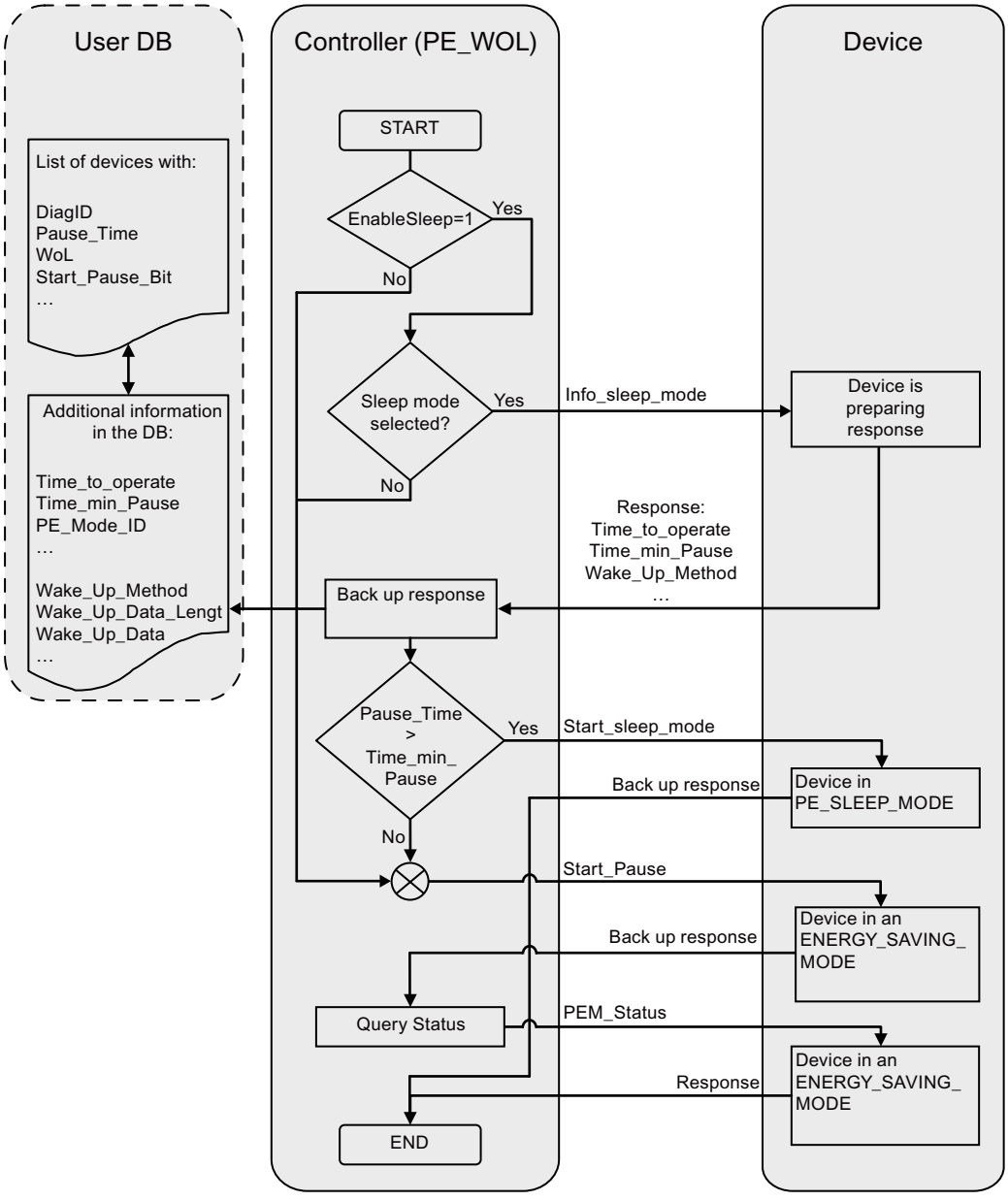
The following flow diagram shows the initialization routine.



Parameter START

Sequence of the CmdStartPause command

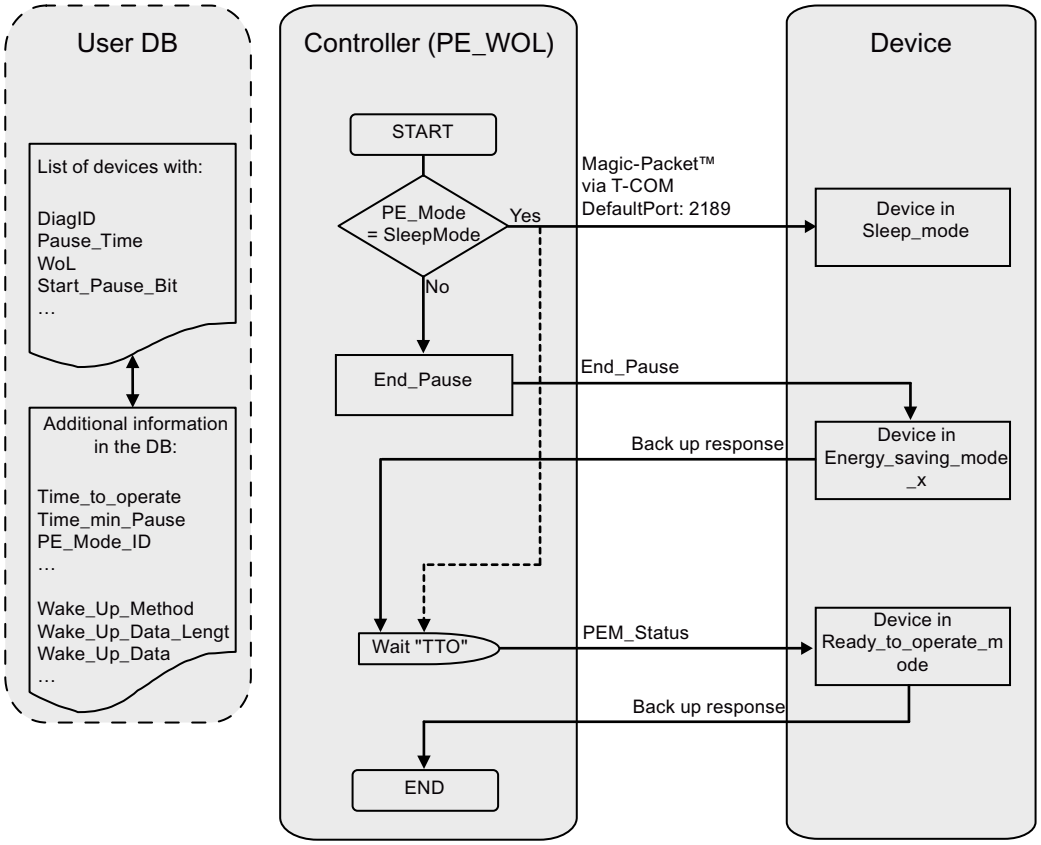
The flow diagram shows the internally used functions and interaction with a device when executing the CmdStartPause command.



Parameter END

Sequence of the CmdEndPause command

The flow diagram shows the internally used functions and interaction with a device when executing the CmdEndPause command.



PENERGY parameter

Data block at the PENERGY parameter

The user DB for the "PE_WOL" PROFIenergy instruction represents a data base for processing multiple devices.

The data block is usually divided into two parts. These are:

- Header 110 bytes
- Device section for a maximum of 256 devices each with 100 bytes (Device). These include:
  - Device-specific data (Device)
  - PROFIenergy-specific data (PE)
  - Data for the job processing (Task)
  - User data (UserData)

The data block operates with optimized access.

Connection parameter "Connection"

The "PE_WOL" instruction reserves a connection resource in the area of "Open User Communication". This is used as a UDP connection. The following parameters must be defined in the data block for this:

- Connection ID ("Connection.id" parameter)
  The connection ID is an integer within the range of 1 to 4095. It is used to identify the communication resources allocated by the firmware, for example the send and receive buffers.
  The connection ID must be unique throughout the CPU.
- Port number that is used for the "Wake on LAN" function ("Header.PortNo" parameter)
  Number of the UDP port via which a "Wake On LAN" packet is sent. These port numbers are part of the communication resources, which are identified and provided by the firmware through the connection ID. The default setting for port 2189 used here is currently not assigned by IANA. The port number is transferred to the connection configuration and applied to parameter "Connection.local_tsap_id[1]". The port number and the length of the remote port are defined using parameters "Connection.rem_tsap_id[1]" and "Connection.rem_tsap_id_LEN" and must be assigned manually.
- Interface ID ("Connection.local_device_id" parameter)
  The interface ID is also part of the connection description. This ID identifies the CPU interface to use for this connection. There are several possible valid values.
  However, these need to be adapted to the employed CPU and interface:
  - B#16#01 for S7-1500 CPUs, ET 200S CPU, or WinAC RTX with Ethernet interface in subslot IF1
  - B#16#02 for CPU 315(F)-2PN/DP or CPU 317(F)-2PN/DP
  - B#16#03 for CPU 319(F)-2PN/DP
  - B#16#05 for CPU 41x(F)-3PN/DP
  - B#16#06 for WinAC RTX with Ethernet interface in subslot IF2
  - B#16#0B for WinAC RTX with Ethernet interface in subslot IF3
  - B#16#0F for WinAC RTX with Ethernet interface in subslot IF4

Structure of the data block

The data block has the following structure:

Name	Data type	Offset	Comment
Header	PE_HEADER	-	Header information
Update (1)	BOOL	-	Signal for indicating a change in the data area. • True= indicates a change by the user. • False = indicates the application of changes.
Initialized	BOOL	-	Signal indicating the completed initialization. • True=Initialization finished. • False= indicates that the block is not initialized.
LinkUp	BOOL	-	Indicates successful configuration of the Ethernet interface. • True=interface ready for use. • False=interface not yet configured.
LinkDown	BOOL	-	Indicates an unconfigured interface. • True=interface is not yet configured. • False=interface is currently being configured or is now configured.
PROFINET_ID (1)	INT	-	ID of the PROFINET I/O system
Reserved	ARRAY [1..37] OF BYTE	-	Reserved
LastDeviceID	INT	-	Contains the highest Device-ID in this PROFINET I/O system.
PortNo (1)	WORD	-	Port number that is used for the "Wake on LAN" function (default= 2189).
Connection	TCON_Param	-	Contains the connection configuration of the "Wake on LAN" connection.

Name	Data type	Offset	Comment
BLOCK_LENGTH	UInt	-	Length of the structure (always B#16#40).
ID (1)	CONN_OUC	-	Connection ID
CONNECTION_TYPE (1)	USINT	-	Connection type (UDP = B#16#13)
ACTIVE_EST (1)	BOOL	-	Active connection establishment (always passive for UDP)
LOCAL_DEVICE_ID (1)	USINT	-	Contains the interface ID (CPU dependent)
LOCAL_TSAP_ID_LEN (1)	USINT	-	Contains the length in bytes of the own/local UDP port
REM_SUBNET_ID_LEN (1)	USINT	-	Unused (always B#16#00)
REM_STADDR_LEN (1)	USINT	-	Contains the length of the remote IP address or B#16#00.
REM_TSAP_ID_LEN (1)	USINT	-	Contains the length in bytes of the remote UDP port. The length specification must be entered manually.
NEXT_STADDR_LEN (1)	USINT	-	Contains the length of the default router address (not relevant).
LOCAL_TSAP_ID (1)	ARRAY[1..16] OF BYTE	-	Contains the own/local port number. The value is applied from parameter PortNo during initialization.
REM_SUBNET_ID (1)	ARRAY[1..6] OF USINT	-	Unused (always B#16#00)
REM_STADDR (1)	ARRAY[1..6] OF USINT	-	Contains the remote IP address.
REM_TSAP_ID (1)	ARRAY[1..16] OF BYTE	-	Contains the remote UDP port number. The port number must be entered manually.
NEXT_STADDR (1)	ARRAY[1..6] OF BYTE	-	irrelevant
SPARE (1)	WORD	-	
Device	ARRAY[1..256] OF PE_DEVICE	-	Array of devices
Device	PE_DEV	-	Contains data for each device
DeviceID	HW_DEVICE	-	Hardware identifier of the device. Assigned by the hardware configuration.
PE_EntityID	HW_IO	-	Hardware identifier of the PROFIenergy entity. Assigned by the hardware configuration.
MACAdr	ARRAY[1..6] OF BYTE	-	Contains the MAC address of the device.
IPAdr	ARRAY[1..4] OF BYTE	-	Contains the IP address of the device.
OrderID_MxLen	Byte	-	Contains the maximum length of the OrderID.
OrderId_ActLen	Byte		Contains the actual length of the OrderID.
OrderID_Data	ARRAY[1..20] OF CHAR		Contains the OrderID of the device.
PE	PE_PE	-	PROFIenergy-specific data.

Name	Data type	Offset	Comment
ModeID	BYTE	-	PE_MODE_ID according to PROFIenergy specification.
Result	BYTE	-	PE ErrorCode according to PROFIenergy specification.
PauseTime (1)	TIME	-	Contains the idle time in ms.
TimeToPause	TIME	-	Contains the time it takes for the device to go into Pause mode.
TimeToOperate	TIME	-	Contains the time it takes for the device to go into operating mode.
MinSleepTime	TIME	-	Contains the minimum time of the device in PE_SLEEP_MODE.
SleepToOperate	TIME	-	Contains the time it takes for the device to go from PE_SLEEP_MODE into operating mode.
StatusOperate	BOOL	-	Indicates the operating mode of the device.
StatusPause	BOOL	-	Indicates the device is in Pause mode.
StatusSleep	BOOL	-	Indicates the PE_SLEEP_MODE of the device.
StatusTransitOK	BOOL	-	Indicates the transition from one energy state to another.
StatusInTransit	BOOL	-	Indicates a current state transition.
StatusTransitNOK	BOOL	-	Indicates that the state change was not successful.
StatusError	BOOL	-	Indicates an error with the device.
StatusRetryEx	BOOL	-	Indicates an unsuccessful execution of a command. No more attempts will be made to execute this command.
CmdStartPause (1)	BOOL	-	Places a START_PAUSE command for this device in the queue.
CmdEndPause (1)	BOOL	-	Places a END_PAUSE command for this device in the queue.
CmdUpdateStatus (1)	BOOL	-	Places a PEM_STATUS command for this device in the queue.
EnableSleep (1)	BOOL	-	Allows PE_SLEEP_MODE for this device. • True= device should go to PE_SLEEP_MODE if idle time is sufficient • False=device may not go into PE_SLEEP_MODE.
Services	WORD	-	Shows all supported PROFIenergy services.
UserData (2)	ARRAY[1..24] OF BYTE	-	User-defined data
Task	PE_TASK	-	Job processing

Name		Data type	Offset	Comment
	Cmd	BYTE	-	Internal bits for job processing
	CmdJ	BYTE	-	Internal bits for job processing
	TimeStart	BOOL	-	Starts a delay time.
	TimeStarted	BOOL	-	The delay time has just started.
	TimeDone	BOOL	-	Shows the expiration of the delay time.
	Done	BOOL	-	Indicates that this job has been completed.
	DelayedCmd	BOOL	-	Indicates that a delayed command is still pending.
	IsV1_0	BOOL	-	Indicates that this device is a Spec. V1.0 device.
	IsWakeOnLAN	BOOL	-	Indicates that this device is woken up by "Wake On LAN".
	RetryCount	BYTE	-	Repetition counter for PE_COMMANDS
	Duration	TIME	-	Contains the value for the delay in ms.
	StartTime	TIME	-	Contains the start time for the delay time.
	MachineState	INT	-	Contains the internal status of the job.

(1) To be filled in by the user.

(2) Available for use by the user.

Parameter STATUS

Parameter STATUS

The output value at the STATUS parameter is divided into three areas:

- Bits 31 to 24: MESSAGE
- Bit 23 to 16: LOCATION
- Bit 15 to 0: INFORMATION

You can learn the meaning of each error code for the three areas in the following tables:

Table 11-55 Possible values for MESSAGE

Error code (W#16#...)	Description
00	No error.
50	Instruction initialized.
51	Determining configuration of the PROFINET I/O system.
52	The instruction could not determine any configured devices in the PROFINET I/O system.
53	Determining the logical addresses of configured devices.
54	Reading interface information out of the devices.
55	Determining I&M data (data record 0 only) of the configured devices.
56	Configuring PROFINET interface for sending the "Wake on LAN" MagicPaket™ via UDP.
57	Determining PROFIenergy capabilities of the connected devices.
62	Invalid PROFINET I/O system ID detected. The causative number is displayed in the INFORMATION field.

Error code (W#16#...)	Description
70	The instruction is initiated and processing jobs. The value in the INFORMATION field contains the number of currently active jobs.
80	The instruction has lost the initialization during processing of jobs. This generally occurs when either the instance DB or the user DB has been reloaded.
FF	Unknown error has occurred.

Table 11-56 Possible values for LOCATION

Error code (W#16#...)	Description
00	The instruction is either not initialized or it is idle.
70	The instruction is waiting for jobs.
71	The instruction has appended a job to the job list.
72	The instruction is preparing to send a job.
73	The instruction is sending a job to a device.
74	The instruction is waiting for a response from the device.
75	The instruction is evaluating the response of the device.
76	The instruction is removing the job from the job list.
FF	Unknown error has occurred.

Table 11-57 Possible values for INFORMATION

Error code (W#16#...)	Description
0000	No additional information available and no jobs active.
0001 -00FF	1 – 255 jobs are currently being processed.
7000	Re-initialization with COM_RST started but not yet completed.
8001	Error at parameter 1
8002	Error at parameter 2
8003	Error at parameter 3
8004	Error at parameter 4
8005	Error at parameter 5. This error is reported if there is no interconnection or an invalid interconnection to the user DB. Possible causes: • The user DB is too small. • The user DB is write-protected. • There is no user DB in the RAM. • The user DB is invalid for the employed CPU.

Error code (W#16#...)	Description
8085 to 80CE	Error when configuring the connection. The error messages of the internally used TCON instruction are output. The description of the error messages can be found in the table for the STATUS (Page 3664) parameter.
8100	An attempt has been made to place more than the maximum of 256 possible jobs. This is a temporary error, which is resolved by completing a few jobs. The placed jobs are not accepted and must be placed again.
8200	An attempt has been made to send an invalid or an unsupported PROFIenergy command (PE_COMMAND).
8400	Connection ID outside the permitted range. Initialization aborted. Check the ID of the connection configuration for "Wake on LAN". See data block at parameter PENERGY > Header > Connection > ID.
84xx	A communication error has occurred. The number of the device that generated the error is output to "xx".
85xx	An error has been returned from device xx. The number of the device that generated the error is output to "xx".
8600	The requested wake-up method is currently not supported.
FFFF	Unknown error has occurred.

See also

PENERGY parameter (Page 3170)

PROFIenergy commands

Structure of the message frames

Structure of the response frame in accordance with the PROFIenergy profile

The table below shows the basic structure of the response frame in accordance with the PROFIenergy profile. The response frame consists of a general section (Header) and a specific section (Service Data Response). The contents of the specific section of the response frame can be found in the description of the relevant PROFIenergy command.

Block definition	Attributes	Value	Data type	Description
BlockHeader	BlockType	801_{hex}	WORD	
	BlockLength		WORD	Number of bytes without consideration of the fields BlockType and BlockLength.
	BlockVersionHigh	1_{hex}	BYTE	
	BlockVersionLow	0_{hex}	BYTE	

Block definition	Attributes	Value	Data type	Description
Response Header	Service_Request_ID	1_{hex} to FF $_{hex}$	BYTE	ID of the PI command executed. The ID of the PE command processed by the PE entity is returned in the response frame: • 01: Start_Pause • 02: End_Pause • 03: Query_Modes • 04: PEM_Status • 05: PE_Identify • 06 to 09: Reserved • 16: Query_Measurement • 11 to CF: Reserved • D0 to FF: Manufacturer-specific
	Request_Reference	1_{hex} to FF $_{hex}$	BYTE	Unique number to identify the query/response pair (returned by the server in the response).
Service Header Response	Status	1_{hex} to FF $_{hex}$	BYTE	Information whether or not the PI command was executed: • 00: Reserved • 01: Completed • 02: Completed with error • 03: Data incomplete • 04 to CF: Reserved • D0 to FF: Depends on the Service_Request_ID
	Data_Structure_ Identifier_RS	1_{hex} to FF $_{hex}$	BYTE	• 00: Reserved • 01 to FF: Data structure depends on the Service_Request_ID • 0xFF - Error
Service Data Response				Depends on the Service-Request-ID: • For the service request IDs, see the CMD and CMD_MODIFIER parameters of the instruction "PE_CMD (Page 3159)". • The specific contents of the response frame can be found in the description of the PE command (see for example the "Start_Pause (Page 3179)" command).

PI Command "Start_Pause"

Description

Use the PE command "Start_Pause" to start the energy-saving mode. The command Start_Pause can be used to:

- Switch the PE from "ready" state (PE_ready_to_operate) to an energy-saving mode (PE_energy_saving_mode).
- The PE entity can automatically switch between the energy-saving modes.
  Energy consumption can increase or decrease when you switch energy-saving mode.

Calling the PI command "Start_Pause"

The command "Start_Pause" is called with the instruction "PE_CMD (Page 3159)" with the following parameters:

Parameter	Value	Description
CMD	1	Call of PE command "Start_Pause".
CMD_MODIFIER	0	There are no additional specifications of the command call for command "Start_Pause".
CMD_PARA_LEN	4	Length of the CMD_PARA parameter of 4 bytes.
CMD_PARA	VARIANT	VARIANT pointer to the value for "Pause_Time" (TIME).

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see instruction "PE_CMD (Page 3159)"):

Attribute	Value	Data type	Description
PE_Mode_ID	1 to 255	BYTE	Identification number of the energy-saving mode
Reserved	0	BYTE	-

PI Command "End_Pause"

Description

The PE command "End_Pause" is used to end the energy-saving mode of the PE entity.

Calling the PE command "End_Pause"

The command "End_Pause" is called with the instruction "PE_CMD (Page 3159)" with the following parameters:

Parameter	Value	Description
CMD	2	Call of PE command "End_Pause".
CMD_MODIFIER	0	There are no additional specifications of the command call for command "End_Pause".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 3159)"):

Attribute	Value	Data type	Description
Time_to_operate	-	DWORD	Expected time it takes to switch to the "PE_ready_to_operate" mode.

PI command "Query_modes" - "List_Energy_Saving_Modes"

Description

Use PE command "Query_modes" and sub-command (modifier) "List_Energy_Saving_Modes" to output all energy-saving modes (PE_Mode_ID) supported by the PE entity.

The query result is written in the form of a response frame to the data block referenced by theRESPONSE_DATA parameter.

Calling the PE command "Query_modes" - "List_Energy_Saving_Modes"

The command "List_Energy_Saving_Modes" is called with the instruction "PE_CMD (Page 3159)" with the following parameters:

Parameter	Value	Description
CMD	3	Call of PE command "Query_modes".
CMD_MODIFIER	1	Specification of the command call: Select the sub-command "List_Energy_Saving_Modes" to output the number and types of energy-saving modes supported.
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 3159)"):

Attribute	Value	Data type	Description
Number_of_PE_Mode_IDs	1	BYTE	The number of energy-saving modes.
PE_Mode_IDs	-	Array [...] of BYTE	Array with the IDs of the supported energy-saving modes. The meaning of the individual IDs depends on the PE entity.

PI command "Query_modes" - "Get_Mode"

Description

You use the PE command "Query_modes" and the sub-command (modifier) "Get_Mode" to output the attributes of the energy-saving mode which is currently enabled.

Calling the PE command "Query_modes" - "Get_Mode"

The call of the command with the instruction "PE_CMD" occurs with the following parameters:

Parameter	Value	Description
CMD	3	Call of the PE command "Query_modes"
CMD_MODIFIER	2	Specification of the command call: Select the sub-command "Get_Mode" to output the status of the mode which is currently enabled.
CMD_PARA_LEN	1	Length of the CMD_PARA parameter of 1 byte.
CMD_PARA	VARIANT	VARIANT pointer to the value for PE_MODE_ID.

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 3159)"):

Attribute	Value	Data type	Description
PE_Mode_ID	• 0 "PE_power_off" mode • 1...254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	ID of the energy-saving mode currently enabled.
PE_Mode_Attributes	Bit 0: • = 0: Only static energy consumer and time values are available. • = 1: Dynamic energy consumer and time values are available. Bit 1 to 7: • Reserved	BYTE	
Time_min_Pause [1]	Time difference without date	DWORD	Minimum pause time for the PI mode. The minimum pause time is the sum of the values of the following attributes: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay See the description "PAUSE_TIME parameter" of the instruction "PE_START_END: Start and exit energy-saving mode (Page 3154)".
Time_to_Pause [1]	Time difference without date	DWORD	Switch off time: Duration from call of energy-saving mode to start of energy-saving mode (transition duration of PE_ready_to_operate to PE_energy_saving_mode). The switch-off time depends on the PE entity.
Time_to_operate [1]	Time difference without date	DWORD	Switch-on time: Duration of the transition from energy-saving mode (PE_energy_saving_mode) to ready to operate mode (PE_ready_to_operate). The PE entity calculates the duration dynamically in the output operation.
Time_min_length_of_stay [1]		DWORD	Minimum active period of the energy-saving mode on the PE entity.
Time_max_length_of_stay [1]		DWORD	Maximum active period of the energy-saving mode on the PE entity.
Mode_Power_Consumption [2]		REAL	Power consumption of the PE entity in active energy-saving mode. Unit: kW

Attribute	Value	Data type	Description
Energy_ Consumption_ to_pause [2]		REAL	Energy consumption of the PE entity during transition from ready-to-operate mode (PE_ready_to_operate) to energy-saving mode (PE_energy_saving_mode) Unit: kWh
Energy_ Consumption_ to_operate [2]		REAL	Energy consumption of the PE entity during transition from energy-saving mode (PE_energy_saving_mode) to ready-to-operate mode (PE_ready_to_operate) Unit: kWh
[1] If the duration is infinite, 0xFFFFFFFF will be output. If the duration is zero, "0" will be output. [2] If the data for energy and power consumption is not defined by the PE entity, "0.0" will be output as the value.			

PI command "PEM_Status"

Description

The PE command "PEM_Status" is used to query the status of a PE entity energy-saving mode that is currently enabled.

Calling the PE command "PEM_Status"

The command "PEM_Status" is called with the instruction "PE_CMD" with the following parameters:

Parameter	Value	Description
CMD	4	Call of PE command "PEM_Status".
CMD_MODIFIER	0	There are no other specifications of the command call for command "PEM_Status".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 3159)"):

Attribute	Value	Data type	Description
PE_Mode_ID_ Source	• 0 "PE_power_off" mode • 1 to 254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	Mode of the PE entity before a PE command is sent.
PE_Mode_ID_ Destination	• 0 "PE_power_off" mode • 1 to 254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	Mode of the PE entity after a PE command is executed.
Time_to_operate	Time difference without date	DWORD	Switch-on time: Duration of the transition from energy-saving mode (PE_energy_saving_mode) to ready to operate mode (PE_ready_to_operate). The PE entity calculates the duration dynamically during output.
Remaining_time_to_ destination	Time difference without date	DWORD	Remaining time to switch to another mode.
Mode_Power_ Consumption		REAL	Power consumption of the PE entity in active energy-saving mode. Unit: kW
Energy_ Consumption_ to_Destination		REAL	Energy consumption for the current PI transition Unit: kWh
Energy_ Consumption_ to_operate		REAL	Energy consumption of the PE entity during transition from energy-saving mode (PE_energy-saving mode) to ready-to-operate mode (PE_ready_to_operate) Unit: kWh

PI command "PE_Identify"

Description

The PE command "PE_Identify" is used to read out the number and description of PE commands supported by the PE entity. The number of specific commands supported depends on the PE entity. As PE_Identify is itself a PE command, at least three supported PE commands will be output upon a positive response: Start_Pause, End_Pause and PE_Identify.

Calling the PE command "PE_Identify"

The command "PE_Identify" is called with the instruction "PE_CMD (Page 3159)" with the following parameters:

Parameter	Value	Description
CMD	5	Call of the command "PE_Identify".
CMD_MODIFIER	0	There are no other specifications of the command call for command "PE_Identify".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 3159)"):

Attribute	Value	Data type	Description
Count [1]	6	BYTE	Number of supported PROFIenergy commands
Start_Pause	1	BYTE	First PE command supported (Service_Request_ID)
End_Pause	2	BYTE	...
Query_Modes	3	BYTE	...
PEM_Status	4	BYTE	...
PE_Identify	5	BYTE	...
Query_Measurement	16	BYTE	Last PE command supported (Service_Request_ID)

[1] The number of commands supported is manufacturer-specific and depends on the PE entity used. The values given are an example of a response frame where all 6 PE commands are supported.

PI Command "Query_Measurement" - "Get_Measurement_list"

Description

Use the PE command "Query_Measurement" and sub-command (modifier) "Get_measurement_list" to query the specific measured values supported by the PE entity. The supported measured values are output as a list in the data block referenced by the RESPONSE_DATA parameter.

Calling the PE command "Query_Measurement" - "Get_Measurement_list"

The command is called with the instruction "PE_CMD (Page 3159)" with the following parameters:

Parameter	Value	Description
CMD	16	Call of the command "Query_Measurement".
CMD_MODIFIER	1	Specification of the command call: Select the sub-command "Get_Measurement_List" to output a list of supported measured values.
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 3159)"):

Attribute	Value	Data type	Description
Count	-	BYTE	Number of measurement IDs
reserved	-	BYTE	
...			
Measurement_ID	-	WORD	First supported Measurement_ID The Measurement_ID is manufacturer-specific. For more information, refer to the manual of the respective PE entity.
Accuracy_Domain	-	BYTE	See "accuracy domain" table.
Accuracy_Class	-	BYTE	See "accuracy classes" table.
Range	-	REAL	Specifies the scale end value of the measured value (only for accuracy domain 1). The attribute range uses the same unit as defined using attribute Measurement_ID (only one unit is used for each Measurement_ID).
...			
Measurement_ID	-	WORD	Last supported Measurement_ID
Accuracy_Domain	-	BYTE	See "accuracy domain" table.
Accuracy_Class	-	BYTE	See "accuracy classes" table.
Range	-	REAL	Specifies the scale end value of the measurement value (only for accuracy domain 1). The attribute range uses the same unit as defined using attribute Measurement_ID (only one unit is used for each Measurement_ID).

Accuracy domains

Accuracy domain	Description
0	Reserved
1	The accuracy deviation is output in percentages of the scale end value. The percentage of the possible deviation is divided into accuracy classes (see table: Accuracy classes of the accuracy domains 1 and 2).

Accuracy domain	Description
2	The accuracy deviation is output in percentages of the current measurement value. The percentage of the possible deviation is divided into accuracy classes (see table: Accuracy classes of the accuracy domains 1 and 2).
3	The measuring accuracy adheres to the standard IEC 61557-12. The function performance classes for performance measurement and monitoring devices (PMD) without external sensors, and the system performance classes for PMD with external sensors, are coded as set out in the "Accuracy classes of accuracy domain 3" table.
4	The entry of the accuracy adheres to the standard EN 50470-3, chapter 8 (see also table: Accuracy classes of accuracy domain 4).

Accuracy classes

Table 11-58 Accuracy classes of the accuracy domains 1 and 2

Accuracy class	0	1	2	3	4	5	6	7	8
Meaning	Reserved	0.01%	0.02%	0.05%	0.1%	0.2%	0.5%	1%	1.5%

Accuracy class	9	10	11	12	13	14	15	>15
Meaning	2%	2.5%	3%	5%	10%	20%	>20%	Not defined

Table 11-59 Accuracy classes of accuracy domain 3

Accuracy class	0	1	2	3	4	5	6	7	8
Meaning	Reserved	0,02	0,05	0,1	0,2	0,5	1	1,5	2

Accuracy class	9	10	11	12	13	14	>13
Meaning	2,5	3	5	10	20	20%	Not defined

Table 11-60 Accuracy classes of accuracy domain 4

Accuracy class	0	1	2	3	4	5	6	>7
Meaning	Reserved	0,5	1,0	1,5	2,0	2,5	3,0	Not defined

PI Command "Query_Measurement" - "Get_Measurement_values"

Description

Use the PE command "Query_Measurement" and sub-command (modifier) "Get_measurement_values" to output a list of measured values supported by the PE entity. The measured values are output as a list in the data block referenced by the RESPONSE_DATA parameter.

Calling the PE command "Query_Measurement" - "Get_Measurement_values"

The command is called with the instruction "PE_CMD (Page 3159)" with the following parameters:

Parameter	Value	Description
CMD	16	Call of the command "Query_Measurement".
CMD_MODIFIER	2	Specification of the command call: Select the command "Get_Measurement_Values" to output a list of supported measurement values.
CMD_PARA_LEN	0	Depends on the number of measurement values. The length of the parameters results from the attribute count and the sum of the lengths of the attributes for the transferred measurement values.
CMD_PARA	VARIANT	VARIANT pointer to data structure with list of measured values to be queried (see "CMD_PARA parameter").

Parameter CMD_PARA

The structure specified with the VARIANT pointer at the CMD_PARA parameter must have the following setup:

Attribute	Value	Data type	Description
Count	-	BYTE	Number of measured values (Measurement-IDs)
reserved	0	BYTE	Not used
Measurement_ID	-	WORD	First measured value queried
...			
Measurement_ID	-	WORD	Last measured value queried

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 3159)"):

Attribute	Value	Data type	Description
Count [1]	-	BYTE	Number of measured values (Measurement-IDs)
reserved	0	BYTE	Not used
Length_of_Structure	2 to 65535	WORD	Length of the structure in bytes

Attribute	Value	Data type	Description
Measurement_Data_Structure_ID	1 = simple value	BYTE	Defines the following structure.
Measurement_ID	0 to 65535	WORD	ID of the supported measurement value.
Status_of_ Measurement_Value	1 to 3	BYTE	Status of the measurement value: • 1: Valid • 2: Not supported • 3: Invalid
Transmission_Data_Type	-	REAL	
End_of_demand	-	TOD	Optional time stamping with data type TimeOfDay.
...			
Length_of_Structure	-	WORD	Length of the structure in bytes
Measurement_Data_Structure_ID	-	BYTE	Defines the following structure.
Measurement_ID	-	WORD	ID of the supported measurement value.
Status_of_Measurement_Value	-	BYTE	Status of the measurement value: • 1: Valid • 2: Not supported • 3: Invalid
Transmission_Data_Type	-	REAL	
End_of_demand	-	TOD	Optional time stamping with data type TimeOfDay.

[1] If the data length of the measurement values queried exceeds the size of the PDU (Protocol Data Unit) of the protocol layer, the data will not be completely transferred and only the supported measurement values will be output.

iDevice / iSlave

PE_I_DEV: Control PROFIenergy commands in the I-Device

Description

The instruction "PE_I_DEV" is used to process the PROFIenergy profile in the intelligent IO device (iDevice). The functions which are executed by the firmware in standard PROFIenergy-compatible IO devices, such as the ET 200S, are implemented in the iDevice by the instruction "PE_I_DEV" and the corresponding auxiliary blocks:

- The instruction "PE_I_DEV" is called cyclically by the user program of the iDevice and receives all PROFIenergy commands.
- The PROFIenergy response is generated by configuring an auxiliary block. The response in the pause is fully programmable. The response data must be provided within 10 seconds; otherwise, "State conflict 0x80B5" will occur at the STATUS parameter of the instruction in the IO controller.

No specific knowledge of the PROFIenergy standard is required to use the instruction.

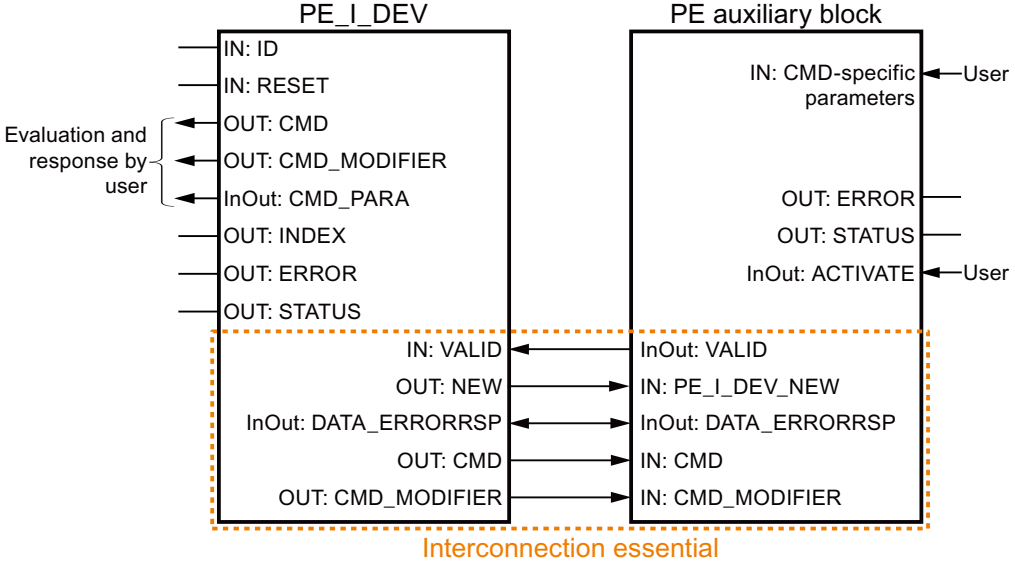
PROFIenergy auxiliary blocks (PE auxiliary blocks)

The PE auxiliary blocks are used to generate the response frame. Enter the response data (in plain text) at the input parameters of the relevant block.

- For each PROFIenergy command, there is a corresponding auxiliary block for a positive response:
  - PE command "Start_Pause": PE_Start_RSP (Page 3194)
  - PE command "End_Pause": PE_End_RSP (Page 3196)
  - PE command "Query_modes" - "List_Energy_Saving_Modes": PE_List_Modes_RSP (Page 3197)
  - PE command "Query_modes" - "Get_Mode": PE_Get_Mode_RSP (Page 3198)
  - PE command "PEM_Status": PE_PEM_Status_RSP (Page 3200)
  - PE command "PE_Identify": PE_Identify_RSP (Page 3202)
  - PE command "Query_Measurement" - "Get_Measurement_list": PE_Measurement_List_RSP (Page 3204)
  - PE command "Query_Measurement" - "Get_Measurement_values": PE_Measurement_Value_RSP (Page 3205)
- Irrespective of the PROFIenergy command, there is an additional shared auxiliary block for a negative response (see PE_Error_RSP (Page 3193)).

Interconnection of the auxiliary blocks

The instruction "PE_I_DEV" and the auxiliary blocks are coordinated. Some of the parameters are simply interconnected. The diagram below shows which of the parameters need to be interconnected.



Parameters

The following table shows the parameters of the instruction "PE_I_DEV":

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L or constant	Resets the instruction.
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Address of the transfer area, which is supplied in the IO controller with the data for PROFIenergy. The hardware identifier can be found in the system constants.
VALID	Input	BOOL	I, Q, M, D, L or constant	The response data for the PROFIenergy controller is ready and can be sent.
CMD_PARA	Output	VARIANT	I, Q, M, D, L	Parameters for: • Get mode: PE_mode_ID • Get measurement values: List of Measurement_IDs (list of IDs of the tags to be read; either one tag or multiple tags may be read at any given time). Maximum length: 234 bytes
DATA_ERRORRSP	InOut	VARIANT	I, Q, M, D, L	Pointer to the data area containing the acknowledgment data for the PROFIenergy controller. This must correspond to the pointer which is also used with the auxiliary blocks.
INDEX	Output	INT	I, Q, M, D, L	Data record number of the PROFIenergy record (set at 0x80A0)
CMD	Output	INT	I, Q, M, D, L	Service-Request-ID of the PROFIenergy command in accordance with the PROFIenergy profile (see "CMD and CMD_MODIFIER parameters"). Further PE commands (Service-Request-IDs) are possible following extensions of the PROFIenergy profile.
CMD_ MODIFIER	Output	INT	I, Q, M, D, L	PROFIenergy sub-command: • Only when CMD=3 or CMD=16; see "CMD and CMD_MODIFIER parameters". • For all other commands: "0". Further sub-commands are possible following extensions of the PROFIenergy profile.
NEW	Output	BOOL	I, Q, M, D, L	New data available from the PROFIenergy controller.
ERROR	Output	BOOL	I, Q, M, D, L	Command completed with error.
STATUS	Output	DWORD	I, Q, M, D, L	Error information (see "STATUS parameter").

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameters CMD and CMD_MODIFIER

CMD	CMD_ MODIFIER	PROFIenergy command	Description
01	0	Start_Pause	Start energy-saving mode or switch to a different energy-saving mode.
02	0	End_Pause	End energy-saving mode.
03	1	Query_Modes - List energy saving Modes	Output the energy-saving modes supported.
	2	Query_Modes - Get Mode	Output attributes of the energy-saving mode currently enabled.
04	0	PEM_Status	Query status of energy-saving mode.
05	0	PE_Identify	Read out the number and description of PE commands supported.
16	1	Query_Measurement - Get_Measurement_List	List of measured values supported by the PE entity.
	2	Query_Measurement - Get_Measurement_Values	Output of the measured values of the PE entity.

Parameter STATUS

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error during reading of data record • B#16#**DF**: Error during writing of data record • B#16#**C0**: Error message from the "PE_I_DEV" instruction or the internally used "PRVREC (Page 3130)" and "RCVREC (Page 3128)" communication instructions.
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific
STATUS[3]	Error_Code_1	Error ID (when Error_Decode = **80**): • **B1**: Write length error (error in write length or insufficient length information with the data type VARIANT).
STATUS[4]	Error_Code_2	For PROFINET errors: Output of IO controller error message If no PROFINET error has occurred, the value in STATUS[4] = "0".

Note

Error messages of the "PRVREC" and "RCVREC" instructions

The "PE_I_DEV" instruction uses the "PRVREC (Page 3130)" and "RCVREC (Page 3128)" instructions for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

For the meaning of the error codes of the "PRVREC" and "RCVREC" instructions, see the description of the corresponding STATUS (Page 3078) parameter.

See also

WRREC: Write data record (Page 3071)

RDREC: Read data record (Page 3069)

Description of PROFIenergy (Page 3152)

Auxiliary blocks of the instruction PE_I_DEV

PE_Error_RSP: Generate negative answer to command

Description

The auxiliary block "PE_Error_RSP" (Response with failure) generates a negative response if the command requested is not supported (either in general or temporarily). Response generation does not depend on the command requested.

Parameter

The following table shows the parameters of the "PE_Error_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
ERROR_CODE	Input	BYTE	I, Q, M, D, L or constant	Error number
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 3189)" instruction.

Parameter	Declaration	Data type	Memory area	Description
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 3189)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

PE_Start_RSP: Generate answer to command at start of pause

Description

The auxiliary block "PE_Start_RSP" (Start Pause) generates the response to the PE command Start_Pause (Page 3179). The instruction returns the energy-saving mode to which the device has switched (PE_MODE_ID parameter).

If the response differs depending on the length of the pause, you can return this fact in the feedback on the energy-saving mode (PE_Mode_ID = 1 for a brief pause, PE_Mode_ID = 2 for a longer pause, etc.).

Parameter

The following table shows the parameters of the "PE_Start_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 3189)" instruction.
PE_MODE_ID	Input	BYTE	I, Q, M, D, L or constant	PE mode that the process assumes
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 3189)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 3189)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

PE_End_RSP: Generate answer to command at end of pause

Description

The auxiliary block "PE_End_RSP" generates the response to the PE command End_Pause (Page 3179). The response returned is the time required to switch from the current mode to "Ready_To_Operate" mode.

Parameter

The following table shows the parameters of the "PE_End_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 3189)" instruction.
Time_to_Operate	Input	DWORD	I, Q, M, D, L or constant	Time required to switch from the current mode to "Ready_To_Operate".
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 3189)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 3189)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

PE_List_Modes_RSP: Generate queried energy savings modes as answer

Description

The auxiliary block "PE_List_Modes_RSP" generates the response to the PE command List_Energy_Saving_Modes (Page 3180). The response generated includes the number and the IDs of energy-saving modes supported.

Parameter

The following table shows the parameters of the "PE_List_Modes_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 3189)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Number_of_PE_Mode_IDs	Input	BYTE	I, Q, M, D, L or constant	Number of energy-saving modes supported. Permitted values: 1 to 254
PE_MODE_ID	Input	VARIANT	I, Q, M, D, L	Points to the area in which the IDs of the energy-saving modes supported (PE_Mode_ID) are stored. Valid range: 1 to 254.

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 3189)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 3189)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

PE_Get_Mode_RSP: Generate queried energy data as answer

Description

The auxiliary block "PE_Get_Mode_RSP" generates the response to the command Get_Mode (Page 3181). The response frame contains the time, performance, or energy data of an energy-saving mode.

Parameter

The following table shows the parameters of the "PE_Get_Mode_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 3189)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
PE_Mode_ID	Input	BYTE	I, Q, M, D, L or constant	ID of the energy-saving mode currently enabled.
Time_min_Pause	Input	DWORD	I, Q, M, D, L or constant	Minimum pause duration for this PE energy-saving mode. This is the sum of the following three parameters: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay
Time_to_Pause	Input	DWORD	I, Q, M, D, L or constant	Time between the edge at the START parameter (see "PE_I_DEV (Page 3189)") and the requested PE energy-saving mode being reached.
Time_to_Operate	Input	DWORD	I, Q, M, D, L or constant	Maximum switch-on time to "PE_ready_to_operate". The "Time_to_operate" parameter can be used directly for the relevant calculations. The value may be a static maximum, or be calculated dynamically by the PE entity.
Time_min_Lenght_of_stay	Input	DWORD	I, Q, M, D, L or constant	Minimum dwell time of the PE entity in this PE mode.
Time_max_Lenght_of_stay	Input	DWORD	I, Q, M, D, L or constant	Maximum dwell time of the PE entity in this PE mode.
Mode_Power_Consumption	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the current PE mode in [kW].
Energy_Consum_to_Pause	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from "PE_ready_to_operate" to the current PE mode in [kWh].
Energy_Consum_to_operate	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from the current PE mode to "PE_ready_to_operate" in [kWh].

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 3189)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 3189)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

PE_PEM_Status_RSP: Generate PEM status as answer

Description

The auxiliary block "PE_PEM_Status_RSP" generates the response to the command PEM_Status (Page 3183).

Parameter

The following table shows the parameters of the "PE_PEM_Status_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 3189)" instruction.
PE_MODE_ID_ Source	Input	BYTE	I, Q, M, D, L or constant	Source and Destination for PEM_STATUS. Values: • 0x00: PE_POWER_OFF • 0x01 – 0xFE: manufacturer-specific • 0xFF: PE_READY_TO_OPERATE
PE_MODE_ID_ Destination	Input	BYTE	I, Q, M, D, L or constant	
Time_to_Operate[1]	Input	DWORD	I, Q, M, D, L or constant	Maximum switch-on time to "PE_ready_to_operate". "Time_to_operate" can be used directly for the relevant calculations. The value may be a static maximum, or be calculated dynamically by the PE entity.
Remaining_time_ to_destination[1]	Input	DWORD	I, Q, M, D, L or constant	Optional: Time remaining until the requested PE mode. Dynamic value or static maximum value
Mode_Power_ Consumption[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the current PE mode in [kW].
Energy_ Consumption_ to_Destination[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the time until the requested PE mode in [kW].
Energy_ Consumption_ to_operate[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from the current PE mode to "PE_ready_to_operate" in [kWh].
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 3189)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.

Parameter	Declaration	Data type	Memory area	Description
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 3189)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range
[1] If the time period is unlimited, the maximum value "0xFFFFFFFF" can be specified. If the time period is "Zero", "0x00" can be used. [2] If no energy consumption value has been defined, "0.0" can be specified.				

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

PE_Identify_RSP: Generate supported PROFIenergy commands as answer

Description

The auxiliary block "PE_Identify_RSP" generates the response to the command PE_Identify (Page 3185). In the response to the command, specify which PROFIenergy commands are supported. Please note that PE_IDENTIFY is itself a PE command and has to be included in the response.

Parameter

The following table shows the parameters of the "PE_Identify_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 3189)" instruction.

Parameter	Declaration	Data type	Memory area	Description
Start_Pause	Input	BOOL	I, Q, M, D, L or constant	One parameter for each of the relevant PROFIenergy commands: • 0: PE command is not supported • 1: PE command is supported
End_Pause	Input	BOOL	I, Q, M, D, L or constant	
Query_Modes	Input	BOOL	I, Q, M, D, L or constant	
PEM_Status	Input	BOOL	I, Q, M, D, L or constant	
PE_Identify	Input	BOOL	I, Q, M, D, L or constant	
Query_ Measurement	Input	BOOL	I, Q, M, D, L or constant	
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 3189)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 3189)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

PE_Measurement_List_RSP: Generate list of supported measured values as answer

Description

The auxiliary block "PE_Measurement_List_RSP" generates the response to the command Get_measurement_list (Page 3185). In the response, specify which measured values (Measurement-IDs) are supported.

Parameter

The following table shows the parameters of the "PE_Measurement_List_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 3189)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Count	Input	BYTE	I, Q, M, D, L or constant	Number of measured values supported (measurement IDs)
Measurement_List	Input	VARIANT	D	Pointer to the array with the Measurement_IDs supported. For information on the structure of the array in accordance with the PROFIenergy profile, see: PI Command "Query_Measurement" - "Get_Measurement_list" (Page 3185)
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 3189)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.

Parameter	Declaration	Data type	Memory area	Description
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 3189)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

PE_Measurement_Value_RSP: Generate queried measured values as answer

Description

The auxiliary block "PE_Measurement_Value_RSP" generates the response to the command Get_measurement_values (Page 3188). In the response, return the values of the requested measurements.

Parameter

The following table shows the parameters of the "PE_Measurement_Value_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 3189)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Count	Input	BYTE	I, Q, M, D, L or constant	Number of measured values (Measurement_Values).

Parameter	Declaration	Data type	Memory area	Description
Measure-ment_Values	Input	VARIANT	D	Pointer to the array with the measured values (Measurement_IDs). For information on the structure of the array in accordance with the PROFIenergy profile, see PI Command "Query_Measurement" - "Get_Measurement_values" (Page 3188)
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 3189)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 3189)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 3189)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

11.6.3.6 Module parameter assignment

Writing and reading data records

Principle

Some modules have a write-only system data area to which your program can transfer data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see following table).

Modules can also have a read-only system data area in which your program can read data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see following table).

Note

There are modules that have both system data areas. These are physically separate areas and the only thing they have in common is their logical structure.

Write-only system data area

The following table shows the structure of the write-only system data area. This table also shows how the permitted size of the individual data records and which instructions can be used to write them.

Data record number	Contents	Size	Can be written with instruction
0	Parameter	-	WR_DPARM (Page 3214)
1	Parameter	-	WR_DPARM (Page 3214)
2 to 127	User data	Each ≤ 240 bytes	WR_DPARM (Page 3214) WR_REC (Page 3123)
128 to 240	Parameter	Each ≤ 240 bytes	WR_DPARM (Page 3214) WR_REC (Page 3123)

Read-only system data area

The following table shows the structure of the read-only system data area. This table also shows the permitted size of the individual data records and which instructions can be used to read them.

Data record number	Contents	Size	Can be read with instruction
0	Module-specific diagnostics data (set as standard for the entire system)	4 bytes	RD_REC (Page 3118)
1	Channel-specific diagnostics data (incl. data record 0)	4 to 220 Bytes	RD_REC (Page 3118)
2 to 127	User data	Each ≤ 240 bytes	RD_REC (Page 3118)
128 to 240	Diagnostics data	Each ≤ 240 bytes	RD_REC (Page 3118)

System resources

If you start several asynchronous data record transfers one after the other with only short intervals between them, the allocation of system resources by the operating system ensures that all the jobs are executed and that they do not interfere with each other.

If the limits of the system resources are reached, this is indicated in RET_VAL. You can remedy this temporary error situation by simply repeating the job.

The maximum number of "simultaneously" active jobs of a single instruction type depends on the CPU.

RD_DPAR: Read module data record

Description

You use the instruction to read the data record with the number INDEX of the addressed component from the configured system data. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the target range RECORD. In this case, the LEN output parameter contains the length of the read data in bytes.

If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR . In this case, the output parameter STATUS contains the error information.

Functional description

The "RD_DPAR" instruction works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling "RD_DPAR" with REQ = 1.

The output parameter BUSY and bytes 2 and 3 of the output parameter STATUS show the status of the job. Bytes 2 and 3 of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

The transfer of the data record is complete when the output parameter BUSY has the value FALSE.

Parameter

The following table shows the parameters of the instruction "RD_DPAR":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Identification number of the CPU or the interface. The number is assigned automatically and is stored in the properties of the CPU or of the interface of the hardware configuration.
INDEX	Input	INT	I, Q, M, D, L or constant	Data record number
RECORD	InOut	VARIANT	I, Q, M, D, L	Target range for the read data record
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The job is not yet completed.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.
STATUS	Output	DWORD	I, Q, M, D, L	• Call ID (bytes 2 and 3) or error code • Byte 1: B#16#00, if no error. Otherwise function ID from DPV1-PDU: In the case of error for data record reading B#16#DE, in the case of error for data record writing B#16#DF. If no DPV1 protocol element is used: B#16#C0. • Byte 4: Manufacturer-specific error ID extension
LEN	Output	INT	I, Q, M, D, L	Length of the read data record information

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

STATUS parameter

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Address specified at parameter LADDR is invalid.	-
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.	-
8093	This instruction is not valid for the module selected with LADDR.	-
80A1	Negative acknowledgment during sending of data record to module (module defective or unplugged during sending).	-
80A2	DP protocol error at layer 2, possible hardware/interface error in DP slave.	Distributed I/O
80A3	DP protocol error at user interface/user.	Distributed I/O
80A4	Communication problem on com bus.	Error occurs between CPU and external DP interface module
80B0	Instruction for module type not possible, or module does not recognize the data record.	-
80B1	The length of the data record to be sent is incorrect.	-
80B2	The configured slot is not assigned.	-

Error code* (W#16#...)	Explanation	Restriction
80B3	Actual module type does not correspond to specified module type.	-
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available or deactivated.	Distributed I/O
80C6	Data record transfer cancelled due to priority class abort (restart or background).	Distributed I/O
80D0	There is no entry for the module.	-
80D1	The data record number is not configured for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-
80D5	The data record is static.	-
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

RD_DPARM: Read data record from configured system data (Page 3213)

RD_DPARA: Read module data record asynchronously

Description

You use the instruction to read the data record with the number RECNUM of a selected module from the configured system data. The read data record is entered in the target range defined by the parameter RECORD .

Functional description

The "RD_DPARA" instruction works asynchronously, that is, its execution extends over multiple calls. You start the reading process by calling the instruction with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

Parameters

The following table shows the parameters of the instruction "RD_DPARA":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Identification number of the CPU or the interface. The number is assigned automatically and is stored in the properties of the CPU or of the interface of the hardware configuration.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error occurred during the transmission, the following cases are distinguished: • RET_VAL contains the length of the actually read data record in bytes if the target range is larger than the read data record. • RET_VAL contains "0" if the length of the read data record is equal to the length of the target range.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The job is not yet completed.
RECORD	Output	VARIANT	I, Q, M, D, L	Target range for the data record read.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Specified logical base address invalid.	-
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.	-
8093	This instruction is not valid for the module selected with LADDR.	-

Error code* (W#16#...)	Explanation	Restriction
80A1	Negative acknowledgment during sending of data record to module (module defective or unplugged during sending).	-
80A2	DP protocol error at layer 2, possible hardware/interface error in DP slave.	Distributed I/O
80A3	DP protocol error at user interface/user.	Distributed I/O
80A4	Communication on PBUS+ disrupted	-
80B0	Instruction for module type not possible, or module does not recognize the data record.	-
80B1	The length of the data record to be sent is incorrect. With RD_DPARM (Page 3213): The target range spanned by RECORD is too short.	-
80B2	The configured slot is not assigned.	-
80B3	Actual module type does not correspond to specified module type.	-
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available or deactivated.	Distributed I/O
80C6	Data record transfer cancelled due to priority class abort (restart or background).	Distributed I/O
80D0	There is no entry for the module.	-
80D1	The data record number is not configured for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-
80D5	The data record is static.	-
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

RD_DPARM: Read data record from configured system data

Description

You use instruction to read the data record with the number RECNUM of the addressed module from the configured system data. The read data record is entered into the target range spanned by the RECORD parameter.

Parameters

The following table shows the parameters of the instruction "RD_DPARM":

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware ID of the CPU or the interface. The number is assigned automatically and is stored in the properties of the CPU or of the interface in the hardware configuration.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	Length of the data record read in bytes if the read data record fits in the target range and no error occurred in the transfer. If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	VARIANT	I, Q, M, D, L	Target range for the data record read.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Specified logical base address invalid.	-

Error code* (W#16#...)	Explanation	Restriction
8092	A type other than BYTE is specified in the VARIANT reference at the RECORD parameter.	-
8093	This instruction is not valid for the module selected with LADDR.	-
80B1	The target range spanned by RECORD is too short.	-
80C3	The required resources (memory, etc.) are currently occupied.	-
80D0	There is no entry for the module.	-
80D1	The data record number is not configured for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-
80D5	The data record is static.	-
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

WR_DPARM: Transfer data record

Description

You use the instruction "WR_DPARM" to transfer the data record with the number RECNUM from the configuration data to the addressed module. With this instruction, it is irrelevant whether the data record is static or dynamic.

Parameters

The following table shows the parameters of the instruction "WR_DPARM":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Write request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware ID of the CPU or of the interface. The number is assigned automatically and is stored in the properties of the CPU or of the interface in the hardware configuration.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet completed.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Address specified at parameter LADDR is invalid.	-
8093	This instruction is not valid for the module selected with LADDR.	-
80A1	Negative acknowledgment during sending of data record to module (module defective or unplugged during sending).	-
80A2	DP protocol error at layer 2, possible hardware/interface error in DP slave.	Distributed I/O
80A3	DP protocol error at user interface/user.	Distributed I/O
80A4	Communication on PBUS+ disrupted	-
80B0	Instruction for module type not possible, or module does not recognize the data record.	-
80B1	The length of the data record to be sent is incorrect.	-
80B2	The configured slot is not assigned.	-
80B3	Actual module type does not correspond to specified module type.	-
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available or deactivated.	Distributed I/O
80C6	Data record transfer cancelled due to priority class abort (restart or background).	Distributed I/O
80D0	There is no entry for the module.	-
80D1	The data record number is not configured for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-

Error code* (W#16#...)	Explanation	Restriction
80D5	The data record is static.	-
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

11.6.3.7 Interrupts

ATTACH: Attach an OB to an interrupt event

Description

You use the instruction "ATTACH" to assign an organization block (OB) to a hardware interrupt event.

- You enter the symbolic or numeric name of the organization block in the OB_NR parameter. This will then be assigned to the event specified in the EVENT parameter.
- You select the hardware interrupt event at the EVENT parameter. Hardware interrupt events already generated are listed in the PLC tags under "System constants".

If the event in the EVENT parameter occurs following error-free execution of the "ATTACH" instruction, the organization block in the OB_NR parameter will be called and its program executed.

With the ADD parameter, you specify whether previous assignments of the organization block to other events should be canceled or retained. If the ADD parameter has the value "0", the existing assignments will be replaced by the current assignment.

Hardware interrupt events

Hardware interrupts can be used if events are not pending long enough to respond to them during runtime. Each hardware interrupt can be assigned to hardware interrupt OBs. These OBs contain the reaction to a particular event.

Hardware interrupts can be created for different events. Examples for this are:

- The detection of rising or falling edges for digital inputs.
- Violation of defined low and high limits for analog inputs.
- External reset, overflow/underflow, direction reversal etc. for high-speed counters.

Functional principle

Each hardware interrupt can be assigned to a hardware interrupt OB, which is placed in the queue for processing when the hardware interrupt event occurs. The assignment of OB and event can occur during configuration or at runtime:

- To assign an event to an OB during configuration, select a hardware interrupt OB for an event in the hardware configuration under "Hardware interrupts".
- Use the ATTACH instruction to make the assignment during runtime. The assignment of event and hardware interrupt OB is made via the EVENT and OB_NR parameters.

Parameters

The following table shows the parameters of the "ATTACH" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_ATT	I, Q, M, D, L or constant	Organization block (numbers up to 32767 are supported.)
EVENT	Input	EVENT_ATT	D, L or constant	Hardware interrupt event to be assigned to the OB. The hardware interrupt event must first be enabled in the hardware device configuration for inputs or high-speed counters.
ADD	Input	BOOL	I, Q, M, D, L or constant	Effects on previous assignments: • ADD=0 (default): This event replaces all previous event assignments for this OB. • ADD=1: This event is added to the previous event assignments for this OB.
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#....)	Description
0	No error
8090	OB does not exist
8091	OB is incorrect type
8093	Event does not exist
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

DETACH: Detach an OB from an interrupt event

Description

You use this instruction to cancel the existing assignment of an organization block to one or more hardware interrupt events during runtime.

You enter the symbolic or numeric name of the organization block in the OB_NR parameter. The assignment of this organization block to the event specified in the EVENT parameter will then be canceled.

- If you have selected an individual hardware interrupt event at the EVENT parameter, the assignment of the OB to this hardware interrupt event is cancelled. All other currently existing assignments remain active. You can select an individual hardware interrupt event using the drop-down list of the operand placeholder.
- If you have not selected a hardware interrupt event, all the events currently assigned to this OB_NR organization block are separated.

Parameters

The following table shows the parameters of the "DETACH" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_ATT	I, Q, M, D, L or constant	Organization block (numbers up to 32767 are supported.)
EVENT	Input	EVENT_ATT	D, L or constant	Hardware interrupt event
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#....)	Description
0	No error
1	No assignment exists (warning)
8090	OB does not exist
8091	OB has incorrect type
8093	Event does not exist
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Cyclic interrupt

SET_CINT: Set cyclic interrupt parameters

Description

You use this instruction to set the parameters for a cyclic interrupt OB. The start time for a cyclic interrupt OB is generated from the respective time interval of the OB and the phase offset.

- The time interval of an OB is the interval at which the OB is periodically called. For example, if the time interval is 100 µs, the OB will be called every 100 µs during program execution.
- The phase offset is a time interval by which the call of a cyclic interrupt OB is offset. You can use the phase offset to process low priority organization blocks in a precise time base.

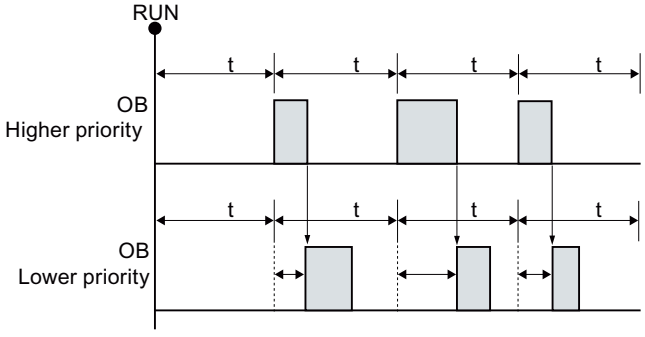
If the OB does not exist or if the time interval used is not supported, a corresponding error alarm is output in the RET_VAL parameter.

A time interval in the CYCLE parameter of "0" means that the OB will not be called.

Functional description

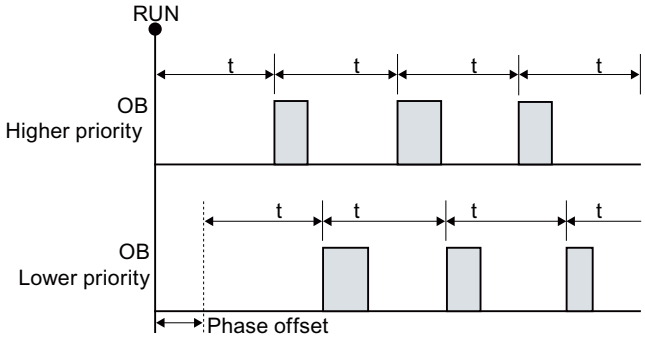
If a lower priority OB and a higher priority OB are called in the same time interval, the lower priority OB will only be called once the higher priority OB has been executed. The call time for the lower priority OB can be offset according to the length of time to execute the higher-priority OB.

OB call without phase offset



If a phase offset is configured for the lower priority OB and the phase offset is greater than the current execution time of the respective higher priority OB, then the block will be called in a fixed time base.

OB call with phase offset



Parameters

The following table shows the parameters of the "SET_CINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_CYCLIC	I, Q, M, D, L or constant	OB number (<32768)
CYCLE	Input	UDINT	I, Q, M, D, L or constant	Time interval in microseconds
PHASE	Input	UDINT	I, Q, M, D, L or constant	Phase offset
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0	No error
8090	OB does not exist or is of the wrong type
8091	Incorrect time interval
8092	Incorrect phase offset
80B2	No event assigned to OB
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

QRY_CINT: Query cyclic interrupt parameters

Description

You can use this instruction to query the current parameters of a cyclic interrupt OB. The cyclic interrupt OB is identified using the OB_NR parameter.

The values of the queried cyclic interrupt parameters correspond to those at the time the "QRY_CINT" instruction is executed.

Parameters

The following table shows the parameters of the "QRY_CINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_CYCLIC (INT)	I, Q, M, D, L or constant	OB number (<32768) or symbolic addressing via the name of the OB (e.g. OB_MyOB)
CYCLE	Output	UDINT	I, Q, M, D, L	Time interval in microseconds
PHASE	Output	UDINT	I, Q, M, D, L	Phase offset
STATUS	Output	WORD	I, Q, M, D, L	Status of the cyclic interrupt: • Bit 0 to bit 4: see parameter STATUS • Other bits: Always "0"
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter STATUS

Bit	Value	Meaning
0	0	Not used (always "0").
1	0	The cyclic interrupt is enabled.
	1	The cyclic interrupt is delayed.
2	0	The cyclic interrupt is not enabled or has expired.
	1	The cyclic interrupt is enabled.
3	0	Not used (always "0").
4	0	An OB with the specified number does not exist.
	1	An OB with the specified number exists.
Other bits		Not used (always "0").

Parameter RET_VAL

If an error occurs, the relevant error code will be displayed in the RET_VAL parameter and the STATUS parameter is set to "0".

Error code* (W#16#....)	Description
0	No error.
8090	OB does not exist or is of the wrong type
80B2	No result assigned to OB.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Time-of-day interrupt

SET_TINT: Set time-of-day interrupt

Description

The instruction "SET_TINT" is used to set the start data and time of the time-of-day interrupt organization blocks from the user program, without having to make settings in the hardware configuration.

- Enter the number of the time-of-day interrupt-OBs at the parameter OB_NR for which you would like to set the start date and the time-of-day.
- Use the parameter SDT and PERIOD to specify when and how often the time-of-day interrupt OB should be called:
  - One-time call: Enter at the parameter SDT the date and the time-of-day and use the value "0" at the parameter PERIOD.
  - Repeated call: Enter the date and the time-of-day of the first call at the parameter SDT. Use the parameter PERIOD to define the time interval in which subsequent calls of the OBs should take place.

Observe the following when setting the start date and start time:

- The date and the time-of-day specification at the parameter SDT refers to the system time.
- The seconds and milliseconds specification is ignored and set to "0" at the start time.
- Only the starting date days 1, 2, ... 28 are possible if you want to specify a monthly time-of-day interrupt OBs. This restriction prevents skipping the monthly call (e.g. 30-day months or in February).
  The setting "end of the month" (W#16#2001) can be used at the parameter PERIOD as an alternative to the 29th 30th and the 31st of the month.

After setting the time-of-day interrupt with "SET_TINT" it still must be activated with the instruction "ACT_TINT".

Note

Additional information about time-of-day interrupt OBs

Other special features for the use of time-of-day interrupt OBs can be found in the descriptions of the organization blocks of the respective CPU:

For the S7-1200: Auto-Hotspot

For the S7-1500: Auto-Hotspot

The settings at the parameters SDT and PERIOD coincide with the settings for time-of-day interrupt in the properties of the time-of-day interrupt OBs.

Parameters

The following table shows the parameters of the "SET_TINT" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OBs • The numbers 10 to 17 are available for the time-of-day interrupt OBs. • An OB number starting with 123 can also be assigned as an alternative. The OB number is displayed in the program block folder and in the system constants.
SDT	Input	DT	D, L or constant	Start date and start time
PERIOD	Input	WORD	I, Q, M, D, L or constant	Execution intervals from the starting point SDT to: • W#16#0000 = single execution • W#16#0201 = once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR (no time-of-day interrupt OB addressed).
8091	Error at the parameter SDT (invalid date and time-of-day specification).
8092	Incorrect input at the parameter PERIOD.
80A1	The set start time is in the past. This error code only occurs at PERIOD = W#16#0000.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

ACT_TINT: Enable time-of-day interrupt (Page 3227)

SET_TINTL: Set time-of-day interrupt

Description

The instruction "SET_TINTL" is used to set the start data and time of the time-of-day interrupt organization blocks from the user program, without having to make settings in the hardware configuration.

- Enter the number of the time-of-day interrupt-OBs at the parameter OB_NR for which you would like to set the start date and the time-of-day.
- Use the parameter SDT and PERIOD to specify when and how often the time-of-day interrupt OB should be called:
  - One-time call: Enter at the parameter SDT the date and the time-of-day and use the value "0" at the parameter PERIOD.
  - Repeated call: Enter the date and the time-of-day of the first call at the parameter SDT. Use the parameter PERIOD to define the time interval in which subsequent calls of the OBs should take place.
- Use the parameter LOCAL to select whether the time specified at the parameter SDT refers to the local time or the system time.
- With the ACTIVATE parameter, you specify whether the settings made for the organization block are to be applied directly (ACTIVATE = true) or only after "ACT_TINT (Page 3227)" for the time-of-day interrupt organization block is called (ACTIVATE = false).

Observe the following when setting the start date and start time:

- The seconds and milliseconds specification is ignored and set to "0" at the start time.
- Only the starting date days 1, 2, ... 28 are possible if you want to specify a monthly time-of-day interrupt OBs. This restriction prevents skipping the monthly call (e.g. 30-day months or in February).
  The setting "end of the month" (W#16#2001) can be used at the parameter PERIOD as an alternative to the 29th 30th and the 31st of the month.

Please observe the following during the use of local time:

- Changeover from daylight saving to standard time: When calling time-of-day interrupt organization blocks with a start time within the second hour during changeover from daylight saving time to standard time, also use a time-delay interrupt during the first hour of the time changeover.
- Changeover from standard to daylight saving time: If you specify the skipped hour as the time-of-day for the daylight saving time changeover day, the error code 16#8091 will be output at a single execution (PERIOD = W#16#0000).

Note

Additional information about time-of-day interrupt OBs

Other special features for the use of time-of-day interrupt OBs can be found in the descriptions of the organization blocks of the respective CPU:

For the S7-1200: Auto-Hotspot

For the S7-1500: Auto-Hotspot

The settings at the parameters SDT and PERIOD coincide with the settings for time-of-day interrupt in the properties of the time-of-day interrupt OBs.

Parameters

The following table shows the parameters of the "SET_TINTL" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OBs • The numbers 10 to 17 are available for the time-of-day interrupt OBs. • An OB number starting with 123 can also be assigned as an alternative. The OB number is displayed in the program block folder and in the system constants.
SDT	Input	DTL	D, L or constant	Start date and start time
LOCAL	Input	BOOL	I, Q, M, D, L or constant	• true: Use local time • false: Use system time
PERIOD	Input	WORD	I, Q, M, D, L or constant	Execution intervals from the starting point SDT to: • W#16#0000 = single execution • W#16#0201 = Once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end
ACTIVATE	Input	BOOL	I, Q, M, D, L or constant	• true: set and activate the time-of-day interrupt • false: set the time-of-day interrupt and only activate at call of "ACT_TINT (Page 3227)"
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR (no time-of-day interrupt OB addressed).
8091	Error at the parameter SDT (invalid date and time-of-day specification).
8092	Incorrect input at the parameter PERIOD.
80A1	The set start time is in the past. This error code only occurs at PERIOD = W#16#0000.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

CAN_TINT: Cancel time-of-day interrupt

Description

The instruction "CAN_TINT" is used to delete the start data and start time of a specified time-of-day interrupt organization block. This deactivates the time-of-day interrupt, and the organization block is no longer called.

If you want to reuse the time-of-day interrupt, the start time must be reset (instruction "SET_TINTL (Page 3224)" or "SET_TINT (Page 3222)").

Thereafter, you must reactivate the time-of-day interrupt:

- If you use the instruction "SET_TINT (Page 3222)" or "SET_TINTL (Page 3224)" with parameter ACTIVE=false to set the time-of-day interrupt, call "ACT_TINT (Page 3227)".
- By using the instruction "SET_TINTL (Page 3224)" you can also activate the time-of-day interrupt directly with the parameter ACTIVE=true.

Parameters

The following table shows the parameters of the instruction "CAN_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OB, whose starting date and time-of-day are to be deleted.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR.
80A0	No starting date/time-of-day defined for the affected time-of-day interrupt OB.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

ACT_TINT: Enable time-of-day interrupt

Description

With the instruction "ACT_TINT" you can activate a time-of-day interrupt organization block from the user program. A prerequisite for the execution of the instruction is that the starting date and time-of-day are already set for the time-of-day interrupt OB.

Parameters

The following table shows the parameters of the instruction "ACT_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OBs • The numbers 10 to 17 are available for the time-of-day interrupt OBs. • An OB number starting with 123 can also be assigned as an alternative. The OB number is displayed in the program block folder and in the system constants.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR (no time-of-day interrupt OB addressed).
80A0	Start date and time-of day not set for the relevant time-of-day interrupt OB.

Error code* (W#16#...)	Description
80A1	The activated time is in the past. The error only occurs if the time-of-day interrupt is only to be executed once.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

QRY_TINT: Query status of time-of-day interrupt

Description

You can use this instruction to display the status of a time-of-day interrupt organization block in the STATUS output parameter.

Parameters

The following table shows the parameters of the "QRY_TINT" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OB whose status is queried The OB number is displayed in the program block folder and in the system constants.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-of-day interrupt (see below)

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred. Exception: Status message "0" on bit 4 (no OB with this number created).
8090	Error at the parameter OB_NR. Possible causes: • The value at the parameter OB_NR is not an OB number (<1 or > 32767) supported by the CPU. • The OB number does not address a time-of-day interrupt OB.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS

If an error occurs (see RET_VAL parameter), "0" is output in the STATUS parameter.

Bit	Value	Meaning
0	0	In Run.
	1	During startup.
1	0	The time-of-day interrupt is enabled.
	1	The time-of-day interrupt is disabled.
2	0	Time-of-day interrupt is not activated or has elapsed.
	1	The time-of-day interrupt is activated.
4	0	An OB with an OB number as specified at OB_NR parameter does not exist.
	1	An OB with an OB number as specified at OB_NR parameter does exist.
6	0	The time-of-day interrupt is based on the system time
	1	The time-of-day interrupt is based on the local time
Other		Always "0"

Time-delay interrupt

Using time-delay interrupts

Definition

After you have called the "SRT_DINT (Page 3230)" instruction, the operating system generates an interrupt after the specified delay time has elapsed, in other words, the assigned time-delay interrupt OB is called.

Prerequisites for the call

Before a time-delay interrupt can be called by the operating system, the following conditions must be met:

- The time-delay interrupt OB must be started by the "SRT_DINT (Page 3230)" instruction.
- The time-delay interrupt OB must not be deselected during configuration.
- The time-delay interrupt OB must exist in the CPU.

Purpose of the instructions "SRT_DINT", "CAN_DINT" and "QRY_DINT"

You use the instructions to

- Start time-delay interrupts ("SRT_DINT (Page 3230)")
- Cancel time-delay interrupts ("CAN_DINT (Page 3231)")
- Query time-delay interrupts ("QRY_DINT (Page 3232)")

Effects on the time-delay interrupt

The following table lists a number of different situations and explains the effect they have on a time-delay interrupt.

If ...	and ...	Then ...
A time-delay interrupt is started (by calling "SRT_DINT (Page 3230)")	The time-delay interrupt has already started,	The delay time is overwritten; the time-delay interrupt is started again.
	The time-delay interrupt OB does not exist at the time of the call,	The operating system generates a priority class error (calls OB 85). If OB 85 does not exist, the CPU changes to STOP.
	The interrupt is started in a startup OB and the delay time elapses before the CPU changes to RUN,	The call of the time-delay interrupt OB is delayed until the CPU is in RUN mode.
The delay time has elapsed,	A previously started time-delay interrupt OB is still being executed,	The operating system generates a time error (calls OB 80). If OB 80 does not exist, the CPU changes to STOP.

Response to warm restart and cold restart

During a warm restart or a cold restart, all the time-delay interrupt settings made in the user program by means of instructions are cleared.

Starting in a startup OB

A time-delay interrupt can be started in a startup OB. Two conditions must be satisfied to call the time-delay OB:

- The delay time must have elapsed.
- The CPU must be in the RUN mode.

If the delay time has elapsed and the CPU is not yet in the RUN mode, the time-delay interrupt OB call is delayed until the CPU is in RUN mode. The time-delay interrupt OB is then called before the first instruction in OB Main [OB 1] is executed.

SRT_DINT: Start time-delay interrupt

Description

The instruction "SRT_DINT" is used to start a time-delay interrupt, which calls a time-delay interrupt OB after the expiry of the delay time specified at the parameter DTIME. The delay time is started when a negative edge is generated in the EN enable input. The signal state of enable input EN must be "0" during delay time countdown. If the delay time countdown is interrupted, the OB configured in the OB_NR parameter is not executed.

Accuracy

The maximum time between the call of the "SRT_DINT" instruction and the start of the time-delay interrupt OB is one millisecond longer than the configured delay time, provided that no interruption events delay the call.

Parameters

The following table shows the parameters of the instruction "SRT_DINT":

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	I, Q, M, D, L or constant	Number of the OB to be executed after a delay time
DTIME	Input	TIME	I, Q, M, D, L or constant	Delay time (1 to 60000 ms) You can realize longer times, for example, by using a counter in a time-delay interrupt OB.
SIGN	Input	WORD	I, Q, M, D, L or constant	Identifier that appears when the time-delay interrupt OB is called in the start event information of the OB.
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
8090	Incorrect parameter OB_NR
8091	Incorrect parameter DTIME
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

CAN_DINT: Cancel time-delay interrupt

Description

You use this instruction to cancel a started time-delay interrupt and, thus, also cancel the call of the time delay interrupt OB that is to be executed after the configured delay time. You specify the number of the organization block whose call is to be canceled in the OB_NR parameter.

Parameters

The following table shows the parameters of the "CAN_DINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	I, Q, M, D, L or constant	Number of the OB whose call will be canceled
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
8090	Incorrect parameter OB_NR
80A0	Time-delay interrupt has not started.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

QRY_DINT: Query time-delay interrupt status

Description

The instruction "QRY_DINT" is used to query the status of the time-delay interrupt.

Parameters

The following table shows the parameters of the instruction "QRY_DINT":

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	I, Q, M, D, L or constant	Number of the OB whose status is being queried.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code. The value "0" is displayed in the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-delay interrupt, see following table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter STATUS

Bit	Value	Meaning
0	0	Operating system in RUN
	1	Operating system in startup
1	0	Time-delay interrupt is enabled by the operating system.
	1	Time-delay interrupt is disabled.
2	0	Time-delay interrupt is not activated or has elapsed.
	1	Time-delay interrupt is activated.
3	-	-
4	0	Time-delay interrupt OB with the specified number does not exist.
	1	Time-delay interrupt OB with the specified number exists.
Other bits		Always "0"

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Incorrect information in the OB_NR parameter
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Synchronous errors

Mask synchronous error events

Introduction

Synchronous errors are programming and access errors. Such errors occur as a result of programming with incorrect operand areas or numbers, or incorrect addresses. **Masking** these synchronous errors means the following:

- Masked synchronous errors do not trigger an error OB call and do not lead to a programmed alternative reaction.
- The CPU "records" the masked errors that have occurred in an error status register.

Masking is carried out by calling the "MSK_FLT (Page 3239)" instruction.

Unmasking errors means canceling a previously set mask and clearing the corresponding bit in the event status register of the current priority class. Masking is canceled as follows:

- By calling the "DMSK_FLT (Page 3240)" instruction.
- Once the current priority class has been completed.

If an error event occurs after it has been unmasked, then the operating system will start the associated error OB.

You can use the "READ_ERR (Page 3240)" instruction to read the masked errors that have occurred.

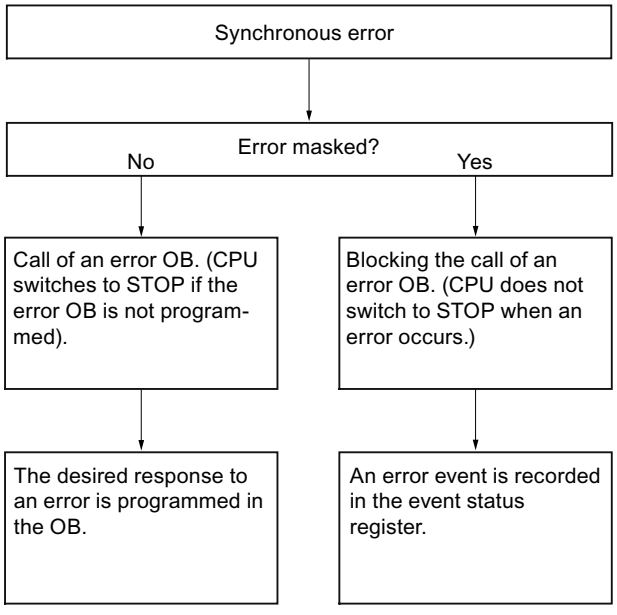
Note

Regardless of masking or unmasking of the error events for the S7-1500, the error event is entered in the diagnostic buffer and the group error LED of the CPU lights up.

Handling errors in general

Different actions are possible if programming and I/O area access errors occur in a user program:

- You can program an error OB that is called by the operating system when the corresponding error occurs.
- You can disable the error OB call individually for each priority class. In this case, the CPU does not change to STOP when an error of this type occurs in the particular priority class. The CPU enters the error in an error register. From this entry, however, you cannot recognize when or how often the error occurred.

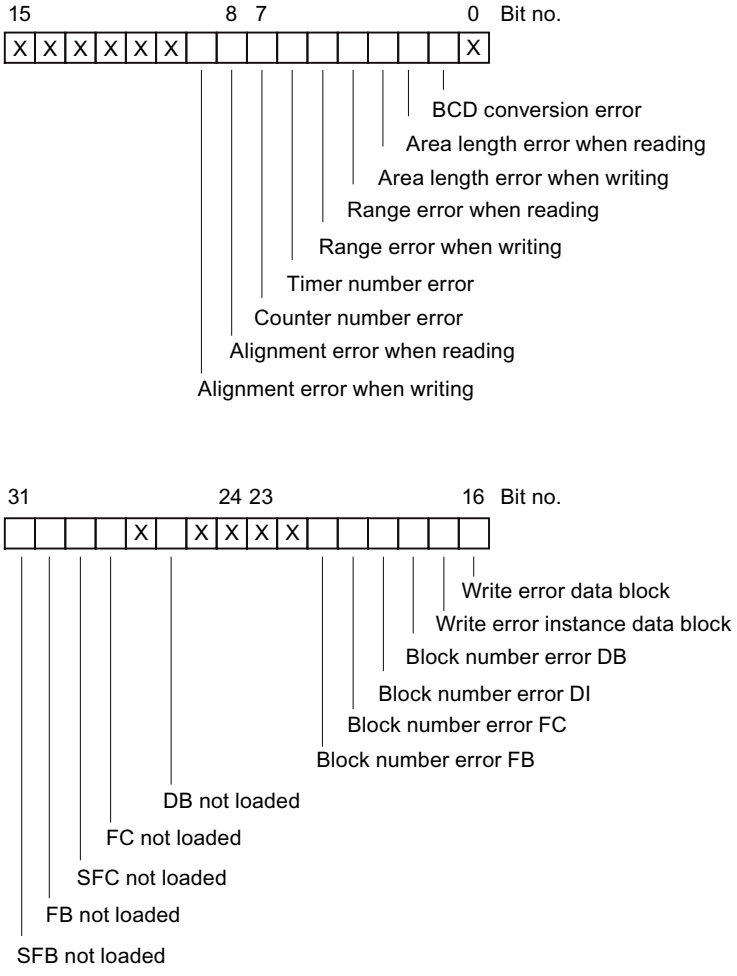


Synchronous errors are assigned to a particular bit pattern known as the **error filter**. You can find this error filter in the input and output parameters of the "MSK_FLT (Page 3239)", "DMSK_FLT (Page 3240)", and "READ_ERR (Page 3240)" instructions.

Synchronous errors are divided into **programming and access errors** which you can mask in two error masks. The error filters are illustrated in the following figures.

Programming error filter

The following figure shows the bit pattern of the error filter for programming errors. The error filter for the programming errors is available in the "PRGFLT_..." parameters (see below "Programming errors, Low-Word" or "Programming errors, High-Word").

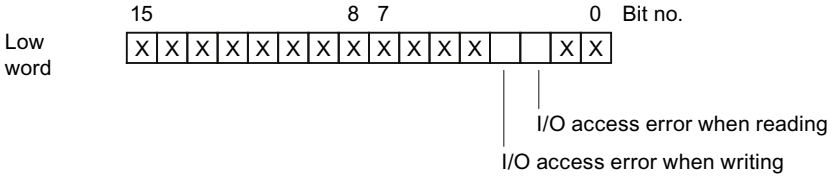


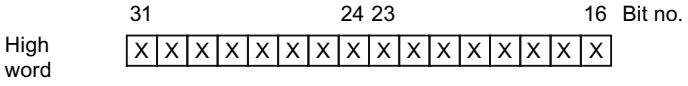
Legend: X Not relevant

The non-relevant bits "x" are set to "0" for the input and output parameters of the instructions "MSK_FLT (Page 3239)", "DMSK_FLT (Page 3240)", "READ_ERR (Page 3240)".

Access error mask

The following figure shows the bit pattern of the error mask for access errors. The error mask for access errors is located in the parameters ACCFLT_...

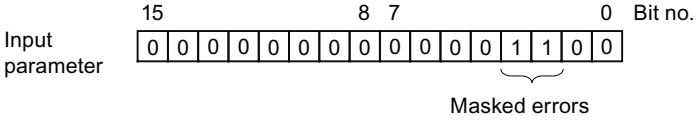


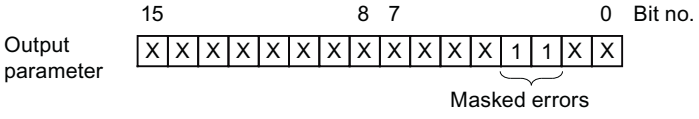


Legend: X Not relevant

Example: The following figure shows how the low word of the error mask for access errors appears with all masked errors.

- As input parameters for "MSK_FLT (Page 3239)"
- As output parameters for "MSK_FLT (Page 3239)"





Legend:
X Not relevant
0 Not masked
1 Masked

Programming error low word

The following table lists the errors assigned to the low word of the error filter for programming errors. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Possible cause of error
BCD conversion error	2521	The value to be converted is not a BCD number (e.g. 5E8).
Area length error when reading	2522	An addressed operand is not located entirely within the possible operand range. Example: MW 320 should be read although the memory area is only 256 bytes long.
Area length error when writing	2523	An addressed operand is not located entirely within the possible operand range. Example: A value should be written to MW 320 although the memory area is only 256 bytes long.
Range error when reading	2524	An incorrect range identifier for the operand was specified during indirect, overlapping range addressing. Example: • Correct: LAR1 P#E 12.0 L W[AR1, P#0.0] • Incorrect: LAR1 P#12.0 L W[AR1, P#0.0] The range error is reported during this operation.
Range error when writing	2525	An incorrect range identifier for the operand was specified during indirect, overlapping range addressing. Example: • Correct: LAR1 P#E 12.0 T W[AR1, P#0.0] • Incorrect: LAR1 P#12.0 T W[AR1, P#0.0] The range error is reported during this operation.
Timer number error	2526	A non-existent timer is accessed. Example: • SI T [MW 0] with MW 0 = 129; the timer 129 should be started although there are only 128 timers available.
Counter number error	2527	A non-existent counter is accessed. • Example: ZV Z [MW 0] with MW 0 = 600; the counter 600 is accessed although only 512 counters are available.

Error	Event ID (W#16#...)	Possible cause of error
Alignment error when reading	2528	A byte, word or double word operand with a bit address ≠ 0 is addressed. Example: • Correct: LAR1 P#M12.0 L B[AR1, P#0.0] • Incorrect: LAR1 P#M12.4 L B[AR1, P#0.0]
Alignment error when writing	2529	A byte, word or double word operand with a bit address ≠ 0 is addressed. Example: • Correct: LAR1 P#M12.0 T B[AR1, P#0.0] • Incorrect: LAR1 P#M12.4 T B[AR1, P#0.0]

Programming error high word

The following table lists the errors assigned to the high word of the error filter for programming errors. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Possible cause of error
Write error data block	2530	The data block to be written to is write-protected.
Write error instance data block	2531	The instance data block to be written to is write-protected.
Block number error DB	2532	The number of the data block is greater than the highest permissible number.
Block number error DI	2533	The number of the instance data block is greater than the highest permissible number.
Block number error FC	2534	The number of a called function (FC) is greater than the highest permissible number.
Block number error FB	2535	The number of a called function block (FB) is greater than the highest permissible number.
DB not loaded	253A	The data block is not loaded.
Instruction not loaded	253C to 253F	The instruction to be called is not loaded.

Access error

The following table contains the errors that are assigned to the error mask for access errors. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Possible cause of error
I/O access error when reading	2942	• The address in the I/O area has no assigned signal module. • Access to this I/O area was not acknowledged within the specified module monitoring time (timeout).
I/O access error when writing	2943	

MSK_FLT: Mask synchronous error events

Description

You use the instruction to control the reaction of the CPU to synchronous errors. This is done by masking the respective synchronous errors (for error filters, see Mask synchronous error events (Page 3233)). When you call "MSK_FLT", you mask the synchronous errors in the current priority class.

If you set individual bits of the synchronous error filter to "1" in the input parameters, other bits that were set previously retain their value of "1". You obtain new error filters that you can read out using the output parameters. The synchronous errors you have masked do not call an OB but are simply entered in an error register. You can read the error register with the "READ_ERR (Page 3240)" instruction.

Parameters

The following table shows the parameters of the instruction "MSK_FLT":

Parameter	Declaration	Data type	Memory area	Description
PRGFLT_SET_MASK	Input	DWORD	I, Q, M, D, L or constant	Programming error to be masked
ACCFLT_SET_MASK	Input	DWORD	I, Q, M, D, L or constant	Access error to be masked
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_MASKED	Output	DWORD	I, Q, M, D, L	Masked programming errors
ACCFLT_MASKED	Output	DWORD	I, Q, M, D, L	Masked access errors

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	None of the errors were already masked.
0001	At least one of the errors was already masked. Nevertheless the other errors will be masked.
-	General error information See also: Getting error ID locally with GetErrorID (Page 2922)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Evaluating errors with output parameter RET_VAL (Page 2195)

DMSK_FLT: Unmask synchronous error events

Description

You use the instruction to unmask the errors masked with "MSK_FLT (Page 3239)". To do this, you must set the corresponding bits of the error filter to "1" in the input parameters. With the "DMSK_FLT" call, you unmask the corresponding synchronous errors of the current priority class. At the same time, the queried entries are cleared in the error register. You can read out the new error filters using the output parameters.

Parameters

The following table shows the parameters of the instruction "DMSK_FLT":

Parameter	Declaration	Data type	Memory area	Description
PRGFLT_RESET_MASK	Input	DWORD	I, Q, M, D, L or constant	Programming errors to be unmasked
ACCFLT_RESET_MASK	Input	DWORD	I, Q, M, D, L or constant	Access errors to be unmasked
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_MASKED	Output	DWORD	I, Q, M, D, L	Still masked programming errors
ACCFLT_MASKED	Output	DWORD	I, Q, M, D, L	Still masked access errors

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	All specified errors were unmasked.
0001	At least one of the errors was not masked. Nevertheless the other errors will be unmasked.
-	General error information See also: GET_ERR_ID: Get error ID locally (Page 2922)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

READ_ERR: Read out event status register

Description

You use the instruction to read the error register. The structure of the error register corresponds to that of the programming and access error filters, which you can program as input parameters with "MSK_FLT (Page 3239)" and "DMSK_FLT (Page 3240)".

In the input parameters, you enter the synchronous errors you want to read from the error register. When you call "READ_ERR", you read the required entries from the error register and at the same time clear these entries.

The error register contains information that tells you which of the masked synchronous errors in the current priority class occurred at least once. If a bit is set, this means that the corresponding masked synchronous error occurred at least once.

Parameters

The following table shows the parameters of the instruction "READ_ERR":

Parameter	Declaration	Data type	Memory area	Description
PRGFLT_QUERY	Input	DWORD	I, Q, M, D, L or constant	Query programming error
ACCFLT_QUERY	Input	DWORD	I, Q, M, D, L or constant	Query access error
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_CLR	Output	DWORD	I, Q, M, D, L	Programming errors that occurred
ACCFLT_CLR	Output	DWORD	I, Q, M, D, L	Access errors that occurred

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	All queried errors are masked.
0001	At least one of the queried errors is not masked.
-	General error information See also: Getting error ID locally with GetErrorID (Page 2922)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Evaluating errors with output parameter RET_VAL (Page 2195)

Asynchronous error event

DIS_IRT: Disable interrupt event

Description

The instruction "DIS_IRT" is used to disable the processing of new interrupts and asynchronous error events. Disabling means that if an interrupting event occurs, the operating system of the CPU reacts as follows:

- It **neither** calls an interrupt OB nor an asynchronous error OB,
- **nor** does it trigger the normal reaction if an interrupt OB or asynchronous error OB is not programmed.

If you disable interrupts and asynchronous error events, this remains in effect for all priority classes. The disable can only be canceled with the "EN_IRT (Page 3243)" instruction or by a warm or a cold restart.

Whether the operating system writes interrupts and asynchronous error events to the diagnostics buffer when they occur depends on the input parameter setting you select for MODE.

Note

Remember that when you program the "DIS_IRT" instruction, all interrupts that occur will be discarded.

Parameters

The following table shows the parameters of the instruction "DIS_IRT":

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Specifies which interrupts and asynchronous errors are disabled.
OB_NR	Input	INT	I, Q, M, D, L or constant	OB number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter MODE

MODE (B#16#...)	Meaning
00	All newly occurring interrupts and asynchronous error events are disabled. (Synchronous errors are not disabled.) Assign the value "0" to the OB_NR parameter. Entries continue to be made in the diagnostics buffer.
01	All newly occurring events belonging to a specified interrupt class are disabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Process interrupts: 40 • Interrupts for DPV1: 50 • Multicomputing interrupt: 60 • Redundancy error interrupts: 70 • Asynchronous error interrupts: 80 Entries continue to be made in the diagnostics buffer.
02	All new occurrences of a specified interrupt are disabled. You designate the interrupt using the OB number. Entries continue to be made in the diagnostics buffer.

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	The OB_NR input parameter contains an invalid value.
8091	The MODE input parameter contains an invalid value.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

EN_IRT: Enable interrupt event

Description

You use the instruction to enable the processing of new interrupts and asynchronous error events that you have previously disabled with the "DIS_IRT (Page 3242)" instruction. This means that if an interrupting event occurs, the operating system of the CPU reacts in one of the following ways:

- It calls an interrupt OB or asynchronous error OB.
  or
- It triggers the standard reaction if the interrupt OB or asynchronous error OB is not programmed.

Parameters

The following table shows the parameters of the instruction "EN_IRT":

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Specifies which interrupts and asynchronous error events will be enabled.
OB_NR	Input	INT	I, Q, M, D, L or constant	OB number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter MODE

MODE	Meaning
0	All newly occurring interrupts and asynchronous error events are enabled.
1	All newly occurring events belonging to a specified interrupt class are enabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Process interrupts: 40 • Interrupts for DPV1: 50 • Multicomputing interrupt: 60 • Redundancy error interrupts: 70 • Asynchronous error interrupts: 80
2	All newly occurring events of a specified interrupt are enabled. You designate the interrupt using the OB number.

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	The OB_NR input parameter contains an invalid value.
8091	The MODE input parameter contains an invalid value.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

DIS_AIRT: Delay execution of higher priority interrupts and asynchronous error events

Description

You use "DIS_AIRT" to delay the processing of interrupt OBs whose priority are higher than the priority of the current organization block.

You can call "DIS_AIRT" multiple times in an organization block. The "DIS_AIRT" calls are counted by the operating system. Processing is delayed more and more each time "DIS_AIRT" is executed. To cancel a delay, you must execute the "EN_AIRT (Page 3245)" instruction. To cancel all delays, the number of "EN_AIRT (Page 3245)" executions must be equal to the number of "DIS_AIRT" calls.

You can query the number of delays in the RET_VAL parameter of the "DIS_AIRT" instruction. If the value in the RET_VAL parameter is "0", there are no delays.

Parameters

The following table shows the parameters of the "DIS_AIRT" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of delays

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Return value	Description
n	"n" shows the number of processing delays, i.e., the number of DIS_AIRT calls, after the instruction is complete (interrupt processing is only enabled again when n = 0; see EN_AIRT (Page 3245)).

EN_AIRT: Enable execution of higher priority interrupts and asynchronous error events

Description

You use "EN_AIRT" to enable processing of organization blocks when interrupts occur that have been delayed by the "DIS_AIRT (Page 3245)" instruction.

When "EN_AIRT" is executed, you cancel a processing delay that was registered by the operating system when "DIS_AIRT (Page 3245)" was called. To cancel all delays, the number of "EN_AIRT" executions must be equal to the number of "DIS_AIRT (Page 3245)" calls. If, for example, you have called "DIS_AIRT (Page 3245)" five times and thereby also delayed the processing five times, you must call the "EN_AIRT" instruction five times in order to cancel all five delays.

You can query the number of interrupt delays that have not yet been enabled after the execution of "EN_AIRT" in the RET_VAL parameter of the "EN_AIRT" instruction. The value "0" in the RET_VAL parameter means that all delays enabled by "DIS_AIRT (Page 3245)" have been cancelled.

Parameters

The following table shows the parameters of the "EN_AIRT" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of configured delays

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Return value/ Error code* (W#16#...)	Description
n	"n" shows the number of processing delays that are not yet enabled after the instruction is complete (interrupt processing is only enabled again when n = 0).
8080	Although the interrupt processing was already enabled, the function was called again.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

11.6.3.8 Alarms

Program_Alarm: Generate program alarm with associated values

Description

The instruction "Generate program alarm with associated values" monitors a signal and generates a signal change at parameter SIG of a program alarm (for definition, see also: Auto-Hotspot). An incoming program alarm is generated with a signal change from 0 to 1 and an outgoing program alarm is generated with a signal change from 1 to 0. The program alarm is triggered synchronously to program execution.

You can append up to ten associated values to the program alarm at the parametersSD_i (0 ≤ i ≤10). The associated values are detected at the time of signal change at parameter SIG and assigned to the program alarm. For more information on configuring associated values, refer to the following sections: Auto-Hotspot, Auto-Hotspot.

Each incoming or outgoing alarm is assigned a time stamp:

- The current system time of the PLC when the signal change occurs is used by default (default value at the TIMESTAMP parameter).
- If you wish to specify a different time stamp, you can create one at the TIMESTAMP parameter.
  The time value must always be specified in system time (i.e. UTC) as this is the time used for time synchronization throughout the plant.
- If an alarm is to be time-stamped with a local time, a conversion module, which converts the local time to system time, must be connected in series. This is the only way to make sure that the time stamp is shown properly in the alarm display.

To use the current system time of the CPU, set the TIMESTAMP parameter to its default value (LDT#1970-01-01-00:00:00.0).

Call of the instruction "Generate program alarm with associated values"

The instruction can only be called in a function block (FB). The block is processed synchronously. This means that an alarm is triggered as soon as the block is exited. If there is an error during processing, an error code is output.

Once the instruction has been inserted in the FB, a multi-instance of the data type "Program_Alarm" is created in the "Static" section of the block interface. You can choose any name for this multi-instance in the dialog which appears. It is also the name of the program alarm.

Finally, add the parameters of the instruction in line with your requirements (see the "Parameters" table).

Configuring the program alarm

The program alarm settings are displayed in the "Properties" window when you select the name of the program alarm in the "Static" section or in the network of the FB. In this window, you can select the alarm class, priority, etc. and edit the alarm text.

The settings that are made here can be edited in the project tree. Go to "PLC alarms" and open the "Program alarms" tab. All existing program alarms are then displayed in the "Alarm types" table.

Parameters

The following table shows the parameters of the instruction "Generate program alarm with associated values":

Parameter	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L, T, C or constant	The signal to be monitored. • Positive signal edge: An incoming program alarm is generated • Negative signal edge: An outgoing program alarm is generated
TIMESTAMP	Input	LDT	M, D, L or constant	This parameter is used to assign an alarm a time stamp, for example, from an input signal with a distributed stamp. The time value must always be specified in system time (i.e. UTC), as this is the time used for time synchronization throughout the plant. • "Not assigned" means that the system time of the CPU will be used as the interrupt time stamp when a signal change occurs (default). • Any system time input will be used as the interrupt time stamp when a signal change occurs. Note: If an interrupt is to be time-stamped with a local time, a conversion module must be connected in series to convert the local time to system time. This is the only way to make sure that the time stamp is shown correctly in the interrupt display.
SD_i	Input	VARIANT	I, Q, M, D, L	i-th associated value (1 ≤ i ≤ 10) You can use binary numbers, integers, floating-point numbers, or character strings as associated values.

Parameter	Declaration	Data type	Memory area	Description
Error	Output	BOOL	I, Q, M, D, L	Status parameter Error Error = TRUE indicates that an error has occurred during processing. The possible error cause is displayed at the Status parameter.
Status	Output	WORD	I, Q, M, D, L	Status parameter Status Display of error information (see "Error and Status parameters").

You can find additional information on valid data types under "See also".

Parameters Error and Status

The following table contains all specific error information that can be output via the Error and Status parameters.

Error	Status*	Explanation
0	0000	No error, or the instruction was not processed because there was no signal edge at the parameter SIG.
1	0085	"Only information" alarm type
1	8001	Invalid static alarm information
1	8002	No valid static alarm information
1	8004	Maximum size of 512 bytes reached for the associated values of the alarm.
1	8005	A positive signal edge is pending at parameter SIG and there is still an alarm pending which has not yet been acknowledged.
1	8007	Outgoing alarm which was not preceded by an incoming alarm.
1	8087	Static alarms are deactivated.
1	8089	Alarm is too long.
1	80Ax	The SD_i parameter has an invalid value.
1	80C1	The CPU cannot generate any alarms at this time because initialization routines are running (this is the case after download in RUN, for example). Try again later.
1	80C2	The maximum number of alarms per time unit has been sent. Try again later.
1	80C3	All dynamic alarm instances are being used. Try again later.
1	80C4	Alarm is being output and cannot be overwritten. Try again later.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Example

In the following example, you generate a program alarm with associated value for a signal change.

Create one tag in a global data block for storing the signal value to be monitored.

Program_Alarm_Tags

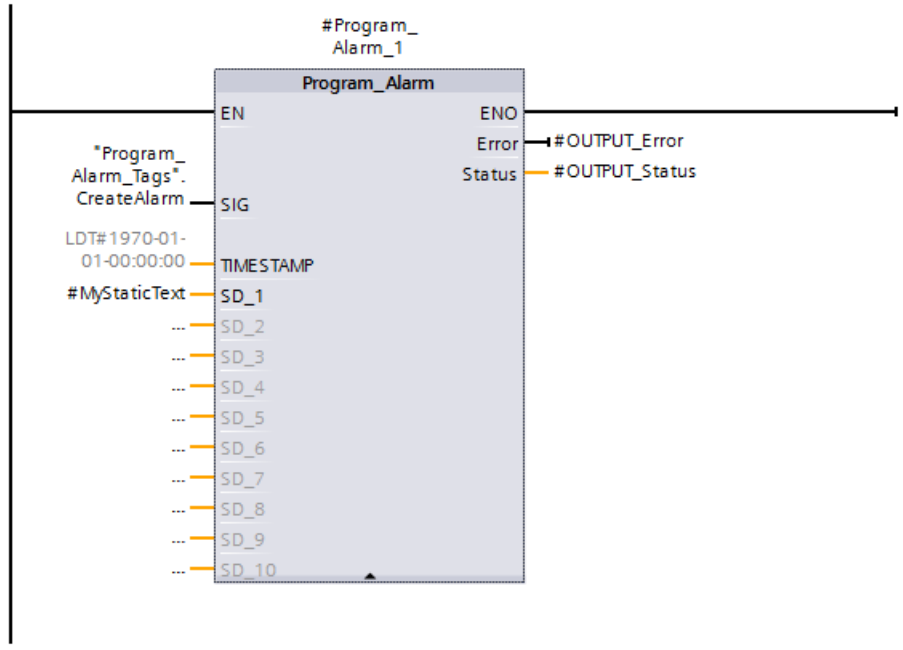
		Name	Data type	Start value
1		Static		
2		CreateAlarm	Bool	false

The instruction is called in a function block. Create four parameters for the function block for interconnection of the instruction.

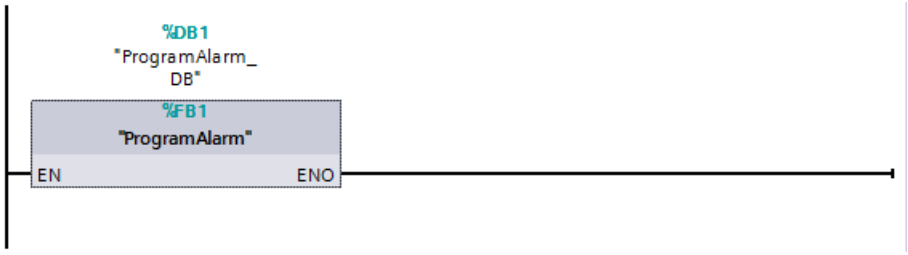
ProgramAlarm

		Name	Data type	Default value
1		Input		
2		<Add new>		
3		Output		
4		<Add new>		
5		InOut		
6		<Add new>		
7		Static		
8		MyStaticText	String	'MyStaticText'
9		Program_Alarm_1	Program_Alarm	
10		Temp		
11		OUTPUT_Error	Bool	
12		OUTPUT_Status	Word	

Interconnect the parameters of the instruction as follows.



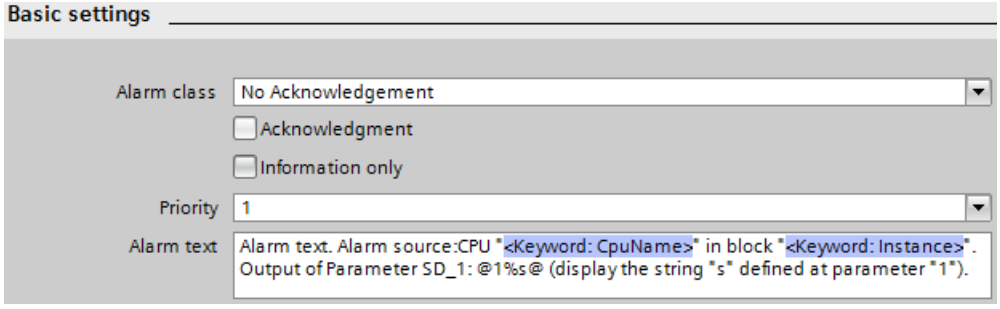
Call the function block in an OB.



A PLC alarm is generated automatically when the instruction is created. Open the PLC alarms dialog and select the alarm to be processed on the Program alarms tab. Create the alarm text, including two key words.

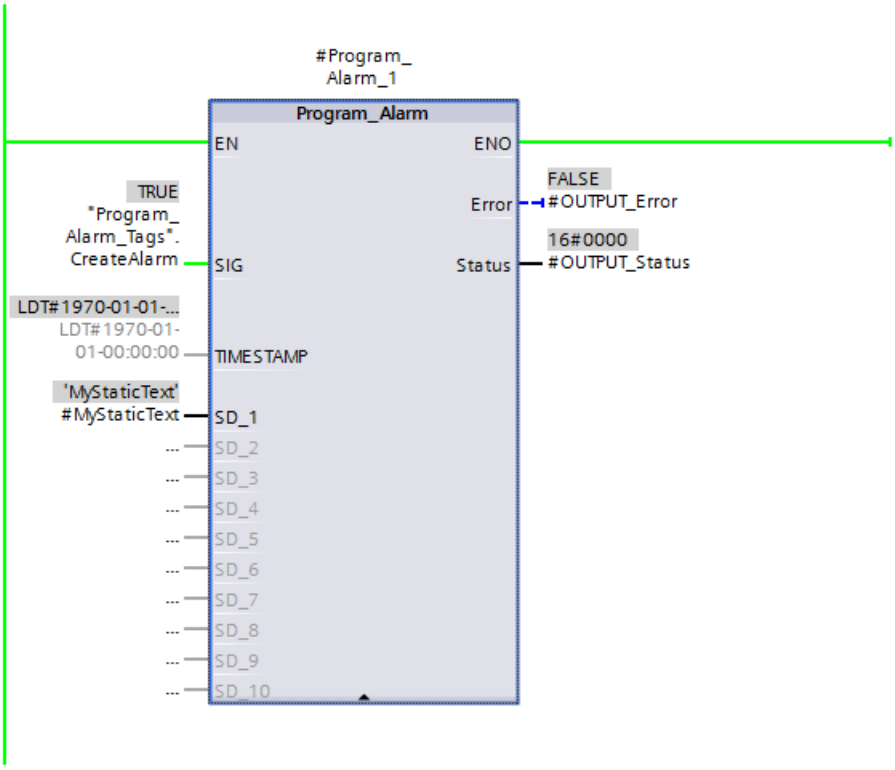
Note: If you right-click in the text box, you can insert a key word, tag, or text list.

As a result of the character sequence "@1%s@", the value of parameter SD_1 ("#MyStaticText") is read out and output as a character string.



If parameter SIG ("CreateAlarm") has the signal state "TRUE", the PLC alarm is output. At output parameter STATUS ("OUTPUT_Status"), the value "0001" indicates that a signal change has occurred. Then, it is indicated that no further processing is taking place (the value is "0000"). At parameter SD_1 ("#MyStaticText"), the associated value of the PLC alarm is output.

A time stamp can be transferred to parameter TIMESTAMP. If not interconnected, parameter TIMESTAMP outputs the local time of the CPU clock. At output parameter ERROR ("OUTPUT_Error"), it is indicated that the processing of the instruction is proceeding error-free.



For the output of the PLC alarm, use the web server of the CPU. In order to use the web server, the following is necessary:

- The web server must be activated in the CPU settings.

Use the Internet browser to open the web server (via the IP address of the CPU) and log on in the web server menu. The CPU outputs the alarm text as long as the signal being monitored ("CreateAlarm") supplies the value "TRUE".

Time	Message text
02:32:09.226 pm	Alarm text. Alarm source:CPU "PLC_2" in block "ProgramAlarm_DB". Output of Par

Example

You can find a detailed application example in the Siemens Industry Online Support (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=99&lang=en&referer=%2fWW%2f&func=cslib.csinfo2&siteid=csius&extranet=standard&viewreg=WW&groupid=4000002) under the following FAQ ID: 98210758.

See also

Overview of the valid data types (Page 1908)

Get_AlarmState: Output alarm status

Description

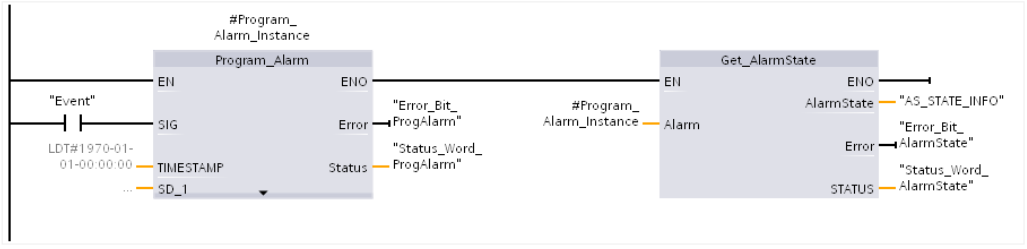
Use the instruction "Output alarm status" to output the alarm status of a program alarm.

There are three possible alarm statuses:

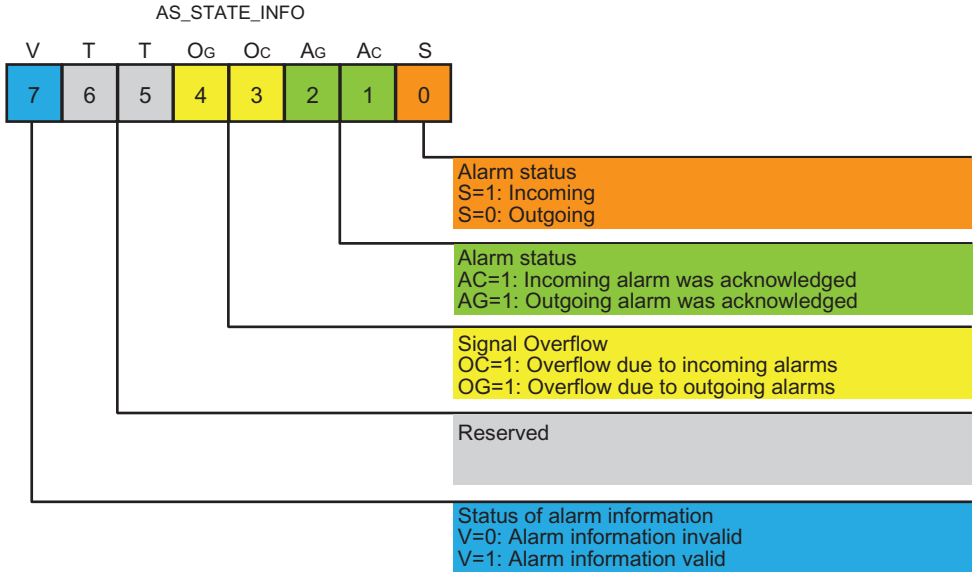
- Incoming
- Outgoing
- Acknowledged

The output of the alarm status always refers to a program alarm that was created using the instruction "Generate program alarm with associated values".

The program alarm is selected with the Alarm input parameter. Specify the instance DB of the instruction "Generate program alarm with associated values" at the parameter Alarm.



The alarm status is returned in a byte via the AlarmState output parameter. The meaning of the individual bits is shown in the following figure:



The execution status of the instruction is shown with the output parameters Error and STATUS.

Parameters

The following table lists the parameters of the instruction "Output alarm status":

Parameter	Declaration	Data type	Memory area	Description
Alarm	Input	ALARM_BASE	D	Instance of the instruction "Generate program alarm with associated values" • Alarm.Messagetype = Alarm_AP, the the bit Ac has either the signal status 0 or 1 and the bit Ag has the signal status 1 – not active: 0x86 (1000 0110) – active/not acknowledged: 0x85 (1000 0101) – active/acknowledged: 0x87 (1000 0111) – outgoing/not acknowledged: 0x84 (1000 0100) • Alarm.Messagetype = Notify_AP, the the bit Ac has either the signal status 0 or 1 and the bit Ag has the signal status 1 – not active: 0x86 (1000 0110) – active: 0x85 (1000 0101) • Alarm.Messagetype = Inforeport_AP, the bits Ac and Ag both have the signal status 1 – Static: 0x86 (1000 0110) • If the alarm is not active or it is an Inforeport, bit S has the signal status 0.
AlarmState	Output	BYTE	I, Q, M, D, L	Status of the alarm as a bit array

Parameter	Declaration	Data type	Memory area	Description
Error	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy STATUS to a free data area.

You can find additional information on valid data types under "See also".

Parameter STATUS

Error code* (W#16#...)	Explanation
8001	Invalid static alarm instance
8002	ID of the alarm is invalid
8003	No active alarms within the alarm class • Alarm_AP: The alarm is outgoing and acknowledged. • Notify_AP: The alarm is outgoing. • Inforeport Bit V is set to signal status 0.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

In the following example, you output the alarm status of a program alarm.

Create four tags in a global data block for storing the data.

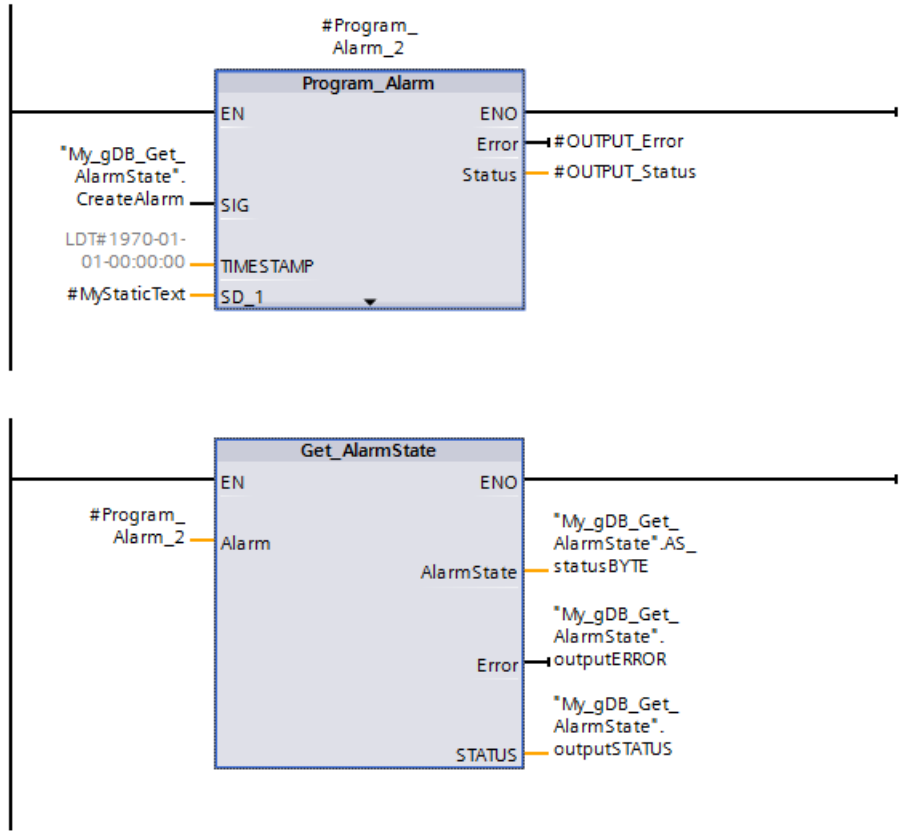
My_gDB_Get_AlarmState

	Name	Data type	Start value
1	Static		
2	AS_statusBYTE	Byte	16#0
3	outputERROR	Bool	false
4	outputSTATUS	Word	16#0
5	CreateAlarm	Bool	false

The "Get_AlarmState" instruction is called together with the "Program_Alarm" instruction in a function block. Create four parameters for the function block for interconnection of the instructions.

7		Static		
8		MyStaticText	String	'MyStaticText'
9		Program_Alarm_2	Program_Alarm	
10		Temp		
11		OUTPUT_Error	Bool	
12		OUTPUT_Status	Word	

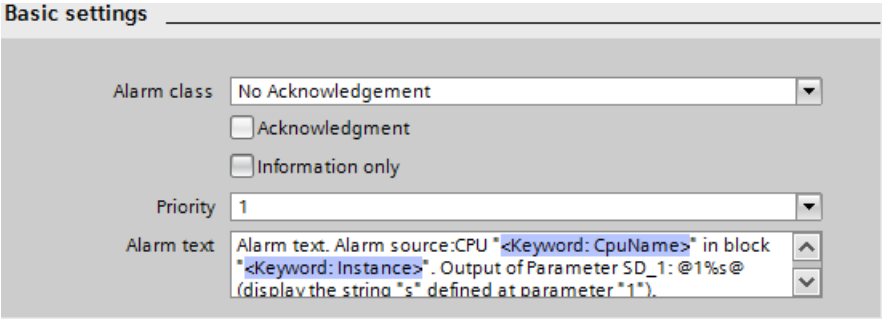
Interconnect the parameters of the instructions as follows.



Call the function block in an OB.

A PLC alarm is generated automatically when the "Program_Alarm" instruction is created. Open the PLC alarms dialog and select the alarm to be processed on the Program alarms tab. Create the alarm text, including two key words.

Select the following settings for the alarm. The alarm belongs to the alarm class "Notify_AP".



If input parameter SIG ("CreateAlarm") of the "Program_Alarm" instruction has the signal state "TRUE", the PLC alarm is output.

In the case of the "Get_AlarmState" instruction, the following occurs: The PLC alarm is communicated to the "Get_AlarmState" instruction using input parameter ALARM. At output parameter AlarmState ("AS_statusBYTE"), it is indicated that the alarm corresponding to the alarm class "Notify_AP" is active.

At output parameters ERROR ("outputERROR") and STATUS ("outputSTATUS"), it is indicated that the processing of the instruction is proceeding error-free.

My_gDB_Get_AlarmState

	Name	Data type	Start value	Monitor value
1	Static			
2	AS_statusBYTE	Byte	16#0	16#85
3	outputERROR	Bool	false	FALSE
4	outputSTATUS	Word	16#0	16#0000
5	CreateAlarm	Bool	false	TRUE

For the output of the PLC alarm, use the web server of the CPU.

Example

You can find a detailed application example in the Siemens Industry Online Support (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=99&lang=en&referer=%2fWW%2f&func=cslib.csinfo2&siteid=csius&extranet=standard&viewreg=WW&groupid=4000002) under the following FAQ ID: 98210758.

See also

Overview of the valid data types (Page 1908)

Program_Alarm: Generate program alarm with associated values (Page 3246)

Gen_UsrMsg: Generate user diagnostic alarms

Description

You use the "Gen_UsrMsg" instruction to generate an alarm that is entered in the diagnostic buffer.

You use the Mode parameter to select whether an incoming or an outgoing alarm is to be generated:

- When Mode = 1: Create an incoming alarm.
- When Mode = 2: Create an outgoing alarm.
- Regardless of whether an incoming or outgoing alarm is generated, the alarm always has the attribute "Information only".

The entry in the diagnostic buffer is created synchronously. Alarm transmission is asynchronous.

If an error occurs during the execution of an instruction, it is output via the parameter RET_VAL.

Content of the alarm

The content of the alarm is defined with a text list:

- Specify the text list you want to use with the parameter TextListID. For this purpose open the dialog "Text lists" in the project navigation. Show the column "ID" in the dialog "Text lists". Apply the ID at the parameter TextListID.
- Use the parameter TextID to select the text list entry you want to write in the diagnostic buffer. For this purpose select an entry from the "Text lists entries" dialog by applying a number from the columns "Range from / range to" at the parameter TextID. The same number must be used from both the "Range from" and "Range to" columns for the text list entry.
- Additional information about text lists can be found here: Text lists (Page 463)

Defining associated values

New associated values can be defined in the text entry and then added to the alarm:

- Associated values are defined by adding the following information in the text list entry: @<No. of the associated value><Element type><Format specification>@
- Use the system data type AssocValues to specify which associated value is added when generating the alarm.
- Additional information on the structure of associated values can be found here: Auto-Hotspot

Parameters

The following table shows the parameters of the "Gen_UsrMsg" instruction:

Parameter	Declaration	Data type	Memory area	Description
Mode	Input	UInt	I, Q, M, D, L or constant	Parameters for selecting the status of the alarm: • 1: incoming alarm • 2: outgoing alarm
TextID	Input	UInt	I, Q, M, D, L or constant	ID of the text list entry that should be used for the alarm text.
TextListID	Input	UInt	I, Q, M, D, L or constant	ID of the text list that contains text list entry.
Ret_Val	Return	Int	I, Q, M, D, L	Error code of the instruction.
AssocValues	InOut	VARIANT	D, L	Pointer to the system data type AssocValues that allows you to define the associated values.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter AssocValues

Use the system data type AssocValues to define which associated values will be sent. A maximum of 8 associated values are possible. The structure of the system data type is created by entering the data type "AssocValues" as a data block.

The associated values are selected by entering the numbers of the associated values for the parameters Value[x]. Note the following:

- The values for TextID and TextListID are treated by the instruction as the associated values to be sent. As a result, "1" and "2" are already assigned as numbers for addressing associated values. Do not use the numbers "1" or "2" to address associated values.
- Address the associated value at parameter Value [1] as number "3", at parameter Value [2] as number "4", and so forth.

Byte	Parameter	Data type	Start value	Description	Number of the associated value
0..1	Value[1]	UINT	0	First associated value of the alarm.	3
2..3	Value[2]	UINT	0	Second associated value of the alarm.	4
4..5	Value[3]	UINT	0	...	5
6..7	Value[4]	UINT	0	...	6
8..9	Value[5]	UINT	0	...	7
10..11	Value[6]	UINT	0	...	8
12..13	Value[7]	UINT	0	...	9
14..15	Value[8]	UINT	0	Eighth associated value of the alarm.	10

Parameter RET_VAL

The following table contains all specific error information that can be output via the RET_VAL parameter.

Error code* (W#16#...)	Explanation
0000	No error
8080	Value in the Mode parameter is not supported.
80C1	Resource bottleneck due to too many parallel calls.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

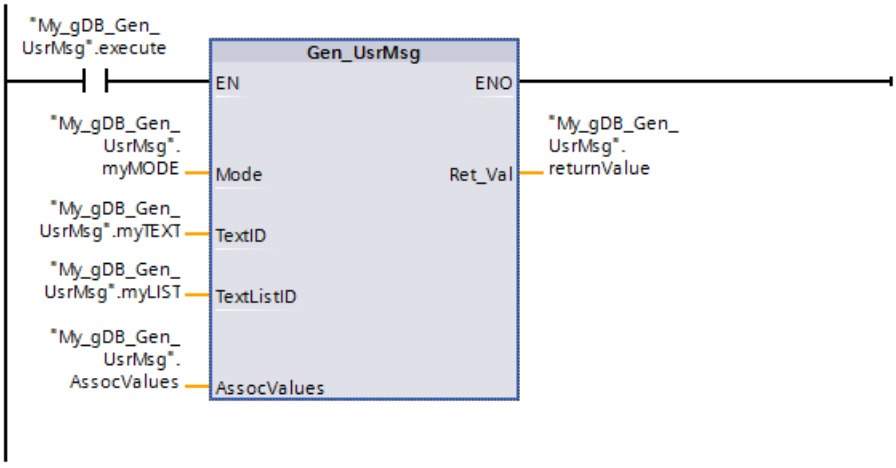
In the following example, you generate an alarm that is entered in the diagnostic buffer.

Create five tags and an "AssocValues" structure (with the AssocValues data type) in a global data block for storing the data.

My_gDB_Gen_UsrMsg

	Name	Data type	Start value
1	Static		
2	myMODE	UInt	1
3	myTEXT	UInt	3
4	myLIST	UInt	512
5	returnValue	Int	0
6	execute	Bool	false
7	AssocValues	AssocValues	
8	Value	Array[1..8] of UInt	
9	Value[1]	UInt	33
10	Value[2]	UInt	411
11	Value[3]	UInt	578
12	Value[4]	UInt	6122
13	Value[5]	UInt	722
14	Value[6]	UInt	8111
15	Value[7]	UInt	9829
16	Value[8]	UInt	10457

Interconnect the parameters of the instruction as follows.



Use the "Text lists" entry to create a Text list and a Text list entry for the alarm. Apply the ID of the text list at parameter TextListID ("myLIST"). Apply the ID (Range ...) of the text list entry at parameter TextID ("myTEXT"). Assign the alarm parameters as follows.

Text lists

Name	Id	Selection	Comment
USER_1	512	Decimal	

Text list entries of USER_1

Range from	Range to	Entry
3	3	This is a user generated message. Return Value[3]: @5I%6d@

If the normally open contact ("execute") supplies the signal state "TRUE", the "Gen_UsrMsg" instruction is executed. According to the value of parameter Mode ("myMODE"), an incoming

alarm is generated. The alarm to be output is made known to the instruction using parameters TextListID ("myLIST") and TextID ("myTEXT"). The associated values of the alarm are transferred using parameter AssocValues ("AssocValues").

When the alarm is generated, the character string "@5I%6d@" contained in the alarm text is interpreted as follows:

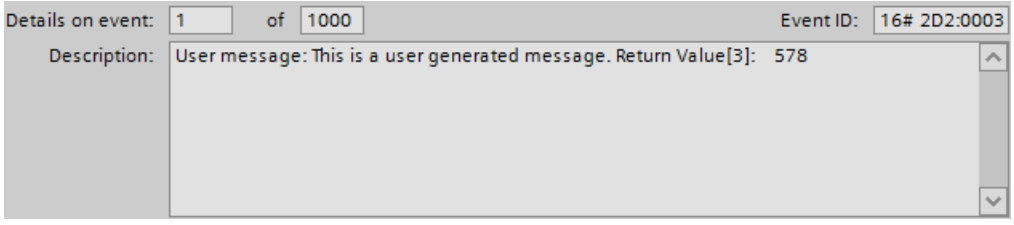
- The associated value with the number "5" is read out in the INT data type. The number corresponds to parameter Value[3] of the "AssocValues" structure.
- The associated value is output as a decimal number. The decimal number is limited to six places.

At output parameter Ret_Val ("returnValue"), it is indicated that the processing of the instruction is proceeding error-free.

My_gDB_Gen_UsrMsg

	Name	Data type	Start value	Monitor value
1	Static			
2	myMODE	UInt	1	1
3	myTEXT	UInt	3	3
4	myLIST	UInt	512	512
5	returnValue	Int	0	0
6	execute	Bool	false	FALSE
7	AssocValues	AssocValues		
8	Value	Array[1..8] of UInt		
9	Value[1]	UInt	33	33
10	Value[2]	UInt	411	411
11	Value[3]	UInt	578	578
12	Value[4]	UInt	6122	6122
13	Value[5]	UInt	722	722
14	Value[6]	UInt	8111	8111
15	Value[7]	UInt	9829	9829
16	Value[8]	UInt	10457	10457

For the output of the alarm, for a CPU of the S7-1500 series, open the entry "Online & Diagnostics > Diagnostic buffer".



Example

You can find a detailed application example in the Siemens Industry Online Support (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=99&lang=en&referer=%2fWW%2f&func=cslib.csinfo2&siteid=csius&extranet=standard&viewreg=WW&groupid=4000002) under the following FAQ ID: 98210758.

11.6.3.9 Diagnostics

RD_SINFO: Read current OB start information

Description

You use the instruction "RD_SINFO" to read the start information

- of the last OB called that has not yet been completely executed, and
- of the last startup OB started.

There is no time stamp in either case. If the call is in OB 100, OB 101 or OB 102, two identical start information messages will be returned.

Parameter

The following table shows the parameters of the "RD_SINFO" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Error information
TOP_SI	Output	VARIANT	D, L	Start information of the current OB
START_UP_SI	Output	VARIANT	D, L	Start information of the startup OB last started

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

SDTs of the TOP_SI parameter

The following table shows the possible SDTs for the TOP_SI parameter:

Organization blocks (OB)	System data types (SDT)	System data type numbers
Any	SI_classic	592
	SI_none	593
ProgramCycleOB	SI_ProgramCycle	594
TimeOfDayOB	SI_TimeOfDay	595
TimeDelayOB	SI_Delay	596
CyclicOB	SI_Cyclic	597
ProcessEventOB	SI_HWInterrupt	598
ProfileEventOB StatusEventOB UpdateEventOB	SI_Submodule	601
SynchronousCycleOB	SI_SynchCycle	602
IOredundancyErrorOB	SI_IORedundancyError	604
CPUredundancyErrorOB	SI_CPURedundancyError	605
TimeErrorOB	SI_TimeError	606
DiagnosticErrorOB	SI_DiagnosticInterrupt	607

Organization blocks (OB)	System data types (SDT)	System data type numbers
PullPlugEventOB	SI_PlugPullModule	608
PeripheralAccessErrorOB	SI_AccessError	609
RackStationFailureOB	SI_StationFailure	610
ServoOB	SI_Servo	611
IpoOB	SI_Ipo	612
StartupOB	SI_Startup	613
ProgrammingErrorOB IOaccessErrorOB	SI_ProgIOAccessError	614

SDTs of the START_UP_SI parameter

The following table shows the possible SDTs for the START_UP_SI parameter:

System data types (SDT)	System data type numbers
SI_classic	592
SI_none	593
SI_Startup	613

Structures

The following tables set out the meaning of the structure elements of the individual structures:

Table 11-61 SI_classic structure

Structure element	Data type	Description
EV_CLASS	BYTE	• Bits 0 to 3: Event ID • Bits 4 to 7: Event class
EV_NUM	BYTE	Event number
PRIORITY	BYTE	Priority class number (Meaning of B#16#FE: OB not available or disabled or cannot be started in current operating mode)
NUM	BYTE	OB number
TYP2_3	BYTE	Data ID 2_3: Identifies the information entered in ZI2_3
TYP1	BYTE	Data ID 1: Identifies the information entered in ZI1
ZI1	WORD	Additional information 1
ZI2_3	DWORD	Additional information 2_3

Table 11-62 SI_none structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)

Table 11-63 SI_ProgramCycle structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 1	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
Remanence	BOOL	For OB_Class = 1

Table 11-64 SI_TimeOfDay structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 10	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
CaughtUp	BOOL	For OB_Class = 10
SecondTime	BOOL	For OB_Class = 10

Table 11-65 SI_Delay structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 20	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Sign	WORD	For OB_Class = 20

Table 11-66 SI_Cyclic structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 30	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92

Table 11-67 SI_HWInterrupt structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 40	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
USI	WORD	For OB_Class = 40
IChannel	USINT	For OB_Class = 40
EventType	BYTE	For OB_Class = 40

Table 11-68 SI_Submodule structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Slot	UINT	For OB_Class = 55, 56, 57
Specifier	WORD	For OB_Class = 55, 56, 57

Table 11-69 SI_SynchCycle structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 61	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
PIP_Input	BOOL	For OB_Class = 61, 91, 92
PIP_Output	BOOL	For OB_Class = 61, 91, 92

Structure element	Data type	Description
IO_System	USINT	For OB_Class = 61, 91, 92
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92
SyncCycleTime	LTIME	Calculated cycle time

Table 11-70 SI_IORedundancyError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 70	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_ANY	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86

Table 11-71 SI_CPURedundancyError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 72	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Switch_Over	BOOL	For OB_Class = 72

Table 11-72 SI_TimeError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 80	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86
Csg_OBnr	OB_ANY	For OB_Class = 80
Csg_Prio	UINT	For OB_Class = 80

Table 11-73 SI_DiagnosticInterrupt structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 82	OB class for "No information" or "Optimized start information"

Structure element	Data type	Description
OB_Nr	UINT	OB number (1 ... 32767)
IO_State	WORD	For OB_Class = 82
LADDR	HW_ANY	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Channel	UINT	For OB_Class = 82
MultiError	BOOL	For OB_Class = 82

Table 11-74 SI_PlugPullModule structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 83	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86

Table 11-75 SI_AccessError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 85	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86
IO_Addr	UINT	For OB_Class = 85
IO_LEN	UINT	For OB_Class = 85

Table 11-76 SI_StationFailure structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 86	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86

Table 11-77 SI_Servo structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 91	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
PIP_Input	BOOL	For OB_Class = 61, 91, 92
PIP_Output	BOOL	For OB_Class = 61, 91, 92
IO_System	USINT	For OB_Class = 61, 91, 92
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92
Synchronous	BOOL	

Table 11-78 SI_Ipo structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 92	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
PIP_Input	BOOL	For OB_Class = 61, 91, 92
PIP_Output	BOOL	For OB_Class = 61, 91, 92
IO_System	USINT	For OB_Class = 61, 91, 92
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92
Reduction	UINT	For OB_Class = 92

Table 11-79 SI_Startup structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 100	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LostRetentive	BOOL	For OB_Class = 100
LostRTC	BOOL	For OB_Class = 100

Table 11-80 SI_ProgIOAcessError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT	OB class for "No information" or "Optimized start information"

Structure element	Data type	Description
OB_Nr	UINT	OB number (1 ... 32767)
BlockNr	UINT	For OB_Class = 121, 122
Reaction	USINT	For OB_Class = 121, 122
Fault_ID	BYTE	For OB_Class = 121, 122
BlockType	USINT	For OB_Class = 121, 122
Area	USINT	For OB_Class = 121, 122
DBNr	DB_ANY	For OB_Class = 121, 122
Csg_OBNr	OB_ANY	For OB_Class = 121, 122
Csg_Prio	USINT	For OB_Class = 121, 122
Width	USINT	For OB_Class = 121, 122

Note

The structure elements specified for the SI_classic structure are identical in content to the temporary tags of any OB created with the block property "Default".

Please note, however, that temporary tags of the individual OBs can have different names and different data types. Also note that the call interface of each OB includes additional information regarding the date and the time of the OB request.

Bits 4 to 7 of the EV_CLASS structure element contain the event class. The following values are possible here:

- 1: Start events from standard OBs
- 2: Start events from synchronous error OBs
- 3: Start events from asynchronous error OBs

The PRIORITY structure element supplies the priority class belonging to the current OB.

Apart from these two elements, NUM is also relevant. NUM contains the number of the current OB or the startup OB that was started last.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
8080	Start information of the current OB does not correspond to the specified user-defined data type
8081	Start information of the current OB does not correspond to the specified system data type
8082	Start information of the last startup OB started does not correspond to the specified user-defined data type
8083	Start information of the last startup OB started does not correspond to the specified system data type

Example

OB 80 is the OB that was called last and that has not yet been completely processed and OB 100 is the start-up OB that was started last.

The following table shows the assignment between the structure elements of the TOP_SI parameter of the "RD_SINFO" instruction and the associated local tags of OB 80.

TOP_SI structure element	Data type	OB 80 - Associated local tag	Data type
EV_CLASS	BYTE	OB80_EV_CLASS	BYTE
EV_NUM	BYTE	OB80_FLT_ID	BYTE
PRIORITY	BYTE	OB80_PRIORITY	BYTE
NUM	BYTE	OB80_OB_NUMBR	BYTE
TYP2_3	BYTE	OB80_RESERVED_1	BYTE
TYP1	BYTE	OB80_RESERVED_2	BYTE
ZI1	WORD	OB80_ERROR_INFO	WORD
ZI2_3	DWORD	OB80_ERR_EV_CLASS	BYTE
		OB80_ERR_EV_NUM	BYTE
		OB80_OB_PRIORITY	BYTE
		OB80_OB_NUM	BYTE

The following table shows the assignment between the structure elements of the START_UP_SI parameter of the "RD_SINFO" instruction and the associated local tags of OB 100.

START_UP_SI structure element	Data type	OB 100 - Local tag	Data type
EV_CLASS	BYTE	OB100_EV_CLASS	BYTE
EV_NUM	BYTE	OB100_STRTUP	BYTE
PRIORITY	BYTE	OB100_PRIORITY	BYTE
NUM	BYTE	OB100_OB_NUMBR	BYTE
TYP2_3	BYTE	OB100_RESERVED_1	BYTE
TYP1	BYTE	OB100_RESERVED_2	BYTE
ZI1	WORD	OB100_STOP	WORD
ZI2_3	DWORD	OB100_STRT_INFO	DWORD

See also

Evaluating errors with output parameter RET_VAL (Page 2195)

RT_INFO: Read out runtime statistics

Description

Use the instruction "RT_INFO" to generate statistics on the runtime of specific organization blocks, communication or the user program.

Use the MODE parameter to select which information is going to be output:

- MODE 1 to 3 return the data on the runtime of a specific organization block whose number was specified at the OB parameter.
- MODE 10 and 11 return the proportions of runtime used for communication and the user program.
- MODE 20 and 21 return the percentages of the runtime allotted to communication and the user program for the last program cycle.
- MODE 23 to 25 output the shortest, longest, and current cycle time of the user program.
- MODE 30 to 32 return data on configured settings of the user program.

All measurements are started once again when the CPU transitions from startup to RUN.

Parameters

The following table shows the parameters of the instruction "RT_INFO":

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	Use the MODE parameter to select the information you want to read out (see table "MODE parameter").
OB	Input	OB_ANY	I, Q, M, D, L or constant	Use the OB parameter to select the OB for which you want to read out information.
RET_VAL	Return	INT	I, Q, M, D, L	Error information (see " RET_VAL parameter").
INFO	InOut	VARIANT	I, Q, M, D, L	Pointer to the area to which the read data is to be written. The data type required for INFO depends on the MODE parameter (see table "MODE parameter").

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter MODE

The following table shows which information is returned for the values at the MODE parameter.

MODE (decimal)	Description	Note	Value at OB parameter	Data type of INFO	Available as of CPU version
1	Runtime of a specific OB	The runtime information always refers to the specified OB. Processes, such as interruptions by OBs of a higher priority or communication, are not included. You can use the instruction "RUNTIME" to measure the runtime of the entire program.	OB number	LTIME	S7-1500 V1.5
2	Maximum runtime of a specific OB		OB number	LTIME	S7-1500 V1.5
3	Minimum runtime of a specific OB		OB number	LTIME	S7-1500 V1.5

MODE (decimal)	Description	Note	Value at OB parameter	Data type of INFO	Available as of CPU version
10	Percentage of total runtime used by higher-priority OBs The runtime of all OBs used in the user program with a higher priority than cyclical program OBs is output (ProgramCycle). These usually include all OB types except startup OBs. The CPU determines which OBs can be used and their priority. This information is included in the basic chapters on programming.	-	Not relevant	UINT	S7-1500 V1.5
11	Percentage of total runtime used by communication The percentage of communication processes in the total runtime of the user program is output.		Not relevant	UINT	S7-1500 V1.5
20	Identical to MODE 10 except that the evaluation is based on the last program cycle executed.	• If the cycle time is too high (> 1 second), the calculation cannot be performed. The instruction outputs the value 65535 (0xFFFF). • If the cycle time is too low (< 1 millisecond), the calculation cannot be performed. In this case, the last cycle with a duration of at least one millisecond is evaluated. By assigning a minimum cycle time in the CPU properties, you can prevent cycle times from falling below 1 ms.	Not relevant	UINT	S7-1500 V1.7
21	Identical to MODE 11 except that the evaluation is based on the last program cycle executed.		Not relevant	UINT	S7-1500 V1.7
23	Longest cycle time Duration of the longest cycle since the last transition from STOP to RUN.	The times correspond to the "Cycle times measured" values in the "Cycle time" dialog of the TIA Portal. You can open the dialog using Online & Diagnostics > Diagnostics > Cycle time.	Not relevant	LTIME	S7-1500 V1.7
24	Shortest cycle time Duration of the shortest cycle since the last transition from STOP to RUN.		Not relevant	LTIME	S7-1500 V1.7
25	Current/last cycle time Duration of the last cycle.		Not relevant	LTIME	S7-1500 V1.7

MODE (decimal)	Description	Note	Value at OB parameter	Data type of INFO	Available as of CPU version
30	Cycle monitoring time Maximum permitted duration of the CPU program. If the cycle time exceeds the cycle monitoring time, the CPU goes to STOP or calls a time error OB.	The times correspond to the "Cycle times assigned" values in the "Cycle time" dialog of the TIA Portal. You can open the dialog using Online & Diagnostics > Diagnostics > Cycle time.	Not relevant	LTIME	S7-1500 V1.5
31	Output of the configured minimum cycle time for the user program. If a minimum cycle time has been assigned in the CPU properties, the operating system delays the start of the new cycle until the minimum cycle time has been reached.		Not relevant	LTIME	S7-1500 V1.5
32	Output of the configured maximum communication load as a percentage	The specification of the percentage of the cycle load allotted to communication is made in the CPU properties under "Communication load".	Not relevant	UINT	S7-1500 V1.5

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

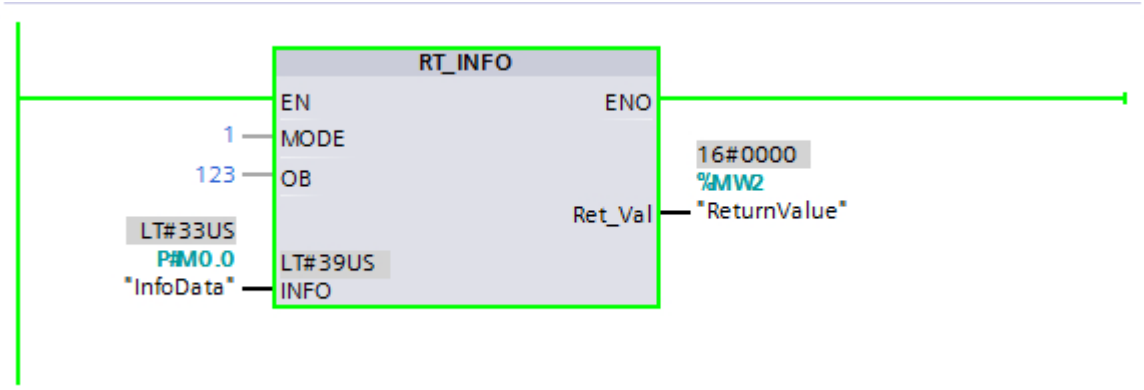
Error code* (W#16#...)	Explanation
0	No error
8080	Value at the MODE parameter is not supported.
8081	Organization block selected at OB parameter does not exist in the user program.
8092	Selected MODE parameter is not supported by this CPU version.
80C3	Insufficient resources. Try to call the instruction again at a later time.
8452	Data type at INFO parameter is incorrect. Check whether the correct data type was selected depending on the MODE parameter.

Example

In the following example, you read out the runtime of a cycle organization block.

- Create a new block of type "Program cycle". Specify the OB number at the OB parameter.
- Enter "1" (read out runtime of a particular OB) at the MODE parameter.
- Specify a tag with the data type LTIME at the INFO parameter ("InfoData" in this case).
- Specify a tag with the data type INT at the Ret_Val parameter to output the error messages of the instruction.

After calling the instruction, the actual measured runtime is written to the "InfoData" tag.



See also

Evaluating errors with output parameter RET_VAL (Page 2195)

LED: Read LED status

Description

You can use the "LED" instruction to read out the status (e.g., "On" or "Off") of a particular module LED.

- With the LADDR parameter, you address the CPU or the interface.
- With the LED parameter, you select the module LED whose current status is to be read out using the instruction.
- The RET_VAL parameter outputs the status of the selected LED when the instruction is called. Depending on the LED selected, only certain status information may be displayed. For example, some LEDs have only color information. Refer to the hardware documentation of the respective module for the available status information of a particular LED.

Parameters

The following table shows the parameters of the "LED" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Hardware identifier of the CPU or interface The number is assigned automatically and is stored in the properties of the CPU or interface in the hardware configuration (*CPU name* + *~Common*).
LED	Input	UINT	I, Q, M, D, L or constant	Identification number of the LED: • 1: STOP/RUN • 2: ERROR • 3: MAINT (maintenance) • 4: Redundant • 5: Link (green) • 6: Rx/Tx (yellow)
RET_VAL	Return	INT	I, Q, M, D, L	Status of LED

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

RET_VAL parameter

RET_VAL	Description
0 to 9	LED status: • 0 = LED does not exist • 1 = Permanently switched off • 2 = Color 1 (e.g., for LED STOP/RUN: green) permanently ON • 3 = Color 2 (e.g., for LED STOP/RUN: orange) permanently ON • 4 = Color 1 flashing at 2 Hz • 5 = Color 2 flashing at 2 Hz • 6 = Colors 1 and 2 flashing alternately at 2 Hz • 7 = LED is active, color 1 • 8 = LED is active, color 2 • 9 = LED exists, but status information not available
8091	Hardware component addressed using the LADDR parameter does not exist.
8092	A hardware component that does not support LEDs was addressed using the LADDR parameter.
8093	The hardware identifier specified at the LED parameter is not defined.
80Bx	The CPU specified in the LADDR parameter does not support the "LED" instruction.

Example

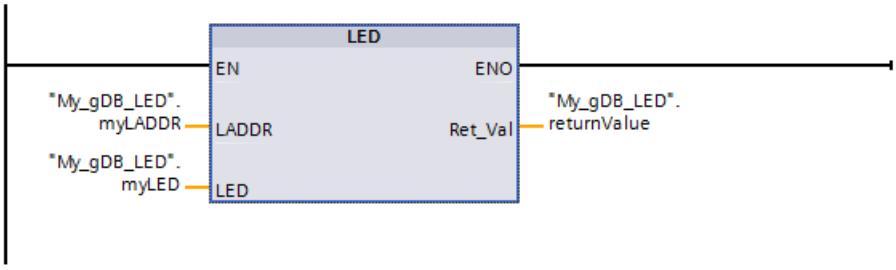
In the following example, you read out the status of the CPU LED.

Create three tags in a global data block for storing the data.

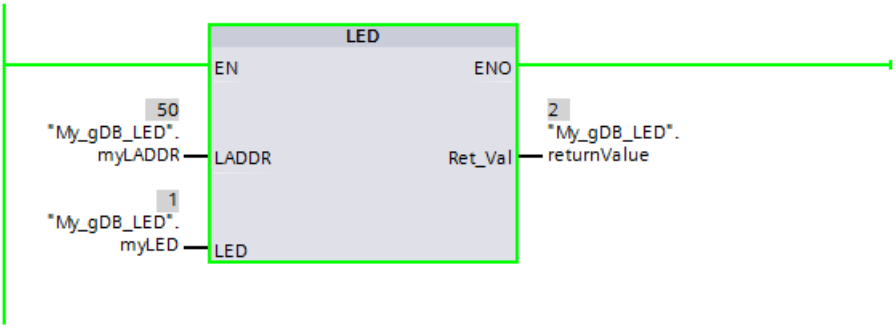
My_gDB_LED

		Name	Data type	Start value
1		Static		
2		myLADDR	HW_IO	50
3		myLED	UInt	1
4		returnValue	Int	0

Interconnect the parameters of the instruction as follows.



The HW identifier of the CPU is made known to the "LED" instruction through parameter LADDR ("myLADDR"). The CPU LED to be monitored is made known through parameter LED ("myLED"). The status of the CPU LED (STOP/RUN) is queried. If the CPU is placed in RUN mode from STOP, the value "6" (green and orange alternating) is indicated at output parameter RET_VAL ("returnValue"). Then, the value "2" (permanently green) is indicated as the LED status ("returnValue").



Note: The meaning of the colors for the STOP/RUN LED are as follows:

Color	Meaning
Red	STOP
Green	RUN
Alternating green and orange	The CPU is loading
Alternating red and green	Program processing is running

Additional information on the meaning of the LED colors can be found in the hardware description of the CPU.

Get_IM_Data: Reading identification and maintenance data

Description

The "Get_IM_Data" instruction reads the identification and maintenance data (I&M) data from a device. Use the LADDR parameter to select the device from which you want to read the I&M via the hardware identifier.

You select the data to be read via the instruction using the IM_TYPE parameter:

- IM_TYPE = 0: I&M 0 data
  The I&M 0 data are the device-specific basic information of a device and contain information such as the manufacturer ID, order number, serial number and the hardware and firmware version. Only read access is possible to the I&M 0 data. The information is also displayed in the TIA Portal with the "Online & Diagnostics" view of a device.
- IM_TYPE = 11: I&M 1 data from the parameter assignment data of the CPU
  The I&M 1 data contains a functional description of the device and the location ID, i.e. the information about how the device is designated within the plant.
- IM_TYPE = 12: I&M 2 data from the parameter assignment data of the CPU
  The I&M 2 data includes the date of installation, i.e. the information about when the device was installed in the plant.
- IM_TYPE = 13: I&M 3 data from the parameter assignment data of the CPU
  The I&M 3 data contains additional information about the installed device. The additional information is free text and can be assigned as desired.

The I&M data read is written to the addressed area defined at the DATA parameter.

The execution status of the read job is displayed via the BUSY, DONE, ERROR output parameters and the two middle bytes of the STATUS output parameter.

Definition: Identification and maintenance data (I&M)

Identification and maintenance (I&M) data refers to information stored in a module, which assists you in checking the plant configuration, locating hardware changes in a plant and eliminating errors.

- Identification data (I-data) is static information about a device that can only be read.
- Maintenance data (M-data) refers to plant-dependent information, such as the installation location or date. Maintenance data are created during configuration and written to the module.

Parameters

The following table shows the parameters of the "Get_IM_Data" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware ID of the device The number is assigned automatically and is stored in the properties of the device or in the hardware configuration.
IM_TYPE	Input	UINT	I, Q, M, D, L or constant	Identification and maintenance data number Possible values: • 0: I&M 0 data • 11: I&M 1 data from the parameter assignment data of the CPU • 12: I&M 2 data from the parameter assignment data of the CPU • 13: I&M 3 data from the parameter assignment data of the CPU
DATA	InOut	VARIANT	I, Q, M, D, L	Area for storing the read identification and maintenance data (see below).
DONE	Output	BOOL	I, Q, M, D, L	The instruction was executed successfully. The I&M data were transferred to the DATA parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Execution of the instruction complete or not yet started. • 1: Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display it, you should therefore copy STATUS to a free data area.

You can find more information on the data types in "Overview of the valid data types (Page 1908)".

Parameter DATA for I&M 0 data

You can use an array (ARRAY of BYTE) or a special data structure to store the I&M 0 data:

- If you address an array (ARRAY of BYTE) at parameter DATA, the read I&M 0 data is copied as a byte sequence to DATA. If the addressed array is longer than the read data, the unneeded bytes are filled with zeros.
- The following "IM0_Data" structure can also be used at the DATA parameter for I&M 0 data:

Parameter		Data type	Bytes	Description
Manufacturer_ID		UINT	2	Manufacturer ID (e.g. "42" for SIEMENS)
Order_ID		CHAR[20]	20	Order number
Serial_Number		CHAR[16]	16	Serial number
Hardware_Revision		UNIT	2	Hardware revision
Software_Revision		STRUCT	4	Firmware revision
	Type	CHAR	1	-
	Functional	UNIT8	1	-
	Bugfix	UNIT8	1	-
	Internal	UINT8	1	-
Revision_Counter		UINT	2	Revision counter
Profile_ID		UNIT	2	Profile
Profile_Specific_Type		UNIT	2	Device class
IM_Version		UNIT	2	I&M version
I&M_Supported		UNIT	2	I&M data (I&M 0-I&M 4) supported at the device end

If a different data type is used at the DATA parameter, the STATUS parameter outputs error code 8093.

DATA parameter for I&M 1, I&M 2, and I&M 3 data

You can use a string (STRING), an array (ARRAY of CHAR/BYTE) or a data structure (STRUCT) to store the I&M data:

- If you address a string (STRING data type) at the the DATA parameter, the length of the string automatically adjusts the length of the read I&M data (up to 254 characters).
- If you address a data structure (ARRAY of CHAR/BYTE or STRUCT) a the DATA parameter, the I&M data read are written to the individual components of the data type used. If the addressed data structure is longer than the read data, the remaining components are filled with zeros.
- If you crate a data structure with a STRUCT data type at the DATA parameter, use a data block without optimized block access (see category "Attributes" of the block properties).

If a data type other than STRING, ARRAY of BYTE/CHAR or STRUCT is used at the DATA parameter, the STATUS parameter generates error code 8093.

Note

Additional information about I&M data

You can find more information about I & M data, for example, on the PROFIBUS & PROFINET International websites (link: http://www.profibus.com (http://www.profibus.com)).

Parameter STATUS

Error code* (W#16#...)	Description
0	No error
7000	No job in progress.
7001	First call of the asynchronous instruction "Get_IM_Data". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
7002	Additional call of the asynchronous instruction "Get_IM_Data". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
807F	Maximum of 10 parallel executions of the instances exceeded: • Avoid more than 10 parallel calls of the instruction. • Repeat the call at a later time, as this is a temporary error.
8091	The device addressed at the LADDR parameter does not exist.
8092	LADDR addresses a device that does not support the output of the I&M data.
8093	Data type at the DATA parameter is not supported.
80B1	The "Get_IM_Data" instruction is not supported by the CPU used.
80B2	Invalid value at the IM_TYPE parameter or the selected IM_TYPE is not supported by the CPU or the addressed device.
8752	The memory area specified at the DATA parameter is too small to store all the I&M data. The read I&M data are stored only up to the maximum length of the specified memory area.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

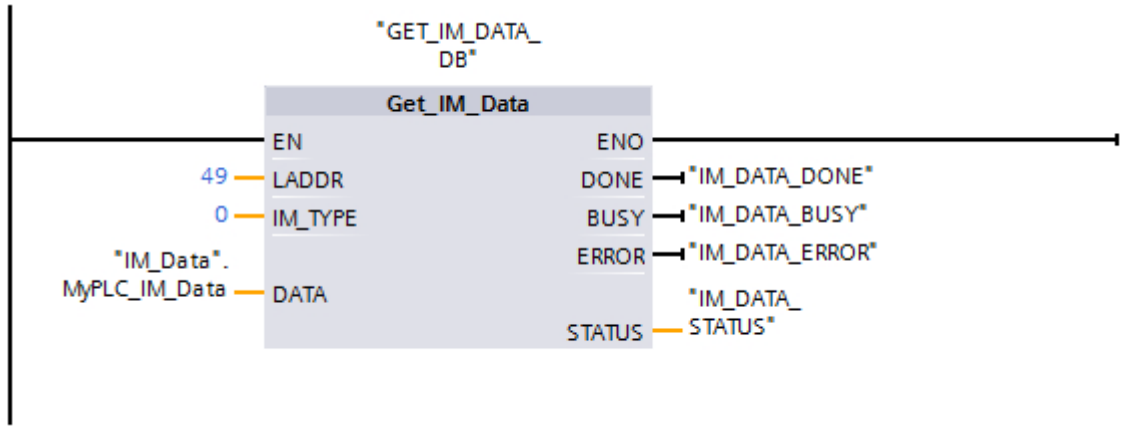
In the following example, you read out the IM0 data of an S7-1500 CPU. The IM0 data are the basic information of a device and contain information such as the manufacturer ID, order number, serial number and the hardware and firmware version.

To store the IM0 data, you create a structure with the IM0_Data data type in a global data block. You can assign any name to the structure ("MyPLC_IM_Data" in this case).

IM_Data

	Name	Datentyp	Startwert
1	Static		
2	MyPLC_IM_Data	IM0_Data	
3	Manufacturer_ID	UInt	0
4	Order_ID	String[20]	''
5	Serial_Number	String[16]	''
6	Hardware_Revision	UInt	0
7	Software_Revision	IM0_Version	
8	Type	Char	' '
9	Functional	USInt	0
10	Bugfix	USInt	0
11	Internal	USInt	0
12	Revision_Counter	UInt	0
13	Profile_ID	UInt	0
14	Profile_Specific_Type	UInt	0
15	IM_Version	Word	16#0
16	IM_Supported	Word	16#0

Enter the hardware identifier of the CPU at the LADDR parameter. The hardware identifier uniquely identifies the product. To determine the hardware identifier of your CPU, open the PLC tag table and the System constants tab. Then, search for the CPU in the Name column. The associated value is the hardware identifier that you enter at the LADDR parameter.



Once the instruction is successfully executed, the IM0 data is written to the data block.

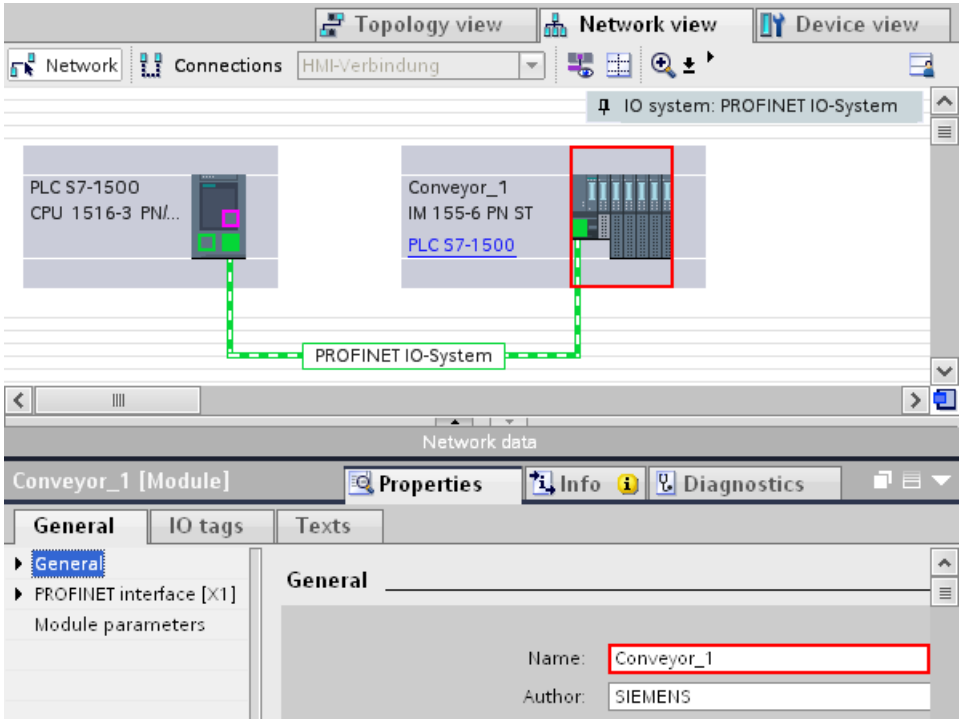
IM_Data

	Name	Datentyp	Startwert	Beobachtungswert
1	Static			
2	MyPLC_IM_Data	IM0_Data		
3	Manufacturer_ID	UInt	0	42
4	Order_ID	String[20]	''	'6ES7 511-1AK00-0AB0 '
5	Serial_Number	String[16]	''	'S C-DOS710132013'
6	Hardware_Revision	UInt	0	3
7	Software_Revision	IM0_Version		
8	Type	Char	' '	'V'
9	Functional	USInt	0	1
10	Bugfix	USInt	0	5
11	Internal	USInt	0	0
12	Revision_Counter	UInt	0	0
13	Profile_ID	UInt	0	0
14	Profile_Specific_Type	UInt	0	0
15	IM_Version	Word	16#0	16#0101
16	IM_Supported	Word	16#0	16#001E

GET_NAME: Reading the name of a module

Description

The instruction "GET_NAME" reads the name of an IO device. The name is displayed in the network view and in the properties of the IO device:



You select the IO device by using the hardware identifier of the PROFINET IO system (at the LADDR parameter) and the device number of the IO device (STATION_NR parameter).

Once the instruction has been executed, the name of the IO device is written to the area addressed with the DATA parameter.

The name that is read depends on the type of IO device:

- For a DP slave or IO device, the name of the head module is output.
- For an I-slave or I-device, the name of the interface module is output.
- The name of the interface is output for an HMI panel.
- The name of the interface module is output for a PC station.
- The name of the Device Access Point (DAP) is displayed (name of the interface or head module) for GSD devices.

The length of the name is output at the LEN parameter. If the name is longer than the area specified at the DATA parameter, only that section which corresponds to the maximum length of the addressed area will be written.

The maximum length for a name is 128 characters.

Note

Name of the CPU readout (Version 1.1)

If you assign a "0" at each of the parameters LADDR and STATION_NR the instruction will output the name of the CPU.

Parameters

The following table shows the parameters of the "GET_NAME" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IOSYSTEM	I, Q, M, D, L or constant	Hardware identifier of the PROFINET IO system. The number can be taken from the system constants or the properties of the IO system.
STATION_NR	Input	UINT	I, Q, M, D, L or constant	Device number of the IO device. The device number can be applied in the Network view from the properties of the IO device under "Ethernet addresses".
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the area to which the name of the module is written.
DONE	Output	BOOL	I, Q, M, D, L	The instruction was executed successfully. Name of the module transferred to the area at the DATA parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Execution of the instruction complete. • 1: Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
LEN	Output	DINT	I, Q, M, D, L	Length of the name of the IO device (number of characters).
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy STATUS to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter STATUS

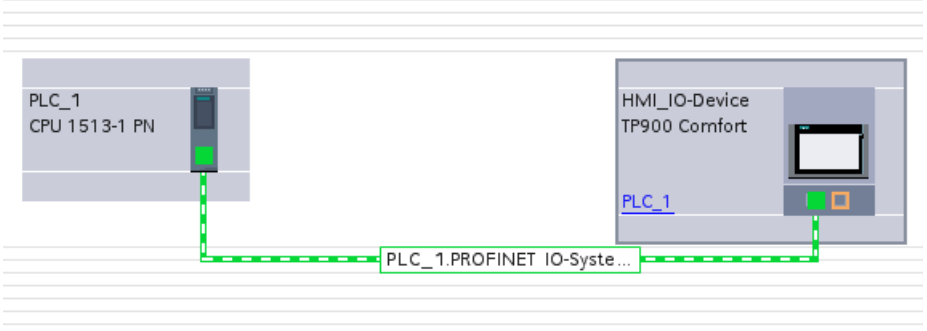
Error code* (W#16#...)	Explanation
0	No error
7000	No job in progress.
7001	First call of the asynchronous instruction "GET_NAME". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
7002	Additional call of the asynchronous instruction "GET_NAME". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
8090	• The LADDR input parameter contains an invalid value. • The hardware identifier specified at the LADDR parameter does not exist in the project.
8092	The value at the LADDR parameter does not address a PROFINET IO system.
8093	Instruction does not support data type at the DATA parameter.
8095	Device number (STATION_NR parameter) does not exist in the selected PROFINET IO system or does not address an IO device.
80B1	The CPU used does not support the instruction.
80C3	Temporary resource error: The CPU is currently processing the maximum possible number of simultaneous block calls. "GET_NAME" cannot be executed unless at least one of the block calls is finished.
8852	The area specified at the DATA parameter is too short for the full name of the IO device. The name will be written up to the maximum possible length. To read the full name, use a longer data area at the DATA parameter. The area must have at least as many characters as there are at the LEN parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

The following example shows how you can read out the station name of a HMI panel.

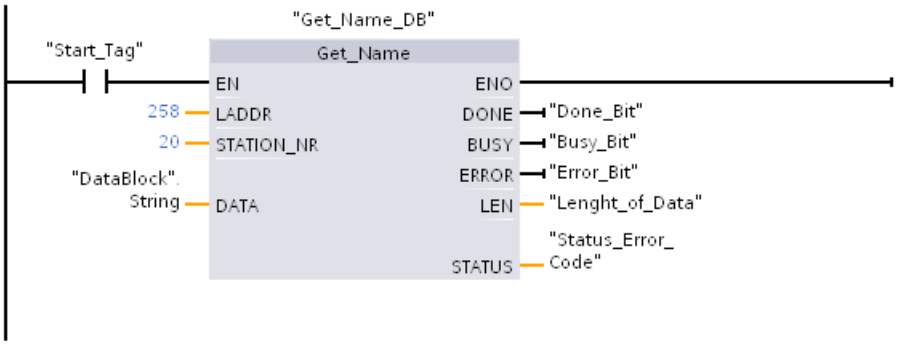
Configuration of the HMI panel:

- The HMI panel with the station name "HMI_IO-Device" is created in the network view and assigned to the same PROFINET IO system as the CPU.
- The operating mode "IO device" was activated in the properties of the hardware configuration for the HMI panel and the CPU was assigned as the IO controller.
- The device number "20" assigned in the properties under "Ethernet addresses".



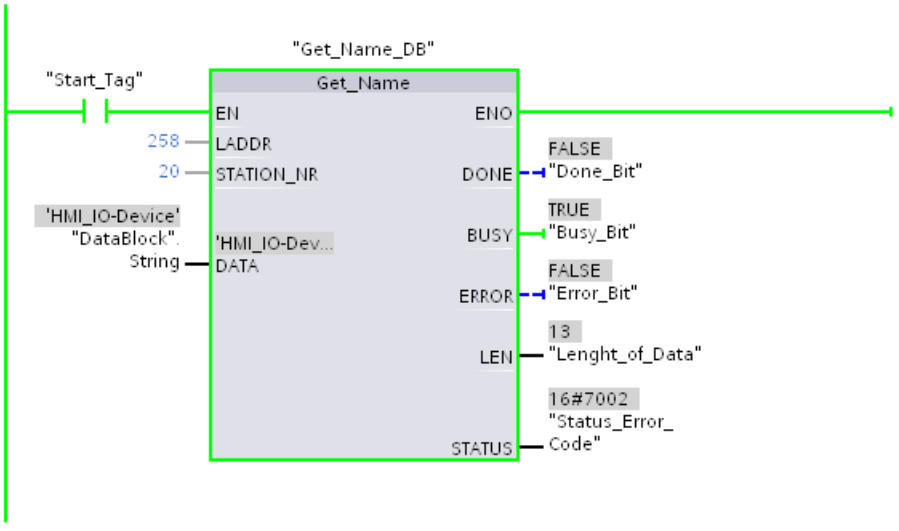
Assigning parameters to the instruction "GET_NAME":

- The hardware ID (258) of the IO system was specified at the parameter LADDR.
- The device number (20) of the HMI panel was specified at the parameter STATION_NR.
- A tag was connected with the data type STRING of a data block at the parameter DATA.
- PLC tags (memory area, flags) were defined for the output parameters of the instruction.



Execution of the instruction:

- After execution of the instruction, the name of the station of the HMI panels (HMI_IO-Device) is written into the data block at the parameter DATA.
- The number of characters of the name is output (13) at the parameter LEN.



GetStationInfo: Read information of an IO device

Description

You can use the "GetStationInfo" instruction to read information of a PROFINET IO device. The instruction also enables you to read the information of an IO device located in a lower-level IO system (connected via CP/CM).

The IO device is addressed via the hardware identifier of the station at the LADDR parameter. The hardware identifier is displayed in the "Devices & Networks" view in the station properties.

Use the MODE parameter to select the information to be read.

At the DATA parameter, specify the data area to which you want to write the read address data. For storing the IP address, use the "IF_CONF_v4" structure. For storing the MAC address, use the "IF_CONF_MAC" structure.

Enable reading of the address data using the REQ control parameter. This requires the IO device to be accessible.

The execution status of the read job is displayed via the BUSY, DONE, ERROR output parameters and the STATUS output parameter.

Note

Address the IO device using only the hardware identifier of the station

The station, the IO device and PROFINET interface have their own hardware identifier. Use only the hardware identifier of the station for the "GetStationInfo" instruction.

If a PROFINET is addressed via the LADDR parameter, for example, the address data is not read and the error code 8092 is generate.

To read the address data of an integrated PROFINET interface or a CM/CP in the central configuration, use the "RDREC" instruction.

Parameters

The following table shows the parameters of the "GetStationInfo" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter Request Activates the reading of the information with REQ = "1".
LADDR	Input	HW_DEVICE	I, Q, M, D, L or constant	Hardware identifier of the station of the IO device The number can be taken from the properties of the station in the Network view or from the "System constants" tab of the standard tag table.
DETAIL	Input	HW_SUB-MODULE	I, Q, M, D, L or constant	The DETAIL parameter is not used. Leave the parameter unconnected.
MODE	Input	UNIT	I, Q, M, D, L or constant	Selection of address data to be read: • MODE = 1: Address parameter according to IPv4 (S7-1500 CPUs as of firmware version V1.1) • MODE = 2: MAC address (S7-1500 CPUs as of firmware version V1.5)

Parameter	Declaration	Data type	Memory area	Description
DATA	InOut	VARIANT	D, L	Pointer to the area to which the address data of the IO device is written. Use the "IF_CONF_v4" structure for MODE = 1 and the "IF_CONF_MAC" structure for MODE = 2.
DONE	Output	BOOL	I, Q, M, D, L	The instruction was executed successfully. The address data was transferred to the DATA parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Execution of the instruction complete. • 1: Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy STATUS to a free data area.

You can find more information on the data types in "Overview of the valid data types (Page 1908)".

Parameter DATA

- Use the "IF_CONF_v4" structure at the DATA parameter to store the address parameter according to IPv4.

Byte	Parameter	Data type	Start value	Description
0 … 1	Id	UINT	30	ID of the "IF_CONF_v4" structure.
2 … 3	Length	UNIT	18	Length of data read in BYTE.
4 ... 5	Mode	UNIT	0	Not relevant for the "GetStationInfo" instruction (left at "0").
6 ... 9	InterfaceAddress	ARRAY [1..4] of BYTE	-	IP address of the IO device in the format IP_V4, e.g. for 192.168.3.10: • addr[1] = 192 • addr[2] = 168 • addr[3] = 3 • addr[4] = 10

Byte	Parameter	Data type	Start value	Description
10 … 13	SubnetMask	ARRAY [1..4] of BYTE	-	Subnet mask of the IO device in the format IP_V4, e.g. for 255.255.255.0: • addr[1] = 255 • addr[2] = 255 • addr[3] = 255 • addr[4] = 0
14 ... 17	DefaultRouter	ARRAY [1..4] of BYTE	-	IP address of the router in the format IP_V4, e.g. for 192.168.3.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 3 • addr[4] = 1

- Use the "IF_CONF_MAC" structure at parameter DATA for storing the MAC address.

Byte	Parameter	Data type	Start value	Description
0 … 1	Id	UINT	3	ID of the "IF_CONF_MAC" structure.
2 … 3	Length	UNIT	12	Length of data read in BYTE.
4 ... 5	Mode	UNIT	0	Not relevant for the "GetStationInfo" instruction (left at "0").
6 ... 11	MACAddress	ARRAY [1..6] of BYTE	-	MAC address of the IO device, e.g. for 08-00-06-12-34-56 • Mac[1] = 8 • Mac[2] = 0 • Mac[3] = 6 • Mac[4] = 12 • Mac[5] = 34 • Mac[6] = 56

Parameter STATUS

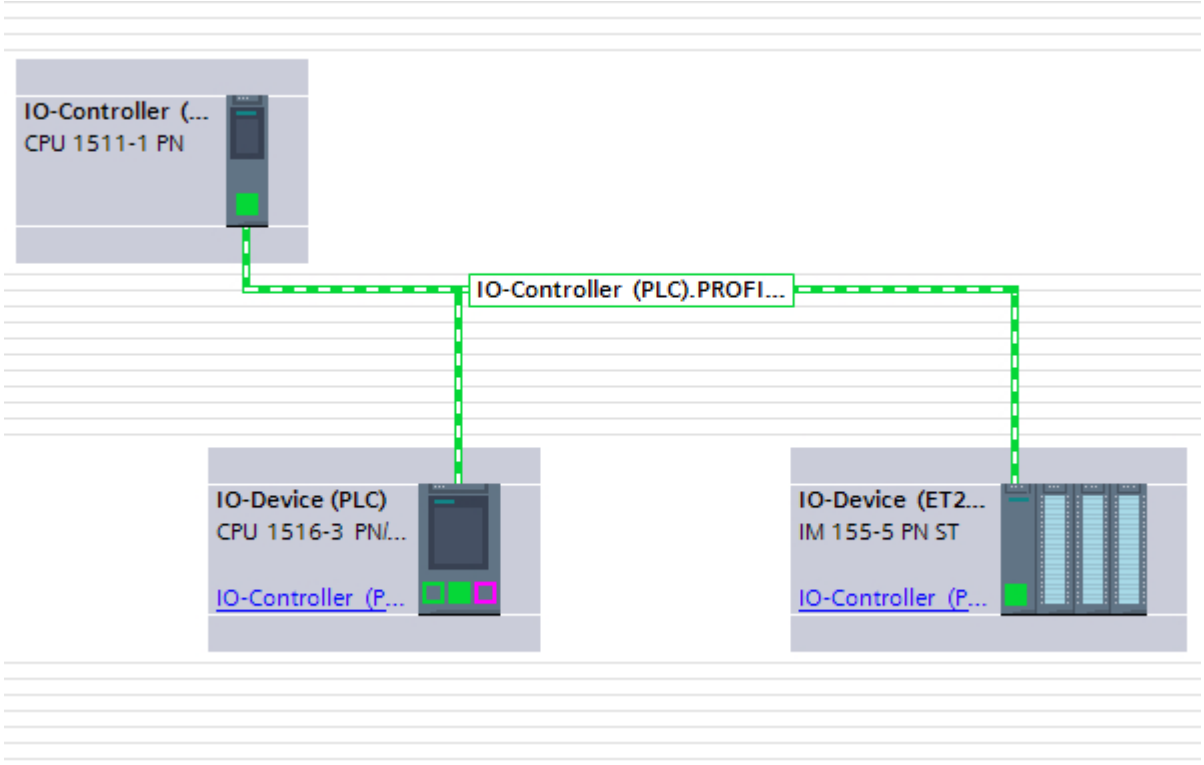
Error code* (W#16#...)	Explanation
0	No error
7000	No job in progress.
7001	First call of the asynchronous instruction "GetStationInfo". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
7002	Additional call of the asynchronous instruction "GetStationInfo". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
8080	Value at the MODE parameter is not supported.
8090	The hardware identifier specified at the LADDR parameter is not configured.
8092	The LADDR parameter does not address a PROFINET IO device.
8093	Invalid data type at the DATA parameter.
80A0	Cannot read requested information.

Error code* (W#16#...)	Explanation
80C0	Addressed IO device is not reachable.
80C3	The maximum number of simultaneous calls of the "GetStationInfo" instruction (10 instances) has been reached.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

Below, you use the „GetStationInfo“ instruction to read the IP address data of an IO device and write the information to a data block. The IP address data includes the IP address, subnet mask and (if used) the address data of the router.

The instruction is executed on the IO controller and reads the IP address information of a lower-level IO device (an ET200MP in this example).



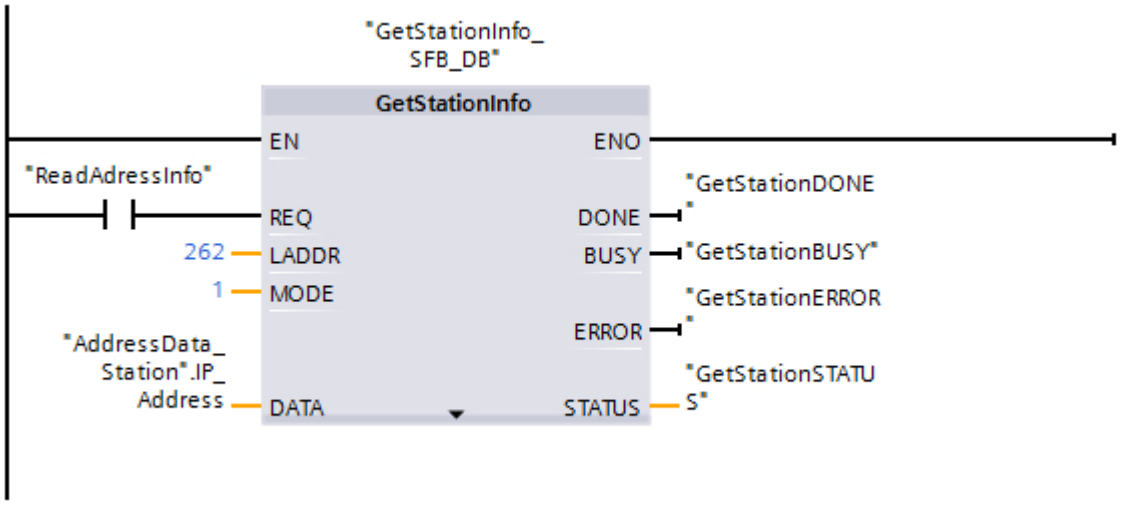
To store the address data, you create a structure with the IF_CONF_v4 data type in a global data block. You can assign any name to the structure ("IP_Address" in this case).

AddressData_Station

	Name	Datentyp	Startwert	Beobachtungswert	Kommentar
1	Static				
2	IP_Address	IF_CONF_v4			
3	Id	UInt	30		
4	Length	UInt	18		
5	Mode	UInt	0		
6	InterfaceAddress	IP_V4			
7	ADDR	Array[1..4] of Byte			IPv4 address
8	ADDR[1]	Byte	0		
9	ADDR[2]	Byte	0		
10	ADDR[3]	Byte	0		
11	ADDR[4]	Byte	0		
12	SubnetMask	IP_V4			
13	ADDR	Array[1..4] of Byte			IPv4 address
14	ADDR[1]	Byte	0		
15	ADDR[2]	Byte	0		
16	ADDR[3]	Byte	0		
17	ADDR[4]	Byte	0		
18	DefaultRouter	IP_V4			
19	ADDR	Array[1..4] of Byte			IPv4 address
20	ADDR[1]	Byte	0		
21	ADDR[2]	Byte	0		
22	ADDR[3]	Byte	0		
23	ADDR[4]	Byte	0		

Then call the "GetStationInfo" instruction:

- You use the IF_CONF_v4 structure at the DATA parameter.
- Enter the hardware identifier of the IO device at the LADDR parameter. The hardware identifier uniquely identifies the product. To determine the hardware identifier of your IO device, open the PLC tag table and the System constants tab. Then, search for the device in the Name column and for the „Hw_Device“ in the Data type column. The associated value is the hardware identifier that you enter at the LADDR parameter.
- Select "1" (read address parameters according to IPv4) for the MODE parameter.



You start reading the address data with REQ= 1 (TRUE). Once the instruction is successfully executed, the IP address data is written to the data block.

DeviceStates: Read module status information in an IO system

Description

You use the instruction "DeviceStates" to query specific status information for all modules in an IO system, which means:

- Either for all IO devices in a PROFINET IO system
- Or for all DP slaves in a DP master system

The Boolean value that is output indicates the modules to which the selected status applies. You can, for example, read which IO devices are currently disabled in a PROFINET IO system.

Information is also displayed as to whether the status information to be read applies to at least one of the IO devices or DP slaves.

The instruction can be called in a cyclic OB as well as in an interrupt OB (e.g. OB82 - diagnostic interrupt).

Parameters

The following table shows the parameters of the "DeviceStates" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IOSYSTEM	I, Q, M, L or constant	Hardware identifier of the PROFINET IO or DP master system (see description below)
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of status information to be read (see description below)
RET_VAL	Return	INT	I, Q, M, D, L	Status of instruction (see description below)
STATE	InOut	VARIANT	I, Q, M, D, L	Buffer for status of the IO devices or the DP slaves (see description below)

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter LADDR

You select the PROFINET IO or DP master system at the LADDR parameter by means of the hardware identifier.

The hardware identifier is available:

- Either in the Network view of the properties of the PROFINET IO or DP master system.
- Or in the PLC tag table in the listed system constants with the data type HW_IOSYSTEM.

Parameter MODE

You use the MODE parameter to read out status information. You can read out one of the following status information items for the entire PROFINET IO or DP master system:

- 1: IO devices/DP slaves are configured
- 2: IO devices/DP slaves are faulty
- 3: IO devices/DP slaves are disabled
- 4: IO devices/DP slaves exist
- 5: IO devices/DP slaves for which a problem has occurred. For example:
  - Maintenance demanded or recommended
  - Not accessible
  - Not available
  - Error occurred

Parameter STATE

With the STATE parameter, the status of the IO devices/DP slaves that was selected with the MODE parameter is output.

If the status selected using MODE applies to an IO device/DP slave, the following bits are set to "1" at the STATE parameter:

- Bit 0 = 1: Group display. The bit n of at least one IO device/DP slave was set to "1".
- Bit n = 1: The status selected with MODE applies to the IO device/DP slave.
  - With a PROFINET IO system, bit n corresponds to the device number of the respective IO device (see properties of the PROFINET interface in the device view and network view)
  - With a PROFIBUS DP system, bit n corresponds to the PROFIBUS address of the DP slave (see properties of the DP slave in the device view and network view)

Use "BOOL" or "Array of BOOL" as data type:

- To only output the bit for group display of the status information, you can use the data type BOOL at the STATE parameter.
- To output the status information for all IO devices/DP slaves, use Array of BOOL with the following length:
  - With PROFINET IO system: 1024 bits
  - With DP master system: 128 bits

Parameter RET_VAL

Error code* (W#16#...)	Description
0	No error
8091	Hardware identifier of the LADDR parameter does not exist. Check (for example, in the system constants) whether the value for LADDR exists in the project.
8092	LADDR does not address a PROFINET IO or DP master system.
8093	Invalid data type at the STATE parameter.
80B1	Instruction "DeviceStates" is not supported by the CPU.
80B2	The selected MODE parameter is not supported by the used CPU for the IO system specified in the LADDR parameter.
8452	The complete status information does not fit in the tag configured in the STATE parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example - Reading out the presence of IO devices in the PROFINET IO master system

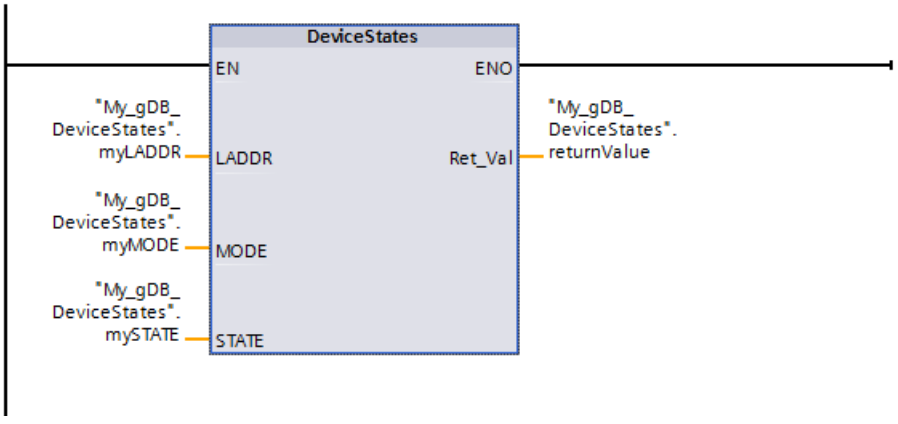
In the following example, you query the existence of IO devices in an IO system. The IO system will consist of two CPUs of the S7-1500 series. The "PLC_14" CPU will contain the program, including the "DeviceStates" instruction. The "PLC_13" CPU will be configured as an IO device.

In the "PLC_14" CPU: Create three tags and a "mySTATE" structure (with the Array of BOOL data type) in a global data block for storing the data.

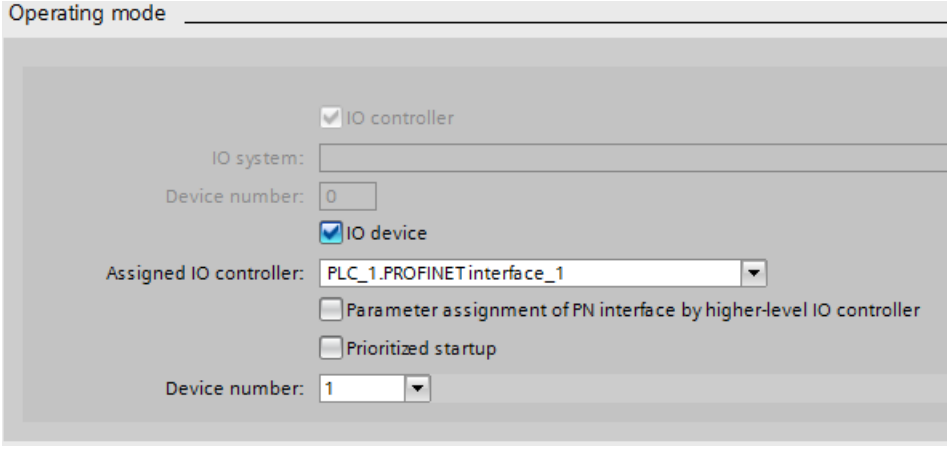
My_gDB_DeviceStates

	Name	Data type	Start value
1	Static		
2	myLADDR	HW_IOSYSTEM	258
3	myMODE	UInt	4
4	returnValue	Int	0
5	mySTATE	Array[0..1023] ...	
6	mySTATE[0]	Bool	false
7	mySTATE[1]	Bool	false
8	mySTATE[2]	Bool	false
9	mySTATE[3]	Bool	false
10	mySTATE[4]	Bool	false
11	mySTATE[5]	Bool	false
12	mySTATE[6]	Bool	false

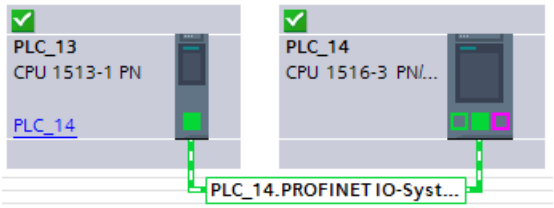
In the "PLC_14" CPU: The instruction is called in a cyclic OB. Interconnect the parameters of the instruction as follows.



In the "PLC_13" CPU: Set up this CPU "PLC_13" as an IO device using the CPU properties. The IO device receives the device number 1.



An IO system is displayed in the Network view.



In the "PLC_14" CPU: The HW identifier of the IO system is made known to the "DeviceStates" instruction through parameter LADDR ("myLADDR"). According to the value "4" of parameter MODE ("myMODE"), the IO system is searched for IO devices.

At parameter STATE ("mySTATE"), the presence is output for the IO devices (based on the value of parameter MODE). Bit 0 serves as a group value and indicates that IO devices are present. Bit 1 indicates that the IO device with the device number 1 is present.

The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

My_gDB_DeviceStates

	Name	Data type	Start value	Monitor value
1	Static			
2	myLADDR	HW_IOSYSTEM	258	16#0102
3	myMODE	UInt	4	4
4	returnValue	Int	0	0
5	mySTATE	Array[0..1023] ...		
6	mySTATE[0]	Bool	false	TRUE
7	mySTATE[1]	Bool	false	TRUE
8	mySTATE[2]	Bool	false	FALSE
9	mySTATE[3]	Bool	false	FALSE
10	mySTATE[4]	Bool	false	FALSE
11	mySTATE[5]	Bool	false	FALSE
12	mySTATE[6]	Bool	false	FALSE

Example - Reading the faulty stations of a PROFINET IO master system

A PROFINET IO system consists of 4 IO devices with the device numbers 1, 2, 3 and 4. The IO device with number 2 is faulty.

The instruction "DeviceStates" is executed for the PROFINET IO system with MODE = 2 (faulty/not faulty).

The following bits are set in the STATE parameter:

- Bit 0 = 1: A fault exists for at least one of the IO devices.
- Bit 1 = 0: IO device with device number 1 is not faulty.
- Bit 2 = 1: IO device with device number 2 is faulty.
- Bit 3 = 0: IO device with device number 3 is not faulty.
- Bit 4 = 0: IO device with device number 4 is not faulty.
- Bit 5 = 0: irrelevant
- Bit 6 = 0: irrelevant
- ...

Example - Reading the faulty stations of a PROFIBUS DP master system

A DP master system consists of 4 DP slaves with the PROFIBUS addresses 3, 4, 5 and 6. The DP slave with address 4 is faulty.

The instruction "DeviceStates" is executed for the DP master system with MODE = 2 (faulty/ not faulty).

The following bits are set in the STATE parameter:

- Bit 0 = 1: A fault exists for at least one of the DP slaves.
- Bit 1 = 0: irrelevant
- Bit 2 = 0: irrelevant
- Bit 3 = 0: DP slave with address 3 is not faulty.
- Bit 4 = 1: DP slave with address 4 is faulty.
- Bit 5 = 0: DP slave with address 5 is not faulty.
- Bit 6 = 0: DP slave with address 6 is not faulty.
- Bit 7 = 0: irrelevant
- Bit 8 = 0: irrelevant
- ...

ModuleStates: Read module status information of a module

Description

You can use the "ModuleStates" instruction to read the status information of the modules of a PROFINET IO device or PROFIBUS DP slave.

The Boolean value that is output indicates the modules to which the selected status applies. You can, for example, read which modules are currently disabled in a PROFINET IO device.

Information is also displayed as to whether the status information to be read applies to at least one of the modules.

The instruction can be called in a cyclic OB as well as in an interrupt OB (e.g. OB82 - diagnostic interrupt).

Parameters

The following table shows the parameters of the "ModuleStates" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_DEVICE	I, Q, M, D, L or constant	Hardware identifier of the station (see description below)
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of module status information to be read (see description below)
RET_VAL	Return	INT	I, Q, M, D, L	Status of instruction (see description below)
STATE	InOut	VARIANT	I, Q, M, D, L	Buffer for the module status (see description below)

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter LADDR

You select the IO device or the DP slave at the LADDR parameter by means of the hardware identifier of the station.

The hardware identifier is available:

- Either in the network view of the properties of the IO device station or DP slave station.
- Or in the PLC tag table for the listed system constants with the data type HW_DEVICE (for an IO device) or with the data type HW_DPSLAVE (for a DP slave).

Parameter MODE

You use the MODE parameter to read out status information. One of the following status information items can be read for the modules:

- 1: Modules are configured
- 2: Modules are faulty
- 3: Modules are disabled
- 4: Modules exist
- 5: There is a problem in the modules. For example:
  - Maintenance demanded or recommended
  - Not accessible
  - Not available
  - Error occurred

Parameter STATE

The STATE parameter outputs the status of the modules selected with the MODE parameter.

If the status selected using MODE applies to a module, the following bits are set to "1":

- Bit 0 = 1: Group display. The bit n of at least one module was set to "1".
- Bit n = 1: The status selected with MODE applies to the module in slot n-1 (example: bit 3 = slot 2).

Use "BOOL" or "Array of BOOL" as data type:

- To only output the bit for group display of the status information, you can use the data type BOOL at the STATE parameter.
- To output the status information for all modules, use Array of BOOL with a length of 128 bits.

Parameter RET_VAL

Error code* (W#16#...)	Description
0	No error
8091	Hardware identifier of the LADDR parameter does not exist. Check (for example, in the system constants) whether the value for LADDR exists in the project.
8092	LADDR does not address an IO device or DP slave.
8093	Invalid data type at the STATE parameter.
80B1	The instruction "ModuleStates" is not supported by the CPU.
80B2	The selected MODE parameter is not supported by the used CPU for the IO device/the DP slave in the LADDR parameter.
8452	The complete status information does not fit in the tag configured in the STATE parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

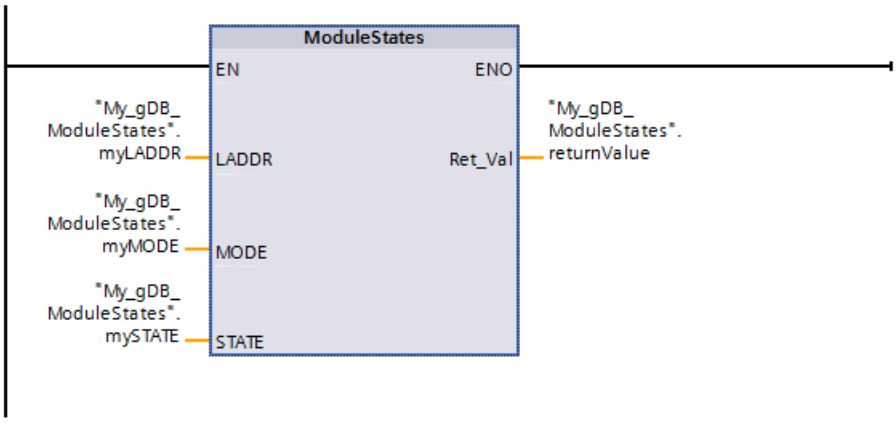
In the following example, you query the existence of modules of a PROFINET IO device. The IO system will consist of two CPUs of the S7-1500 series. The "PLC_14" CPU will contain the program, including the "ModuleStates" instruction. The "PLC_13" CPU will be configured as an IO device.

In the "PLC_14" CPU: Create three tags and a "mySTATE" structure (with the Array of BOOL data type) in a global data block for storing the data.

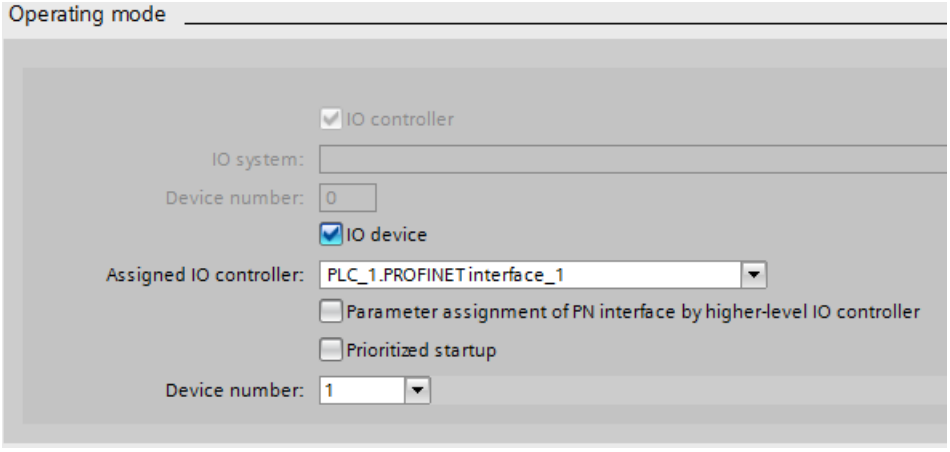
My_gDB_ModuleStates

	Name	Data type	Start value
1	Static		
2	myLADDR	HW_DEVICE	260
3	myMODE	UInt	4
4	returnValue	Int	0
5	mySTATE	Array[0..127] of Bool	
6	mySTATE[0]	Bool	false
7	mySTATE[1]	Bool	false
8	mySTATE[2]	Bool	false
9	mySTATE[3]	Bool	false
10	mySTATE[4]	Bool	false
11	mySTATE[5]	Bool	false
12	mySTATE[6]	Bool	false

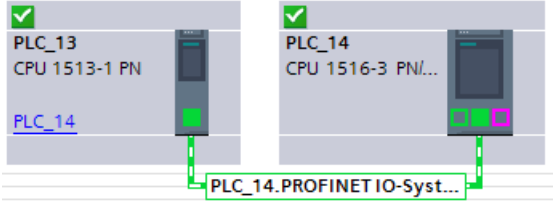
In the "PLC_14" CPU: The instruction is called in a cyclic OB. Interconnect the parameters of the instruction as follows.



In the "PLC_13" CPU: Set up this CPU "PLC_13" as an IO device using the CPU properties.



An IO system is displayed in the Network view.



In the "PLC_14" CPU: A module is located in slot 1 of the IO device.

In the "PLC_14" CPU: The HW identifier of the IO device is made known to the "ModuleStates" instruction through parameter LADDR ("myLADDR"). According to the value "4" of parameter MODE ("myMODE"), the IO device is searched for modules.

At parameter STATE ("mySTATE"), the presence is output for the modules (based on the value of parameter MODE). Bit 0 serves as a group value and indicates that modules are present. Bit 2 indicates that a module is present in slot 1.

The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

My_gDB_ModuleStates

	Name	Data type	Start value	Monitor value
1	Static			
2	myLADDR	HW_DEVICE	260	16#0104
3	myMODE	UInt	4	4
4	returnValue	Int	0	0
5	mySTATE	Array[0..127] of Bool		
6	mySTATE[0]	Bool	false	TRUE
7	mySTATE[1]	Bool	false	FALSE
8	mySTATE[2]	Bool	false	TRUE
9	mySTATE[3]	Bool	false	FALSE
10	mySTATE[4]	Bool	false	FALSE
11	mySTATE[5]	Bool	false	FALSE
12	mySTATE[6]	Bool	false	FALSE

Example

An IO device includes 4 modules in slots 1 to 4. The module in slot 2 is faulty.

The instruction "ModuleStates" is executed for the IO device with MODE = 2 (faulty/not faulty).

The following bits are set in the STATE parameter:

- Bit 0 = 1: A fault exists for at least one module.
- Bit 1 = 0: Slot number 0 (used by IO device)
- Bit 2 = 0: Module in slot number 1 is not faulty.
- Bit 3 = 1: Module in slot number 2 is faulty.
- Bit 4 = 0: Module in slot number 3 is not faulty.
- Bit 5 = 0: Module in slot number 4 is not faulty.
- Bit 6 = 0: irrelevant
- Bit 7 = 0: irrelevant

GEN_DIAG: Generate diagnostics information

Description

The "GEN_DIAG" instruction generates diagnostics information for hardware components from other manufacturers for use in TIA Portal diagnostics. The GSD(GSDL/GSDML) file supplied by the manufacturer must first be installed before the instruction can be used.

The instruction generates all diagnostic events (including those for maintenance).

- Use the LADDR parameter to select the hardware component for which you want to generate a diagnostic event.
- Use the MODE parameter to specify whether the event is to be outgoing or incoming.
- Use the DiagEvent parameter to define the diagnostic event in the DiagnosticDetail structure. The structure is created automatically in the local interface of the block when you define a tag at the DiagEvent parameter.

The diagnostics information is provided synchronously. Diagnostics information transfer and alarm output are asynchronous.

Notice

Failsafe-specific errors messages are not valid

If you define failsafe-specific diagnostics information at the DiagEvent parameter, the instruction will check this information and output the error code 80A1.

Parameters

The following table shows the parameters of the "GEN_DIAG" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_ANY	I, Q, M, D, L or constant	Identification number of the hardware component
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of incoming/outgoing information: • 1: The diagnostic event specified is an incoming event • 2: The diagnostic event specified is an outgoing event • 3: All diagnostic events are outgoing. There are therefore no hardware component faults (green diagnostics symbol). The DiagEvent parameter is not evaluated when MODE = 3.

Parameter	Declaration	Data type	Memory area	Description
DiagEvent	InOut	DiagnosticDetail	L	Specifies the diagnostic event (see "DiagEvent parameter").
RET_VAL	Return	INT	I, Q, M, D, L	Status of instruction / error message (see "RET_VAL parameter")

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

DiagEvent parameter

The structure DiagnosticDetail is a system data type for specifying the diagnostic event. Its structure is as follows:

Parameter		Data type	Description
DiagnosticDetail		Struct	
	ChannelInfo	WORD	Channel properties (0...7)
	ALID	UINT	Local ID of the alarm. The ID uniquely identifies the alarm.
	TextID	UINT	ID of an alarm text in a text list.
	ChannelNumber	UINT	Manufacturer-specific channel number (0x0000 — 0x7FFF)
	Addval_0	DWORD	Placeholder for additional information The value / list of values depends on the connection error.
	TextID2	UINT	Texts for CPU response (mode, OB calls, etc.).
	LADDR	HW_ANY	Identical to the LADDR parameter.
	TextListId	UINT	• 0: No text list • ≠0: ID of the text list
	ChannelDirection	UINT	• 0000: irrelevant • FFF1: Input • FFF2: Output • FFF3: Input/Output
	Addval_1	DWORD	Placeholder for additional information on the channel error (depends on the GSD file). For channel error types, see also: IEC 61158 (PROFINET IO Type 10 and PROFIBUS DP Type 3).

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error
8080	Value in the MODE parameter is not supported.
8090	Identification for hardware component not available at the LADDR parameter.

Error code* (W#16#...)	Explanation
8091	Diagnostics information cannot be generated for the hardware component addressed at the LADDR parameter.
80A1	• Content of the DiagnosticsDetail structure at the DiagEvent parameter is invalid or inconsistent. • Failsafe-specific diagnostics information defined at the DiagEvent parameter (not valid).
80A4	Hardware component addressed cannot be accessed.
80C1	Insufficient resources for parallel execution.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

GET_DIAG: Read diagnostic information

Description

You can use the "GET_DIAG" instruction to read the diagnostic information of a hardware object. The hardware object is selected using the parameter LADDR. With the MODE parameter, you select which diagnostic information is to be read.

Parameters

The following table shows the parameters of the "GET_DIAG" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	Use the MODE parameter to select which diagnostic data is to be output.
LADDR	Input	HW_ANY (WORD)	I, Q, M, L or constant	Hardware identifier of the device.
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction
CNT_DIAG	Output	UINT	I, Q, M, D, L	Reserved (always "0").
DIAG	InOut	VARIANT	I, Q, M, D, L	Diagnostic information corresponds to the selected mode, see table below
DETAIL	InOut	VARIANT	I, Q, M, D, L	Parameter is hidden. Do not use this parameter!

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter MODE

Depending on the value at the MODE parameter, different diagnostic data is output at the DIAG, CNT_DIAG and DETAIL output parameters.

MODE	Description	DIAG	CNT_DIAG
0	Output of all supported diagnostic information for a module as DWORD, where Bit X=1 indicates that mode X is supported.	Bits of the DWORD data type: • Bit 0 = 1: MODE 0 is supported • Bit 1 = 1: MODE 1 is supported • Bit 2 = 1: MODE 2 is supported • Bit 3 = 1: Not relevant	0
1	Output of the diagnostic status of the addressed hardware object.	Structure DIS (see below for description): • MaintenanceState • ComponentStateDetail • OwnState • IOState • OperatingState	0
2	Output of the status of all subordinate modules of the addressed hardware object.	Structure DNN (see below for description): • SubordinateState • SubordinateIOState • DNNmode	0

DIS structure

With parameter MODE = 1, the diagnostic information is output in accordance with the DIS structure. In this case, enter the system data type "DIS" as data type for the tag declaration.

The following table shows the meaning of the individual parameter values.

Parameter	Data type	Value	Description
MaintenanceState	DWORD	Enum	
		0	No maintenance required
		1	The module or device is disabled.
		2	-
		3	-
		4	-
		5	Maintenance required
		6	Maintenance demanded
		7	Error
		8	Status unknown / error in subordinate module
		9	-
		10	Inputs/outputs are not available.

Parameter	Data type	Value	Description
ComponentState-Detail	DWORD	Bit array	Status of the module sub-modules: • Bit 0 to 15: Status message of the module • Bit 16 to 31: Status message of the CPU
		0 to 2 (enum)	Additional information: • Bit 0: No additional information • Bit 1: Transfer not permitted
		3	Bit 3 = 1: At least one channel supports qualifiers for diagnostics
		4	Bit 4 = 1: Maintenance required for at least one channel or one component.
		5	Bit 5 = 1: Maintenance demanded for at least one channel or one component.
		6	Bit 6 = 1: Error in at least one channel or one component.
		7 to 10	Reserved (always = 0)
		11 to 14	• Bit 11 = 1: PNIO - sub-module correct • Bit 12 = 1: PNIO - replacement module • Bit 13 = 1: PNIO - incorrect module • Bit 14 = 1: PNIO - module disconnected
		15	Reserved (always = 0)
		16 to 31	Status information for modules generated by the CPU: • Bit 16 = 1: Module disabled • Bit 17 = 1: CiR operation active • Bit 18 = 1: Input not available • Bit 19 = 1: Output not available bit • 20 = 1: Overflow diagnostics buffer bit • 21 = 1: Diagnostics not available bit • 22 - 31: Reserved (always 0)
OwnState	UINT	Enum	The value of the parameter Ownstate describes the maintenance status of the module.
		0	No fault
		1	The module or device is disabled.
		2	Maintenance required
		3	Maintenance demanded
		4	Error
		5	The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
		6	Inputs/outputs are not available.
		7	-

Parameter	Data type	Value	Description
IOState	WORD	Bit array	I/O status of the module
		0	Bit 0 = 1: No maintenance required
		1	Bit 1 = 1: The module or device is disabled.
		2	Bit 2 = 1: Maintenance required
		3	Bit 3 = 1: Maintenance demanded
		4	Bit 4 = 1: Error
		5	Bit 5 = 1: The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
		6	Inputs/outputs are not available.
		7	Qualifier; bit 7 = 1, if bit 0, 2, or 3 are set
		8 to 15	Reserved (always = 0)
OperatingState	UINT	Enum	
		0	-
		1	In STOP / Firmware update
		2	In STOP / reset memory
		3	In STOP / self start
		4	In STOP
		5	Memory reset
		6	In START
		7	In RUN
		8	-
		9	In HOLD
		10	-
		11	-
		12	Module defective
		13	-
		14	No power
		15	CiR
		16	In STOP / without ODIS
		17	In
		18	
		19	
		20	

DNN structure

With parameter MODE = 2, the diagnostic information details are output in accordance with the DNN structure. In this case, enter the system data type "DNN" as data type for the tag declaration.

The following table shows the meaning of the individual parameter values.

Parameter	Data type	Value	Description
SubordinateState	UINT	Enum	Status of the subordinate module (see parameter OwnState of the DIS structure)
SubordinateIOState	WORD	Bitarray	Status of the inputs and outputs of the subordinate module (see parameter IO State of the DIS structure)
DNNmode	WORD	Bitarray	• Bit 0 = 0: Diagnostics enabled • Bit 0 = 1: Diagnostics disabled • Bit 1 to 15: Reserved

Parameter RET_VAL

Error code* (W#16#...)	Description
0	No error
n	The data area at the DETAIL parameter is too small. Not all details of the diagnostic data can be output.
8080	Value in the MODE parameter is not supported.
8081	Type at the DIAG parameter is not supported with the selected mode (parameter MODE).
8082	Type at the DETAIL parameter is not supported by the selected mode (parameter MODE).
8090	LADDR does not exist
80C1	Insufficient resources for parallel execution.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

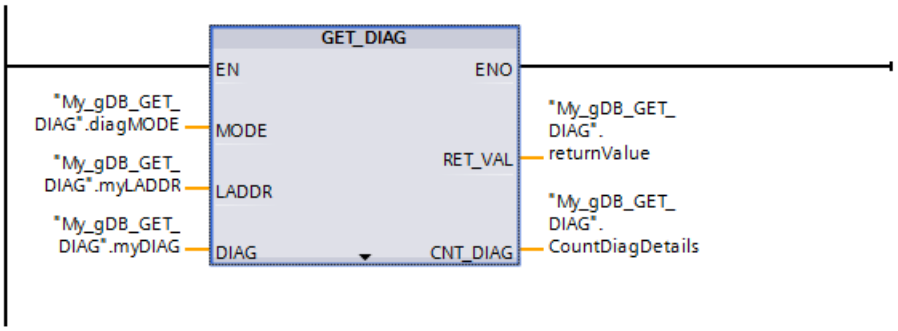
In the following example, you read out the diagnostic information of a CPU.

Create four tags and a "myDIAG" structure (with the DIS data type) in a global data block for storing the data.

My_gDB_GET_DIAG

	Name	Data type	Start value
1	Static		
2	diagMODE	UInt	1
3	myLADDR	HW_ANY	50
4	returnValue	Int	0
5	CountDiagDetails	UInt	0
6	myDIAG	DIS	
7	MaintainanceState	DWord	16#0
8	ComponentStateDetail	DWord	16#0
9	OwnState	UInt	0
10	IOState	Word	16#0
11	OperatingState	UInt	0

Interconnect the parameters of the instruction as follows.



The HW identifier of the CPU is made known to the "GET_DIAG" instruction through parameter LADDR ("myLADDR"). According to the value "1" of parameter MODE ("diagMODE"), the following applies:

- The instruction reads the status of the addressed hardware object (of the CPU).
- At parameter DIAG ("myDIAG"), the diagnostic information is output in a structure (DIS data type).

To understand the diagnostic information, the hex values must be converted to binary code. At parameter DIAG ("myDIAG"), the following is indicated:

- MaintenanceState: According to the value "0", the CPU requires no maintenance.
- ComponentStateDetail: According to the hex value "0000_8000", bit 15 is active.
- OwnState: According to the value "0", no fault has occurred.
- IOState: According to the hex value "0001", there is no maintenance required.
- OperatingState: Outputs "0".

The output parameter RET_VAL ("returnValue") indicates that processing took place without errors. At output parameter CNT_DIAG ("CountDiagDetails"), it is indicated that "0" diagnostic details of parameter DETAIL have been output.

My_gDB_GET_DIAG

	Name	Data type	Start value	Monitor value
1	Static			
2	diagMODE	UInt	1	1
3	myLADDR	HW_ANY	50	16#0032
4	returnValue	Int	0	0
5	CountDiagDetails	UInt	0	0
6	myDIAG	DIS		
7	MaintainanceState	DWord	16#0	16#0000_0000
8	ComponentStateDetail	DWord	16#0	16#0000_8000
9	OwnState	UInt	0	0
10	IOState	Word	16#0	16#0001
11	OperatingState	UInt	0	0

Note: For example, you could individually read out bit 3 (channel diagnostics yes/no) from the ComponentStateDetail tag.

- Address the bit as follows: ComponentStateDetail.%X3.

11.6.3.10 Pulse

CTRL_PWM: Pulse-width modulation

Description

You can use the "CTRL_PWM" instruction to enable and disable a pulse output supported by the CPU using the software.

- Enter the hardware ID of the pulse generator that you want to control with the instruction at the input PWM.
- The pulse output is enabled when the bit in the ENABLE input of the instruction is set.
  - If ENABLE has a value of TRUE the pulse generator will generate a pulse with the properties defined in the device configuration.
  - When the bit in the ENABLE input is reset or the CPU changes to STOP, the pulse output is disabled and no more pulses are generated.

BUSY always has the value of FALSE for S7-1200 since S7-1200 activates the pulse generator if the instruction "CTRL_PWM" is executed.

The ENO enable output is set only when the EN enable input has signal state "1" and no errors have occurred during execution of the instruction.

Note

Use of the force table for PWM and PTO

Digital inputs and outputs that are used for PWM and PTO cannot be forced. Digital inputs and outputs that were assigned via device configuration cannot be controlled by either the force table or the monitoring table.

Requirement

A prerequisite for the correct execution of the instruction is that the specified pulse generator is activated in the hardware configuration.

Open the properties of the module in the device view. Go to "Pulse generators (PTO/PWM)", open the desired PTO/PWM and activate the function the "Activate this pulse generator" under "General".

Go to "Assigning parameters" and set the Pulse options.

Note

Pulse output parameters are assigned exclusively in the device configuration and not using the "CTRL_PWM" instruction. Any change of parameters that is intended to have an effect on the CPU must therefore be made while the CPU is in STOP mode. Changing the duration of the pulse is an exception.

Changing the pulse duration from the user program

The pulse duration setting made in the dialog "Pulse options" can be changed from the user program.

The value set for "Initial pulse duration" is written in the output bytes of the pulse generator. The start address and the end address are displayed in the properties of the pulse generator under "I/O-addresses".

To change the pulse duration, write the desired values in the output word address specified in the device configuration.

Example:

- A value of 500 (decimal) is used for the "Initial pulse duration". The start address of the PTO/PWM is "1000" and the end address is "1001".
- The binary value of "0000000111110100" (=500 decimal) is written in both of the output bytes.
  - Start address (AB1000): 0000_0001 (BIN)
  - End address (AB1001): 1111_0100 (BIN)

Please note that the pulse duration always depends on the parameters set for the Pulse duration format (hundredths, thousandths, ...).

Parameters

The following table shows the parameters of the "CTRL_PWM" instruction:

Parameter	Declaration	Data type	Memory area	Description
PWM	Input	HW_PWM	I, Q, M, D, L or constant	Hardware ID of the pulse generator The hardware ID can be found in the properties of the pulse generator in the Device view. The hardware IDs of the pulse generators are also listed in the system constants.
ENABLE	Input	BOOL	I, Q, M, D, L or constant	The pulse output is enabled when ENABLE = TRUE and disabled when ENABLE = FALSE.
BUSY	Output	BOOL	I, Q, M, D, L	Processing status
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction (see below).

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter STATUS

Error code* (W#16#...)	Description
0	No errors
80A1	Hardware ID of the pulse generator is invalid.

Error code* (W#16#...)	Description
80D0	Pulse generator with the specified hardware ID is not activated. Activate the pulse generator in the CPU properties under "Pulse generators (PTO/PWM)".
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

11.6.3.11 Recipes and data logging

Recipe functions

RecipeExport: Exporting recipes

Description

The "RecipeExport" instruction exports the recipe data from a data block to a CSV file on the memory card of the CPU.

The export is triggered by the "REQ" parameter. The BUSY parameter is set to "1" during the export. During the export, the CSV file is created in the "Recipes" folder in the main directory of the memory card. The name of the data block is used as file name of the created CSV file. If a CSV file with an identical name already exists, it is overwritten during the export.

After the execution of the instruction, BUSY is reset to "0" and the completion of the operation is indicated with "1" at the DONE parameter. If an error occurs during execution, this is signaled by the parameters ERROR and STATUS.

Parameters

The following table shows the parameters of the "RecipeExport" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Control parameter REQUEST: Activates the export on a rising edge.
RECIPE_DB	InOut	VARIANT	D	Pointer to the recipe data block. For information on the structure of the data block, refer to: Structure of a recipe DB (Page 3315)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: The instruction is not executed. • 1: The instruction is executed.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See the "STATUS" parameter table.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter STATUS

Error code* (W#16#...)	Explanation
0	No error occurred
7000	No job processing active
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8090	File name of the CSV file contains invalid characters. The file name of the CSV file is identical to the name of the data block.
8091	The data structure referenced with RECIPE_DB cannot be processed.
8092	Data structure at RECIPE_DB parameter exceeds 5000 bytes.
80B3	Insufficient memory space on the memory card or the internal load memory.
80B4	The memory card is write-protected.
80C0	CSV files is temporarily locked.
80C1	Data block temporarily locked.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

RecipeImport: Importing recipes

Description

The "RecipeImport" instruction imports the recipe data from a CSV file on the memory card into the data block at the RECIPE_DB parameter. The values in the data block are overwritten in the process.

Note the following when importing the CSV file:

- The CSV file must be located in the "Recipes" directory of the memory card.
- The name of the CSV file must match the name of the data block at the RECIPE_DB parameter.

- The first line (header) of the CSV file contains the name of the recipe components (see also: Structure of a recipe DB (Page 3315)). The first line is ignored during import. The names of the recipe components in the CSV file and the data block are not reconciled during import.
- In each case the first value in each line of the CSV file is the index number of the recipe. The individual recipes are imported in the order of the index. For this, the index in the CSV file has to be in ascending order and may contain no gaps (if this is not the case, the error message 80B0 is output at the STATUS parameter.
- The CSV file may not contain more recipe data records than provided for in the data block. The maximum number of data records is indicated by the array limits in the data block.

The import is triggered by the "REQ" parameter. The BUSY parameter is set to "1" during the import. After the execution of the instruction, BUSY is reset to "0" and the completion of the operation is indicated with "1" at the DONE parameter. If an error occurs during execution, this is signaled by the parameters ERROR and STATUS.

Parameters

The following table shows the parameters of the "RecipeImport" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Control parameter REQUEST: Activates the import on a positive edge.
RECIPE_DB	InOut	VARIANT	D	Pointer to the recipe data block. For information on the structure of the data block, refer to: Structure of a recipe DB (Page 3315)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: The instruction is not executed. • 1: The instruction is executed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See the "STATUS" parameter table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter STATUS

Error code* (W#16#...)	Explanation
0	No error occurred
7000	No job processing active
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8090	The file name contains invalid characters.
8092	No matching CSV file found for the import. Possible cause: The name of the CSV file does not match the name of the recipe DB.
80C0	CSV file is temporarily locked.
80C1	Data block is temporarily locked.
80B0	Numbering in the index of the CSV file is not continuous, not ascending or exceeds the maximum number (array limit) in the data block.
80B1	Structure of the recipe data block and the CSV file do not match: The CSV file contains too many fields.
80B2	Structure of the recipe data block and the CSV file do not match: The CSV file contains too few fields.
80D0 +n	Structure of the recipe data block and the CSV file do not match: Data type in field n does not match (n<=46).
80FF	Structure of the recipe data block and the CSV file do not match: Data type in field n does not match (n>46).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Structure of a recipe DB

Introduction

The following sections describes the structure of a recipe DB based on a simple example. The recipe DB consists of five data records, three of which are used. The fourth and fifth data records are left free for later expansions. Each data record contains one recipe, which is made up of the recipe name and eight ingredients.

product-name	water	barley	wheat	hops	yeast	wa-terTmp	mashTmp	mash-Time	QTest
Pils	10	9	3	280	39	40	30	100	0
Lager	10	9	3	150	33	50	30	120	0
BlackB-eer	10	9	3	410	47	60	30	90	1
Not_used	0	0	0	0	0	0	0	0	0
Not_used	0	0	0	0	0	0	0	0	0

Structure of the recipe data block

The recipe data are implemented as follows in a global data block:

- The template for all recipes is the PLC data type "Beer_Recipe" with the recipe components "procutname", "water", etc. with the corresponding data types.
- In a global data block, the PLC data type is used as Array [1.. 5] of "Beer_Recipe". The array limits (1 to 5 in this case) indicate the maximum number of recipes that the DB may contain.
- The values for the recipe components are added as start value in the data block.
- The global DB is interconnected with the instruction via the InOut parameter RECIPE_DB.

Recipe_DB

	Name	Data type	Offset	Start value
1	Static			
2	Products	Array [1 .. 5] of "Beer_Recipe"	...	
3	Products[1]	"Beer_Recipe"	...	
4	Products[2]	"Beer_Recipe"	...	
5	Products[3]	"Beer_Recipe"	...	
6	productname	String[20]	...	'BlackBeer'
7	water	UInt	...	10
8	barley	UInt	...	9
9	wheat	UInt	...	3
10	hops	UInt	...	410
11	yeast	UInt	...	47
12	waterTmp	UInt	...	60
13	mashTmp	UInt	...	30
14	mashTime	UInt	...	90
15	QTest	UInt	...	1
16	Products[4]	"Beer_Recipe"	...	
17	Products[5]	"Beer_Recipe"	...	

Exporting to CSV file

After the execution of the "RecipeExport (Page 3312)" instruction the data of the DB are written to a CSV file with the following structure:

Recipe_DB.csv

```
index,productname,water,barley,wheat,hops,yeast,waterTmp,mashTmp,mashTime,QTest
1,”Pils”,10,9,3,280,39,40,30,100,0
2,”Lager”,10,9,3,150,33,50,30,120,0
3,”BlackBeer”,10,9,3,410,47,60,30,90,1
4,”Not_used”,0,0,0,0,0,0,0,0,0
5,”Not_used”,0,0,0,0,0,0,0,0,0
```

Display in Excel

The CSV file can be opened in Excel to make reading and editing easier. If the commas are not recognized as separators when you open the file, use the Excel import function to output the data in structured form:

	A	B	C	D	E	F	G	H	I	J	K
1	index	product	water	barley	wheat	hops	yeast	waterTmp	mashTmp	mashTime	QTest
2	1	"Pils"	10	9	3	280	39	40	30	100	0
3	2	"Lager"	10	9	3	150	33	50	30	120	0
4	3	"BlackBeer"	10	9	3	410	47	60	30	90	1
5	4	"Not_used"	0	0	0	0	0	0	0	0	0
6	5	"Not_used"	0	0	0	0	0	0	0	0	0

Editing the CSV file

The CSV file can be uploaded to the PC/programming device by means of the Web server and edited. After editing, the modified file can be reloaded in the CPU. You must delete the existing CSV file for this.

You can use the "RecipeImport (Page 3313)" instruction to re-import the modified data from the CSV file into the data block.

Note that the modified data must still be compatible with the data block. In other words:

- No changes may be made to the structure in the table (for example, by adding ingredients in a new column).
- If you add data records to the file, make sure when importing into the data block that the array limits through which you specify the maximum number of data records correspond to at least the number of data records.
- An index is automatically generated during export to the CSV file. If you create additional data records, add consecutive index numbers accordingly.
- The values in the table cells must correspond to the data types used in the data block with respect to format as well as length.
  - Example 1: If data type INT was used in the data block, you can only use integers in the table.
  - Example 2: If data type SINT was used in the data block, you can only use integers with the values -128 to +127 in the table.

  Always keep in mind the permitted data types and data areas as listed in the table below when you make changes in the table.

Data type		Format	Note
Floating-point numbers	LReal	+9.9999999999999E+999	Always written exponentially
	Real	+9.9999999E+99	Always written exponentially
Signed integers	LInt	+9999999999999999999	Value range for signed integers: -9223372036854775808 .. +9223372036854775807
	DInt	+9999999999	Value range of integers: -2147483648 to +2147483647
	Int	+99999	Value range of integers: -32768 to +32767
	SInt	+999	Value range of integers: -128 to +127
Unsigned integers	ULInt	+99999999999999999999	Value range of integers: 0 to +18446744073709551615
	UDInt	+9999999999	Value range of integers: 0 to +4294967295
	UInt	+99999	Value range of integers: 0 to +65535
	USInt	+999	Value range of integers: 0 to +255
Binary numbers	LWord	+99999999999999999999	Value range of integers: 0 to +18446744073709551615
	DWord	+9999999999	Value range of integers: 0 to +4294967295
	Word	+99999	Value range of integers: 0 to + 65535
	Byte	+999	Value range of integers: 0 to +255
	Bool	9	Value range: 0 or 1

Data type		Format	Note
Date and time	LTIME	dddddd:hh:mm:ss.999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	TIME	hhh:mm:ss.999	ISO format with milliseconds
	S5TIME	hhh:mm:ss.999	ISO format with milliseconds
	LDT	YYYY-MM-DD hh:mm:ss.999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	DTL	YYYY-MM-DD hh:mm:ss.999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	DT	YYYY-MM-DD hh:mm:ss.999	ISO format with milliseconds
	DATE	YYYY-MM-DD	ISO format
	LTime_Of_Day	hh:mm:ss.999_999_999	ISO format
	TOD	hh:mm:ss.999	ISO format
Characters	WString	"abcd"	• Character string inside double quotation marks. When a recipe DB is imported, the double quotation marks may be omitted. • Current length • A character string of data type WString consists of elements of the WChar data type. When a recipe DB is exported, the contents of characters of the WChar data type are limited to 16#FF (example: 16#1255 is changed to 16#55).
	String	"abcd"	• Character string inside double quotation marks. When a recipe DB is imported, the double quotation marks may be omitted. • Current length
	WChar	"a"	Individual character inside double quotation marks. When a recipe DB is imported, the double quotation marks may be omitted. When a recipe DB is exported, the contents of characters of the WChar data type are limited to 16#FF (example: 16#1255 is changed to 16#55).
	Char	"a"	Individual character inside double quotation marks. When a recipe DB is imported, the double quotation marks may be omitted.

Data logging

Data logging - Overview

Saving process values

The data logging instructions are used in the user program to save process values to data logs. Data logs can be saved on the Memory Card (MC) or in the internal load memory. The data logs are saved in CSV format (Comma Separated Value).

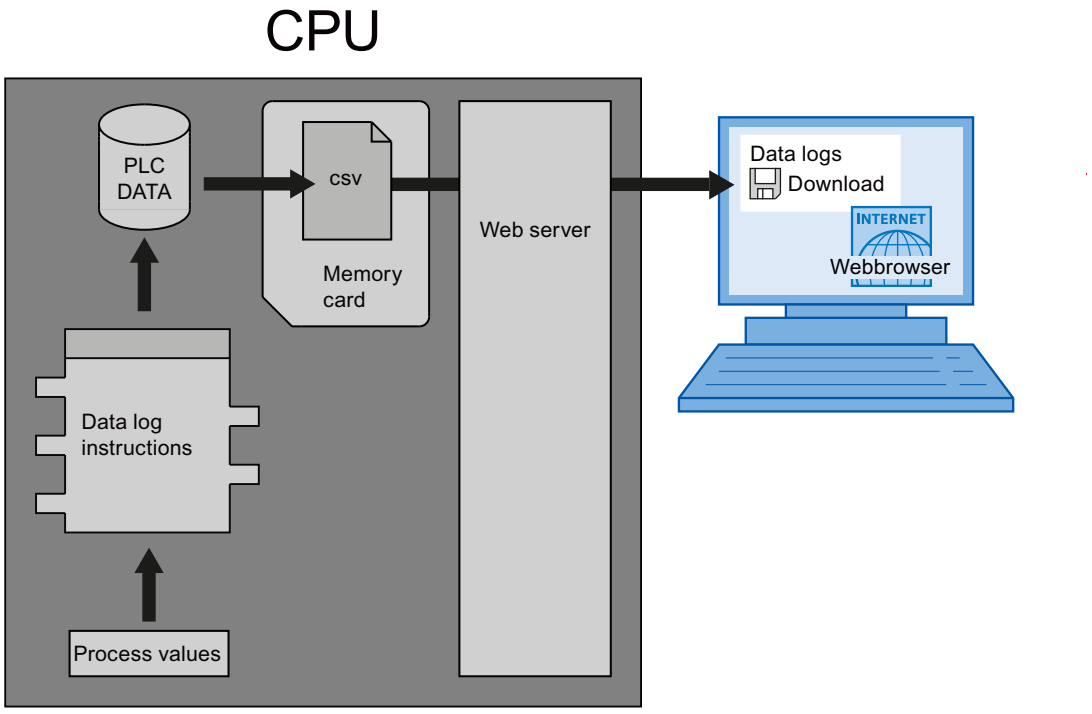
The data types are converted into a character string according to the following rules:

Data type		Format	Note
Floating-point numbers	LReal	+9.9999999999999E+999	Always written exponentially
	Real	+9.9999999E+99	Always written exponentially
Signed integers	LInt	+9999999999999999999	Value range for signed integers: -9223372036854775808 .. +9223372036854775807
	DInt	+9999999999	Value range of integers: -2147483648 to +2147483647
	Int	+99999	Value range of integers: -32768 to +32767
	SInt	+999	Value range of integers: -128 to +127
Unsigned integers	ULInt	+99999999999999999999	Value range of integers: 0 to +18446744073709551615
	UDInt	+9999999999	Value range of integers: 0 to +4294967295
	UInt	+99999	Value range of integers: 0 to + 65535
	USInt	+999	Value range of integers: 0 to +255
Binary numbers	LWord	+99999999999999999999	Value range of integers: +00000000000000000000 to +18446744073709551615
	DWord	+9999999999	Value range of integers: +0000000000 to +4294967295
	Word	+99999	Value range of integers: +00000 to + 65535
	Byte	+999	Value range of integers: +000 to +255
	Bool	9	Value range: 0 or 1
Date and time	LTIME	dddddd:hh:mm:ss. 999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	TIME	hhh:mm:ss.999	ISO format with milliseconds
	S5TIME	hhh:mm:ss.999	ISO format with milliseconds
	LDT	YYYY-MM-DD hh:mm:ss. 999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	DTL	YYYY-MM-DD hh:mm:ss. 999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	DT	YYYY-MM-DD hh:mm:ss.999	ISO format with milliseconds
	DATE	YYYY-MM-DD	ISO format
	LTime_Of_Day	hh:mm:ss.999_999_999	ISO format
	TOD	hh:mm:ss.999	ISO format

Data type		Format	Note
Characters	WString	"abcd"	• Character string inside double quotation marks • The current length is padded with spaces up to the maximum length. • A character string of data type WString consists of elements of the WChar data type. When a data log is saved, the contents of characters of the WChar data type are limited to 16#FF (example: 16#1255 is changed to 16#55).
	String	"abcd"	• Character string inside double quotation marks • The current length is padded with spaces up to the maximum length.
	WChar	"a"	Individual character inside double quotation marks. When a data log is saved, the contents of characters of the WChar data type are limited to 16#FF (example: 16#1255 is changed to 16#55).
	Char	"a"	Individual character inside double quotation marks

The data logging instructions are used in your program to create or open a data log, to write an entry and to close the data log file.

You decide by the creation of the data buffer which program values are stored in the data log during the creation of the data buffer. The data buffer is used as a memory for new data log entries. New values have to be written to the buffer before "DataLogWrite (Page 3334)" is called. During execution of the "DataLogWrite (Page 3334)" instruction, data is written from the buffer to a data log record.



Data log files can be copied to the PC as follows:

- If the PROFINET interface is connected to the PC, you use a Web browser to access the data logs via the Web server. The CPU can be in RUN or STOP mode for this. If the CPU is in "RUN" mode, the program continues running while the Web server is transferring data.
- If there is a memory card in the CPU, you can remove this card and insert it into a standard slot for SD (Secure Digital) cards or MMC (MultiMediaCard) cards on a PC or programming device. Use File Manager to transfer the data log files from the memory card to the PC. The CPU goes to "STOP" when you remove the memory card.

Properties of the data log

Writing of the data records of a data log is carried out in accordance with the principle of a ring buffer. New data records are added until the maximum number of data records is reached (RECORD parameter). The next data record then overwrites the "oldest" data record of the data log.

If you want to prevent data records from being overwritten, use the "DataLogNewFile (Page 3340)" instruction to create a new data log file based on the current data log. New data records are then written into the new data log.

Creating data logs

With the "DataLogCreate (Page 3323)" instruction, you can create a new data log file in the ""\DataLogs" directory of the load memory.

- The name assigned at the NAME parameter is the designation for the data log and is also used the file name for the CSV file. The file is stored in the directory "DataLogs".
- The block parameter DATA specifies the data buffer for the new data log object and the columns and data types in the data log. The columns and data types of a data record in the data log are generated by the elements in the structure declaration or the array declaration of this data buffer. Each element of a structure or of an array corresponds to a column in a line in the data log.
- You can use the HEADER block parameter to assign a header text in the header to each column.
- The "DataLogCreate (Page 3323)" instruction returns an ID. This ID is used by the other data logging instructions as a reference for the created data log.

Opening data logs

You use the instruction "DataLogOpen" (S7-1200 and S7-1500) to open an existing data log on the memory card. A data log must be open before you can write new data records to it.

Data log opens automatically when you execute the instructions "DataLogCreate (Page 3323)" and "DataLogNewFile (Page 3340)".

Up to 10 data logs can be open simultaneously. You can select the data log to be opened using the ID or name of the data log.

- If you specify the ID and the name of the data log in the ID and NAME parameters, respectively, the data log will be identified based on the ID. The data log name will not be compared.
- If you select the data log using the NAME parameter and no ID is specified, the ID will be displayed in the ID parameter when the data log is opened.
- If you select the data log using the ID parameter and no name is specified, the name will be displayed in the NAME parameter when the data log is opened.

You use the MODE parameter to specify whether the data records of the data log are to be deleted upon opening.

Writing to data logs

A data log must be open ("DataLogOpen (Page 3329)" instruction) before a data record can be written to a data log. The "DataLogWrite (Page 3334)" instruction writes a data record to the data log.

Closing data logs

You use the "DataLogClose (Page 3337)" instruction to close an open data log. You select the data log using the ID parameter.

The data log is closed automatically when the CPU switches to STOP and upon restart.

Deleting data logs

You use the instruction "DataLogDelete (Page 3338)" (S7-1500) to delete a data log file from the memory card. The data log and the data records it contains can only be deleted if the log was created with the instruction "DataLogCreate (Page 3323)".

Select the data log to be deleted using the NAME and ID parameters. The ID parameter is evaluated first. If there is a data log with the relevant ID, the NAME parameter will not be evaluated. If the value "0" is used at the ID parameter, a value with the data type STRING must be used at the NAME parameter.

Clearing data logs

You use the instruction "DataLogClear (Page 3333)" (S7-1500) to delete all the data records in an existing data log. The instruction does not delete the optional header of the CSV file (see the description of the HEADER parameter of the instruction "DataLogCreate (Page 3323)").

You use the ID parameter to select the data log whose data records are to be deleted. Before you can delete data records, the data log must be open.

New file for data logs

You use the instruction "DataLogNewFile (Page 3340)" (S7-1200) or "DataLogTypedNewFile (Page 3341)" (S7-1500) to create a new data log with the same properties as an existing data log. This allows you to retain the contents of an existing data log.

When called, the instruction creates a new data log on the memory card or in the internal load memory with the name defined at the NAME parameter. You use the ID parameter to specify the ID of the old data log whose properties you want to apply to the new data log. The ID of the new data log is then output via the ID parameter.

You specify the file size of the new data log with the RECORDS parameter of the instruction.

You can run a consistency check for "DataLogTypedNewFile (Page 3341)" (S7-1500).

DataLogCreate: Create data log

Description

With the "DataLogCreate" instruction you create a data log.

The data log is saved on the memory card or in the internal load memory in the directory "\DataLogs". The amount of data that can be stored in a data log is dependent on the available space on the memory card or the storage space in the internal load memory of the CPU.

You specify the maximum number of data records that can be stored in a data log in the RECORDS parameter. If the specified maximum number of data records in the data log is reached, the oldest data record will be overwritten. To avoid overwriting of existing data records, use the "DataLogNewFile (Page 3340)" instruction. The instruction can be used to create a new data log with the same structure when the number specified at the RECORDS parameter is reached (return value 1 at the STATUS parameter of the "DataLogWrite (Page 3334)" instruction). The data records are then saved in the new data log.

You can specify the name for the data log in the NAME parameter. The data log is created in CSV (Comma Separated Value) format. With the HEADER parameter, you can create an (optional) header for the data log.

Once the data log is created, it is opened automatically. This means that data can be written.

Parameters

The following table shows the parameters of the "DataLogCreate" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction The data log is created with a rising edge at the parameter REQ.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Maximum number of data records in the data log If the instruction "DataLogWrite (Page 3334)" writes more records than specified in this parameter, the oldest record is overwritten.

Parameter	Declaration	Data type	Memory area	Description
FORMAT	Input	UInt	I, Q, M, L, D or constant	Data format: • 0: Internal (not supported) • 1: CSV (Comma separated values)
TIMESTAMP	Input	UInt	I, Q, M, L, D or constant	Time stamp: • 0: No time stamp • 1: Date and time The extra columns in the header are automatically added if time-stamping is activated.
NAME	Input	VARIANT	L, D	Name of the data log The specified name is also used as a file name for the csv file. The restrictions for Windows file names apply when assigning the name. The following characters must not be used: "\", "/", ":", "*", "?", "<", ">", "|", "space"
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log (only output) The ID of the data log is required for further data logging instructions to address the created data log.
HEADER	InOut	VARIANT	L, D	Header of the data log (optional) The parameter is hidden after the instruction is added. The header is written as the first line in the CSV file.
DATA	InOut	VARIANT	L, D	Pointer to the data structure that is to be written as data record when executing the instruction "DataLogWrite (Page 3334)".
DONE	Output	BOOL	I, Q, M, L, D	Status parameter: • 0: Processing not yet complete. • 1: Processing of instruction finished successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Status parameter: • 0: Processing of the instruction has not started, completed, or canceled. • 1: Processing of the instruction is in progress.
ERROR	Output	BOOL	I, Q, M, L, D	Status parameter: • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Detailed status information: Detailed error and status information is output at the parameter STATUS. The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

HEADER parameter

The HEADER parameter is a VARIANT pointer to a data block that defines a header for the CSV file (header). The header is always the first line in the CSV file representation.

- When creating a header, remember that the individual columns must be separated by a comma (S7-1200) or a semicolon (S7-1500).
- A STRING, Array of BYTE, or Array of CHAR data type can be used for the individual column names. A longer character string is possible with the Array [...] of type data type than with the STRING data type. When the STRING data type is used, the length is limited to 254 bytes.

If no header is to be created, do not specify a value in the HEADER parameter.

Parameter DATA

The DATA parameter is a VARIANT pointer to a structure or an array in a data block. An element of a structure or an array corresponds to a column in the data log with a specific data type.

Note the following when creating the data block:

- The number of columns must correspond to the number of columns defined in the HEADER parameter.
- Each element of the structure or the array corresponds to a column entry in the CSV file. Therefore, no nesting structures (STRUCT in STRUCT) may be used when using the data type STRUCT.
- The data structure can contain up to 256 elements. If there are more than 256 elements, error code 8C52 is output at the STATUS parameter.
- The tags of the data block can be set as retentive or non-retentive tags. However, the retentivity setting must be the same for all tags of the data block.

Parameter STATUS (S7-1200)

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.
8090	Invalid file name (see description of the NAME parameter).
8093	Data log already exists.
8097	File length exceeds the file system limit.

Error code* (W#16#...)	Description
80A2	Write error returned from file system.
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
80C1	Too many data logs are open.
80C3	Resources are not sufficient. Possible causes: • Multiple calling of instructions with different parameters. • More than 10 data logs open simultaneously. At least one data log must be closed in order to continue because the instruction "DataLogCreate" automatically opens the the log that is being created.
8253	The RECORDS parameter has an invalid value.
8453	Invalid format selection.
8553	Invalid time stamp.
8B51	Impermissible data type at parameter HEADER or length exceeds the maximum size.
8C20	String with length specification other than 254 used.
8C24	Invalid assignment at parameter DATA (for example, bit memory used as memory area).
8C51	Impermissible data type at parameter DATA / data structure cannot be used.
8C52	Structure at the parameter DATA contains more than 256 elements.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Parameter STATUS (S7-1500)

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
8070	Complete internal instance memory is assigned.
8090	Invalid file name (see description of the NAME parameter).
8091	"NAME" parameter is not a string.
8093	Data log already exists.
8097	File length exceeds the file system limit.
80A2	Write error returned from file system.
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
80C1	Too many data logs are open.
80C3	Resources are not sufficient. Possible causes: • Multiple calling of instructions with different parameters. • More than 10 data logs open simultaneously. At least one data log must be closed in order to continue because the instruction "DataLogCreate" automatically opens the the log that is being created.
8253	The RECORDS parameter has an invalid value.
8353	Invalid format selection.

Error code* (W#16#...)	Description
8453	Invalid time stamp.
8B24	Invalid assignment at parameter HEADER (for example, bit memory used as memory area).
8B51	Impermissible data type at parameter HEADER or length exceeds the maximum size.
8C24	Invalid assignment at parameter DATA (for example, bit memory used as memory area).
8C51	Impermissible data type at parameter DATA / data structure cannot be used.
8C52	Structure at the parameter DATA contains more than 256 elements.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

In the following example, a simple data log is created with a time stamp and three process values.

Tags in the global data block

The values for the input parameters of the data log are stored in the global data block "DataLogDB":

- DataLogName (String): The tag contains the name of the data log which is also used as the file name for the CSV file.
- DataLogID (DInt): The ID of the data log is written in this tag when the instruction is called.
  - The ID is automatically assigned by the instruction.
  - Use the tag "DataLogID" in other DataLog instructions in order to address this data log.
- MyHeader (String): This tag contains the header of the DataLog, i.e. the column headings for the process values. A semicolon is used as a delimiter for the columns when using a S7-1500.
- MyData (Struct): This tag contains the three process values that are written in the data log. Each time a record is written (instruction "DataLogWrite (Page 3334)") the current values are written in a new record.

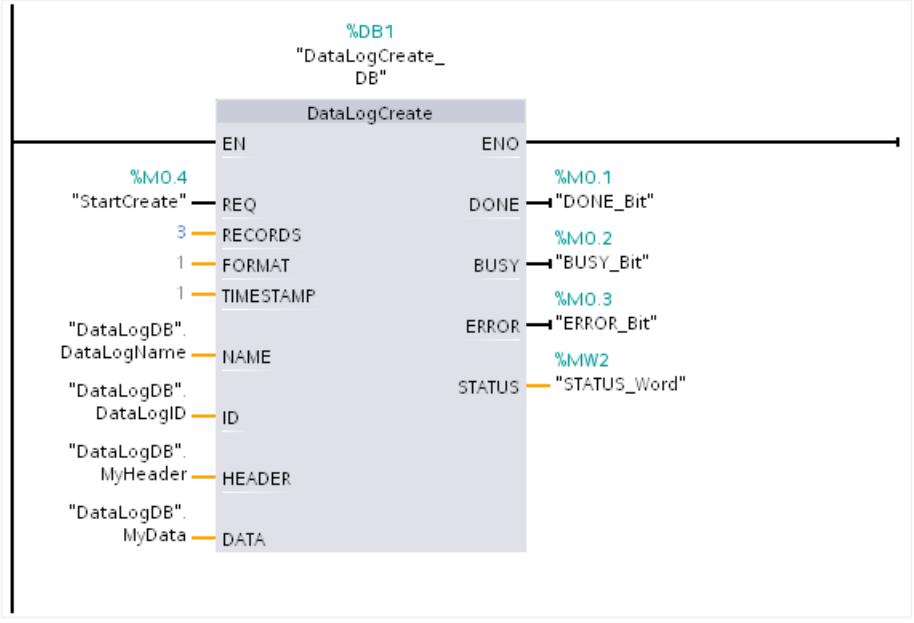
DataLogDB

	Name	Data type	Start value	Retain
1	Static			
2	DataLogName	String	'MyDataLog'	
3	DataLogID	DInt	0	
4	MyHeader	String	'Value1;Value2;Value3'	
5	MyData	Struct		
6	ProcessValue1	Int	2	
7	ProcessValue2	Int	3	
8	ProcessValue3	Int	4	

Calling the "DataLogCreate" instruction

The instruction is called with the following input parameters:

- REQ (BOOL): With REQ = "1" the data log is created.
- RECORD (3): A maximum of three data records can be written in the data log. Thereafter, the oldest data record is overwritten.
- FORMAT (1): The data log is created as a CSV file.
- TIMESTAMP (1): Activated. Two additional columns are automatically created (date and time) for the data log. The current time stamp is written in the data record each time "DataLogWrite (Page 3334)" is executed.
- NAME (VARIANT): Pointer to the tag "DataLogName" in the data block "DataLogDB".
- ID (VARIANT): Pointer to the tag "DataLogID" in the data block "DataLogDB".
- HEADER (VARIANT): Pointer to the tag "MyHeader" in the data block "DataLogDB".
- DATA (VARIANT): Pointer to the tag "MyData" in the data block "DataLogDB".



Readout of the data log via the web server

You can readout the created data log with the web server:

- Activate the web server in the properties of the CPU. The authorization "...to read files" must be activated for the web server in order to readout the data log.
- You can access the web server via the Internet browser by entering the IP address of the CPU as a URL.
- Under "Filebrowser" the directory "\DataLogs" was automatically created which contains the data log.
- The DataLog only contains one entry "//END" if the instruction "DataLogWrite (Page 3334)" has still not been executed. The first data record is written after the initial execution of "DataLogWrite (Page 3334)".

	A	B	C	D	E	F
1	SeqNo	Date	Time	ProcessValue1	ProcessValue2	ProcessValue3
2	1	01.07.2013	12:33:42.917	2	3	4
3						
4						
5						

See also

Data logging - Overview (Page 3318)

DataLogOpen: Open data log

DataLogOpen: Open data log

Description

You use the "DataLogOpen" instruction to open an existing data log on the memory card. A data log must be open before you can write new data records to it.

Data log opens automatically when you execute the instructions "DataLogCreate (Page 3323)" and "DataLogNewFile (Page 3340)".

Up to 10 data logs can be open simultaneously. You can select the data log to be opened using the ID or name of the data log.

- If you specify the ID and the name of the data log in the ID and NAME parameters, respectively, the data log will be identified based on the ID. The data log name will not be compared.
- If you select the data log using the NAME parameter and no ID is specified, the ID will be displayed in the ID parameter when the data log is opened.
- If you select the data log using the ID parameter and no name is specified, the name will be displayed in the NAME parameter when the data log is opened.

You use the MODE parameter to specify whether the data records of the data log are to be deleted upon opening.

Parameters

The following table shows the parameters of the "DataLogOpen" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execution of the instruction upon a rising edge.
MODE	Input	UInt	I, Q, M, L, D or constant	Mode for opening the data log: • MODE= "0" Data records of the data log are retained • MODE= "1" Data records of the data log are deleted, but the header is retained
NAME	Input	VARIANT	L, D	(File) name of the data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log.
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter STATUS

Error code* (W#16#...)	Description
0	No errors.
2	Warning: Data log file has already been opened by this application.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.
8090	There are inconsistencies between the data log definition and existing data log data.
8091	A data type other than String was used at the NAME parameter.
8092	Data log does not exist.
80B4	The memory card is write-protected.

Error code* (W#16#...)	Description
80C0	The data log file is locked.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Data logging - Overview (Page 3318)

DataLogOpen: Open data log

Description

You use the "DataLogOpen" instruction to open an existing data log on the memory card. A data log must be open to write new data records to it.

Data log opens automatically when you execute the instructions "DataLogCreate (Page 3323)" and "DataLogNewFile (Page 3340)".

Up to 10 data logs can be open simultaneously. You can select the data log to be opened using the ID or name of the data log.

- If you specify the ID and the name of the data log in the ID and NAME parameters, respectively, the data log will be identified based on the ID. The data log name will not be compared.
- If you select the data log using the NAME parameter and no ID is specified, the ID will be displayed in the ID parameter when the data log is opened.
- If you select the data log using the ID parameter and no name is specified, the name will be displayed in the NAME parameter when the data log is opened.

You use the MODE parameter to specify whether the data records of the data log are to be deleted upon opening.

The parameter DATA enables a consistency check between the data log to be opened and the definition of the data log of the instruction "DataLogCreate (Page 3323)". The consistency check can only be executed if the data log was created with the instruction "DataLogCreate (Page 3323)":

- If you are using the same pointer at the DATA parameter as at the DATA parameter of the "DataLogCreate (Page 3323)" instruction, a check is performed to determine whether the data types match. If this is not the case, the error code W#16#8090 is output at the STATUS parameter.
- If the data log to be opened was not created with "DataLogCreate (Page 3323)", a consistency check is not possible. In this case, enter the value "NULL" in the DATA parameter.

Parameters

The following table shows the parameters of the "DataLogOpen" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
MODE	Input	UInt	I, Q, M, L, D or constant	Mode for opening the data log: • MODE= "0" Data records of the data log are retained • MODE= "1" Data records of the data log are deleted, but the header is retained
NAME	Input	VARIANT	L, D	(File) name of the data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log.
DATA	InOut	VARIANT	L, D	With consistency check: Pointer to the data area at the DATA parameter of the "DataLogCreate (Page 3323)" instruction.
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter STATUS

Error code* (W#16#...)	Description
0	No errors.
2	Warning: Data log file has already been opened by this application.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.

Error code* (W#16#...)	Description
8090	Data types inconsistent. The data log at parameter ID uses different data types than specified at parameter DATA.
8091	A data type other than String was used at the NAME parameter.
8092	Data log does not exist.
80B4	The memory card is write-protected.
80C1	Too many files open.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

See also

Data logging - Overview (Page 3318)

DataLogClear: Empty data log

Description

The "DataLogClear" instruction deletes all data records in an existing data log. The instruction does not delete the optional header of the CSV file (see the description of the HEADER parameter of the instruction "DataLogCreate (Page 3323)").

You use the ID parameter to select the data log whose data records are to be deleted.

Requirement

Before you can delete data records, the data log must be open (see "DataLogOpen (Page 3329) instruction").

Parameters

The following table shows the parameters of the "DataLogClear" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
ID	InOut	DWORD	I, Q, M, D, L	Object ID of the data log
DONE	Output	BOOL	I, Q, M, D, L	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Execution of the instruction not yet complete.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output at the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter STATUS

Error code* (W#16#...)	Explanation
0000	No error.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8080	The data log file selected with the ID parameter cannot be processed with the "DataLogClear" instruction.
8092	Data log does not exist.
80A2	Write error signaled back by the file system.
80B0	Data log is not open.
80B4	The memory card is write-protected.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

See also

Data logging - Overview (Page 3318)

DataLogWrite: Write data log

Description

The instruction "DataLogWrite" is used to write a data record to an existing data log. The ID parameter is used to select the data log to which the data record is to be written. To create a new data record, the data log must be open. The instruction creates a new data record in the format that was specified in the DATA parameter when the data log was created.

Before calling the "DataLogWrite" instruction, transfer the data to the tag that you interconnected at the DATA parameter of the "DataLogCreate" instruction. When the "DataLogWrite" instruction is executed, the transferred data is copied to the data log.

Notice

Data log data loss when the power supply to the CPU is interrupted

If the power supply is interrupted during execution of the "DataLogWrite" instruction, the data record to be transferred is lost.

Parameters

The following table shows the parameters of the "DataLogWrite" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter STATUS (S7-1200)

Error code* (W#16#...)	Description
0	No errors
0001	Last possible data record created at the end of the file. Creating another data record will overwrite an older data record.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".

Error code* (W#16#...)	Description
8070	Complete internal instance memory is assigned.
8092	Data log does not exist.
80A2	Write error signaled back by the file system.
80B0	Data log is not open.
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
80C0	Data log is locked.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Parameter STATUS (S7-1500)

Error code* (W#16#...)	Description
0	No errors
0001	Last possible data record created at the end of the file. Creating another data record will overwrite an older data record.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
8070	Complete internal instance memory is assigned.
8092	Data log does not exist.
80A2	Write error signaled back by the file system.
80B0	Data log is not open.
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
80C0	Data log is locked.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

See also

Data logging - Overview (Page 3318)

DataLogClose: Close data log

Description

You use the "DataLogClose" instruction to close an open data log. You select the data log using the ID parameter.

Note

Closing data logs automatically

The data log is closed automatically when the CPU goes to STOP or if there is a restart.

Parameters

The following table shows the parameters of the "DataLogClose" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execute function on a rising edge.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

STATUS (S7-1200) parameter

Error code* (W#16#...)	Description
0	No errors
1	Data log is not open
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0

Error code* (W#16#...)	Description
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8092	Data log does not exist.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

STATUS (S7-1500) parameter

Error code* (W#16#...)	Description
0	No errors
1	Data log is not open
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
8070	Complete internal instance memory is assigned.
8092	Data log does not exist.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

DataLogDelete: Delete data log

Description

You use the "DataLogDelete" instruction to delete a data log file from the memory card. The data log and the data records it contains can only be deleted if it was created with the "DataLogCreate" or "DataLogNewFile" instruction.

Parameters

The following table shows the parameters of the "DataLogDelete" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
NAME	Input	VARIANT	L, D	File name of the data log
DELFILE	Input	BOOL	I, Q, M, D, L	• 0: Data log is retained. • 1: Data log is deleted.
ID	InOut	DWORD	I, Q, M, D, L	Object ID of the data log

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Deletion of the data log is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output at the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameters NAME and ID

Select the data log to be deleted using the NAME and ID parameters. The ID parameter is evaluated first. If there is a data log with the relevant ID, the NAME parameter will not be evaluated. If the value "0" is used at the ID parameter, a value with the data type STRING must be used at the NAME parameter.

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8091	A data type other than STRING is being used at the NAME parameter.
8092	Data log does not exist.
80A2	Write error signaled back by the file system.
80B4	The memory card is write-protected.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

See also

DataLogCreate: Create data log (Page 3323)

DataLogNewFile: Data log in new file

DataLogNewFile: Data log in new file

Description

You use the "DataLogNewFile" instruction to create a new data log with the same properties as an existing data log. This allows the contents of an existing data log to be retained.

When called, the instruction creates a new data log on the memory card or in the internal load memory with the name defined at the NAME parameter. You use the ID parameter to specify the ID of the old data log whose properties you want to apply to the new data log. The ID of the new data log is then output via the ID parameter.

You specify the file size of the new data log with the RECORDS parameter of the instruction.

Once the new data log is created, it is opened automatically. This means that data can be written.

Parameters

The following table shows the parameters of the "DataLogNewFile" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execution of the instruction upon a rising edge.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Number of data records in the data log.
NAME	Input	VARIANT	L, D	File name of the new data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log • In: ID of the existing data log • Out: ID of the new data log
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter STATUS

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.
8090	Invalid file name.
8091	Data type at parameter NAME is not STRING.
8092	Source data log does not exist.
8093	New data log already exists.
8097	File length exceeds the file system limit.
80A0	Data types inconsistent. The data log at parameter ID uses different data types than specified at parameter DATA.
80A2	Write error signaled back by the file system.
80B3	Load memory not sufficient.
80B4	The memory card is write-protected.
80C1	Too many files open.
8253	The RECORDS parameter has an invalid value.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

DataLogNewFile: Data log in new file

Description

You use the "DataLogNewFile" instruction to create a new data log with the same properties as an existing data log. This allows the contents of an existing data log to be retained.

When called, the instruction creates a new data log on the memory card or in the internal load memory with the name defined at the NAME parameter. You use the ID parameter to specify the ID of the old data log whose properties you want to apply to the new data log. The ID of the new data log is then output via the ID parameter.

You specify the file size of the new data log with the RECORDS parameter of the instruction.

The parameter DATA enables a consistency check between the new data log to be created and the definition of the data log of the instruction "DataLogCreate (Page 3323)". The

consistency check can only be executed if the data log was created with the instruction "DataLogCreate (Page 3323)":

- If you are using the same pointer at the DATA parameter as at the DATA parameter of the "DataLogCreate (Page 3323)" instruction, a check is performed to determine whether the data types match. If the data types do not match, the error code W#16#80A0 is output at the STATUS parameter.
- If the data log to be opened was not created with "DataLogCreate (Page 3323)", a consistency check is not possible. In this case, enter the value "NULL" in the DATA parameter.

Once the new data log is created, it is opened automatically. This means that data can be written.

Parameters

The following table shows the parameters of the "DataLogNewFile" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Number of data records in the data log.
NAME	Input	VARIANT	L, D	File name of the new data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log • In: ID of the existing data log • Out: ID of the new data log
DATA	InOut	VARIANT	L, D	Data type for consistency check
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter STATUS

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.
8090	Invalid file name
8091	Data type at parameter NAME is not STRING.
8092	Source data log does not exist.
8093	New data log already exists.
8097	File length exceeds the file system limit.
80A0	Data types inconsistent. The data log at parameter ID uses different data types than specified at parameter DATA.
80A2	Write error signaled back by the file system.
80B3	Load memory not sufficient.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
80C1	Too many files open.
8253	The RECORDS parameter has an invalid value.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example program for working with data logs

Introduction

The following example program shows the essential functions of the data log instructions. For detailed information about individual instructions, open the corresponding help description using the corresponding links.

General notes on using data logs

- The data log created is automatically opened when the "DataLogCreate" and "DataLogNew" instructions are executed.
- Data logs are automatically closed after the CPU switches from RUN to STOP or after a restart of the CPU.
- For the "DataLogWrite" instruction to be executed, the data log must be open.
- For S7-1200 CPUs, a maximum of eight data logs can be open simultaneously. For S7-1500 CPUs, a maximum of ten data logs can be open simultaneously.

Example program

The contents of the data logs are defined in a data block (DB) in this example. The DB is used to create a data log (DataLogCreate (Page 3323)) and to provide process values for writing a data record (DataLogWrite (Page 3334)).

The three entries of the "MyData" structure are used as process values: MyCount, MyTemerature and MyPressure. In the data block, these three values are stored temporarily and then transferred with the "DataLogWrite (Page 3334)" instruction to a data log as a data record.

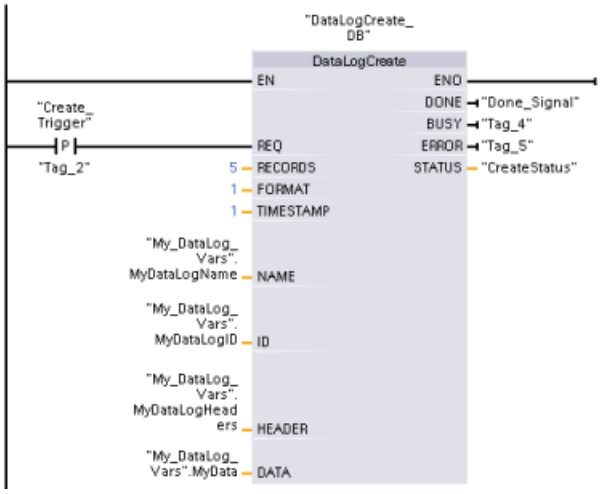
My_Datalog_Vars

	Name	Datentyp	Startwert
1	Static		
2	MyNewDataLogName	String	'MyNEWDataLog'
3	MyDataLogName	String	'MyDataLog'
4	MyDataLogID	DWord	0
5	MyDataLogHeaders	String	'Count,Temperature,Pressure'
6	MyData	Struct	
7	MyCount	Int	0
8	MyTemperature	Real	0.0
9	MyPressure	Real	0.0

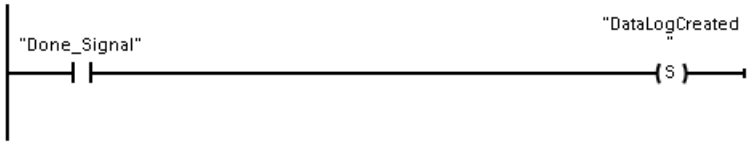
A data record is composed later from the following six entries:

1. The data record number (assigned automatically)
2. The date (assigned automatically when "1" is used for DataLogCreate at the TIMESTAMP parameter).
3. The time of day (assigned automatically when "1" is used for DataLogCreate at the TIMESTAMP parameter).
4. The current value of "MyCount" from the "MyData" structure.
5. The current value of "MyTemperature" from the "MyData" structure.
6. The current value of "MyPressure" from the "MyData" structure.

Network 1: A rising edge at REQ starts the creation of the data log with the DataLogCreate (Page 3323) instruction.

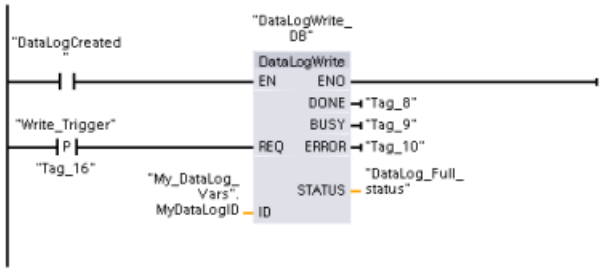


Network 2: Acquire the DONE output of DataLogCreate (Page 3323), because it is only valid for one cycle.

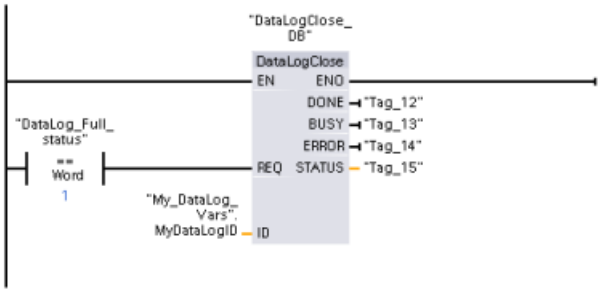


Network 3: A rising edge triggers the point in time at which new process values are stored in the MyData structure. This step is used to store the desired process values temporarily in the data block.

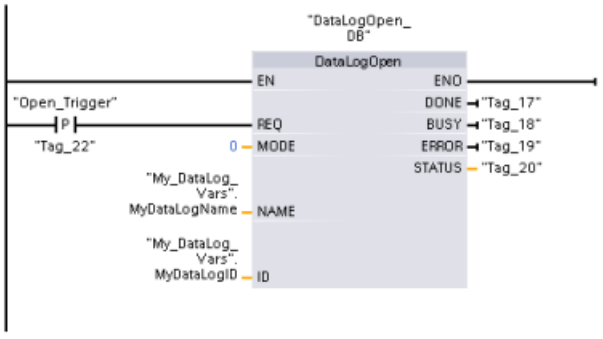
Network 4: Once the execution of DataLogCreate (Page 3323) is complete, (DONE parameter =1, see Network 1), the EN input of DataLogWrite (Page 3334) is set. The reason for this is that a generation process extends over several cycles and must be completed in order for a write operation to be performed. Writing is triggered for a data record by a rising edge at the REQ input.



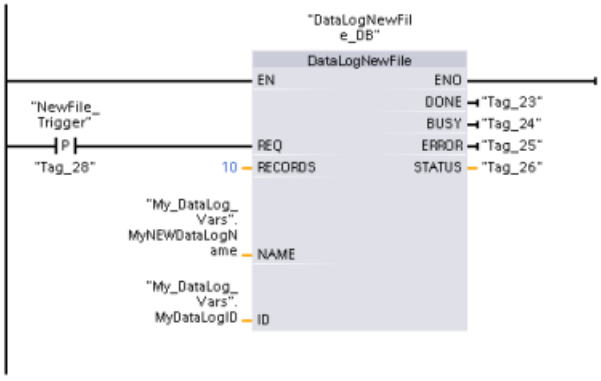
Network 5: Close the data log after the last data record has been written. The data log was created for 5 data records (see Network 1). That means, after 5 data records, 0001 is output on the DataLogWrite (Page 3334) instruction at the STATUS parameter (last possible data record created at the end of the file. Creating another data record overwrites an older data record). In this case, the REQ input is set and the DataLogClose (Page 3337) instruction is executed. When the data log is closed, no more data records can be written.



Network 6: The data log must be opened again with the DataLogOpen (Page 3331) instruction in order to write a data record again at a later time. If another data record is now written with DataLogWrite (Page 3334), the oldest data record is overwritten.



Network 7: If you do not want to overwrite the older data records, you can use the DataLogNewFile (Page 3341) instruction to create a new data log with the same structure. To do this, enter the ID of the existing data log whose structure you want to copy at the ID parameter of the instruction. Once the DataLogNewFile (Page 3341) instruction has been executed, a new and unique ID value for the new data log is assigned.

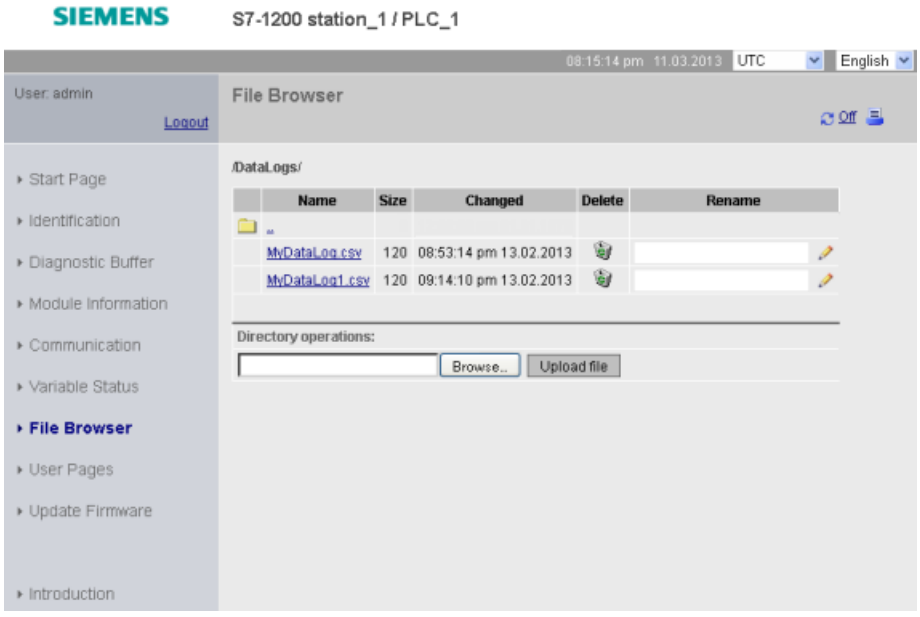


Note that the call for DataLogNewFile (Page 3341) also extends over several cycles. Therefore, similar to the DataLogCreate (Page 3323) instruction, you should also query the DONE bit to prevent premature execution of DataLogWrite (Page 3334) (see Networks 1, 2 and 4).

Result

Opening of written data via the web server

The data logs created with the example program can be displayed via the web server. To do this, open the web server using an Internet browser and open the "\DataLogs" directory.



Note

The "Delete" and "Rename" options are only available if you are logged on with editing rights. You assign the rights in the hardware configuration in the web server properties of the CPU.

Contents of the CSV files

- "5" is set as the maximum number of data records when creating the data log with the DataLogCreate instruction. If this number is not exceeded, all written data records are included in the data log.

	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	1	9/30/2010	20:26:56	1	9.86E+01	3.52E+01
3	2	9/30/2010	20:28:43	2	1.00E+02	3.73E+01
4	3	9/30/2010	20:29:03	3	9.99E+01	3.68E+01
5	4	9/30/2010	20:29:21	4	9.95E+01	3.64E+01
6	5	9/30/2010	20:30:19	5	9.92E+01	3.74E+01
7						

- If another data record is added to this, the oldest data record (Record 1) is overwritten.

	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	6	9/30/2010	20:32:03	6	9.86E+01	3.58E+01
3	2	9/30/2010	20:28:43	2	1.00E+02	3.73E+01
4	3	9/30/2010	20:29:03	3	9.99E+01	3.68E+01
5	4	9/30/2010	20:29:21	4	9.95E+01	3.64E+01
6	5	9/30/2010	20:30:19	5	9.92E+01	3.74E+01
7						

See also

DataLogClear: Empty data log (Page 3333)

DataLogDelete: Delete data log (Page 3338)

11.6.3.12 Data block functions

CREATE_DB: Create data block

Description

The instruction "CREATE_DB" is used to create a new data block in the load and/or work memory.

The instruction "CREATE_DB" does not change the checksum of the user program.

Number of the data block

The data block created is assigned a number from the range defined at the LOW_LIMIT (low limit) and UP_LIMIT (high limit) parameters. "CREATE_DB" assigns the smallest possible number from the specified range to the DB. You cannot assign the numbers of the DBs already contained in the user program.

To create a DB with a specific number, enter the same number for the high and low limit of the range to be specified. If a DB with the same number already exists in the work memory and/

or load memory, or if the DB exists as a copied version, the instruction will be terminated and an error message will be generated at the RET_VAL parameter.

Start values of the data block

Using the SRCBLK parameter, you can define start values for the DB to be created. The SRCBLK parameter is a pointer to a DB or a DB area from which you apply the start values. The DB addressed at the SRCBLK parameter must have been generated with standard access ("Optimized block access" attribute disabled).

- If the area specified at the SRCBLK parameter is larger than the DB generated, the values up to the length of the DB generated will be applied as start values.
- If the area specified at the SRCBLK parameter is smaller than the DB generated, the remaining values will be filled with "0".

To ensure data consistency, you must not change this data area while "CREATE_DB" is being executed (which means as long as the BUSY parameter has the value TRUE).

Functional description

The "CREATE_DB" instruction works asynchronously, which means its execution extends over multiple calls. You start the job by calling "CREATE_DB" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193)

Parameters

The following table shows the parameters of the "CREATE_DB" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Request to create the data block
LOW_LIMIT	Input	UINT	I, Q, M, D, L or constant	Low limit of the area ### used by "CREATE_DB" to assign a number to your DB 60000)
UP_LIMIT	Input	UINT	I, Q, M, D, L or constant	High limit of the area ### used by "CREATE_DB" to assign a number to your DB (largest possible DB number: 60999)
COUNT	Input	UDINT	I, Q, M, D, L or constant	The count value specifies the number of bytes which you want to reserve for the DB generated. The number of bytes must be an even number. The maximum length is 65534 bytes.

<table>
<tr><th>Parameter</th><th>Declaration</th><th>Data type</th><th>Memory area</th><th>Description</th></tr>
<tr><td rowspan="9">ATTRIB</td><td rowspan="9">Input</td><td rowspan="9">BYTE</td><td rowspan="9">I, Q, M, D, L or constant</td><td>You use the first 4 bits of the byte at parameter ATTRIB to define the properties of the data block *:</td></tr>
<tr><td>• Bit 0 = 0: Attribute "Only store in load memory" is not set.
• Bit 0 = 1: Attribute "Only store in load memory" is set. With this setting, the DB takes up no space in the work memory and is not included in the program. The DB cannot be accessed with bit commands. When bit 0 = 1, the selection for bit 2 is irrelevant.</td></tr>
<tr><td>• Bit 1 = 0: Attribute "Data block write-protected in the device" is not set.
• Bit 1 = 1: Attribute "Data block write-protected in the device" is set.</td></tr>
<tr><td>• Bit 2 = 0: DB is retentive (only for DBs generated in the load memory). The DB is regarded as retentive if at least one value has been set as retentive.
• Bit 2 = 1: DB is not retentive</td></tr>
<tr><td>• Bit 3= 0: Creation of the DB either in the load memory or in the work memory (selection using bit 0, see above)
• Bit 3= 1: Creation of the DB both in the load memory and in the work memory (bit 0 irrelevant)
To ensure compatibility with STEP 7 V5.x, bits 0 and 3 must be used in combination:</td></tr>
<tr><td><table><tr><th>Bit 0</th><th>Bit 3</th><th>DB generation</th></tr><tr><td>0</td><td>0</td><td>In work memory only</td></tr><tr><td>1</td><td>0</td><td>In load memory only</td></tr><tr><td>0</td><td>1</td><td>Work and load memory</td></tr><tr><td>1</td><td>1</td><td>Work and load memory</td></tr></table></td></tr>
<tr><td>• Bit 4 = 0 - No start values specified (input values at the SRCBLK parameter will be ignored).
• Bit 4 = 1 - Specify start values (values correspond to the DB addressed by the SRCBLK parameter).</td></tr>
<tr><td>SRCBLK</td><td>Input</td><td>VARIANT</td><td>D</td><td>Pointer to the data block whose values will be used to initialize the data block to be generated.</td></tr>
<tr><td>RET_VAL</td><td>Return</td><td>INT</td><td>I, Q, M, D, L</td><td>Error information</td></tr>
<tr><td>BUSY</td><td>Output</td><td>BOOL</td><td>I, Q, M, D, L</td><td>BUSY= 1: The process is not yet complete.</td></tr>
<tr><td>DB_NUM</td><td>Output</td><td>DB_DYN (UINT)</td><td>I, Q, M, D, L</td><td>Number of the DB created.</td></tr>
<tr><td colspan="5">* The properties selected here correspond to the attributes in the properties of a data block.</td></tr>
</table>

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
0081	The target range is greater than the source range. The complete source range is written to the target range. The remaining bytes of the target range are filled with 0.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source range is larger than the target range. The complete target range is written. The remaining bytes of the source range are ignored.
8092	The "Create data block" function is currently unavailable because • The "Compress user memory" function is currently active. • The maximum number of blocks on your CPU has already been reached.
8093	No data block or a data block that is not in the work memory is specified for the SRCBLK parameter.
8094	A not yet supported attribute was specified for the ATTRIBparameter.
80A1	DB number error: • The number is "0" • Low limit > high limit
80A2	DB length error: • The length is "0" • The length is an odd number • The length is greater than permitted by the CPU
80A3	The data block at the SRCBLK parameter was not created with standard access.
80B1	There is no DB number free.
80B2	Not enough work memory.
80B4	The memory card is write-protected.
80BB	Not enough load memory.
80C3	The maximum number of simultaneously active "CREATE_DB" instructions has already been reached.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

READ_DBL: Read from data block in the load memory

Description

With the instruction you copy a DB or an area of a DB in load memory (Micro Memory Card) to the data area of a destination DB. The destination DB must be relevant for execution; that is, it must not be created with the attribute UNLINKED. The content of the load memory is not changed during the copy process.

To ensure data consistency, you must not change the target range while "READ_DBL" is being executed (i.e., as long as the BUSY parameter has the value TRUE).

The following restrictions apply to the SRCBLK and DSTBLK parameters (source and destination blocks):

- You must be able to divide the length of the VARIANT pointer by eight.
- For a VARIANT pointer of type STRING, the length must be equal to 1.
- The source and destination block must have been created with the same block access, i.e. both must use either the access type "Optimized" or "Standard".

Note

""READ_DBL" is processed asynchronously. Therefore, it is not suitable for frequent (or cyclical) reading of tags in the load memory.

Once started, a job is always completed. If the maximum number of simultaneously active "READ_DBL" instructions is reached and you call "READ_DBL" once again at this time in a priority class having higher priority, error code W#16#80C3 will be returned. Consequently it does not make sense to restart the high-priority job right away.

Functional description

The "READ_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "READ_DBL" with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193)

Parameters

The following table shows the parameters of the "READ_DBL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
SRCBLK	Input	VARIANT	D	Pointer to data block in the load memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
DSTBLK	Output	VARIANT	D	Pointer to the data block in the work memory that is to be written to

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
0081	The target range is greater than the source range. The source range is written completely to the target range; the remaining bytes of the target range will not be changed.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8x51	Note: This error code is only for S7-1200 CPUs. Data type error in the data block.
8081	The source range is larger than the target range. The complete target range is written. The remaining bytes of the source range are ignored.
8082	Destination DB type different from source DB type (optimized/standard access).
8093	Note: This error code is only for S7-1500 CPUs. No data block or a data block that is not in the work memory is specified for the DSTBLK parameter.
80B1	Note: This error code is only for S7-1500 CPUs. At parameter DSTBLK, only data blocks that are located in the load memory are permitted.
8xB1	Note: This error code is only for S7-1200 CPUs. No data block is specified for the SRCBLK parameter, or the data block specified there is not a load memory object.
80B4	Memory card is write-protected or DSTBLK points to a DB with F-attribute (that must not be written)
8xC0	Note: This error code is only for S7-1200 CPUs. The destination DB is currently being processed by another instruction or a communication function.
80C3	The maximum number of simultaneously active "READ_DBL" instructions has already been reached.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

See also

GET_ERR_ID: Get error ID locally (Page 2922)

WRIT_DBL: Write to data block in the load memory

Description

The instruction "WRIT_DBL" is used to transfer the contents of a DB or a DB area from the work memory to a DB or a DB area in the load memory (Micro Memory Card). The source DB must be relevant for execution, which means it must not be created with the Only store in load memory attribute.

To ensure data consistency, it is not permitted to change the source range while "WRIT_DBL" is being executed (i.e., as long as the BUSY parameter has the value TRUE).

The following restrictions apply to the SRCBLK and DSTBLK parameters (source and destination blocks):

- For a VARIANT pointer of type BOOL, the length must be divisible by 8.
- For a VARIANT pointer of type STRING, the length must be equal to 1.
- The source and destination blocks must have been created with the same block access, i.e. both must use "Optimized block access" or optimized access must be disabled for both.

The "WRIT_DBL" instruction does not change the checksum of the user program if you write a DB that was created using an instruction. However, when a loaded DB is written, the first entry in this DB changes the checksum of the user program.

Note

"WRIT_DBL" is not suitable for frequent (or cyclical) writing of tags in the load memory. This is because the Micro Memory Card technology limits the number of write accesses that can be made to a Micro Memory Card.

Functional description

The "WRIT_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "WRIT_DBL" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

Parameters

The following table shows the parameters of the instruction "WRIT_DBL":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Write request
SRCBLK	Input	VARIANT	D	Pointer to the DB in the work memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.
DSTBLK	Output	VARIANT	D	Pointer to the data block in the load memory that is to be written to

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
0081	The target range is greater than the source range. The source range is written completely to the target range; the remaining bytes of the target range will not be changed.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8x51	Note: This error code is only for S7-1200 CPUs. Data type error in the data block.
8081	The source range is larger than the target range. The complete target range is written. The remaining bytes of the source range are ignored.
8082	Destination DB type different from source DB type (optimized/non-optimized access).
8093	Note: This error code is only for S7-1500 CPUs. No data block or a data block that is not in the work memory is specified for the SRCBLK parameter.
80B1	Note: This error code is only for S7-1500 CPUs. At parameter DSTBLK, only data blocks that are located in the load memory are permitted.
8xB1	Note: This error code is only for S7-1200 CPUs. For parameter DSTBLK, no data block is specified or the data block specified there is not in the load memory.
80B4	• The memory card is write-protected. • DB with F-attribute must not be read.
80BB	Insufficient load memory available.
8xC0	Note: This error code is only for S7-1200 CPUs. The destination DB is currently being processed by another instruction or a communication instruction.
80C3	Note: This error code is only for S7-1500 CPUs. The maximum number of simultaneously active "WRIT_DBL" instructions has already been reached.
General error codes	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

ATTR_DB: Read data block attribute

Description

You use the instruction "ATTR_DB" to obtain information about a data block (DB) located in the work memory of the CPU. The instruction determines the attributes set at the ATTRIB parameter for the DB selected.

The length cannot be read for data blocks with optimized access, the DB_LENGH parameter contains the length "0" for DBs with optimized access.

Data blocks for Motion Control cannot be read with the "ATTR_DB" instruction. The error code 80B2 is output for this.

Parameters

The following table shows the parameters of the "ATTR_DB" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request for block attributes
DB_NUM-BER	Input	DB_ANY (UINT)	I, Q, M, D, L or constant	Number of the DB to be tested
RET_VAL	Output	INT	I, Q, M, D, L	Error information
DB_LENGTH	Output	UDINT	I, Q, M, D, L	Number of data bytes which the selected DB contains.
ATTRIB	Output	BYTE	I, Q, M, D, L	DB properties:
				• Bit 0*= 0: Attribute "Only store in load memory" is not set. • Bit 0*= 1: Attribute "Only store in load memory" is set.
				• Bit 1 = 0: Attribute "Data block write-protected in the device" is not set. • Bit 1 = 1: Attribute "Data block write-protected in the device" is set.
				If bit 0 = 1, then bit 2 is irrelevant and gets the value 1. • Bit 2 = 0: Retentive - The DB is regarded as retentive if at least one value has been set as retentive. • Bit 2 = 1: Not retentive - The complete DB is not retentive.
				• Bit 3*= 0: The DB is either in the load memory (bit 0 = 1) or in the work memory (bit 0 = 0). • Bit 3*= 1: The DB is generated in both the load and the work memory
* The relationship between bit 0 and bit 3 is explained in the parameters of the instruction "CREATE_DB: Create data block (Page 3348)".				

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
80A1	Error in input parameter DB_NUMBER: the actual parameter selected • Is "0" • Is greater than the maximum permitted DB number for the CPU used.

Error code* (W#16#...)	Explanation
80B1	The DB with the specified number does not exist on the CPU.
80B2	Data blocks of Motion Control technology objects cannot be read with the "ATTR_DB" instruction.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

DELETE_DB: Delete data block

Description

You use the instruction "DELETE_DB" to delete a data block (DB) that was created from the user program by calling the instruction "CREATE_DB (Page 3348)".

If the data block was not created with "CREATE_DB", the error code W#16#80B5 is output at the RET_VAL parameter.

The selected data block is not deleted immediately, but rather at the cycle control point after execution of the cycle OB.

Functional description

The "DELETE_DB" instruction works asynchronously, that is, its execution extends over multiple calls. You start the interrupt transfer by calling the instruction with REQ = 1.

Output parameter BUSY and bytes 2 and 3 of output parameter RET_VAL show the status of the job.

The deletion of the data block is complete when output parameter BUSY has the value FALSE.

Parameters

The following table shows the parameters of the "DELETE_DB" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ =1: Request to delete the DB with the number in parameter DB_NUMBER
DB_NUMBER	Input	UINT	I, Q, M, D, L or constant	Number of the DB to be deleted
RET_VAL	Output	INT	I, Q, M, D, L	Error information (see "RET_VAL parameter")
BUSY	Output	BOOL	I, Q, M, D, L	BUSY= 1: The process is not yet complete.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
80A1	Error in input parameter DB_NUMBER: • The value at the parameter is "0". • The value at the parameter is greater than the maximum permitted DB number for the CPU used.
80B1	The DB with the specified number does not exist on the CPU.
80B4	The DB cannot be deleted because the memory card of the CPU is write-protected.
80B5	The DB was not created using "CREATE_DB".
80BB	Not enough load memory.
80C3	The "Delete a DB" function cannot be executed at this time due to a temporary resource bottleneck.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

11.6.3.13 Addressing

Instructions for address conversion

Description

There are different options for addressing a module (IO address, hardware identifier, slot).

You can convert the address data with the following instructions:

- GEO2LOG: Determine hardware identifier from slot (Page 3359)
- LOG2GEO: Determine slot from hardware identifier (Page 3361)
- LOG2MOD: Determine the hardware identifier from addressing of STEP 7 V5.5 SPx (Page 3361)
- IO2MOD: Determine hardware identifier from an IO address (Page 3362)
- RD_ADDR: Determine IO addresses from the hardware identifier (Page 3364)

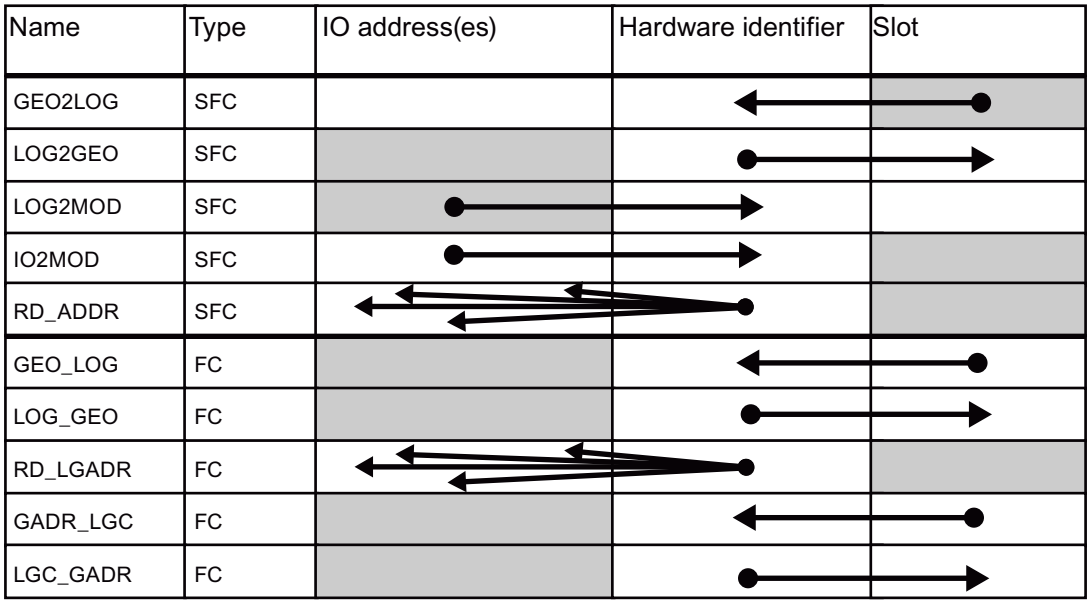
The following instructions are supported in addition for migrated projects:

- GEO_LOG: Determine hardware identifier from slot (Page 3366)
- LOG_GEO: Determine slot from hardware identifier (Page 3368)
- RD_LGADR: Determine IO addresses from the hardware identifier (Page 3369)

- GADR_LGC: Determine hardware identifier from slot and offset in the user data address area (Page 3370)
- LGC_GADR: Determine slot from hardware identifier (Page 3372)

Type of address conversion

The figure below shows which instruction runs which address conversion.



GEO2LOG: Determine hardware identifier from slot

Description

You use the "GEO2LOG" instruction to determine hardware identifier based on slot information that you define using the system data type GEOADDR.

Depending on the type of hardware you define at the parameter HWTYPE the following information is evaluated from the other parameters GEOADDR:

- When HWTYPE = 1 (IO system):
  - Only IOSYSTEM is evaluated. The other parameters of GEOADDR are not taken into consideration.
  - The hardware identifier of the IO system is output.
- When HWTYPE = 2 (IO device):
  - IOSYSTEM and STATION are evaluated. The other parameters of GEOADDR are not taken into consideration.
  - The hardware identifier of the IO device is output.

- With HWTYPE = 4 (module):
  - IOSYSTEM, STATION and SLOT are evaluated. The SUBSLOT parameter of GEOADDR is not taken into consideration.
  - The hardware identifier of the module is output.
- With HWTYPE = 5 (submodule):
  - All parameters of GEOADDR are evaluated.
  - The hardware identifier of the submodule is output.

The AREA parameter of the GEOADDR system data type is not evaluated.

Parameters

The following table shows the parameters of the "GEO2LOG" instruction:

Parameter	Declaration	Data type	Memory area	Description
GEOADDR	Input	VARIANT	D, L	Pointer to the structure of the GEOADDR system data type. The system data type contains the slot information from which the hardware ID is determined. See also: System data type GEOADDR (Page 3365)
RET_VAL	Return	INT	I, Q, M, D, L	Output of error information.
LADDR	Output	HW_ANY	I, Q, M, D, L	Hardware identifier of the assembly or the module. The number is automatically assigned and is stored in the properties in the hardware configuration.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8091	Invalid value for in GEOADDR for HWTYPE.
8094	Invalid value for in GEOADDR for IOSYSTEM.
8095	Invalid value for in GEOADDR for STATION.
8096	Invalid value for in GEOADDR for SLOT.
8097	Invalid value for in GEOADDR for SUBSLOT.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

See also

Instructions for address conversion (Page 3358)

LOG2GEO: Determine slot from hardware identifier

Description

You use the "LOG2GEO" instruction to determine the module slot belonging to a hardware identifier.

Parameters

The following table shows the parameters of the "LOG2GEO" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_ANY	I, Q, M, D, L or constant	Hardware identifier of the module whose slot you want to find. The hardware ID is assigned automatically and is stored in the properties of the module in the hardware configuration and the system constants.
RET_VAL	Output	INT	I, Q, M, D, L	Output of error information.
GEOADDR	InOut	VARIANT	D	Pointer to the GEOADDR system data type. The slot information is written in the system data type GEOADDR. See also: System data type GEOADDR (Page 3365)

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8090	The address specified at the LADDR parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

LOG2MOD: Determine the hardware identifier from addressing of STEP 7 V5.5 SPx

Description

You use the "LOG2MOD" instruction to determine the hardware identifier for an IO (sub)module from the addressing of STEP 7 5.5 SPx (IO data address or diagnostic address).

The hardware identifier is used at the LADDR input parameter for addressing of various instructions. You can convert the addressing parameters from STEP 7 5.5 SPx by calling "LOG2MOD" beforehand.

Parameter

The following table shows the parameters of the "LOG2MOD" instruction:

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Identifier of the address area as in STEP 7 5.5 SPx: • B#16#00: Bit15 of ADDR specifies whether an input (Bit15=0) or output address (Bit15=1) exists. • B#16#54= Peripheral input (PI) • B#16#55= Peripheral output (PQ)
ADDR	Input	WORD	I, Q, M, D, L or constant	Logical address of the IO data of the module as offset (corresponding addressing in STEP 7 5.5 SPx) or diagnostic address.
RET_VAL	Return	INT	I, Q, M, D, L	Error code of the instruction.
HWID	Output	HW_IO	I, Q, M, D, L	Determined hardware identifier of the IO (sub)module.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0	No error occurred.
8093	• Specified address is not used by any hardware components. • Specified value at IOID parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

IO2MOD: Determine hardware identifier from an IO address

Description

The "IO2MOD" instruction determines the hardware identifier of the module from an IO address (I, Q, PI, PQ) of a module.

Enter the IO address at the ADDR parameter. If a series of IO addresses is used at this parameter, only the first address is evaluated to determine the hardware identifier. If the first address is correctly specified, the length for the address specification at the ADDR is of no significance. If an address area is used that encompasses several modules or non-used addresses, the hardware identifier of the first module can also be determined.

If no IO address of a module is specified at parameter ADDR, the error code 8090 is output at parameter RET_VAL.

Note

Input of IO address in SCL

You cannot program using the IO access ID "%QWx:P" in SCL. In this case, use the symbolic tag name or the absolute address in the process image.

Parameters

The following table shows the parameters of the "IO2MOD" instruction:

Parameter	Declaration	Data type	Memory area	Description
ADDR	Input	VARIANT	I, Q, M, D, L	IO address (I, Q, PI, PQ) within a module. Make sure that slice access is not used for the parameter ADDR. If this is the case, incorrect values are output at the LADDR parameter.
RET_VAL	Return	INT	I, Q, M, D, L	Error code of the instruction.
LADDR	Output	HW_IO	I, Q, M, D, L	Determined hardware identifier (logical address) of the IO module.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8090	IO address specified at ADDR parameter is not used by any hardware component.
8092	Invalid data type used at parameter ADDR (for example, WCHAR or WSTRING).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

RD_ADDR: Determine IO addresses from the hardware identifier

Description

The "RD_ADDR" instruction determines the length and the start address of the inputs or outputs based on the hardware identifier of a sub(module).

- Use the LADDR parameter to select the input or output module based on the hardware identifier.
- The following output parameters are used depending on whether it is an input module or output module:
  - In the case of an input module the determined values are output at the PIADDR and PICOUNT parameters.
  - In the case of an output module the determined values are output at the PQADDR and PQCOUNT parameters.
- The PIADDR and PQADDR parameters each contain the start address of the I/O addresses of the module.
- The PICOUNT and PQCOUNT parameters each contain the number of bytes of the inputs our outputs (1 byte for 8 inputs/outputs, 2 bytes for 16 inputs/outputs).

Note

Addressing packed modules

If packed modules (not the first module of the packed module group) are addressed, the displayed data deviates from the hardware configuration. "0" is returned for PIADDR or PQADDR and PICOUNT or PQCOUNT. RET_VAL does not indicate an error (16#0000).

Parameters

The following table shows the parameters of the "RD_ADDR" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the (sub)module.
RET_VAL	Return	INT	I, Q, M, D, L	Error code of the instruction.
PIADDR	Output	UDINT	I, Q, M, D, L	Start address of the input module.
PICOUNT	Output	UINT	I, Q, M, D, L	Number of bytes of the inputs.
PQADDR	Output	UDINT	I, Q, M, D, L	Start address of the output module.
PQCOUNT	Output	UINT	I, Q, M, D, L	Number of bytes of the outputs.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8090	Hardware identifier of the module at the LADDR parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

System data type GEOADDR

Geographical address

The system data type GEOADDR contains the geographical address of a module, i.e. the slot information.

- Geographical address for PROFINET IO
  For PROFINET IO the geographical address was composed of the ID of the PROFINET IO system, the device number, the slot number and the submodule (if a sub-module is used).
- Geographical address for PROFIBUS DP
  For PROFIBUS DP, the geographical address consists of the ID of the DP master system, the station number and the slot number.

The slot information of the modules can be found in the hardware configuration of each module.

System data type GEOADDR

The structure GEOADDR is automatically created if you enter "GEOADDR" as the data type in a data block.

The system data type GEOADDR has the following structure:

Parameter name	Data type	Description
GEOADDR	STRUCT	
HWTYPE	UINT	Hardware type: • 1: IO system (PROFINET/PROFIBUS) • 2: IO device/DP slave • 3: Rack • 4: Module • 5: Submodule If a hardware type is not supported by the instruction, a HWTYPE "0" is output.
AREA	UINT	Area ID: • 0 = CPU • 1 = PROFINET IO • 2 = PROFIBUS DP • 3 = AS-i
IOSYSTEM	UINT	PROFINET IO system (0=central unit in the rack)
STATION	UINT	• Number of the rack if area identifier AREA = 0 (central module). • Station number if area identifier AREA > 0.
SLOT	UINT	Slot number
SUBSLOT	UINT	Number of the submodule. This parameter has the value "0" if no submodule is available or can be plugged.

Legacy

GEO_LOG: Determine hardware identifier from slot

Description

The corresponding module slot of a signal module is known. Use the "GEO_LOG" instruction to determine the corresponding hardware identifier of the module from this.

Parameters

The following table shows the parameters of the instruction "GEO_LOG":

Parameter	Declaration	Data type	Memory area	Description
MASTER	Input	INT	I, Q, M, D, L or constant	Area ID: • 0, if the slot is located in a centralized configuration. • 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	Input	INT	I, Q, M, D, L or constant	• If MASTER = 0: Number of the rack • If MASTER > 0: Station number of the field device
SLOT	Input	INT	I, Q, M, D, L or constant	Slot number
SUBSLOT	Input	INT	I, Q, M, D, L or constant	The parameter SUBSLOT is not evaluated by the instruction.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
LADDR	Output	HW_IO	I, Q, M, D, L	Hardware identifier of the module

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8094	No subnet was configured with the specified SUBNETID .
8095	Illegal value for STATION parameter
8096	Illegal value for SLOT parameter
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Instructions for address conversion (Page 3358)

LOG_GEO: Determine slot from hardware identifier

Description

You use the "LOG_GEO" instruction to determine the module slot belonging to a hardware identifier.

Parameters

The following table shows the parameters of the instruction "LOG_GEO":

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module for which the slot is to be determined.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
AREA	Output	INT	I, Q, M, D, L	Area ID: indicates how the remaining output parameters are to be interpreted: • 0: central device • 2: PROFIBUS DP / PROFINET IO
MASTER	Output	INT	I, Q, M, D, L	With AREA = 0: • 0: If the slot is located in one of the racks (central device). With AREA = 2: • 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	Output	INT	I, Q, M, D, L	• With MASTER = 0: Number of the rack • With MASTER > 0: Station number of the field device
SLOT	Output	INT	I, Q, M, D, L	Slot number
SUBSLOT	Output	INT	I, Q, M, D, L	The SUBSLOT parameter is not output by the instruction (always "0").
OFFSET	Output	INT	I, Q, M, D, L	The OFFSET parameter is not output by the instruction (always "0").

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Instructions for address conversion (Page 3358)

RD_LGADR: Determine IO addresses from the hardware identifier

Description

Based on the hardware identifier, you can determine the logical addresses of a module, a central submodule or a submodule for PNIO with the "RD_LGADR" instruction.

- You can specify the hardware ID of the submodule at the LADDR parameter.
- The addresses are written to the PEADDR and PAADDR parameters is ascending order.
  - For an input module, only the PEADDR parameter is written. For an output module, the PAADDR parameter is written.
  - An Array of WORD is used to store the addresses in each case.
- The number of addresses is output at the PECOUNT parameter (for an input module) and PACOUNT parameter (for an output module).

Parameters

The following table shows the parameters of the instruction "RD_LGADR":

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ)
LADDR	Input	HW_ANY	I, Q, M, D, L or constant	Hardware identifier of the module or the submodule.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PEADDR	Output	ANY	I, Q, M, D, L	Field for the PI addresses with the Array of WORD data type
PECOUNT	Output	INT	I, Q, M, D, L	Number of returned PI addresses

Parameter	Declaration	Data type	Memory area	Description
PAADDR	Output	ANY	I, Q, M, D, L	Field for the PQ addresses with the Array of WORD data type
PACOUNT	Output	INT	I, Q, M, D, L	Number of returned PQ addresses

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid or illegal value for the IOID parameter.
80A0	Error in the output parameter PEADDR: The data type of the array elements is not WORD.
80A1	Error in the output parameter PAADDR: The data type of the array elements is not WORD.
80A2	Error in the output parameter PEADDR: The specified array could not accommodate all the logical addresses.
80A3	Error in the output parameter PAADDR: The specified array could not accommodate all the logical addresses.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)

See also

Instructions for address conversion (Page 3358)

GADR_LGC: Determine hardware identifier from slot and offset in the user data address area

Description

You use the "GADR_LGC" instruction to determine the hardware identifier of a signal module. The hardware identifier is determined from the module slot and the offset in the user data address area of the module.

Note

Output of the diagnostic address

If you use the "GADR_LGC" instruction on power modules or modules with packed addresses, then the diagnostic address will be returned.

Note

Restriction on use

The "GADR_LGC" instruction cannot be used for modules behind gateways (e.g., IE/PB link). Use the "GEO2LOG" instruction instead.

Parameters

The following table shows the parameters of the instruction "GADR_LGC":

Parameter	Declaration	Data type	Memory area	Description
SUBNETID	Input	BYTE	I, Q, M, D, L or constant	Area ID: • 0: If the slot is located in the central module • 1 to 32: DP master system ID of the corresponding distributed I/O system if the slot is in a distributed I/O device. • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
RACK	Input	WORD	I, Q, M, D, L or constant	• Number of the rack, if area identifier is 0 • Device number of the distributed I/O device if the area identifier > 0
SLOT	Input	WORD	I, Q, M, D, L or constant	Slot no.
SUBSLOT	Input	BYTE	I, Q, M, D, L or constant	Sub-module slot (if no sub-module can be inserted, 0 must be entered here)
SUBADDR	Input	WORD	I, Q, M, D, L or constant	Offset in the user data address area of the module
RET_VAL	Return	INT	I, Q, M, D, L	Error information
IOID	Output	BYTE	I, Q, M, D, L	The IOID output parameter is not written (always "0").
LADDR	Output	HW_MODULE	I, Q, M, D, L	Hardware identifier of the module

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8093	Illegal value for SUBNETID parameter
8094	No subnet was configured with the specified SUBNETID .
8095	Illegal value for RACK parameter.

Error code* (W#16#...)	Explanation
8096	Illegal value for SLOT parameter.
8097	Illegal value for SUBSLOT parameter.
8098	Illegal value for SUBADDR parameter.
8099	The slot is not configured.
809A	The sub-address of the selected slot is not configured (only possible with central I/O for CPU and IM).
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

See also

Instructions for address conversion (Page 3358)

LGC_GADR: Determine slot from hardware identifier

Description

You use the "LGC_GADR" instruction to determine the module slot belonging to a hardware identifier.

Note

The "LGC_GADR" instruction cannot be used on a module with packed addresses (ET 200S).

Note

Restriction on use

The "LGC_GADR" instruction cannot be used for modules behind gateways (e.g., IE/PB link). Use the "LOG2GEO" instruction instead.

Parameters

The following table shows the parameters of the instruction "LGC_GADR":

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Is not evaluated.
LADDR	Input	HW_MODULE	I, Q, M, D, L or constant	Hardware identifier of the module
RET_VAL	Return	INT	I, Q, M, D, L	Error information
AREA	Output	BYTE	I, Q, M, D, L	Area ID: indicates how the remaining output parameters are to be interpreted: • 0: Central module • 2: PROFIBUS DP

Parameter	Declaration	Data type	Memory area	Description
RACK	Output	WORD	I, Q, M, D, L	Rack number: • With central module AREA= 0): – Rack number • For PROFIBUS DP (AREA=2): – Low byte: Station number – High byte: DP master system ID
SLOT	Output	WORD	I, Q, M, D, L	Slot number: • With central module AREA= 0): – Slot number • For PROFIBUS DP (AREA=2): – Slot no. in the station
SUBADDR	Output	WORD	I, Q, M, D, L	Is not output (always "0").

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid or illegal value for the IOID parameter.
8093	This instruction is not permitted for the module selected by the parameters IOID and LADDR .
General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

See also

Instructions for address conversion (Page 3358)

11.6.4 Technology

11.6.4.1 S7-1200 Motion Control

S7-1200 Motion Control as of V4

MC_Power

MC_Power: Enable, disable axes as of V4

Description

The Motion Control instruction "MC_Power" enables or disables an axis.

Requirements

- The positioning axis technology object has been configured correctly.
- There is no pending enable-inhibiting error.

Override response

Execution of "MC_Power" cannot be aborted by a Motion Control command.

Disabling the axis (input parameter "Enable" = FALSE) aborts all Motion Control commands for the associated technology object in accordance with the selected "StopMode".

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Axis technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The axis is enabled.
				FALSE	All current jobs are interrupted in accordance with the "StopMode" configured. The axis is stopped and disabled.

Parameter	Declaration	Data type	Default value	Description	
StopMode	INPUT	INT	0	0	Emergency stop If a request to disable the axis is pending, the axis brakes at the configured emergency deceleration. The axis is disabled after reaching standstill.
				1	Immediate stop If a request to disable the axis is pending, this set-point zero is output and the axis is disabled. The axis is braked depending on the configuration in the drive, and is brought to a standstill. With drive connection via PTO (Pulse Train Output): When you disable the axis, the pulse output is stopped with a frequency-dependent deceleration: • Output frequency ≥ 100 Hz Deceleration: max. 30 ms • Output frequency < 100 Hz Deceleration: 30 ms up to max. 1.5 s at 2 Hz
				2	Emergency stop with jerk control If a request to disable the axis is pending, the axis brakes at the configured emergency deceleration. If the jerk control is activated, the configured jerk is taken into account. The axis is disabled after reaching standstill.
Status	OUTPUT	BOOL	FALSE	Status of axis enable	
				FALSE	The axis is disabled. The axis does not execute Motion Control commands and does not accept any new commands (exception: MC_Reset command). For drive connection via PTO (Pulse Train Output): The axis is not homed. Upon disabling, the status does not change to FALSE until the axis reaches a standstill.
				TRUE	The axis is enabled. The axis is ready to execute Motion Control commands. Upon axis enabling, the status does not change to TRUE until the signal "Drive ready" is pending. If the "Drive ready" drive interface was not configured in the axis configuration, the status changes to TRUE immediately.
Busy	OUTPUT	BOOL	FALSE	TRUE	"MC_Power" is active.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction "MC_Power" or in the associated technology object. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

Note

If the axis is switched off due to an error, it will be enabled again automatically after the error has been eliminated and acknowledged. This requires that the input parameter "Enable" has retained the value TRUE during this process.

Enabling an axis with configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE.
   The enable output for "Drive enabled" changes to TRUE to enable the power to the drive.
   The CPU waits for the "Drive ready" signal of the drive.
   When the "Drive ready" signal is available at the configured ready input of the CPU, the axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Enabling an axis without configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE. The axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Disabling an axis

To disable an axis, you can follow the steps described below:

1. Bring the axis to a standstill.
   You can identify when the axis is at a standstill in the tag of the technology object <Axis name>.StatusBits.StandStill.
2. Set input parameter "Enable" to FALSE after standstill is reached.
3. If output parameters "Busy" and "Status" and tag of technology object <Axis name>.StatusBits.Enable indicate the value FALSE, disabling of the axis is complete.

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

MC_Power: Function chart as of V4 (Page 3378)

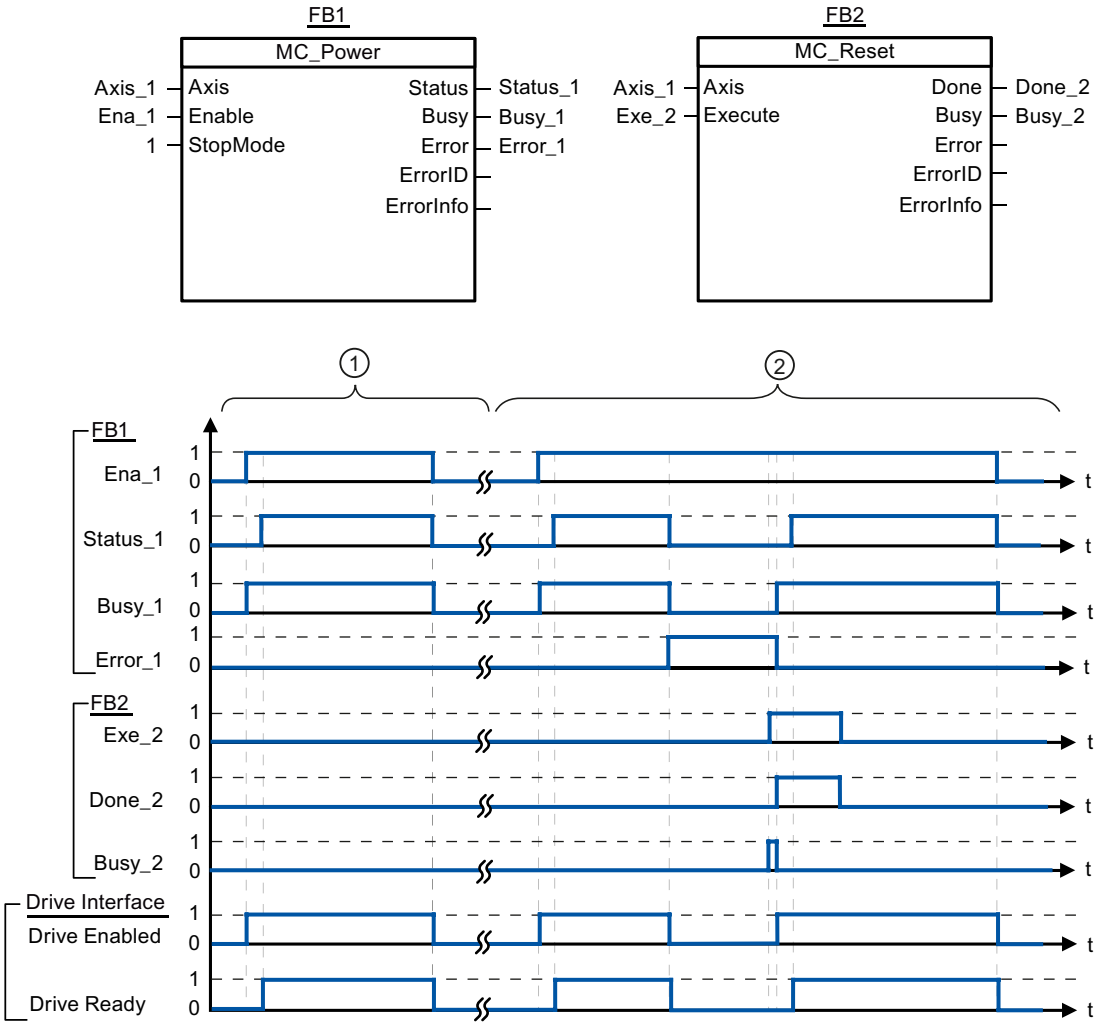
MC_Reset: Acknowledge fault as of V4 (Page 3379)

MC_Home: Home axes, set reference point as of V4 (Page 3381)

MC_Halt: Stop axes as of V4 (Page 3385)
MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)
MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)
MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)
MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)
MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)
MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)
MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)
MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)
S7-1200 Motion Control V1...3 (Page 3410)

MC_Power: Function chart as of V4

Function chart



①	An axis is enabled and then disabled again. When the drive has signaled "Drive ready" back to the CPU, the successful enable can be read out via "Status_1".
②	Following an axis enable, an error has occurred that caused the axis to be disabled. The error is eliminated and acknowledged with "MC_Reset". The axis is then enabled again.

See also

MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Reset

MC_Reset: Acknowledge fault as of V4

Description

Motion Control instruction "MC_Reset" can be used to acknowledge "Operating error with axis stop" and "Configuration error". The errors that require acknowledgment can be found in the "List of ErrorIDs and ErrorInfos" under "Remedy".

The axis configuration can be downloaded to the work memory after a download in RUN mode.

Requirements

- The positioning axis technology object has been configured correctly.
- The cause of a pending configuration error requiring acknowledgment has been eliminated (for example, acceleration in positioning axis technology object has been changed to a valid value).

Override response

The MC_Reset command cannot be aborted by any other Motion Control command.

The new MC_Reset command does not abort any other active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Restart	INPUT	BOOL	FALSE	TRUE	Download the axis configuration from the load memory to the work memory. The command can only be executed when the axis is disabled. Refer to the notes on Download to the CPU (Page 7434).
				FALSE	Acknowledges pending errors
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

Acknowledging an error requiring acknowledgment with MC_Reset

To acknowledge an error, follow these steps:

1. Check the requirements indicated above.
2. Start the acknowledgment of the error with a rising edge at input parameter "Execute".
3. If output parameter "Done" indicates the value TRUE and tag of technology object <Axis name>.StatusBits.Error the value FALSE, the error has been acknowledged.

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

Download to CPU (Page 7434)

MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Home: Home axes, set reference point as of V4 (Page 3381)

MC_Halt: Stop axes as of V4 (Page 3385)

MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)

MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)

MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)

MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)

MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)

MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)

MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)

MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)

S7-1200 Motion Control V1...3 (Page 3410)

MC_Home

MC_Home: Home axes, set reference point as of V4

Description

Motion Control instruction "MC_Home" is used to match the axis coordinates to the real, physical drive position. Homing is required for absolute positioning of the axis. The following types of homing can be executed:

- Active homing (Mode = 3)
  The homing procedure is executed automatically.
- Passive homing (Mode = 2)
  During passive homing, the "MC_Home" Motion Control instruction does not carry out any homing motion. The travel required for this step must be implemented by the user via other Motion Control instructions. When the homing switch is detected, the axis is homed.
- Direct homing absolute (Mode = 0)
  The current axis position is set to the value of parameter "Position".
- Direct homing relative (Mode = 1)
  The current axis position is offset by the value of parameter "Position".

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.
- No MC_CommandTable command may be active upon start with Mode = 0, 1, or 2.

Override response

The override response depends on the selected mode:

Mode = 0, 1

The MC_Home command cannot be aborted by any other Motion Control command.

The MC_Home command does not abort any active Motion Control commands. Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 2

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 2, 3

The new MC_Home command aborts the following active Motion Control command:

- MC_Home command Mode = 2

Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 3

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Home command aborts the following active Motion Control commands:

- MC_Home command Mode = 2, 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description
Axis	INPUT	TO_Axis	-	Axis technology object
Execute	INPUT	BOOL	FALSE	Start of the job with a positive edge
Position	INPUT	REAL	0.0	• Mode = 0, 2, and 3 Absolute position of axis after completion of the homing operation • Mode = 1 Correction value for the current axis position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$

Parameter	Declaration	Data type	Default value	Description	
Mode	INPUT	INT	0	Homing mode	
				0	Direct homing (absolute) New axis position is the position value of parameter "Position".
				1	Direct homing (relative) New axis position is the current axis position + position value of parameter "Position".
				2	Passive homing Homing according to the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
				3	Active homing Homing procedure in accordance with the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
Done	OUTPUT	BOOL	FALSE	TRUE	Command completed
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the job was aborted by another job.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

Resetting the "Homed" status

The "Homed" status of a technology object (<AxisName>.StatusBits.HomingDone) is reset under the following conditions:

- **Drive connection via PTO (Pulse Train Output):**
  - Start a "MC_Home" command for active homing
    (After successful completion of the homing operation, the "Homed" status is reset.)
  - Disabling of axis by the "MC_Power" Motion Control instruction
  - Changeover between automatic mode and manual control
  - After POWER OFF -> POWER ON of the CPU
  - After CPU restart (RUN-STOP -> STOP-RUN)
- **Technology objects with incremental actual values:**
  - Start a "MC_Home" command for active homing
    (After successful completion of the homing operation, the "Homed" status is reset.)
  - Error in the encoder system, or encoder failure
  - Restart of the technology object
  - After POWER OFF → POWER ON of the CPU
  - Memory reset
  - Modification of the encoder configuration
- **Technology objects with absolute actual values:**
  - Restoration of the CPU factory settings
  - Modification of the encoder configuration
  - Replacement of the CPU

Homing an axis

To home the axis, follow these stops:

1. Check the requirements indicated above.
2. Provide the necessary input parameters with values and start the homing operation with a rising edge at input parameter "Execute".
3. If output parameter "Done" and technology object tag <Axis name>.StatusBits.HomingDone indicate the value TRUE, homing is complete.

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Reset: Acknowledge fault as of V4 (Page 3379)

MC_Halt: Stop axes as of V4 (Page 3385)

MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)
MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)
MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)
MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)
MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)
MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)
MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)
MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)
S7-1200 Motion Control V1...3 (Page 3410)

MC_Halt

MC_Halt: Stop axes as of V4

Description

The "MC_Halt" Motion Control instruction stops all movements and brings the axis to a standstill with the configured deceleration. The standstill position is not defined.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_Halt command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Halt command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command

- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

MC_Halt: Function chart as of V4 (Page 3387)

MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Reset: Acknowledge fault as of V4 (Page 3379)

MC_Home: Home axes, set reference point as of V4 (Page 3381)

MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)

MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)

MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)

MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)

MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)

MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)

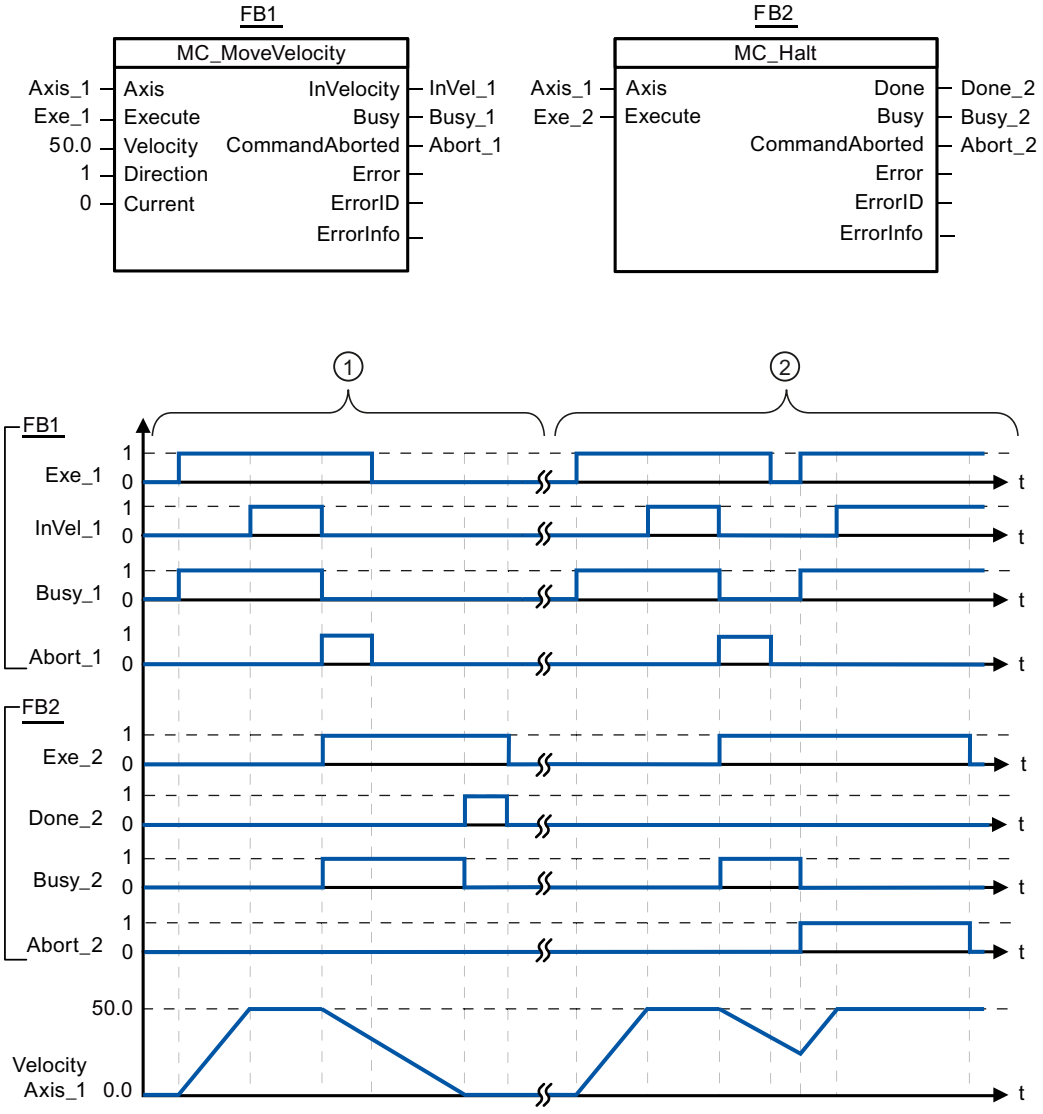
MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)

MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)

S7-1200 Motion Control V1...3 (Page 3410)

MC_Halt: Function chart as of V4

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is braked by an MC_Halt command until it comes to a standstill. The axis standstill is signaled via "Done_2".
②	While an MC_Halt command is braking the axis, this command is aborted by another motion command. The abort is signaled via "Abort_2".

See also

MC_Halt: Stop axes as of V4 (Page 3385)

MC_MoveAbsolute

MC_MoveAbsolute: Absolute positioning of axes as of V4

Description

The "MC_MoveAbsolute" Motion Control instruction starts an axis positioning motion to move it to an absolute position.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.
- The axis is homed.

Override response

The MC_MoveAbsolute command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveAbsolute command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning-Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	Absolute target position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the target position to be approached. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Absolute target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

MC_MoveAbsolute: Function chart as of V4 (Page 3390)

MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Reset: Acknowledge fault as of V4 (Page 3379)

MC_Home: Home axes, set reference point as of V4 (Page 3381)

MC_Halt: Stop axes as of V4 (Page 3385)

MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)

MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)

MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)

MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)

MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)

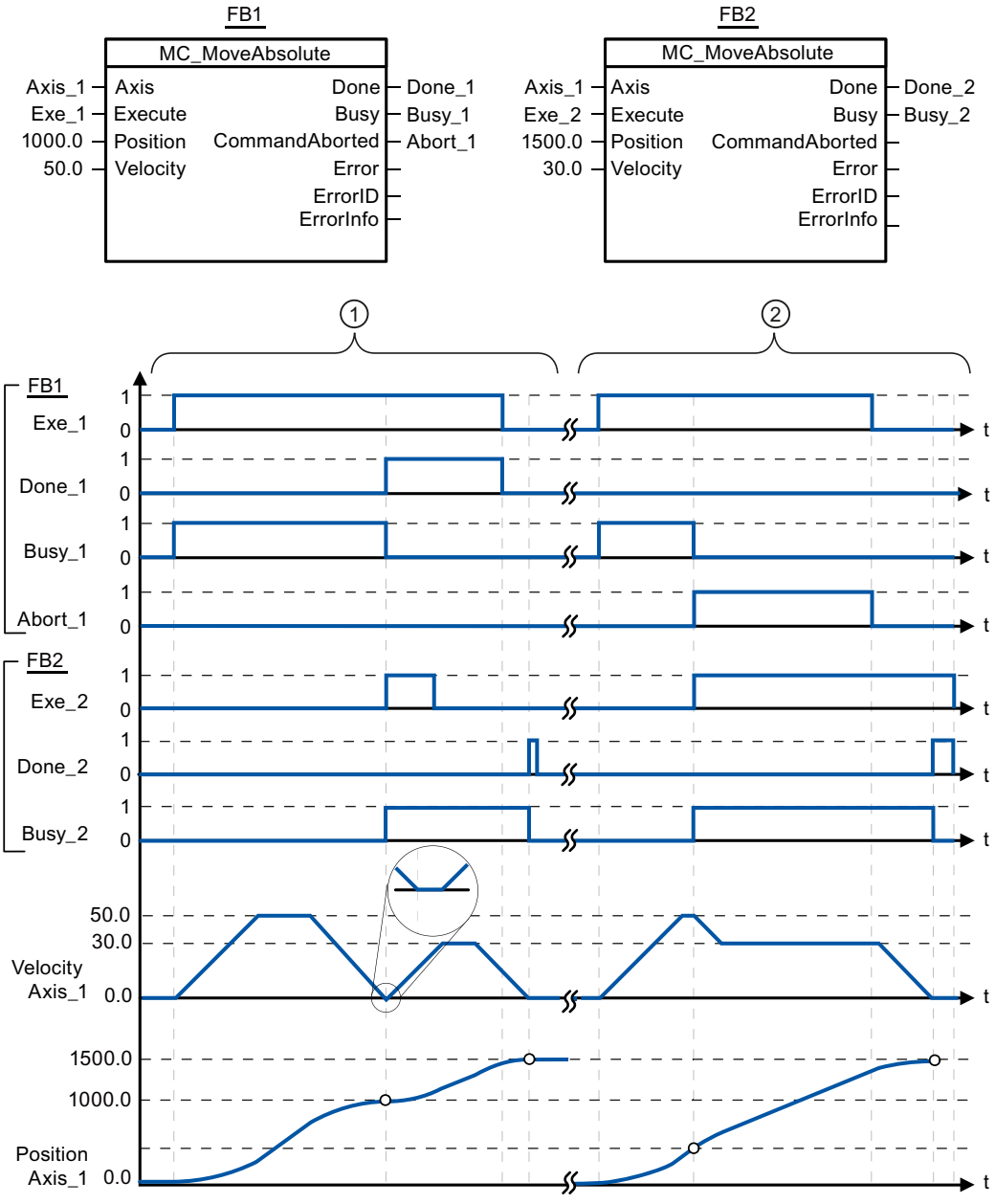
MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)

MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)

S7-1200 Motion Control V1...3 (Page 3410)

MC_MoveAbsolute: Function chart as of V4

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An axis is moved to absolute position 1000.0 with an MC_MoveAbsolute command. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveAbsolute command, with target position 1500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveAbsolute command is aborted by another MC_MoveAbsolute command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)

MC_MoveRelative

MC_MoveRelative: Relative positioning of axes as of V4

Description

The "MC_MoveRelative" Motion Control instruction starts a positioning motion relative to the start position.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveRelative command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveRelative command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning-Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Distance	INPUT	REAL	0.0	Travel distance for the positioning operation Limit values: $-1.0e^{12} \leq$ Distance $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the distance to be traveled. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

MC_MoveRelative: Function chart as of V4 (Page 3394)

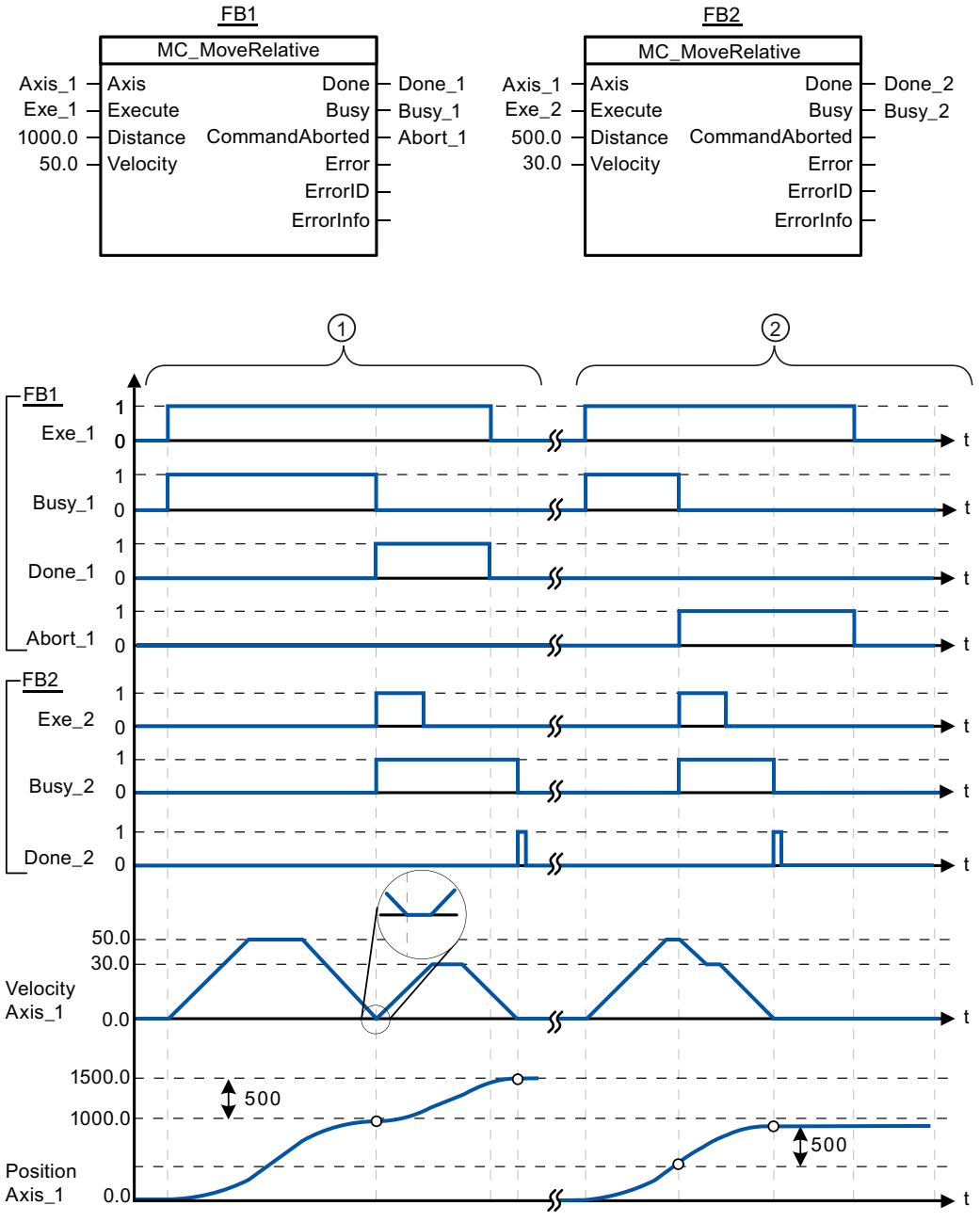
MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Reset: Acknowledge fault as of V4 (Page 3379)

MC_Home: Home axes, set reference point as of V4 (Page 3381)
MC_Halt: Stop axes as of V4 (Page 3385)
MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)
MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)
MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)
MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)
MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)
MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)
MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)
S7-1200 Motion Control V1...3 (Page 3410)

MC_MoveRelative: Function chart as of V4

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	The axis is moved by an MC_MoveRelative command by the distance ("Distance") 1000.0. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveRelative command, with travel distance 500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveRelative command is aborted by another MC_MoveRelative command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity by the new distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)

MC_MoveVelocity

MC_MoveVelocity: Move axes at preset rotational speed as of V4

Description

The Motion Control instruction "MC_MoveVelocity" moves the axis constantly at the specified velocity.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

MC_MoveVelocity can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveVelocity command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command

- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Velocity	INPUT	REAL	10.0	Velocity specification for axis motion Limit values: Start/stop velocity ≤ |Velocity ≤ maximum velocity (Velocity = 0.0 is permitted)	
Direction	INPUT	INT	0	Direction specification	
				0	Direction of rotation corresponds to the sign of the value in parameter "Velocity"
				1	Positive direction of rotation (The sign of the value in parameter "Velocity" is ignored)
				2	Negative direction of rotation (The sign of the value in parameter "Velocity" is ignored)
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	"Maintain current velocity" is deactivated. The values of parameters "Velocity" and "Direction" are used.
				TRUE	"Maintain current velocity" is activated. The values in parameters "Velocity" and "Direction" are not taken into account. When the axis resumes motion at the current velocity, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUTPUT	BOOL	FALSE	TRUE	• "Current" = FALSE: The velocity specified in parameter "Velocity" was reached. • "Current" = TRUE: The axis travels at the current velocity at the start time.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

Note

PLCopen Version 2.0

The Motion Control instruction "MC_MoveVelocity" is compatible to PLCopen Version 2.0 as of V4.

The "InVelocity" and "Busy" parameters show their status regardless of the "Execute" parameter until the command is overridden or stopped by an error. For more information, refer to the section " Tracking active commands (Page 7450).

Behavior with zero setpoint velocity (Velocity = 0.0)

An MC_MoveVelocity command with "Velocity" = 0.0 (such as an MC_Halt command) aborts active motion commands and stops the axis with the configured deceleration.

When the axis comes to a standstill, output parameter "InVelocity" indicates TRUE for at least one program cycle.

"Busy" indicates the value TRUE during the deceleration process and changes to FALSE together with "InVelocity". If parameter "Execute" = TRUE is set, "InVelocity" and "Busy" are latched.

When the "MC_MoveVelocity" command is started, status bit "SpeedCommand" is set in the technology object. Status bit "ConstantVelocity" is set upon axis standstill. Both bits are adapted to the new situation when a new motion command is started.

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

MC_MoveVelocity: Function chart as of V4 (Page 3398)

MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Reset: Acknowledge fault as of V4 (Page 3379)

MC_Home: Home axes, set reference point as of V4 (Page 3381)

MC_Halt: Stop axes as of V4 (Page 3385)

MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)

MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)

MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)

MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)

MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)

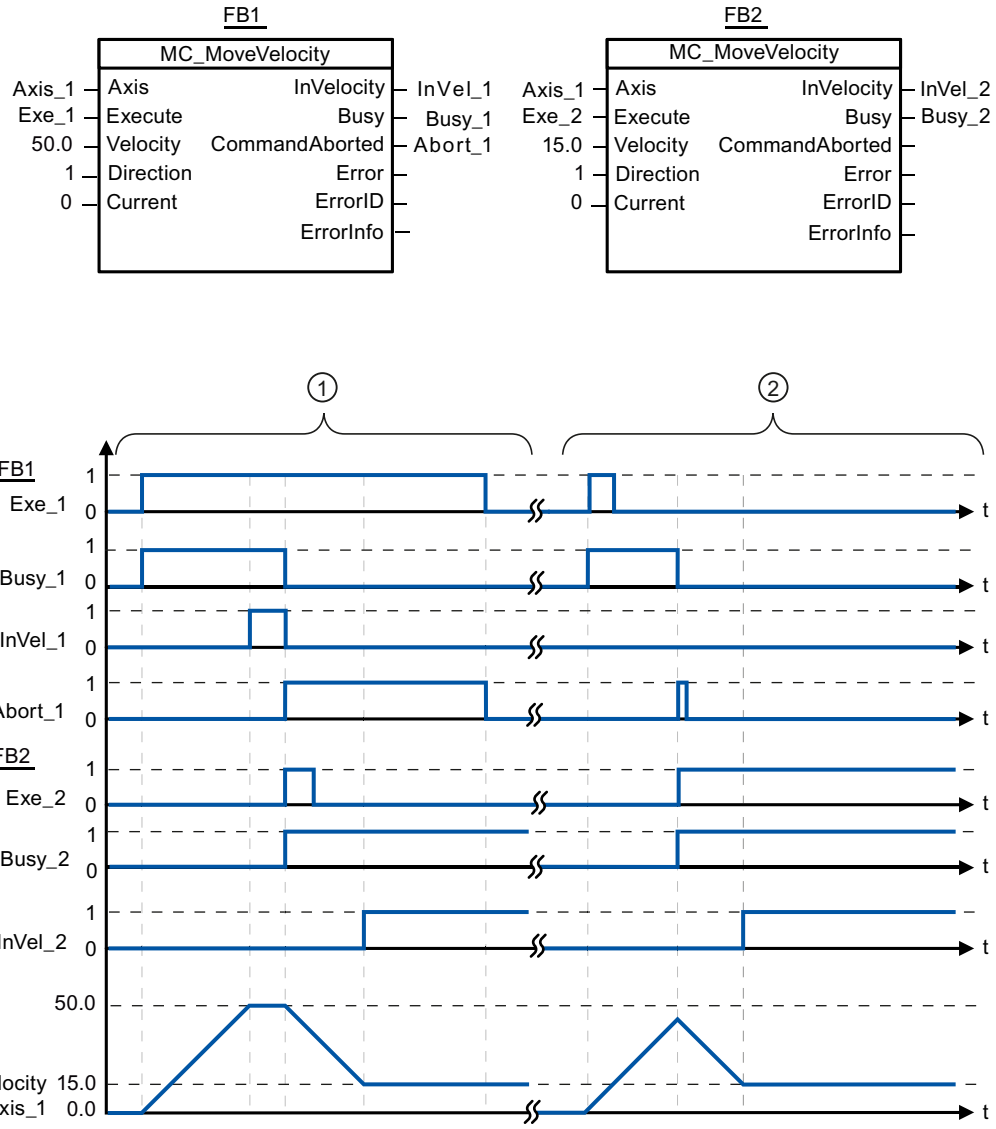
MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)

MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)

S7-1200 Motion Control V1...3 (Page 3410)

MC_MoveVelocity: Function chart as of V4

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An active MC_MoveVelocity command signals via "InVel_1" that its target velocity has been reached. It is then aborted by another MC_MoveVelocity command. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.
②	An active MC_MoveVelocity command is aborted by another MC_MoveVelocity command prior to reaching its target velocity. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.

See also

MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)

MC_MoveJog

MC_MoveJog: Move axes in jog mode as of V4

Description

Motion control instruction "MC_MoveJog" moves the axis constantly at the specified velocity in jog mode. You use this Motion Control instruction, for example, for testing and commissioning purposes.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveJog command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveJog command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".	
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".	
If both parameters are simultaneously TRUE, the axis stops with the configured deceleration. An error is indicated in parameters "Error", "ErrorID", and "ErrorInfo".					
Velocity	INPUT	REAL	10.0	Preset velocity for jog mode Limit values: Start/stop velocity ≤ velocity ≤ maximum velocity	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity specified in parameter "Velocity" was reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

MC_MoveJog: Function chart as of V4 (Page 3401)

MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Reset: Acknowledge fault as of V4 (Page 3379)

MC_Home: Home axes, set reference point as of V4 (Page 3381)

MC_Halt: Stop axes as of V4 (Page 3385)

MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)

MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)

MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)

MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)

MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)

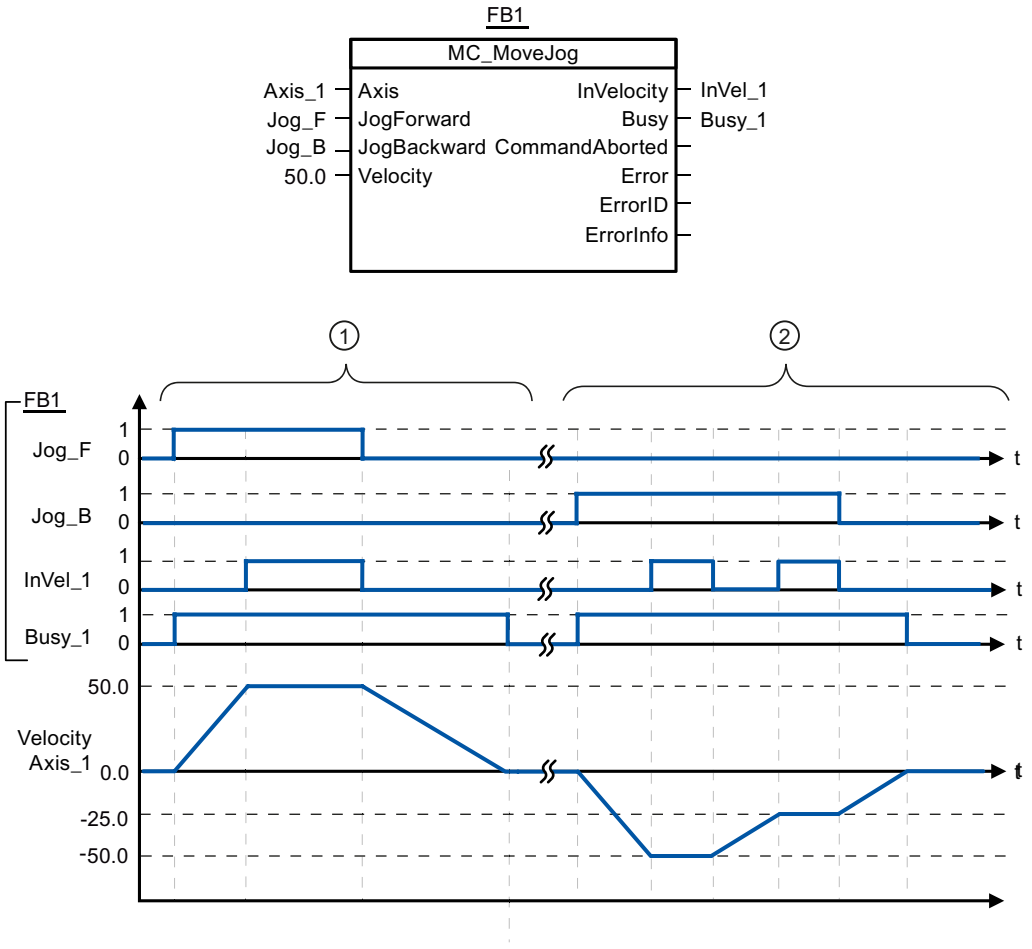
MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)

MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)

S7-1200 Motion Control V1...3 (Page 3410)

MC_MoveJog: Function chart as of V4

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is moved in the positive direction in jog mode via "Jog_F". When the target velocity 50.0 is reached, this is signaled via "InVel_1". After "Jog_F" is reset, the axis is braked to a standstill.
②	The axis is moved in the negative direction in jog mode via "Jog_B". When the target velocity -50.0 is reached, this is signaled via "InVel_1". When "Jog_B" is set, the value at parameter "Velocity" changes to 25.0. "InVel_1" is reset and the axis is braked. When the new target velocity -25.0 is reached, this is signaled via "InVel_1". After "Jog_B" is reset, the axis is braked to a standstill.

See also

MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)

MC_CommandTable

MC_CommandTable: Run axis commands as motion sequence as of V4

Description

The Motion Control instruction "MC_CommandTable" combines multiple individual axis control commands in one movement sequence. "MC_CommandTable" is available for axes with drive connection via PTO (Pulse Train Output).

Requirements

- The positioning axis technology object has been inserted and correctly configured.
- The drive is connected via PTO (Pulse Train Output).
- The command table technology object has been inserted and correctly configured.
- The axis is enabled.

Override response

The MC_CommandTable command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_CommandTable command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The active Motion Control command is canceled by the start of the first "Positioning Relative", "Positioning Absolute", "Velocity set point" or "Halt" command.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
CommandTable	INPUT	TO_CommandTable	-	Command table technology object	
Execute	INPUT	BOOL	FALSE	Command table start with positive edge	
StartStep	INPUT	INT	1	Defines the step at which the execution of the command table should begin Limit values: 1 ≤ StartStep ≤ EndStep	
EndStep	INPUT	INT	32	Defines the step up to which the execution of command table should take place Limit values: StartStep ≤ EndStep ≤ 32	
Done	OUTPUT	BOOL	FALSE	TRUE	Command table has been successfully executed
Busy	OUTPU	BOOL	FALSE	TRUE	The command table is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The command table was canceled by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command table. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	
CurrentStep	OUTPUT	INT	0	Step in command table currently being executed	
StepCode	OUTPUT	WORD	16#0000	User-defined numerical value / bit pattern of the step currently being executed	

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Reset: Acknowledge fault as of V4 (Page 3379)

MC_Home: Home axes, set reference point as of V4 (Page 3381)

MC_Halt: Stop axes as of V4 (Page 3385)

MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)

MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)

MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)

MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)

MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)
MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)
MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)
S7-1200 Motion Control V1...3 (Page 3410)

MC_ChangeDynamic

MC_ChangeDynamic: Change dynamic settings of axis as of V4

Description

Motion Control instruction "MC_ChangeDynamic" allows you to change the following settings of the axis:

- Change the ramp-up time (acceleration) value
- Change the ramp-down time (deceleration) value
- Change the emergency stop ramp-down time (emergency stop deceleration) value
- Change the smoothing time (jerk) value

For the effectiveness of the change, refer to the description of the tag (Page 7498).

Requirements

The positioning axis technology object has been configured correctly.

Override response

A MC_ChangeDynamic command cannot be aborted by any other Motion Control command.

A new MC_ChangeDynamic command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
ChangeRampUp	INPUT	BOOL	FALSE	TRUE	Change ramp-up time in line with input parameter "RampUpTime"
RampUpTime	INPUT	REAL	5.00	Time (in seconds) to accelerate axis from standstill to configured maximum velocity without jerk limit. The change will influence the tag <Axis name>. Config.DynamicDefaults.Acceleration. For the effectiveness of the change, refer to the description of this tag.	
ChangeRampDown	INPUT	BOOL	FALSE	TRUE	Change ramp-down time to correspond to input parameter "RampDownTime"

Parameter	Declaration	Data type	Default value	Description	
RampDownTime	INPUT	REAL	5.00	Time (in seconds) to decelerate axis from the configured maximum velocity to standstill without jerk limiter. The change will influence the tag <Axis name>. Config.DynamicDefaults.Deceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeEmergency	INPUT	BOOL	FALSE	TRUE	Change emergency stop ramp-down time in line with input parameter "EmergencyRampTime"
EmergencyRampTime	INPUT	REAL	2.00	Time (in seconds) to decelerate the axis from configured maximum velocity to standstill without jerk limiter in emergency stop mode. The change will influence the tag <Axis name>. Config.DynamicDefaults.EmergencyDeceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeJerkTime	INPUT	BOOL	FALSE	TRUE	Change smoothing time according to the input parameter "JerkTime"
JerkTime	INPUT	REAL	0.25	Smoothing time (in seconds) used for the axis acceleration and deceleration ramps The change will influence the tag <Axis name>. Config.DynamicDefaults.Jerk . For the effectiveness of the change, refer to the description of this tag.	
Done	OUTPUT	BOOL	FALSE	TRUE	The changed values have been written to the technology data block. The description of the tags will show when the change becomes effective.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

Note

At the input parameters "RampUpTime", "RampDownTime", "EmergencyRampTime" and "JerkTime", values can be entered which exceed the admissible limits of the resulting parameters: "Acceleration", "Deceleration", "Emergency stop deceleration" and "Jerk".

Ensure that your inputs are within the valid range, taking into consideration the equations and limits in section "Dynamic (Page 7381)".

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

Changing the configuration of dynamics in the user program (Page 7387)

Changing the homing configuration in the user program (Page 7394)

Tags of the positioning axis technology object as of V4 (Page 7498)

MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Reset: Acknowledge fault as of V4 (Page 3379)
MC_Home: Home axes, set reference point as of V4 (Page 3381)
MC_Halt: Stop axes as of V4 (Page 3385)
MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)
MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)
MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)
MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)
MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)
MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)
MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)
S7-1200 Motion Control V1...3 (Page 3410)

MC_ReadParam

MC_ReadParam: Continuously read motion data of a positioning axis as of V4

Description

The Motion Control instruction "MC_ReadParam" enables continuous reading of motion data and status messages of an axis. The current value of the corresponding tags is determined at the start of the command.

The following motion data and status messages can be read:

- As of technology version V4:
  - Setpoint position of the axis
  - Setpoint and actual velocity of the axis
  - Current distance of axis from target position
  - Target position of the axis
- Additional as of technology version V5:
  - Actual position of the axis
  - Actual velocity of the axis
  - Current following error
  - Drive status
  - Encoder status
  - Status bits
  - Error bits

Requirements

The positioning axis technology object has been configured correctly.

Override response

A MC_ReadParam command cannot be aborted by any other Motion Control command.

A new MC_ReadParam command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Enable	INPUT	BOOL	FALSE	TRUE	Read the tag specified with the "Parameter" and store the value in the destination address specified with "Value".
				FALSE	Do not update assigned motion data
Parameter	INPUT	VARIANT (REAL)	-	VARIANT pointer to the value to be read. The following tags are permitted: • <Axis name>.Position • <Axis name>.Velocity • <Axis name>.ActualPosition • <Axis name>.ActualVelocity • <Axis name>.StatusPositioning.<Tag name> • <Axis name>.StatusDrive.<Tag name> • <Axis name>.StatusSensor.<Tag name> • <Axis name>.StatusBits.<Tag name> • <Axis name>.ErrorBits.<Tag name> The description of the tags named and the tag structures can be found in the Appendix Tags of the positioning axis technology object as of V4 (Page 7498).	
Value	INOUT	VARIANT (REAL)	-	VARIANT pointer to the target tag or destination address to which the read value is to be written.	
Valid	OUTPUT	BOOL	FALSE	TRUE	The read value is valid.
				FALSE	The read value is invalid.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

S7-1200 Motion Control V1...3 (Page 3410)
MC_Power: Enable, disable axes as of V4 (Page 3374)
MC_Reset: Acknowledge fault as of V4 (Page 3379)
MC_Home: Home axes, set reference point as of V4 (Page 3381)
MC_Halt: Stop axes as of V4 (Page 3385)
MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)
MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)
MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)
MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)
MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)
MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)
MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)

MC_WriteParam

MC_WriteParam: Write tag of positioning axis as of V4

Description

Motion Control instruction "MC_WriteParam" enables the writing of tags of the positioning axis technology object in the user program. In contrast to the value assignment of the tags in the user program, "MC_WriteParam" can also change values of read-only tags.

You can learn about the tags, the conditions under which they can be written and the time at which they take effect in the description of the technology object tags (Page 7498).

With drive connection via PROFIdrive/analog output, some parameters require a restart of the technology object after writing with "MC_WriteParam". If a restart is required, this will be indicated in the tags of the technology object <Axis name>.StatusBits.RestartRequired. The change of the parameter value becomes effective with these parameters after the restart with the enabling of the technology object (MC_Power.Status = TRUE).

Requirements

- The positioning axis technology object has been configured correctly.
- To write tags that are read-only in the user program, the axis must be disabled.

Override response

A MC_WriteParam command cannot be aborted by any other Motion Control command.

A new MC_WriteParam command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Parameter	INPUT	VARIANT (BOOL, INT, DINT, REAL)	-	VARIANT pointer to the technology object tag (Page 7498) positioning axis (destination address) to be written	
Value	INPUT	VARIANT (BOOL, INT, DINT, REAL)	-	VARIANT pointer to the value to be written (source address)	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Value was written
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7477) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7477) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 7440)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

Tags of the positioning axis technology object as of V4 (Page 7498)

S7-1200 Motion Control V1...3 (Page 3410)

MC_Power: Enable, disable axes as of V4 (Page 3374)

MC_Reset: Acknowledge fault as of V4 (Page 3379)

MC_Home: Home axes, set reference point as of V4 (Page 3381)

MC_Halt: Stop axes as of V4 (Page 3385)

MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)

MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)

MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)

MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)

MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)

MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)

MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)

S7-1200 Motion Control V1...3

MC_Power

MC_Power: Enable, disable axis V1...3

Description

The Motion Control instruction "MC_Power" enables or disables an axis.

Requirements

- The axis technology object has been configured correctly.
- There is no pending enable-inhibiting error.

Override response

Execution of "MC_Power" cannot be aborted by a Motion Control command.

Disabling the axis (input parameter "Enable" = FALSE) aborts all Motion Control commands for the associated technology object in accordance with the selected "StopMode".

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Enable	INPUT	BOOL	FALSE	TRUE	Motion Control attempts to enable the axis.
				FALSE	All current jobs are interrupted in accordance with the "StopMode" configured. The axis is stopped and disabled.
StopMode	INPUT	INT	0	0	Emergency stop If a request to disable the axis is pending, the axis brakes at the configured emergency stop deceleration. The axis is disabled after reaching standstill.
				1	Immediate stop If a request to disable the axis is pending, this axis is disabled without deceleration. The pulse output is stopped immediately.
				2	Emergency stop with jerk control If a request to disable the axis is pending, the axis brakes at the configured emergency stop deceleration. If the jerk control is activated, the configured jerk is taken into account. The axis is disabled after reaching standstill.

Parameter	Declaration	Data type	Default value	Description	
Status	OUTPUT	BOOL	FALSE	Status of axis enable	
				FALSE	The axis is disabled. The axis does not execute Motion Control commands and does not accept any new commands (exception: MC_Reset command). The axis is not homed. Upon disabling, the status does not change to FALSE until the axis reaches a standstill.
				TRUE	The axis is enabled. The axis is ready to execute Motion Control commands. Upon axis enabling, the status does not change to TRUE until the signal "Drive ready" is pending. If the "Drive ready" drive interface was not configured in the axis configuration, the status changes to TRUE immediately.
Busy	OUTPUT	BOOL	FALSE	TRUE	"MC_Power" is active.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction "MC_Power" or in the associated technology object. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7555) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7555) for parameter "ErrorID"	

Note

If the axis is switched off due to an error, it will be enabled again automatically after the error has been eliminated and acknowledged. This requires that the input parameter "Enable" has retained the value TRUE during this process.

Enabling an axis with configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE.
   The enable output for "Drive enabled" changes to TRUE to enable the power to the drive.
   The CPU waits for the "Drive ready" signal of the drive.
   When the "Drive ready" signal is available at the configured Ready input of the CPU, the axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Enabling an axis without configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE. The axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Disabling an axis

To disable an axis, you can follow the steps described below:

1. Bring the axis to a standstill.
   You can identify when the axis is at a standstill in the tag of the technology object <Axis name>.StatusBits.StandStill.
2. Set input parameter "Enable" to FALSE after standstill is reached.
3. If output parameters "Busy" and "Status" and the tag of technology object <Axis name>.StatusBits.Enable indicate the value FALSE, disabling of the axis is complete.

See also

Overview of the Motion Control statements (Page 7440)

S7-1200 Motion Control as of V4 (Page 3374)

ErrorIDs and ErrorInfos (Page 7555)

MC_Power: Function chart V1...3 (Page 3413)

MC_Reset: Acknowledge fault V1...3 (Page 3414)

MC_Home: Home axes, set reference point V1...3 (Page 3415)

MC_Halt: Halt axes V1...3 (Page 3419)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 3422)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 3425)

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 3428)

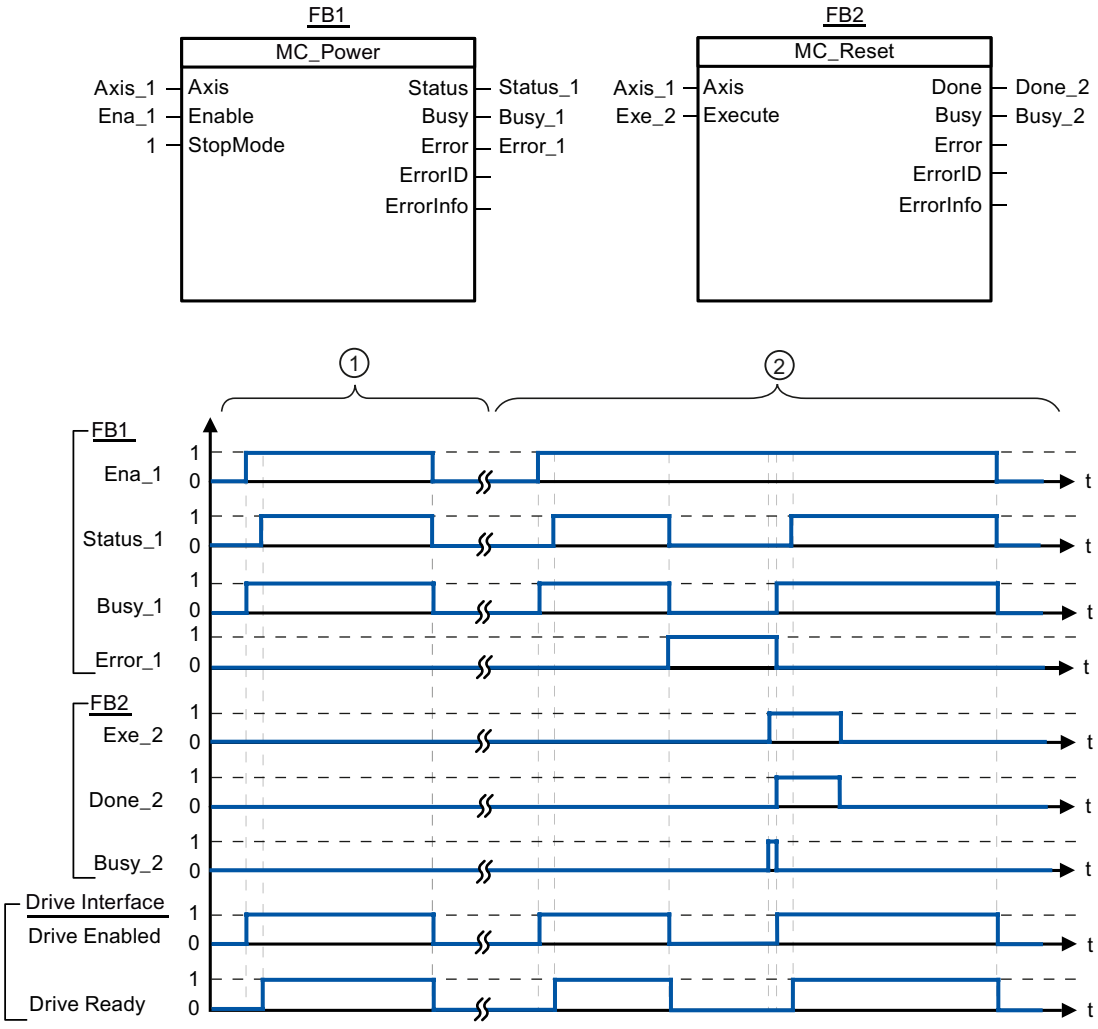
MC_MoveJog: Move axes in jog mode V1...3 (Page 3432)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3435)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3437)

MC_Power: Function chart V1...3

Function chart



①	An axis is enabled and then disabled again. When the drive has signaled "Drive ready" back to the CPU, the successful enable can be read out via "Status_1".
②	Following an axis enable, an error has occurred that caused the axis to be disabled. The error is eliminated and acknowledged with "MC_Reset". The axis is then enabled again.

See also

MC_Power: Enable, disable axis V1...3 (Page 3410)

MC_Reset

MC_Reset: Acknowledge fault V1...3

Description

Motion Control instruction "MC_Reset" can be used to acknowledge "Operating error with axis stop" and "Configuration error". The errors that require acknowledgement can be found in the "List of ErrorIDs and ErrorInfos" under "Remedy".

From version V3.0, the axis configuration can be downloaded to the work memory in RUN operating mode.

Requirements

- The axis technology object has been configured correctly.
- The cause of a pending configuration error requiring acknowledgement has been eliminated (for example, acceleration in positioning axis technology object has been changed to a valid value).

Override response

The MC_Reset command cannot be aborted by any other Motion Control command.

The new MC_Reset command does not abort any other active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Restart	INPUT	BOOL	FALSE	(From version V3.0)	
				TRUE	Download the axis configuration from the load memory to the work memory. The command can only be executed when the axis is disabled. Refer to the notes on Download to the CPU (Page 7434).
				FALSE	Acknowledges pending errors
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7555) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7555) for parameter "ErrorID"	

Acknowledging an error requiring acknowledgement with MC_Reset

To acknowledge an error, follow these steps:

1. Check the requirements indicated above.
2. Start the acknowledgement of the error with a rising edge at input parameter "Execute".
3. If output parameter "Done" indicates the value TRUE and tag of technology object <Axis name>.StatusBits.Error the value FALSE, the error has been acknowledged.

See also

Overview of the Motion Control statements (Page 7440)

S7-1200 Motion Control as of V4 (Page 3374)

ErrorIDs and ErrorInfos (Page 7555)

Download to CPU (Page 7434)

MC_Power: Enable, disable axis V1...3 (Page 3410)

MC_Home: Home axes, set reference point V1...3 (Page 3415)

MC_Halt: Halt axes V1...3 (Page 3419)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 3422)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 3425)

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 3428)

MC_MoveJog: Move axes in jog mode V1...3 (Page 3432)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3435)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3437)

MC_Home

MC_Home: Home axes, set reference point V1...3

Description

Motion Control instruction "MC_Home" is used to match the axis coordinates to the real, physical drive position. Homing is required for absolute positioning of the axis. The following types of homing can be executed:

- Active homing (Mode = 3)
  The homing procedure is executed automatically.
- Passive homing (Mode = 2)
  During passive homing, the "MC_Home" Motion Control instruction does not carry out any homing motion. The traversing motion required for this must be implemented by the user via other Motion Control instructions. When the homing switch is detected, the axis is homed.

- Direct homing absolute (Mode = 0)
  The current axis position is set to the value of parameter "Position".
- Direct homing relative (Mode = 1)
  The current axis position is offset by the value of parameter "Position".

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.
- No MC_CommandTable command may be active upon start with Mode = 0, 1, or 2.

Override response

The override response depends on the selected mode:

Mode = 0, 1

The MC_Home command cannot be aborted by any other Motion Control command.

The MC_Home command does not abort any active Motion Control commands. Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 2

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 2, 3

The new MC_Home command aborts the following active Motion Control command:

- MC_Home command Mode = 2

Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 3

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Home command aborts the following active Motion Control commands:

- MC_Home command Mode = 2, 3
- MC_Halt command
- MC_MoveAbsolute command

- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	• Mode = 0, 2, and 3 Absolute position of axis after completion of the homing operation • Mode = 1 Correction value for the current axis position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$	
Mode	INPUT	INT	0	Homing mode	
				0	Direct homing absolute New axis position is the position value of parameter "Position".
				1	Direct homing relative New axis position is the current axis position + position value of parameter "Position".
				2	Passive homing Homing according to the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
				3	Active homing Homing procedure in accordance with the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
Done	OUTPUT	BOOL	FALSE	TRUE	Command completed
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7555) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7555) for parameter "ErrorID"	

Note

Axis homing is lost under the following conditions:

- Disabling of axis by the "MC_Power" Motion Control instruction
- Changeover between automatic mode and manual control
- Upon start of active homing. After successful completion of the homing operation, axis homing is again available.
- After POWER OFF -> POWER ON of the CPU
- After CPU restart (RUN-STOP -> STOP-RUN)

Homing an axis

To home the axis, follow these stops:

1. Check the requirements indicated above.
2. Initialize the necessary input parameters with values, and start the homing operation with a rising edge at input parameter "Execute"
3. If output parameter "Done" and technology object tag <Axis name>.StatusBits.HomingDone indicate the value TRUE, homing is complete.

See also

Overview of the Motion Control statements (Page 7440)

S7-1200 Motion Control as of V4 (Page 3374)

ErrorIDs and ErrorInfos (Page 7555)

MC_Power: Enable, disable axis V1...3 (Page 3410)

MC_Reset: Acknowledge fault V1...3 (Page 3414)

MC_Halt: Halt axes V1...3 (Page 3419)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 3422)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 3425)

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 3428)

MC_MoveJog: Move axes in jog mode V1...3 (Page 3432)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3435)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3437)

MC_Halt

MC_Halt: Halt axes V1...3

Description

The "MC_Halt" Motion Control instruction stops all movements and brings the axis to a standstill with the configured deceleration. The standstill position is not defined.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_Halt command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Halt command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.

Parameter	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7555) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7555) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 7440)

S7-1200 Motion Control as of V4 (Page 3374)

ErrorIDs and ErrorInfos (Page 7555)

MC_Halt: Function chart V1...3 (Page 3421)

MC_Power: Enable, disable axis V1...3 (Page 3410)

MC_Reset: Acknowledge fault V1...3 (Page 3414)

MC_Home: Home axes, set reference point V1...3 (Page 3415)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 3422)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 3425)

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 3428)

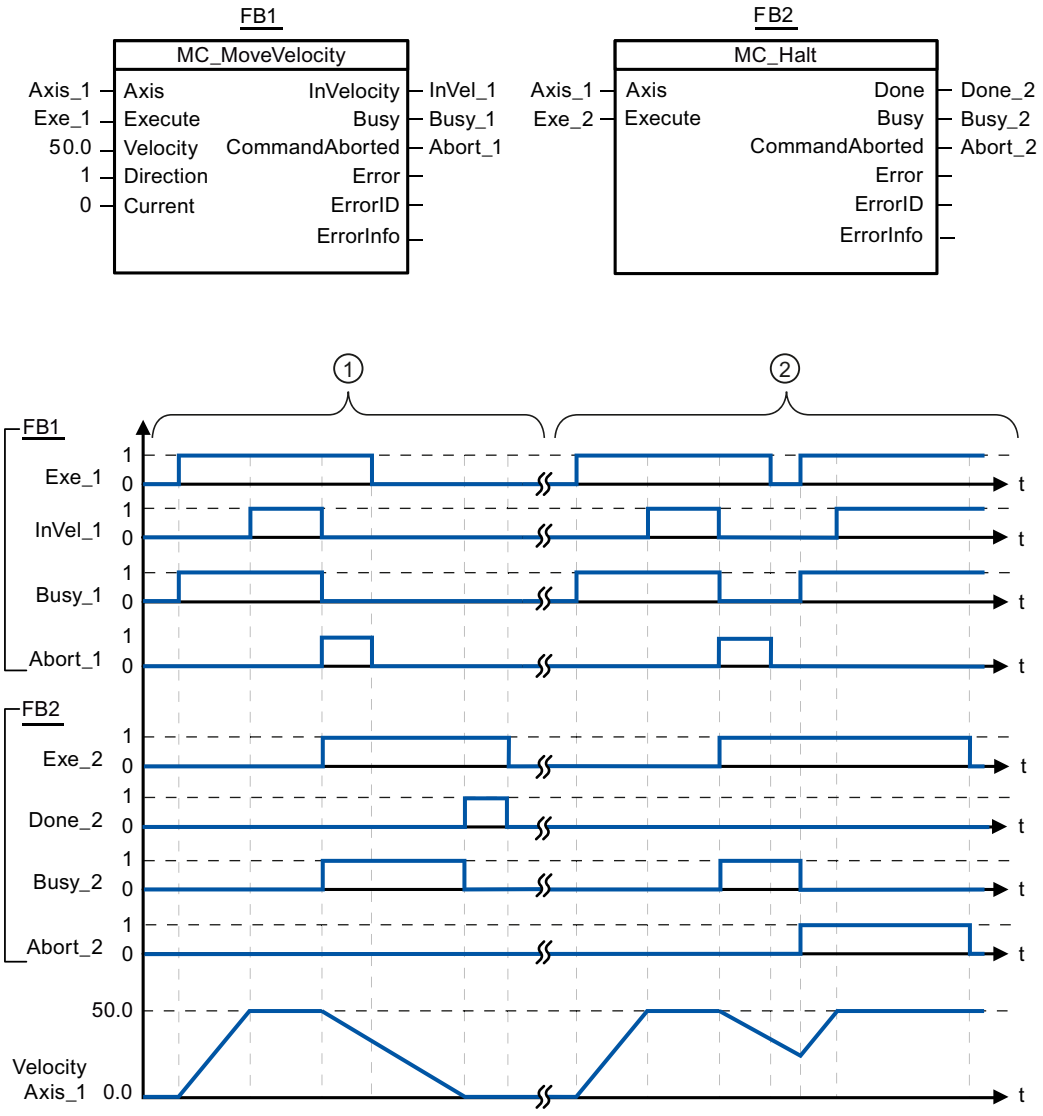
MC_MoveJog: Move axes in jog mode V1...3 (Page 3432)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3435)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3437)

MC_Halt: Function chart V1...3

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is braked by an MC_Halt command until it comes to a standstill. The axis standstill is signaled via "Done_2".
②	While an MC_Halt command is braking the axis, this command is aborted by another motion command. The abort is signaled via "Abort_2".

See also

MC_Halt: Halt axes V1...3 (Page 3419)

MC_MoveAbsolute

MC_MoveAbsolute: Absolute positioning of axes V1...3

Description

The "MC_MoveAbsolute" Motion Control instruction starts an axis positioning motion to move it to an absolute position.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.
- The axis is homed.

Override response

The MC_MoveAbsolute command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveAbsolute command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

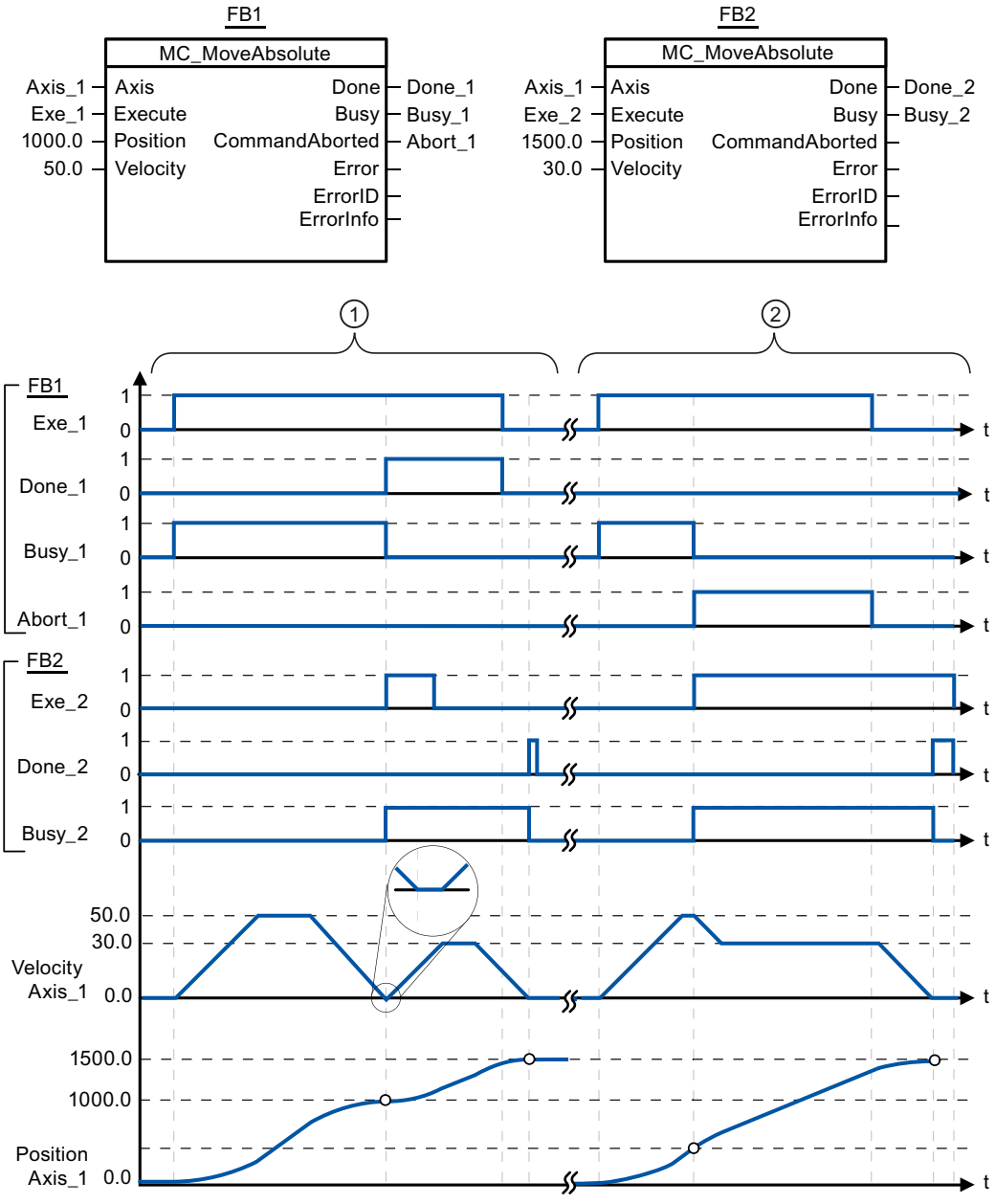
Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	Absolute target position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the target position to be approached. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Absolute target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7555) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7555) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 7440)
S7-1200 Motion Control as of V4 (Page 3374)
ErrorIDs and ErrorInfos (Page 7555)
MC_MoveAbsolute: Function chart V1...3 (Page 3424)
MC_Power: Enable, disable axis V1...3 (Page 3410)
MC_Reset: Acknowledge fault V1...3 (Page 3414)
MC_Home: Home axes, set reference point V1...3 (Page 3415)
MC_Halt: Halt axes V1...3 (Page 3419)
MC_MoveRelative: Relative positioning of axes V1...3 (Page 3425)
MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 3428)
MC_MoveJog: Move axes in jog mode V1...3 (Page 3432)
MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3435)
MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3437)

MC_MoveAbsolute: Function chart V1...3

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An axis is moved to absolute position 1000.0 with an MC_MoveAbsolute command. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveAbsolute command, with target position 1500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveAbsolute command is aborted by another MC_MoveAbsolute command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 3422)

MC_MoveRelative

MC_MoveRelative: Relative positioning of axes V1...3

Description

The "MC_MoveRelative" Motion Control instruction starts a positioning motion relative to the start position.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveRelative command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveRelative command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command

- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Distance	INPUT	REAL	0.0	Travel distance for the positioning operation Limit values: $-1.0e^{12} \leq$ Distance $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the distance to be traveled. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7555) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7555) for parameter "ErrorID"	

See also

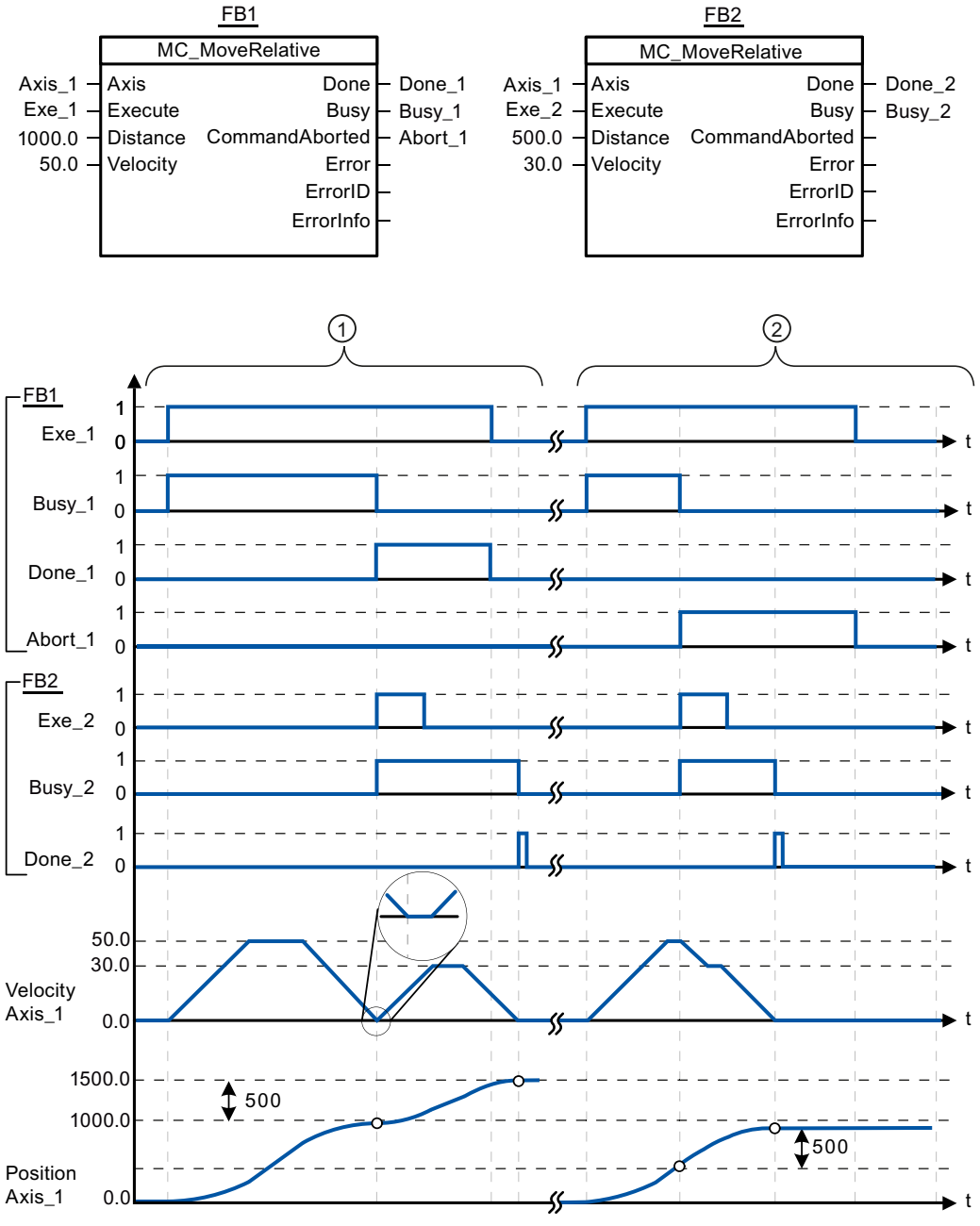
Overview of the Motion Control statements (Page 7440)
S7-1200 Motion Control as of V4 (Page 3374)
ErrorIDs and ErrorInfos (Page 7555)
MC_MoveRelative: Function chart V1...3 (Page 3427)
MC_Power: Enable, disable axis V1...3 (Page 3410)
MC_Reset: Acknowledge fault V1...3 (Page 3414)
MC_Home: Home axes, set reference point V1...3 (Page 3415)
MC_Halt: Halt axes V1...3 (Page 3419)
MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 3422)
MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 3428)
MC_MoveJog: Move axes in jog mode V1...3 (Page 3432)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3435)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3437)

MC_MoveRelative: Function chart V1...3

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	The axis is moved by an MC_MoveRelative command by the distance ("Distance") 1000.0. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveRelative command, with travel distance 500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveRelative command is aborted by another MC_MoveRelative command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity by the new distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveRelative: Relative positioning of axes V1...3 (Page 3425)

MC_MoveVelocity

MC_MoveVelocity: Move axes at preset rotational speed V1...3

Description

Motion control instruction "MC_MoveVelocity" moves the axis constantly at the specified velocity.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

MC_MoveVelocity can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveVelocity command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Velocity	INPUT	REAL	10.0	Velocity specification for axis motion Limit values: Start/stop velocity ≤ |Velocity ≤ maximum velocity (Velocity = 0.0 is permitted)	
Direction	INPUT	INT	0	Direction specification	
				0	Direction of rotation corresponds to the sign of the value in parameter "Velocity"
				1	Positive direction of rotation (The sign of the value in parameter "Velocity" is ignored)
				2	Negative direction of rotation (The sign of the value in parameter "Velocity" is ignored)
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	"Maintain current velocity" is deactivated. The values of parameters "Velocity" and "Direction" are used.
				TRUE	"Maintain current velocity" is activated. The values in parameters "Velocity" and "Direction" are not taken into account. When the axis resumes motion at the current velocity, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUTPUT	BOOL	FALSE	TRUE	• "Current" = FALSE: The velocity specified in parameter "Velocity" was reached. • "Current" = TRUE: The axis travels at the current velocity at the start time.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.

Parameter	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7555) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7555) for parameter "ErrorID"	

Behavior with zero setpoint velocity (Velocity = 0.0)

An MC_MoveVelocity command with "Velocity" = 0.0 (such as an MC_Halt command) aborts active motion commands and stops the axis with the configured deceleration.

When the axis comes to a standstill, output parameter "InVelocity" indicates TRUE for at least one program cycle.

"Busy" indicates the value TRUE during the deceleration process and changes to FALSE together with "InVelocity". If parameter "Execute" = TRUE is set, "InVelocity" and "Busy" are latched.

When the "MC_MoveVelocity" command is started, status bit "SpeedCommand" is set in the technology object. Status bit "ConstantVelocity" is set upon axis standstill. Both bits are adapted to the new situation when a new motion command is started.

See also

Overview of the Motion Control statements (Page 7440)

S7-1200 Motion Control as of V4 (Page 3374)

ErrorIDs and ErrorInfos (Page 7555)

MC_MoveVelocity: Function chart V1...3 (Page 3431)

MC_Power: Enable, disable axis V1...3 (Page 3410)

MC_Reset: Acknowledge fault V1...3 (Page 3414)

MC_Home: Home axes, set reference point V1...3 (Page 3415)

MC_Halt: Halt axes V1...3 (Page 3419)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 3422)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 3425)

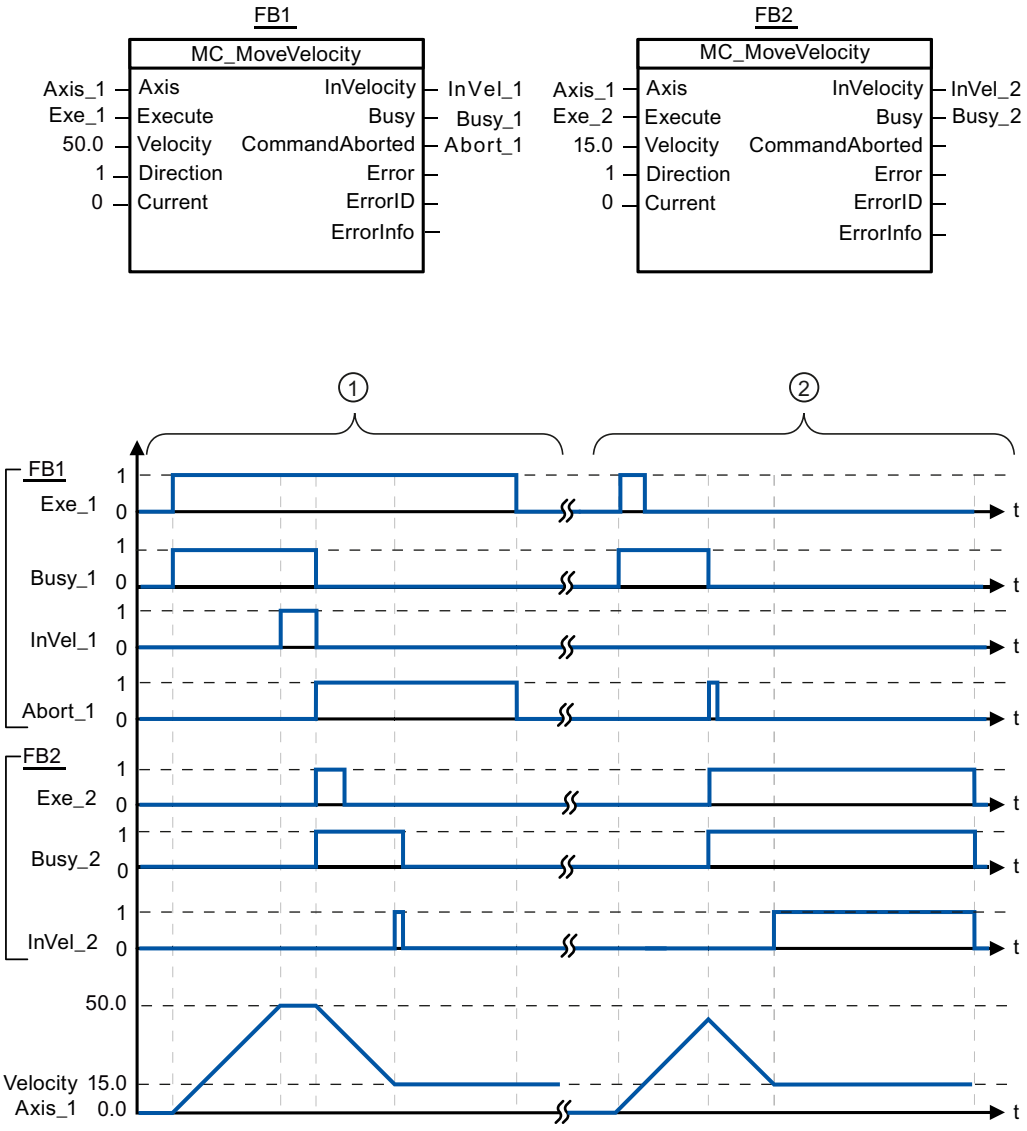
MC_MoveJog: Move axes in jog mode V1...3 (Page 3432)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3435)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3437)

MC_MoveVelocity: Function chart V1...3

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An active MC_MoveVelocity command signals via "InVel_1" that its target velocity has been reached. It is then aborted by another MC_MoveVelocity command. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.
②	An active MC_MoveVelocity command is aborted by another MC_MoveVelocity command prior to reaching its target velocity. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.

See also

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 3428)

MC_MoveJog

MC_MoveJog: Move axes in jog mode V1...3

Description

Motion control instruction "MC_MoveJog" moves the axis constantly at the specified velocity in jog mode. You use this Motion Control instruction, for example, for testing and commissioning purposes.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveJog command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveJog command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".	
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".	
If both parameters are simultaneously TRUE, the axis stops with the configured deceleration. An error is indicated in parameters "Error", "ErrorID", and "ErrorInfo".					
Velocity	INPUT	REAL	10.0	Preset velocity for jog mode	
				Limit values, instruction version V1.0: Start/stop velocity ≤ |Velocity ≤ maximum velocity	
				Limits, instruction version V2.0: Start/stop velocity ≤ velocity ≤ maximum velocity	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity specified in parameter "Velocity" was reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7555) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7555) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 7440)

S7-1200 Motion Control as of V4 (Page 3374)

ErrorIDs and ErrorInfos (Page 7555)

MC_MoveJog: Function chart V1...3 (Page 3434)

MC_Power: Enable, disable axis V1...3 (Page 3410)

MC_Reset: Acknowledge fault V1...3 (Page 3414)

MC_Home: Home axes, set reference point V1...3 (Page 3415)

MC_Halt: Halt axes V1...3 (Page 3419)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 3422)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 3425)

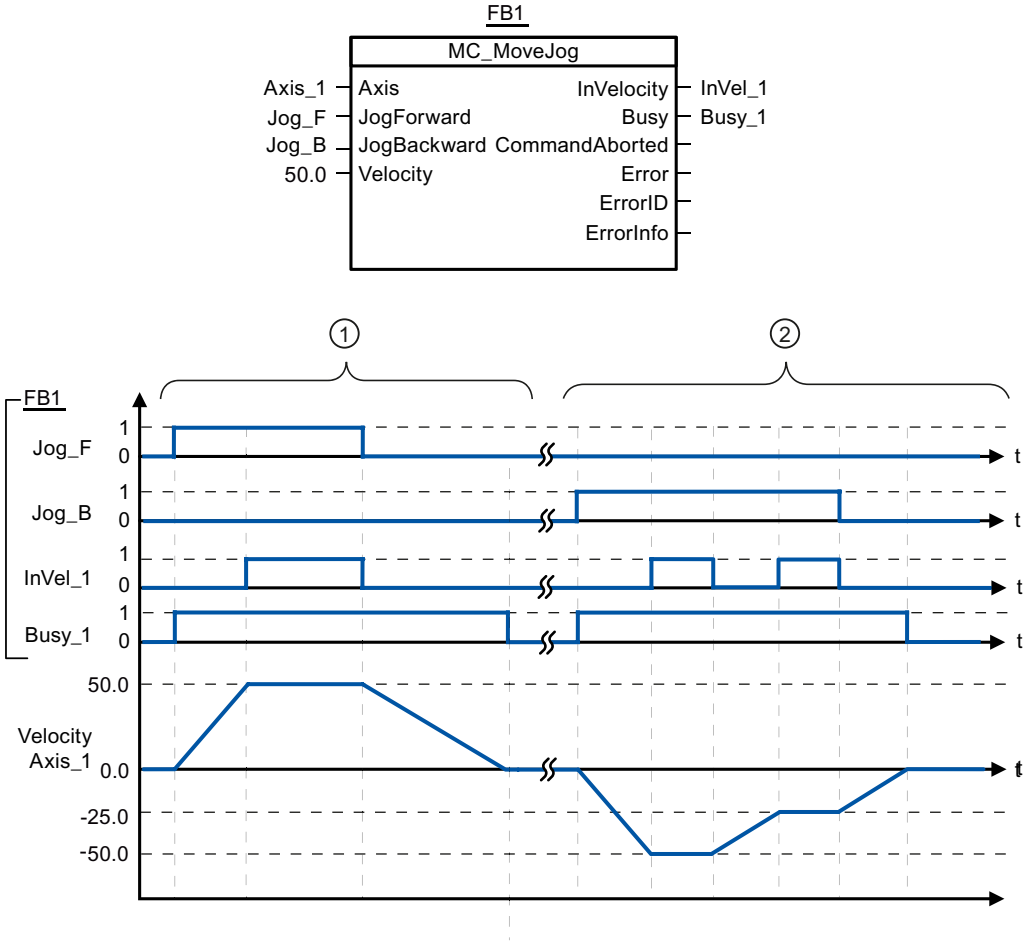
MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 3428)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3435)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3437)

MC_MoveJog: Function chart V1...3

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is moved in the positive direction in jog mode via "Jog_F". When the target velocity 50.0 is reached, this is signaled via "InVelo_1". After " Jog_F" is reset, the axis brakes again until it comes to a standstill.
②	The axis is moved in the negative direction in jog mode via "Jog_B". When the target velocity 50.0 is reached, this is signaled via "InVelo_1". After " Jog_B" is reset, the axis brakes again until it comes to a standstill.

See also

MC_MoveJog: Move axes in jog mode V1...3 (Page 3432)

MC_CommandTable

MC_CommandTable: Run axis commands as motion sequence V2...3

Description

The Motion Control instruction "MC_CommandTable" combines multiple individual axis control commands in one movement sequence.

Requirements

- The axis technology object has been inserted in V2 and correctly configured.
- The command table technology object has been inserted and correctly configured.
- The axis is enabled.

Override response

The MC_CommandTable command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_CommandTable command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The active Motion Control command is cancelled by the start of the first "Positioning Relative", "Positioning Absolute", "Velocity set point" or "Halt" command.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
CommandTable	INPUT	TO_CommandTable_1	-	Command table technology object	
Execute	INPUT	BOOL	FALSE	Command table start with positive edge	
StartStep	INPUT	INT	1	Defines the step at which the execution of the command table should begin Limit values: 1 ≤ StartStep ≤ EndStep	
EndStep	INPUT	INT	32	Defines the step up to which the execution of command table should take place Limit values: StartStep ≤ EndStep ≤ 32	
Done	OUTPUT	BOOL	FALSE	TRUE	Command table has been successfully executed
Busy	OUTPU	BOOL	FALSE	TRUE	The command table is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The command table was cancelled by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command table. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7555) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7555) for parameter "ErrorID"	
CurrentStep	OUTPUT	INT	0	Step in command table currently being executed	
StepCode	OUTPUT	WORD	16#0000	User-defined numerical value / bit pattern of the step currently being executed	

See also

Overview of the Motion Control statements (Page 7440)

S7-1200 Motion Control as of V4 (Page 3374)

ErrorIDs and ErrorInfos (Page 7555)

MC_Power: Enable, disable axis V1...3 (Page 3410)

MC_Reset: Acknowledge fault V1...3 (Page 3414)

MC_Home: Home axes, set reference point V1...3 (Page 3415)

MC_Halt: Halt axes V1...3 (Page 3419)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 3422)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 3425)

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 3428)

MC_MoveJog: Move axes in jog mode V1...3 (Page 3432)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3437)

MC_ChangeDynamic

MC_ChangeDynamic: Change dynamic settings of axis V2...3

Description

Motion Control instruction "MC_ChangeDynamic" allows you to change the following settings of the axis:

- Change the ramp-up time (acceleration) value
- Change the ramp-down time (deceleration) value
- Change the emergency stop ramp-down time (emergency stop deceleration) value
- Change the smoothing time (jerk) value

For the effectiveness of the change, refer to the description of the tag (Page 7568).

Requirements

- The axis technology object has been inserted in Version V2.
- The axis technology object has been configured correctly.

Override response

A MC_ChangeDynamic command cannot be aborted by any other Motion Control command.

A new MC_ChangeDynamic command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
ChangeRampUp	INPUT	BOOL	FALSE	TRUE	Change ramp-up time in line with input parameter "RampUpTime"
RampUpTime	INPUT	REAL	5.00	Time (in seconds) to accelerate axis from standstill to configured maximum velocity without jerk limit. The change will influence the tag <Axis name>. Config.DynamicDefaults.Acceleration. For the effectiveness of the change, refer to the description of this tag.	
ChangeRampDown	INPUT	BOOL	FALSE	TRUE	Change ramp-down time in line with input parameter "RampDownTime"
RampDownTime	INPUT	REAL	5.00	Time (in seconds) to decelerate axis from the configured maximum velocity to standstill without jerk limiter. The change will influence the tag <Axis name>. Config.DynamicDefaults.Deceleration . For the effectiveness of the change, refer to the description of this tag.	

Parameter	Declaration	Data type	Default value	Description	
ChangeEmergency	INPUT	BOOL	FALSE	TRUE	Change emergency stop ramp-down time in line with input parameter "EmergencyRampTime"
EmergencyRampTime	INPUT	REAL	2.00	Time (in seconds) to decelerate the axis from configured maximum velocity to standstill without jerk limiter in emergency stop mode. The change will influence the tag <Axis name>. Config.DynamicDefaults.EmergencyDeceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeJerkTime	INPUT	BOOL	FALSE	TRUE	Change smoothing time according to the input parameter "JerkTime"
JerkTime	INPUT	REAL	0.25	Smoothing time (in seconds) used for the axis acceleration and deceleration ramps The change will influence the tag <Axis name>. Config.DynamicDefaults.Jerk . For the effectiveness of the change, refer to the description of this tag.	
Done	OUTPUT	BOOL	FALSE	TRUE	The changed values have been written to the technology data block. The description of the tags will show when the change becomes effective.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 7555) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 7555) for parameter "ErrorID"	

Note

At the input parameters "RampUpTime", "RampDownTime", "EmergencyRampTime" and "JerkTime", values can be entered which exceed the admissible limits of the resulting parameters: "Acceleration", "Deceleration", "Emergency stop deceleration" and "Jerk".

Please note the equations and limits in "Axis technology object" -> "Configuring the technology object" -> "Dynamics" and ensure that the values you input are within the valid range.

See also

Overview of the Motion Control statements (Page 7440)

S7-1200 Motion Control as of V4 (Page 3374)

ErrorIDs and ErrorInfos (Page 7555)

Tag of the axis technology object V1...3 (Page 7568)

MC_Power: Enable, disable axis V1...3 (Page 3410)

MC_Reset: Acknowledge fault V1...3 (Page 3414)

MC_Home: Home axes, set reference point V1...3 (Page 3415)

MC_Halt: Halt axes V1...3 (Page 3419)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 3422)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 3425)

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 3428)
MC_MoveJog: Move axes in jog mode V1...3 (Page 3432)
MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3435)

11.6.4.2 High-speed counters

CTRL_HSC: Control high-speed counters

Parameter

Parameters	Declaration	Data type	Memory area	Description
EN	INPUT	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	OUTPUT	BOOL	I, Q, M, D, L	Enable output
HSC	INPUT	HW_HSC	I, Q, M or constant	Hardware address of the high-speed counter (HW ID)
DIR	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new count direction (see NEW_DIR)
CV	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new count value (see NEW_CV)
RV	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new reference value (see NEW_RV)
PERIOD	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new period of a frequency measurement (see NEW_PERIOD)
NEW_DIR	INPUT	INT	I, Q, M, D, L or constant	Count direction loaded when DIR = TRUE.
NEW_CV	INPUT	DINT	I, Q, M, D, L or constant	Count value loaded when CV = TRUE.
NEW_RV	INPUT	DINT	I, Q, M, D, L or constant	Reference value loaded when RV = TRUE.
NEW_PERIOD	INPUT	INT	I, Q, M, D, L or constant	Period of the frequency measurement loaded when PERIOD = TRUE.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Processing status
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status of the operation

Description

With the "Control high-speed counter" instruction, you can make parameter settings and control the high-speed counters supported by the CPU by loading new values into the counter. Execution of the instruction requires that a high-speed counter to be controlled is enabled. You cannot execute multiple "Control high-speed counter" instructions simultaneously in the program for a given high-speed counter.

You can load the following parameter values into a high-speed counter using the "Control high-speed counter" instruction:

- Count direction (NEW_DIR): The count direction defines whether a high-speed counter counts up or down. The count direction is defined by the following values at the NEW_DIR input: 1 = up, -1 = down.
  A change to the count direction with the "Control high-speed counter" instruction is only possible when direction control is set in the parameters by the program. The count direction specified at the NEW_DIR input is loaded into a high-speed counter when the bit at the DIR input is set.
- Count value (NEW_CV): The count value is the initial value at which a high-speed counter starts counting. The count value can be in the range -2147483648 to 2147483647.
  The count value specified at the NEW_CV input is loaded into a high-speed counter when the bit at the CV input is set.
- Reference value (NEW_RV): You can compare the reference value with the current counter value to trigger an alarm. Similar to the counter value, the reference value can be in the range -2147483648 to 2147483647.
  The reference value specified at the NEW_RV input is loaded into a high-speed counter when the bit at the RV input is set.
- Period of the frequency measurement (NEW_PERIOD): The period of the frequency measurement is specified by the following values at the NEW_PERIOD input: 10 = 0.01s, 100 = 0.1s, 1000 = 1s.
  The time period can be updated if the "Measure frequency" function for the specified high-speed counter is configured. The time period specified at the NEW_PERIOD input is loaded into a high-speed counter when the bit at the PERIOD input is set.

The "Control high-speed counter" instruction is only executed if the signal state at the EN input is "1". As long as the operation is executing, the bit at the BUSY output is set. Once the operation has executed completely, the bit at the BUSY output is reset.

The ENO enable output is set only when the EN enable input has signal state "1" and no errors occur during execution of the operation.

When inserting the "Control high-speed counter" instruction, an instance data block is created in which the operation data is saved.

STATUS parameter

At the STATUS output, you can query whether errors occurred during execution of the "Control high-speed counter" instruction. The following table shows the meaning of the values output at the STATUS output:

Error code (hexadecimal)	Description
0	No error
80A1	Hardware identifier of the high-speed counter invalid

Error code (hexadecimal)	Description
80B1	Count direction (NEW_DIR) invalid
80B2	Count value (NEW_CV) invalid
80B3	Reference value (NEW_RV) invalid
80B4	Period of the frequency measurement (NEW_PERIOD) invalid
80C0	Multiple access to the high-speed counter
80D0	The high-speed counter (HSC) is not enabled in the CPU hardware configuration.

CTRL_HSC_EXT: Control high-speed counter (extended)

Parameters

Parameters	Declaration	Data type	Memory area	Description
EN	INPUT	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	OUTPUT	BOOL	I, Q, M, D, L	Enable output
HSC	INPUT	HW_HSC	I, Q, M or constant	Hardware address of the high-speed counter (HW ID)
CTRL	INOUT	VARIANT	M, D	Use of a system data type (SDT)
DONE	OUTPUT	BOOL	I, Q, M, D, L	Feedback after successful processing instruction
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Processing status
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Feedback for faulty processing of the instruction
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status of the operation

Description

With the "Control high-speed counter (extended)" instruction, you can control and assign parameters to the high-speed counters supported by the CPU via the software by loading new values into the counters. Execution of the instruction requires that a high-speed counter to be controlled is enabled. You cannot execute multiple "Control high-speed counter (extended)" instructions simultaneously in the program for a given high-speed counter.

The "Control high-speed counter (extended)" instruction is only executed if the signal state at the EN input is "1". As long as the operation is executing, the bit at the BUSY output is set. Once the operation has executed completely, the bit at the BUSY output is reset.

The ENO enable output is set only when the EN enable input has signal state "1" and no errors occur during execution of the operation.

When the "Control high-speed counter (extended)" instruction is inserted, an instance data block is created in which the operation data is saved.

Using the system data type HSC_Period

The "Control high-speed counter (extended)" instruction supports the system data type SDT 381 "HSC_Period" for the period measurement.

Byte	Parameters	Data type	Description
0 ... 3	ElapsedTime	UDINT	Time between the rising edges of Edge_Count
4 ... 7	EdgeCount	UDINT	Number of the rising edges within the Elapsed_Time. If Edge_Count = 0, the Elapsed_Time is the time since the last rising edge.
8.0	EnHSC	BOOL	Use as an enable input via gate control: • FALSE: Measurement stopped • TRUE: Measurement enabled
8.6	EnPeriod	BOOL	Update of the period • FALSE: No update • TRUE: Update period
10 ... 11	NewPeriod	INT	Interval of the period measurement in milliseconds. Permissible values are 10, 100 and 1000.

STATUS parameter

At the STATUS output, you can query whether errors occurred during execution of the "Control high-speed counter (extended)" instruction. The following table shows the meaning of the values output at the STATUS output:

Error code (hexadecimal)	Description
0	No error
80A1	Hardware identifier of the high-speed counter invalid
80C0	Multiple access to the high-speed counter
80D0	The high-speed counter (HSC) is not enabled in the CPU hardware configuration.

11.6.4.3 PID Control

PID_Compact

New features of PID_Compact

PID_Compact V2.2

- **Use with S7-1200**
  As of PID_Compact V2.2, the instruction with V2 functionality can also be used on S7-1200 with firmware version 4.0 or higher.

PID_Compact V2.0

- **Reaction to error**
  The reaction to error has been completely overhauled. PID_Compact now reacts in a more fault-tolerant manner in the default setting. This reaction is set when copying PID_Compact V1.X from an S7-1200 CPU to an S7-1500 CPU.

  Notice

  Your system may be damaged.

  If you use the default setting, PID_Compact remains in automatic mode when the process value limits are exceeded. This may damage your system.

  It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

  The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. Use ErrorAck to acknowledge the errors and warnings without restarting the controller or clearing the integral action. Switching operating modes no longer clears errors that are no longer pending.
  You can configure the reaction to error with SetSubstituteOutput and ActivateRecoverMode.
- **Substitute output value**
  You can configure a substitute output value that is to be output if an error occurs.
- **Switching the operating mode**
  You specify the operating mode at the Mode in/out parameter and use a rising edge at ModeActivate to start the operating mode. The sRet.i_Mode tag has been omitted.
- **Multi-instance capability**
  You can call up PID_Compact as multi-instance DB. No technology object is created in this case and no parameter assignment interface or commissioning interface is available. You must assign parameters for PID_Compact directly in the multi-instance DB and commission it via a watch table.
- **Startup characteristics**
  The operating mode specified at the Mode parameter is also started on a falling edge at Reset and during a CPU cold restart, if RunModeByStartup = TRUE.
- **ENO characteristics**
  ENO is set depending on the operating mode.
  If State = 0, then ENO = FALSE.
  If State ≠ 0, then ENO = TRUE.
- **Setpoint value specification during tuning**
  You configure the permitted fluctuation of the setpoint during tuning at the CancelTuningLevel tag.
- **Value range for output value limits**
  The value 0.0 no longer has to fall within the output value limits.
- **Pre-assigning the integral action**
  Using the tags IntegralResetMode and OverwriteInitialOutputValue, you can determine the pre-assignment of the integral action when switching from "Inactive" operating mode to "Automatic mode".
- **Switching a disturbance variable on**
  You can switch a disturbance variable on at the Disturbance parameter.

- **Default value of PID parameters**
  The following default settings have been changed:
  - Proportional action weighting (PWeighting) from 0.0 to 1.0
  - Derivative action weighting (DWeighting) from 0.0 to 1.0
  - Coefficient for derivative delay (TdFiltRatio) from 0.0 to 0.2
- **Renaming tags**
  The static tags have been given new names that are compatible with PID_3Step.

PID_Compact V1.2

- **Manual mode on CPU startup**
  If ManualEnable = TRUE when the CPU starts, PID_Compact starts in manual mode. A rising edge at ManualEnable is not necessary.
- **Pretuning**
  If the CPU is switched off during pretuning, pretuning starts again when the CPU is switched back on.

PID_Compact V1.1

- **Manual mode on CPU startup**
  When the CPU starts up, PID_Compact only switches to manual mode with a rising edge at ManualEnable. Without rising edge, PID_Compact starts in the last operating mode in which ManualEnable was FALSE.
- **Reaction to reset**
  A rising edge at Reset resets the errors and warnings and clears the integral action. A falling edge at Reset triggers a switchover to the most recently active operating mode.
- **Default of process value high limit**
  The default value of r_Pv_Hlm has been changed to 120.0.
- **Monitoring the sampling time**
  - An error is no longer output when the current sampling time is >= 1.5 x current mean value or when the current sampling time is <= 0.5 x current mean value. The sampling time may deviate much more in automatic mode.
  - PID_Compact is compatible with FW, V2.0 or higher.
- **Access to tags**
  The following tags can now be used in the user program.
  - i_Event_SUT
  - i_Event_TIR
  - r_Ctrl_Ioutv
- **Troubleshooting**
  PID_Compact now outputs the correct pulses when the shortest ON time is not equal to the shortest OFF time.

Compatibility with CPU and FW

The following table shows which version of PID_Compact can be used on which CPU.

CPU	FW	PID_Compact
S7-1200	≥ V4.x	V2.2 V1.2
S7-1200	≥ V3.X	V1.2 V1.1
S7-1200	≥ V2.X	V1.2 V1.1
S7-1200	≥ V1.X	V1.0
S7-1500	≥ V1.5	V2.2 V2.1 V2.0
S7-1500	≥ V1.1	V2.1 V2.0
S7-1500	≥ V1.0	V2.0

CPU processing time and memory requirement PID_Compact V2.x

CPU processing time

Typical CPU processing times of the PID_Compact technology object as of Version V2.0, depending on CPU type.

CPU	Typ. CPU processing time PID_Compact V2.x
CPU 1211C ≥ V4.0	300 µs
CPU 1215C ≥ V4.0	300 µs
CPU 1217C ≥ V4.0	300 µs
CPU 1505S ≥ V1.0	45 µs
CPU 1510SP-1 PN ≥ V1.6	85 µs
CPU 1511-1 PN ≥ V1.5	85 µs
CPU 1512SP-1 PN ≥ V1.6	85 µs
CPU 1516-3 PN/DP ≥ V1.5	50 µs
CPU 1518-4 PN/DP ≥ V1.5	4 µs

Memory requirement

Memory requirement of an instance DB of the PID_Compact technology object as of Version V2.0.

	Memory requirement of the instance DB of PID_Compact V2.x
Load memory requirement	Approx. 12000 bytes

Total work memory requirement	788 bytes
Retentive work memory requirement	44 bytes

PID_Compact V2

Description of PID_Compact V2

Description

The PID_Compact instruction provides a PID controller with integrated tuning for actuators with proportional action.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Substitute output value with error monitoring

For a more detailed description of the operating modes, see the State parameter.

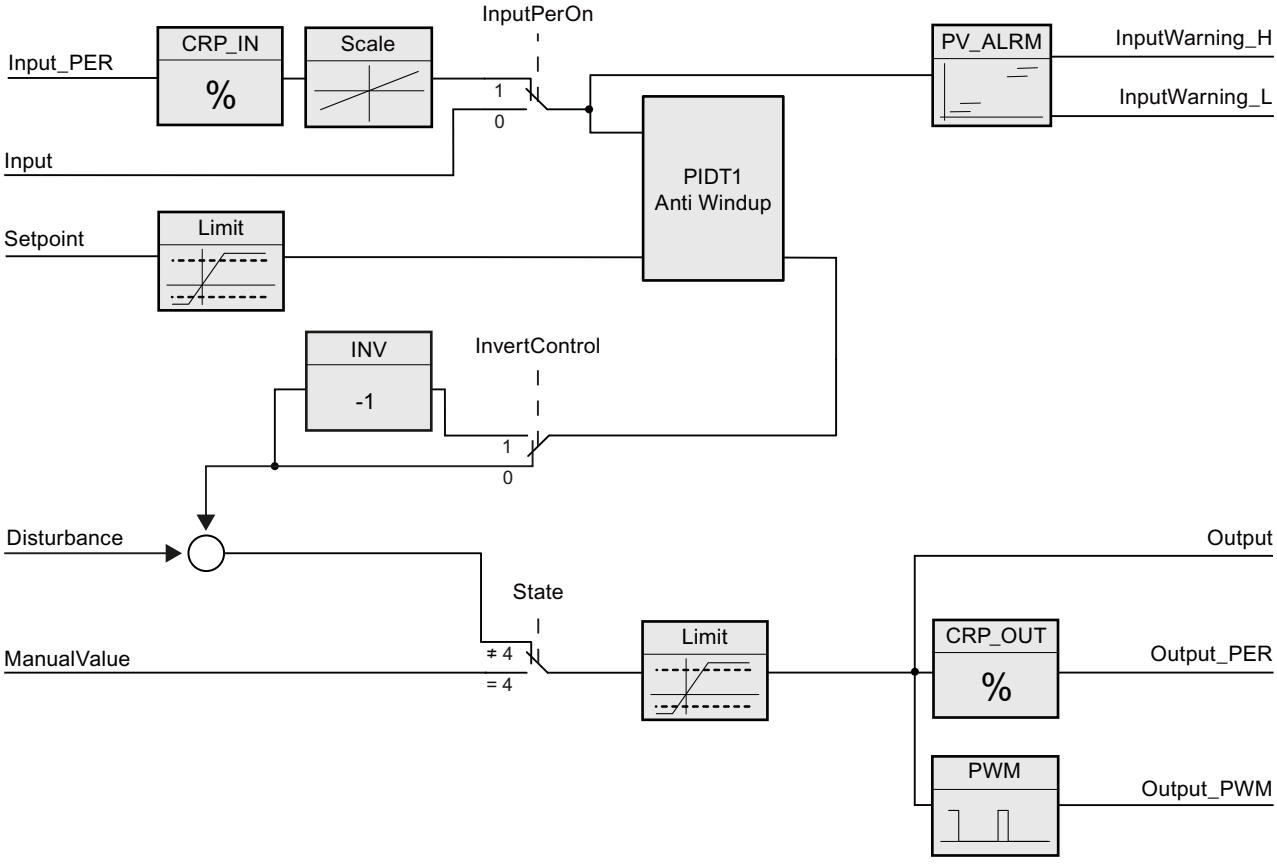
PID algorithm

PID_Compact is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation:

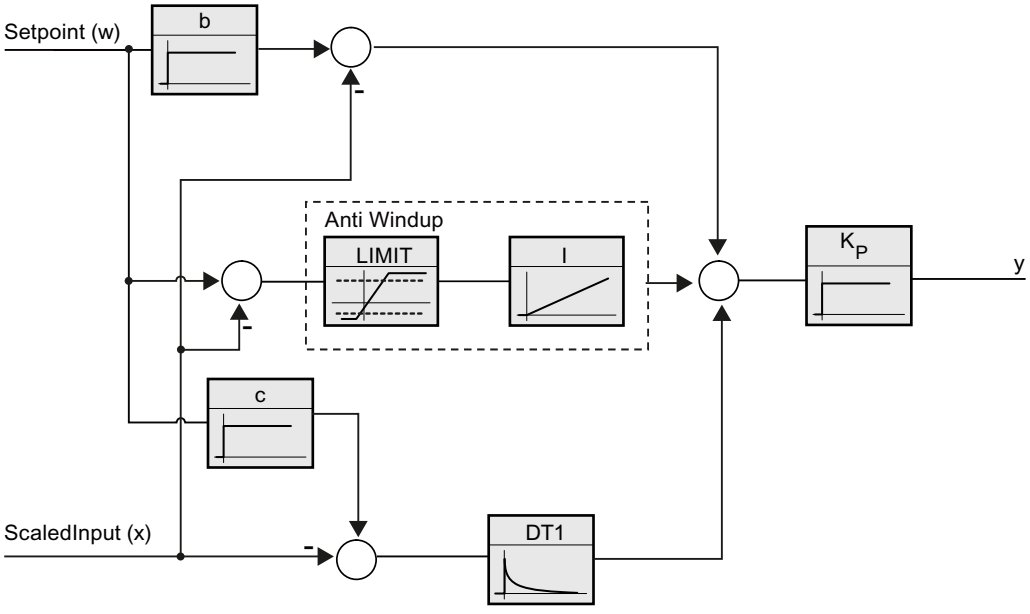
$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
T_D	Derivative action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
c	Derivative action weighting

Block diagram of PID_Compact



Block diagram of PIDT1 with anti-windup



Call

PID_Compact is called in the constant time scale of a cycle interrupt OB.

If you call PID_Compact as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_Compact directly in the multi-instance DB and commission it via a watch table.

Download to device

The actual values of retentive variables are only updated when you download PID_Compact completely.

Downloading technology objects to device (Page 7204)

Startup

When the CPU starts up, PID_Compact starts in the operating mode that is saved in the Mode in/out parameter. To switch to "Inactive" operating mode during startup, set RunModeByStartup = FALSE.

Reaction to error

In automatic mode and during commissioning, the reaction to error depends on the SetSubstituteOutput and ActivateRecoverMode variables. In manual mode, the reaction is independent of SetSubstituteOutput and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the reaction additionally depends on the error that occurred.

SetSubstituteOutput	ActivateRecoverMode	Configuration editor > output value > Set Output to	Reaction
Not relevant	FALSE	Zero (inactive)	Switch to "Inactive" mode (State = 0) The value 0.0 0 is transferred to the actuator.
FALSE	TRUE	Current output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The current output value is transferred to the actuator while the error is pending.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The value at SubstituteOutput is transferred to the actuator while the error is pending.

In manual mode, PID_Compact uses ManualValue as output value, unless ManualValue is invalid. If ManualValue is invalid, SubstituteOutput is used. If ManualValue and SubstituteOutput are invalid, Config.OutputLowerLimit is used.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.

PID_Compact V2 mode of operation

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warning = 0040h), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID_Compact automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID_Compact checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. Output, ManualValue, and SubstituteOutput are limited to these values. The output value limits must match the control logic.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0%
Output_PER	-100.0 to 100.0%
Output_PWM	0.0 to 100.0%

Rule:

OutputUpperLimit > OutputLowerLimit

Substitute output value

In the event of an error, PID_Compact can output a substitute output value that you define at the tag SubstituteOutput. The substitute output value must be within the output value limits.

Monitoring signal validity

The values of the following parameters are monitored for validity when used:

- Setpoint
- Input

- Input_PER
- Disturbance
- ManualValue
- SubstituteOutput
- Output
- Output_PER
- Output_PWM

Monitoring of the sampling time PID_Compact

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_Compact instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (Error = 0800h).

The error occurs during tuning if:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

The error occurs in automatic mode if:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID_Compact in OB1. You must then accept a lower control quality due to the deviating sampling time.

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

If you use Output_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The cycle time should be at least 10 times the PID algorithm sampling time.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_Compact does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

Input parameters of PID_Compact V2

Table 11-81

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A tag of the user program is used as the source of the process value. If you are using the Input parameter, then Config.InputPerOn = FALSE must be set.
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using the Input_PER parameter, then Config.InputPerOn = TRUE must be set.
Disturbance	REAL	0.0	Disturbance variable or precontrol value
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge activates "manual mode", while State = 4, Mode remain unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog. • A TRUE -> FALSE edge activates the operating mode that is specified by Mode. We recommend that you change the operating mode using ModeActivate only.
ManualValue	REAL	0.0	Manual value This value is used as the output value in manual mode. Values from Config.OutputLowerLimit to Config.OutputUpperLimit are permitted.
ErrorAck	BOOL	FALSE	• FALSE -> TRUE edge ErrorBits and Warning are reset.
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Switch to "Inactive" mode – ErrorBits and Warnings are reset. – Integral action is cleared (PID parameters are retained) • As long as Reset = TRUE, PID_Compact remains in "Inactive" mode (State = 0). • TRUE -> FALSE edge PID_Compact switches to the operating mode that is saved in the Mode parameter.
ModeActivate	BOOL	FALSE	• FALSE -> TRUE edge PID_Compact switches to the operating mode that is saved in the Mode parameter.

Output parameters of PID_Compact V2

Table 11-82

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
The "Output", "Output_PER", and "Output_PWM" outputs can be used concurrently.			
Output	REAL	0.0	Output value in REAL format
Output_PER	INT	0	Analog output value
Output_PWM	BOOL	FALSE	Pulse-width-modulated output value The output value is formed by by variable On and Off times.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit). The setpoint is limited to Config.SetpointUpperLimit .
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached (Setpoint ≤ Config.SetpointLowerLimit). The setpoint is limited to Config.SetpointLowerLimit .
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 3462) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Substitute output value with error monitoring
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 3466) shows which error messages are pending. ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck.

In/out parameters of PID_Compact V2

Table 11-83

Parameter	Data type	Default	Description
Mode	INT	4	At Mode, specify the operating mode to which PID_Compact is to switch. Options are: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode The operating mode is activated by: • Rising edge at ModeActivate • Falling edge at Reset • Falling edge at ManualEnable • Cold restart of CPU if RunModeByStartup = TRUE Mode is retentive. A detailed description of the operating modes can be found in Parameters State and Mode V2 (Page 3462).

See also

Parameters State and Mode V2 (Page 3462)

Static tags of PID_Compact V2

You must not change variables that are not listed. These are used for internal purposes only.

Table 11-84

Tag	Data type	Default	Description
IntegralResetMode	INT	1	The tag IntegralResetMode determines how PIDCtrl.IntegralSum is pre-assigned when switching from "Inactive" operating mode to "Automatic mode". This setting only works for one cycle. Options are: • IntegralResetMode = 0: Smoothing The value of IntegralSum is pre-assigned so that the switchover is bumpless. • IntegralResetMode = 1: Deleting The value of IntegralSum is deleted. Any control deviation will cause a jump change of the output value. • IntegralResetMode = 2: Holding The value of IntegralSum is not changed. You can define a new value using the user program. • IntegralResetMode = 3: Pre-assigning The value of IntegralSum is automatically pre-assigned so that Output is calculated with reference to the value OverwriteInitialOutputValue. This setting is useful, for example, for an override controller.
OverwriteInitialOutputValue	REAL	0.0	If IntegralResetMode = 3, the value of IntegralSum is automatically pre-assigned so that Output = OverwriteInitialOutputValue in the next cycle.
RunModeByStartup	BOOL	TRUE	Activate operating mode at Mode parameter after CPU restart If RunModeByStartup = TRUE, PID_Compact starts in the operating mode saved in the Mode parameter after CPU startup. If RunModeByStartup = FALSE, PID_Compact remains in "Inactive" mode after CPU startup.
LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., °C, or °F.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
ActivateRecoverMode	BOOL	TRUE	The Tag ActivateRecoverMode V2 (Page 3468) determines the reaction to error.
Warning	DWORD	0	Tag Warning V2 (Page 3469) shows the warnings since Reset = TRUE or ErrorAck =TRUE. Warning is retentive.
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)

Tag	Data type	Default	Description
CurrentSetpoint	REAL	0.0	CurrentSetpoint always displays the current setpoint. This value is frozen during tuning.
CancelTuningLevel	REAL	10.0	Permissible fluctuation of setpoint during tuning. Tuning is not canceled until: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel
SubstituteOutput	REAL	0.0	Substitute output value When the following conditions are met, the substitute output value is used: • An error has occurred in automatic mode. • SetSubstituteOutput = TRUE • ActivateRecoverMode = TRUE
SetSubstituteOutput	BOOL	TRUE	If SetSubstituteOutput = TRUE and ActivateRecoverMode = TRUE, the substitute output value configured is output as long as an error is pending. If SetSubstituteOutput = FALSE and ActivateRecoverMode = TRUE, the actuator remains at the current output value as long as an error is pending. If ActivateRecoverMode = FALSE, SetSubstituteOutput is not effective. If SubstituteOutput is invalid (ErrorBits = 20000h), the substitute output value cannot be output.
Config.InputPerOn	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.InvertControl	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.InputUpperLimit	REAL	120.0	High limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). This pre-assignment ensures that an error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_Compact reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. InputLowerLimit < InputUpperLimit

Tag	Data type	Default	Description
Config.InputUpperWarning	REAL	3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.402822e +38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	High limit of output value For details, see OutputLowerLimit OutputUpperLimit > OutputLowerLimit
Config.OutputLowerLimit	REAL	0.0	Low limit of output value For Output and Output_PER, the range of values from -100.0 to +100.0, including zero, is valid. At -100.0, Output_PER = -27648; at +100.0, Output_PER = 27648. For Output_PWM, the value range 0.0 to +100.0 applies. The output value limits must match the control logic. OutputLowerLimit < OutputUpperLimit
Config.SetpointUpperLimit	REAL	3.402822e+38	High limit of setpoint If you configure SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is used as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit	REAL	-3.402822e +38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime	REAL	0.0	The minimum ON time of the pulse width modulation in seconds is rounded to MinimumOnTime = n×CycleTime.Value
Config.MinimumOffTime	REAL	0.0	The minimum OFF time of the pulse width modulation in seconds is rounded to MinimumOffTime = n×CycleTime.Value

Tag	Data type	Default	Description
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
CycleTime.StartEstimation	BOOL	TRUE	If CycleTime.StartEstimation = TRUE, the automatic determination of the cycle time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If CycleTime.EnEstimation = TRUE, the PID_Compact sampling time is calculated. If CycleTime.EnEstimation = FALSE, the PID_Compact sampling time is not calculated and you need to correct the configuration of CycleTime.Value manually.
CycleTime.EnMonitoring	BOOL	TRUE	If CycleTime.EnMonitoring = FALSE, the PID_Compact sampling time is not monitored. If it is not possible to execute PID_Compact within the sampling time, no error (ErrorBits=0800h) is output and PID_Compact does not switch to "Inactive" mode.
CycleTime.Value	REAL	0.1	PID_Compact sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time [s]
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time [s]
CtrlParamsBackUp.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	1.0	Saved proportional action weighting factor
CtrlParamsBackUp.DWeighting	REAL	1.0	Saved derivative action weighting factor
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm

Tag	Data type	Default	Description
PIDSelfTune.SUT.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If SUT.CalculateParams = TRUE, the parameters for pretuning are recalculated according to these properties. This enables you to change the parameter calculation method without having to repeat controller tuning. SUT.CalculateParams is set to FALSE after the calculation.
PIDSelfTune.SUT.TuneRule	INT	0	Methods used to calculate parameters during pretuning: • SUT.TuneRule = 0: PID according to Chien, Hrones and Reswick • SUT.TuneRule = 1: PI according to Chien, Hrones and Reswick
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning: • State = 0: Initialize pretuning • State = 100: Calculate standard deviation • State = 200: Determine point of inflection • State = 9900: Pretuning successful • State = 1: Pretuning not successful
PIDSelfTune.TIR.RunIn	BOOL	FALSE	With the RunIn tag, you can specify that fine tuning can also be performed without pretuning. • RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If the requirements for pretuning are not met, PID_Compact reacts as when RunIn = TRUE. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_Compact switches to the mode from which tuning was started. • RunIn = TRUE The pretuning is skipped. PID_Compact tries to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. RunIn is set to FALSE after fine tuning.
PIDSelfTune.TIR.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If TIR.CalculateParams = TRUE, the parameters for fine tuning are recalculated according to these properties. This enables you to change the parameter calculation method without having to repeat controller tuning. TIR.CalculateParams is set to FALSE after the calculation.

Tag	Data type	Default	Description
PIDSelfTune.TIR.TuneRule	INT	0	Methods used to calculate parameters during fine tuning: • TIR.TuneRule = 0: PID automatic • TIR.TuneRule = 1: PID rapid • TIR.TuneRule = 2: PID slow • TIR.TuneRule = 3: Ziegler-Nichols PID • TIR.TuneRule = 4: Ziegler-Nichols PI • TIR.TuneRule = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of fine tuning: • State = -100 Fine tuning is not possible. Pretuning will be performed first. • State = 0: Initialize fine tuning • State = 200: Calculate standard deviation • State = 300: Attempt to reach the setpoint • State = 400: Attempt to reach the setpoint with existing PID parameters (if pretuning was successful) • State = 500: Determine oscillation and calculate parameters • State = 9900: Fine tuning successful • State = 1: Fine tuning not successful
PIDCtrl.IntegralSum	REAL	0.0	Current integral action
Retain.CtrlParams.Gain	REAL	1.0	Active proportional gain To invert the control logic, use the Config.InvertControl tag. Negative values at Gain also invert the control logic. We recommend you use only InvertControl to set the control logic. The control logic is also inverted if InvertControl = TRUE and Gain < 0.0. Gain is retentive.
Retain.CtrlParams.Ti	REAL	20.0	• CtrlParams.Ti > 0.0: Active integral action time • CtrlParams.Ti = 0.0: Integral action is deactivated Ti is retentive.
Retain.CtrlParams.Td	REAL	0.0	• CtrlParams.Td > 0.0: Active derivative action time • CtrlParams.Td = 0.0: Derivative action is deactivated Td is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.TdFiltRatio	REAL	0.2	Active derivative delay coefficient The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient • 0.0: Derivative action is effective for one cycle only and therefore almost not effective. • 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant. • > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed. TdFiltRatio is retentive.
Retain.CtrlParams.PWeighting	REAL	1.0	Active proportional action weighting The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Proportional action for setpoint change is fully effective • 0.0: Proportional action for setpoint change is not effective The proportional action is always fully effective when the process value is changed. PWeighting is retentive.
Retain.CtrlParams.DWeighting	REAL	1.0	Active derivative action weighting The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Derivative action is fully effective upon setpoint change • 0.0: Derivative action is not effective upon setpoint change The derivative action is always fully effective when the process value is changed. DWeighting is retentive.
Retain.CtrlParams.Cycle	REAL	1.0	Active sampling time of the PID algorithm CtrlParams.Cycle is calculated during tuning and rounded to an integer multiple of CycleTime.Value. Cycle is retentive.

Note

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller.

See also

Tag ActivateRecoverMode V2 (Page 3468)

Tag Warning V2 (Page 3469)

Downloading technology objects to device (Page 7204)

Changing the PID_Compact V2 interface

The following table shows what has changed in the PID_Compact instruction interface.

PID_Compact V1	PID_Compact V2	Change
Input_PER	Input_PER	Data type from Word to Int
	Disturbance	New
	ErrorAck	New
	ModeActivate	New
Output_PER	Output_PER	Data type from Word to Int
Error	ErrorBits	Renamed
	Error	New
	Mode	New
sb_RunModeByStartup	RunModeByStartup	Function
	IntegralResetMode	
	OverwriteInitialOutputValue	New
	SetSubstituteOutput	New
	CancelTuningLevel	New
	SubstituteOutput	New

The following table shows which variables have been renamed.

PID_Compact V1.x	PID_Compact V2
sb_GetCycleTime	CycleTime.StartEstimation
sb_EnCyclEstimation	CycleTime.EnEstimation
sb_EnCyclMonitoring	CycleTime.EnMonitoring
sb_RunModeByStartup	RunModeByStartup
si_Unit	PhysicalUnit
si_Type	PhysicalQuantity
sd_Warning	Warning
sBackUp.r_Gain	CtrlParamsBackUp.Gain
sBackUp.r_Ti	CtrlParamsBackUp.Ti
sBackUp.r_Td	CtrlParamsBackUp.Td
sBackUp.r_A	CtrlParamsBackUp.TdFiltRatio
sBackUp.r_B	CtrlParamsBackUp.PWeighting
sBackUp.r_C	CtrlParamsBackUp.DWeighting
sBackUp.r_Cycle	CtrlParamsBackUp.Cycle
sPid_Calc.r_Cycle	CycleTime.Value

PID_Compact V1.x	PID_Compact V2
sPid_Calc.b_RunIn	PIDSelfTune.TIR.RunIn
sPid_Calc.b_CalcParamSUT	PIDSelfTune.SUT.CalculateParams
sPid_Calc.b_CalcParamTIR	PIDSelfTune.TIR.CalculateParams
sPid_Calc.i_CtrlTypeSUT	PIDSelfTune.SUT.TuneRule
sPid_Calc.i_CtrlTypeTIR	PIDSelfTune.TIR.TuneRule
sPid_Calc.r_Progress	Progress
sPid_Cmpt.r_Sp_Hlm	Config.SetpointUpperLimit
sPid_Cmpt.r_Sp_Llm	Config.SetpointLowerLimit
sPid_Cmpt.r_Pv_Norm_IN_1	Config.InputScaling.LowerPointIn
sPid_Cmpt.r_Pv_Norm_IN_2	Config.InputScaling.UpperPointIn
sPid_Cmpt.r_Pv_Norm_OUT_1	Config.InputScaling.LowerPointOut
sPid_Cmpt.r_Pv_Norm_OUT_2	Config.InputScaling.UpperPointOut
sPid_Cmpt.r_Lmn_Hlm	Config.OutputUpperLimit
sPid_Cmpt.r_Lmn_Llm	Config.OutputLowerLimit
sPid_Cmpt.b_Input_PER_On	Config.InputPerOn
sPid_Cmpt.b_LoadBackUp	LoadBackUp
sPid_Cmpt.b_InvCtrl	Config.InvertControl
sPid_Cmpt.r_Lmn_Pwm_PPTm	Config.MinimumOnTime
sPid_Cmpt.r_Lmn_Pwm_PBTm	Config.MinimumOffTime
sPid_Cmpt.r_Pv_Hlm	Config.InputUpperLimit
sPid_Cmpt.r_Pv_Llm	Config.InputLowerLimit
sPid_Cmpt.r_Pv_HWrn	Config.InputUpperWarning
sPid_Cmpt.r_Pv_LWrn	Config.InputLowerWarning
sParamCalc.i_Event_SUT	PIDSelfTune.SUT.State
sParamCalc.i_Event_TIR	PIDSelfTune.TIR.State
sRet.i_Mode	sRet.i_Mode has been omitted. The operating mode is changed using Mode and ModeActivate.
sRet.r_Ctrl_Gain	Retain.CtrlParams.Gain
sRet.r_Ctrl_Ti	Retain.CtrlParams.Ti
sRet.r_Ctrl_Td	Retain.CtrlParams.Td
sRet.r_Ctrl_A	Retain.CtrlParams.TdFiltRatio
sRet.r_Ctrl_B	Retain.CtrlParams.PWeighting
sRet.r_Ctrl_C	Retain.CtrlParams.DWeighting
sRet.r_Ctrl_Cycle	Retain.CtrlParams.Cycle

Parameters State and Mode V2

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID_Compact switches to the operating mode saved in the Mode in-out parameter.

When the CPU is switched on or switches from Stop to RUN mode, PID_Compact starts in the operating mode that is saved in the Mode parameter. To leave PID_Compact in "Inactive" mode, set RunModeByStartup = FALSE.

Meaning of values

State / Mode	Description of operating mode
0	Inactive In "Inactive" operating mode, the output value 0.0 is always output, regardless of Config.OutputUpperLimit and Config.OutputLowerLimit. Pulse width modulation is off.
1	Pretuning The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning. Pretuning requirements: • Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) • ManualEnable = FALSE • Reset = FALSE • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * | Config.InputUpperLimit - Config.InputLowerLimit| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint and the process value lie within the configured limits. The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp. The controller switches to automatic mode following successful pretuning. If pretuning is unsuccessful, the switchover of the operating mode is dependent on ActivateRecoverMode. The phase of pretuning is indicated with PIDSelfTune.SUT.State.

State / Mode	Description of operating mode
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are recalculated based on the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning. PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp. Requirements for fine tuning: • No disturbances are expected. • The setpoint and the process value lie within the configured limits. • ManualEnable = FALSE • Reset = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started from: • Automatic mode (State = 3) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_Compact controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual mode (State = 4) If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. If the process value for pretuning is already too near the setpoint or PIDSelfTune.TIR.RunIn = TRUE, an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start. The controller switches to automatic mode following successful fine tuning. If fine tuning is unsuccessful, the switchover of the operating mode is dependent on ActivateRecoverMode. The "Fine tuning" phase is indicated with PIDSelfTune.TIR.State.
3	Automatic mode In automatic mode, PID_Compact corrects the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate. The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor. The ActivateRecoverMode tag is taken into consideration in automatic mode.

State / Mode	Description of operating mode
4	Manual mode In manual mode, you specify a manual output value in the ManualValue parameter. You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only. The switchover from manual mode to automatic mode is bumpless. Manual mode is also possible when an error is pending.
5	Substitute output value with error monitoring The control algorithm is deactivated. The SetSubstituteOutput tag determines which output value is output in this operating mode. • SetSubstituteOutput = FALSE: Last valid output value • SetSubstituteOutput = TRUE: Substitute output value You cannot activate this operating mode using Mode = 5. In the event of an error, it is activated instead of "Inactive" operating mode if all the following conditions are met: • Automatic mode (Mode = 3) • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode is effective. As soon as the errors are no longer pending, PID_Compact switches back to automatic mode.

ENO characteristics

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

Automatic switchover of operating mode during commissioning

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID_Compact automatically switches the operating mode in the event of an error. The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started

Cycle no.	Mode	State	Action
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. At the start of pretuning or fine tuning, PID_Compact has saved the value of State in the Mode in/out parameter. PID_Compact therefore switches to the operating mode from which tuning was started.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode.

See also

Output parameters of PID_Compact V2 (Page 3452)

Parameter ErrorBits V2

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0003h, for example, indicates that the errors 0001h and 0002h are pending simultaneously.

In manual mode, PID_Compact uses ManualValue as output value. The exception is Errorbits = 10000h.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0004	Error during fine tuning. Oscillation of the process value could not be maintained. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0008	Error at start of pretuning. The process value is too close to the setpoint. Start fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0010	The setpoint was changed during tuning. You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
0020	Pretuning is not permitted during fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact remains in fine tuning mode.
0080	Error during pretuning. Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0100	Error during fine tuning resulted in invalid parameters. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0400	Calculation of output value failed. Check the PID parameters. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0800	Sampling time error: PID_Compact is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
10000	Invalid value at ManualValue parameter. Value has an invalid number format. If ActivateRecoverMode = TRUE before an error occurred, PID_Compact uses SubstituteOutput as the output value. As soon as you specify a valid value in ManualValue, PID_Compact uses it as the output value.
20000	Invalid value at SubstituteOutput tag. Value has an invalid number format. PID_Compact uses the output value low limit as the output value. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_Compact switches back to automatic mode.
40000	Invalid value at Disturbance parameter. Value has an invalid number format. If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_Compact remains in automatic mode. If pretuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_Compact switches to the operating mode saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled.

Tag ActivateRecoverMode V2

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

Automatic mode

Notice

Your system may be damaged.

If ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode even if there is an error and the process limit values are exceeded. This may damage your system.

It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

ActivateRecoverMode	Description
FALSE	PID_Compact automatically switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_Compact switches between the calculated output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more of the following errors occur, PID_Compact stays in automatic mode: • 0001h: The "Input" parameter is outside the process value limits. • 0800h: Sampling time error • 40000h: Invalid value at parameter Disturbance. If one or more of the following errors occur, PID_Compact switches to "Substitute output value with error monitoring" mode: • 0002h: Invalid value at Input_PER parameter. • 0200h: Invalid value at Input parameter. • 0400h: Calculation of output value failed. • 1000h: Invalid value at Setpoint parameter. If the following error occurs, PID_Compact switches to "Substitute output value with error monitoring" mode and moves the actuator to Config.OutputLowerLimit: • 20000h: Invalid value at SubstituteOutput tag. Value has an invalid number format. This characteristics are independent of SetSubstituteOutput. As soon as the errors are no longer pending, PID_Compact switches back to automatic mode.

Pretuning and fine tuning

ActivateRecoverMode	Description
FALSE	PID_Compact automatically switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If the following error occurs, PID_Compact remains in the active mode: • 0020h: Pretuning is not permitted during fine tuning. The following errors are ignored: • 10000h: Invalid value at ManualValue parameter. • 20000h: Invalid value at SubstituteOutput tag. When any other error occurs, PID_Compact cancels the tuning and switches to the mode from which tuning was started.

Manual mode

ActivateRecoverMode is not effective in manual mode.

Tag Warning V2

If several warnings are pending simultaneously, the values of the Warning tag are displayed with binary addition. The display of warning 0003h, for example, indicates that the warnings 0001h and 0002h are pending simultaneously.

Warning (DW#16#....)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRule = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
1000	The substitute output value cannot be reached because it is outside the output value limits.

The following warnings are deleted as soon as the cause is eliminated:

- 0001h
- 0004h
- 0008h

- 0040h
- 0100h

All other warnings are cleared with a rising edge at Reset or ErrorAck.

PID_Compact V1

Description of PID_Compact V1

Description

The PID_Compact instruction provides a PID controller with integrated tuning for automatic and manual mode.

Call

PID_Compact is called in the constant interval of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

Download to device

The actual values of retentive tags are only updated when you download PID_Compact completely.

Downloading technology objects to device (Page 7204)

Startup

At the startup of the CPU, PID_Compact starts in the operating mode that was last active. To retain PID_ Compact in "Inactive" mode, set sb_RunModeByStartup = FALSE.

Monitoring of the sampling time PID_Compact

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_Compact instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. If the current sampling time deviates too much from this mean value, Error = 0800 hex occurs and PID_Compact switches to "Inactive" mode.

PID_Compact, Version 1.1 or higher is set to "Inactive" mode during controller tuning under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

In automatic mode, PID_Compact, Version 1.1 or higher, is set to "Inactive" mode under the following conditions:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

During controller tuning and in automatic mode, PID_Compact 1.0 is set to "Inactive" operating mode under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value
- Current sampling time >= 1.5 x current mean value
- Current sampling time <= 0.5 x current mean value

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

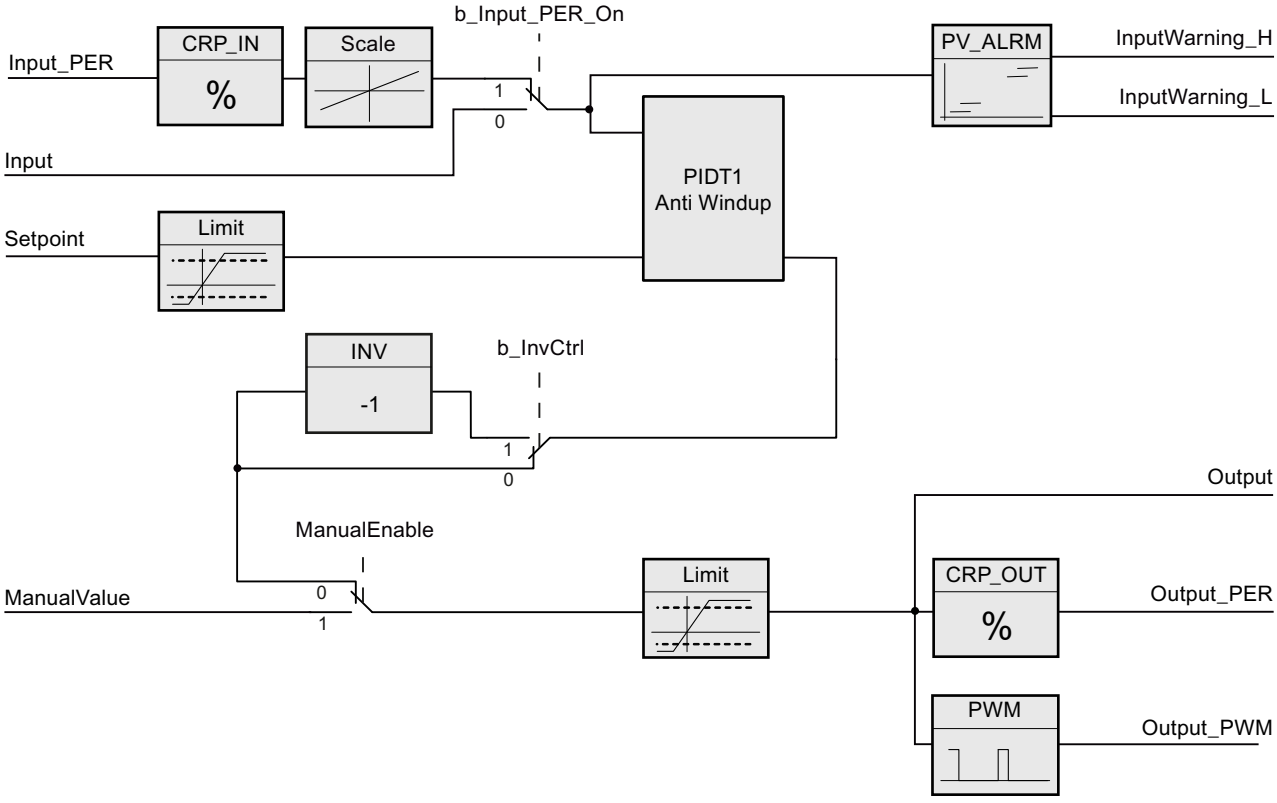
PID algorithm

PID_Compact is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The following equation is used to calculate the output value.

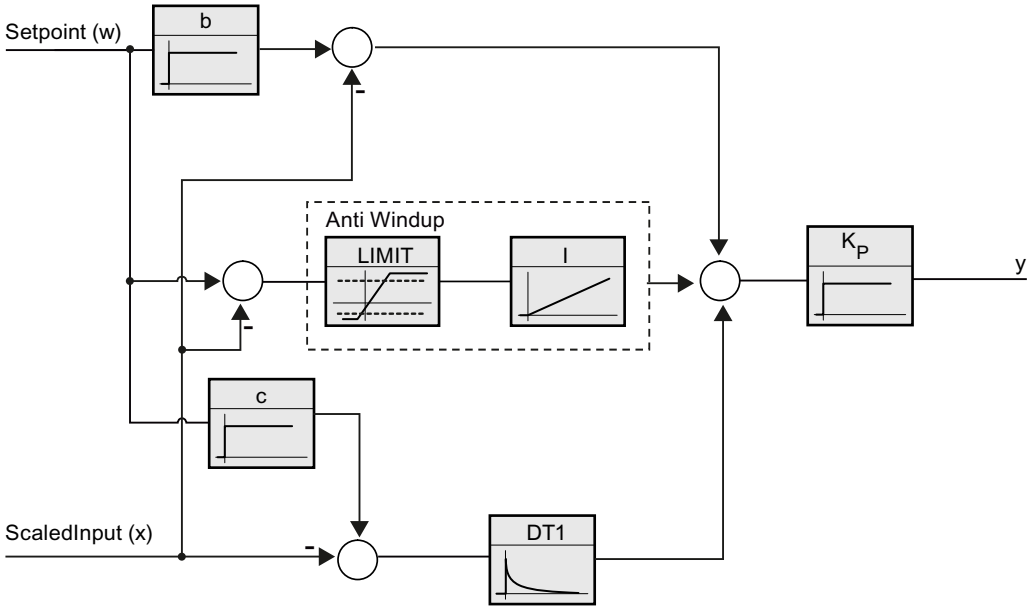
$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (T1 = a × T_D)
	Derivative action time
c	Derivative action weighting

Block diagram of PID_Compact



Block diagram of PIDT1 with anti-windup



Reaction to error

If errors occur, they are output in parameter Error, and PID_Compact changes to "Inactive" mode. Reset the errors using the Reset parameter.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_Compact does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

See also

Controller type (Page 7244)

Input parameters of PID_Compact V1

Table 11-85

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A variable of the user program is used as source for the process value. If you are using parameter Input, then sPid_Cmpt.b_Input_PER_On = FALSE must be set.
Input_PER	WORD	W#16#0	Analog input as the source of the process value If you are using parameter Input_PER, then sPid_Cmpt.b_Input_PER_On = TRUE must be set.
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge selects "Manual mode", while State = 4, sRet.i_Mode remains unchanged. • A TRUE -> FALSE edge selects the most recently active operating mode, State =sRet.i_Mode A change of sRet.i_Mode will not take effect during ManualEnable = TRUE. The change of sRet.i_Mode will only be considered upon a TRUE -> FALSE edge at ManualEnable . **PID_Compact V1.2 und PID_Compact V1.0** If at start of the CPU ManualEnable = TRUE, PID_Compact starts in manual mode. A rising edge (FALSE > TRUE) at ManualEnable is not necessary. **PID_Compact V1.1** At the start of the CPU, PID_Compact only switches to manual mode with a rising edge (FALSE->TRUE) at ManualEnable . Without rising edge, PID_Compact starts in the last operating mode in which ManualEnable was FALSE.
ManualValue	REAL	0.0	Manual value This value is used as the output value in manual mode.
Reset	BOOL	FALSE	The Reset parameter (Page 3483) restarts the controller.

Output parameters of PID_Compact V1

Table 11-86

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Output of the scaled process value
Outputs "Output", "Output_PER", and "Output_PWM" can be used concurrently.			
Output	REAL	0.0	Output value in REAL format
Output_PER	WORD	W#16#0	Analog output value
Output_PWM	BOOL	FALSE	Pulse-width-modulated output value The output value is formed by minimum On and Off times.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the setpoint absolute high limit is reached. The setpoint in the CPU is limited to the configured setpoint absolute high limit. The configured process value absolute high limit is the default for the setpoint high limit. If you set sPid_Cmpt.r_Sp_Hlm to a value within the process value limits, this value is used as the setpoint high limit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the setpoint absolute low limit has been reached. In the CPU, the setpoint is limited to the configured setpoint absolute low limit. The configured process value absolute low limit is the default setting for the setpoint low limit. If you set sPid_Cmpt.r_Sp_Llm to a value within the process value limits, this value is used as the setpoint low limit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 3479) shows the current operating mode of the PID controller. To change the operating mode, use variable sRet.i_Mode. • State = 0: Inactive • State = 1: pretuning • State = 2: fine tuning • State = 3: Automatic mode • State = 4: Manual mode
Error	DWORD	W#16#0	The Error parameter (Page 3482) indicates the error messages. Error = 0000: No error pending.

Static tags of PID_Compact V1

You must not change tags that are not listed. These are used for internal purposes only.

Table 11-87

Tag	Data type	Default	Description
sb_GetCycleTime	BOOL	TRUE	If sb_GetCycleTime = TRUE, the automatic determination of the cycle time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
sb_EnCyclEstimation	BOOL	TRUE	If sb_EnCyclEstimation = TRUE, the sampling time PID_Compact is calculated.
sb_EnCyclMonitoring	BOOL	TRUE	If sb_EnCyclMonitoring = FALSE, the sampling time PID_Compact is not monitored. If it is not possible to execute PID_Compact within the sampling time, an 0800 error is not output and PID_Compact does not change to "Inactive" mode.
sb_RunModeByStartup	BOOL	TRUE	Activate Mode after CPU restart If sb_RunModeByStartup = FALSE, the controller will remain inactive after a CPU startup. After a CPU startup and if sb_RunModeByStartup = TRUE, the controller will return to the most recently active operating mode.
si_Unit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
si_Type	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
sd_Warning	DWORD	DW#16#0	Variable sd_warning (Page 3484) displays the warnings generated since the reset, or since the last change of the operating mode.
sBackUp.r_Gain	REAL	1.0	Saved proportional gain You can reload values from the sBackUp structure with sPid_Cmpt.b_LoadBackUp = TRUE.
sBackUp.r_Ti	REAL	20.0	Saved integral action time [s]
sBackUp.r_Td	REAL	0.0	Saved derivative action time [s]
sBackUp.r_A	REAL	0.0	Saved derivative delay coefficient
sBackUp.r_B	REAL	0.0	Saved proportional action weighting factor
sBackUp.r_C	REAL	0.0	Saved derivative action weighting factor
sBackUp.r_Cycle	REAL	1.0	Saved sampling time of PID algorithm
sPid_Calc.r_Cycle	REAL	0.1	Sampling time of the PID_Compact instruction r_Cycle is determined automatically and usually equivalent to the cycle time of the calling OB.

Tag	Data type	Default	Description
sPid_Calc.b_RunIn	BOOL	FALSE	• b_RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If the requirements for pretuning are not met, PID_Compact reacts like b_RunIn = TRUE. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode. • b_RunIn = TRUE The pretuning is skipped. PID_3Compact tries to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. b_RunIn is set to FALSE after fine tuning.
sPid_Calc.b_CalcParamSUT	BOOL	FALSE	The parameters for pretuning will be recalculated if b_CalcParamSUT = TRUE. This enables you to change the parameter calculation method without having to repeat controller tuning. b_CalcParamSUT will be set to FALSE after calculation.
sPid_Calc.b_CalcParamTIR	BOOL	FALSE	The parameters for fine tuning will be recalculated if b_CalcParamTIR = TRUE. This enables you to change the parameter calculation method without having to repeat controller tuning. b_CalcParamTIR will be set to FALSE after calculation.
sPid_Calc.i_CtrlTypeSUT	INT	0	Methods used to calculate parameters during pretuning: • i_CtrlTypeSUT = 0: PID according to Chien, Hrones and Reswick • i_CtrlTypeSUT = 1: PI according to Chien, Hrones and Reswick
sPid_Calc.i_CtrlTypeTIR	INT	0	Methods used to calculate parameters during fine tuning: • i_CtrlTypeTIR = 0: PID automatic • i_CtrlTypeTIR = 1: PID rapid • i_CtrlTypeTIR = 2: PID slow • i_CtrlTypeTIR = 3: Ziegler-Nichols PID • i_CtrlTypeTIR = 4: Ziegler-Nichols PI • i_CtrlTypeTIR = 5: Ziegler-Nichols P
sPid_Calc.r_Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)

Tag	Data type	Default	Description
sPid_Cmpt.r_Sp_Hlm	REAL	+3.402822e +38	High limit of setpoint If you set sPid_Cmpt.r_Sp_Hlm outside the process value limits, the configured process value absolute high limit is used as the setpoint high limit. If you set sPid_Cmpt.r_Sp_Hlm within the process value limits, this value is used as the setpoint high limit.
sPid_Cmpt.r_Sp_Llm	REAL	-3.402822e +38	Low limit of the setpoint If you set sPid_Cmpt.r_Sp_Llm outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. If you set sPid_Cmpt.r_Sp_Llm within the process value limits, this value is used as the setpoint low limit.
sPid_Cmpt.r_Pv_Norm_IN_1	REAL	0.0	Scaling Input_PER low Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_IN_2	REAL	27648.0	Scaling Input_PER high Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_OUT_1	REAL	0.0	Scaled low process value Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_OUT_2	REAL	100.0	Scaled high process value Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Lmn_Hlm	REAL	100.0	Output value high limit for output parameter "Output"
sPid_Cmpt.r_Lmn_Llm	REAL	0.0	Low output value limit for output parameter "Output"
sPid_Cmpt.b_Input_PER_On	BOOL	TRUE	If b_Input_PER_On = TRUE, then parameter Input_PER is used. If b_Input_PER_On = FALSE, then parameter Input is used.
sPid_Cmpt.b_LoadBackUp	BOOL	FALSE	Activate the back-up parameter set. If an optimization has failed, you can reactivate the previous PID parameters by setting this bit.
sPid_Cmpt.b_InvCtrl	BOOL	FALSE	Invert control logic With b_InvCtrl = TRUE, a rising control deviation reduces the output value.
sPid_Cmpt.r_Lmn_Pwm_PPTm	REAL	0.0	The minimum ON time of the pulse width modulation in seconds is rounded to r_Lmn_Pwm_PPTm = r_Cycle or r_Lmn_Pwm_PPTm = n*r_Cycle

Tag	Data type	Default	Description
sPid_Cmpt.r_Lmn_Pwm_PBTm	REAL	0.0	The minimum OFF time of the pulse width modulation in seconds is rounded to r_Lmn_Pwm_PBTm = r_Cycle or r_Lmn_Pwm_PBTm = n*r_Cycle
sPid_Cmpt.r_Pv_Hlm	REAL	120.0	High limit of the process value At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and the PID_Compact switches to "Inactive" mode. r_Pv_Hlm > r_Pv_Llm
sPid_Cmpt.r_Pv_Llm	REAL	0.0	Low limit of the process value r_Pv_Llm < r_Pv_Hlm
sPid_Cmpt.r_Pv_HWrn	REAL	+3.402822e +38	Warning high limit of the process value If you set r_Pv_HWrn outside the process value limits, the configured process value absolute high limit is used as the warning high limit. If you set r_Pv_HWrn within the process value limits, this value is used as the warning high limit. r_Pv_HWrn > r_Pv_LWrn r_Pv_HWrn ≤ r_Pv_Hlm
sPid_Cmpt.r_Pv_LWrn	REAL	-3.402822e +38	Warning low limit of the process value If you set r_Pv_LWrn outside the process value limits, the configured process value absolute low limit is used as the warning low limit. If you set r_Pv_LWrn within the process value limits, this value is used as the warning low limit. r_Pv_LWrn < r_Pv_HWrn r_Pv_LWrn ≥ r_Pv_LWrn
sParamCalc.i_Event_SUT	INT	0	Variable i_Event_SUT (Page 3485) indicates the current phase of "pretuning":
sParamCalc.i_Event_TIR	INT	0	Variable i_Event_TIR (Page 3485) indicates the current phase of "fine tuning":
sRet.i_Mode	INT	0	The operating mode is changed edge-triggered. The following operating mode is enabled on a change to • i_Mode = 0: "Inactive" (controller stop) • i_Mode = 1: "Pretuning" mode • i_Mode = 2: "Fine tuning" mode • i_Mode = 3: "Automatic mode" • i_Mode = 4: "Manual mode" i_Mode is retentive.
sRet.r_Ctrl_Gain	REAL	1.0	Active proportional gain Gain is retentive.

Tag	Data type	Default	Description
sRet.r_Ctrl_Ti	REAL	20.0	• r_Ctrl_Ti > 0.0: active integral action time • r_Ctrl_Ti = 0.0: Integral action is disabled r_Ctrl_Ti is retentive.
sRet.r_Ctrl_Td	REAL	0.0	• r_Ctrl_Td > 0.0: Active derivative action time • r_Ctrl_Td = 0.0: Derivative action is disabled r_Ctrl_Td is retentive.
sRet.r_Ctrl_A	REAL	0.0	Active derivative delay coefficient r_Ctrl_A is retentive.
sRet.r_Ctrl_B	REAL	0.0	Active proportional action weighting r_Ctrl_B is retentive.
sRet.r_Ctrl_C	REAL	0.0	Active derivative action weighting r_Ctrl_C is retentive.
sRet.r_Ctrl_Cycle	REAL	1.0	Active sampling time of the PID algorithm r_Ctrl_Cycle is calculated during controller tuning and rounded to an integer multiple of r_Cycle. r_Ctrl_Cycle is retentive.

Note

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller. The "Inactive" mode is forced by setting variable "sRet.i_Mode" to "0".

See also

Downloading technology objects to device (Page 7204)

Parameters State and sRet.i_Mode V1

Correlation of the parameters

The State parameter indicates the current operating mode of the PID controller. You cannot modify the State parameter.

You need to modify the sRet.i_Mode tag to change the operating mode. This also applies when the value for the new operating mode is already in sRet.i_Mode. First set sRet.i_Mode = 0 and then sRet.i_Mode = 3. Provided the current operating mode of the controller supports this change, State is set to the value of sRet.i_Mode.

When PID_Compact automatically switches the operating mode, the following applies: State ! = sRet.i_Mode.

Examples:

- Successful pretuning
  State = 3 and sRet.i_Mode = 1
- Error
  State = 0 and sRet.i_Mode remains at the same value, e.g sRet.i_Mode = 3
- ManualEnalbe = TRUE
  State = 4 and sRet.i_Mode remain at the previous value, for example, sRet.i_Mode = 3

Note

You wish to repeat successful fine tuning without exiting automatic mode with i_Mode = 0.

Setting sRet.i_Mode to an invalid value such as 9999 for one cycle has no effect on State. Set Mode = 2 in the next cycle. You can generate a change to sRet.i_Mode without first switching to "inactive" mode.

Meaning of values

State / sRet.i_Mode	Description of the operating mode
0	Inactive The controller is switched off. The controller was in "inactive" mode before pretuning was performed. The PID controller will change to "inactive" mode when running if an error occurs or if the "Deactivate controller" icon is clicked in the commissioning window.
1	Pretuning The pretuning determines the process response to a jump of the output value and searches for the point of inflection. The optimized PID parameters are calculated as a function of the maximum rate of rise and dead time of the controlled system. Pretuning requirements: • The controller is in inactive mode or manual mode • ManualEnable = FALSE • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * |sPid_Cmpt.r_Pv_Hlm - sPid_Cmpt.r_Pv_Llm| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint may not be changed during pretuning. The higher the stability of the process value, the easier it is to calculate the PID parameters and increase precision of the result. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. PID parameters are backed up before they are recalculated and can be reactivated with sPid_Cmpt.b_LoadBackUp. There is a change to automatic mode following successful pretuning and to "inactive" mode following unsuccessful pretuning. The phase of pretuning is indicated with Tag i_Event_SUT V1 (Page 3485).

State / sRet.i_Mode	Description of the operating mode
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized based on the amplitude and frequency of this oscillation. The differences between the process response during pretuning and fine tuning are analyzed. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning. PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. PID parameters are backed up before they are recalculated and can be reactivated with sPid_Cmpt.b_LoadBackUp. Requirements for fine tuning: • No disturbances are expected. • The setpoint and the process value lie within the configured limits. • The setpoint may not be changed during fine tuning. • ManualEnable = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started in: • Automatic mode (State = 3) Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning. PID_Comact will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual (State = 4) mode If the requirements for pretuning are met, pretuning is started. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode. An attempt is made to reach the setpoint with a minimum or maximum output value if the process value for pretuning is already too near the setpoint or sPid_Calc.b_RunIn = TRUE. This can produce increased overshoot. The controller will change to "automatic mode" after successfully completed "fine tuning" and to "inactive" mode if "fine tuning" has not been successfully completed. The "Fine tuning" phase is indicated with Tag i_Event_TIR V1 (Page 3485).

State / sRet.i_Mode	Description of the operating mode
3	Automatic mode In automatic mode, PID_Compact corrects the controlled system in accordance with the parameters specified. The controller changes to automatic mode if one the following conditions is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Change of variable sRet.i_Mode to the value 3. After CPU startup or change from Stop to RUN mode, PID_Compact will start in the most recently active operating mode. To retain PID_Compact in "Inactive" mode, set sb_RunModeByStartup = FALSE.
4	Manual mode In manual mode, you specify a manual output value in the ManualValue parameter. This operating mode is enabled if sRet.i_Mode = 4, or at the rising edge on ManualEnable. If ManualEnable changes to TRUE, only State will change. sRet.i_Mode will retain its current value. PID_Compact will return to the previous operating mode upon a falling edge at ManualEnable. The change to automatic mode is bumpless.

See also

Output parameters of PID_Compact V1 (Page 3474)

Pretuning (Page 7254)

Fine tuning (Page 7256)

"Manual" mode (Page 7257)

Tag i_Event_SUT V1 (Page 3485)

Tag i_Event_TIR V1 (Page 3485)

Parameter Error V1

If several errors are pending simultaneously, the values of the error codes are displayed with binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending simultaneously.

Error (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > sPid_Cmpt.r_Pv_Hlm or • Input < sPid_Cmpt.r_Pv_Llm You cannot move the actuator again until you eliminate the error.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input.
0004	Error during fine tuning. Oscillation of the process value could not be maintained.
0008	Error at start of pretuning. The process value is too close to the setpoint. Start fine tuning.
0010	The setpoint was changed during tuning.
0020	Pretuning is not permitted in automatic mode or during fine tuning.

Error (DW#16#...)	Description
0080	Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic.
0100	Error during tuning resulted in invalid parameters.
0200	Invalid value at "Input" parameter: Value has an invalid number format.
0400	Calculation of output value failed. Check the PID parameters.
0800	Sampling time error: PID_Compact is not called within the sampling time of the cyclic interrupt OB.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format.

See also

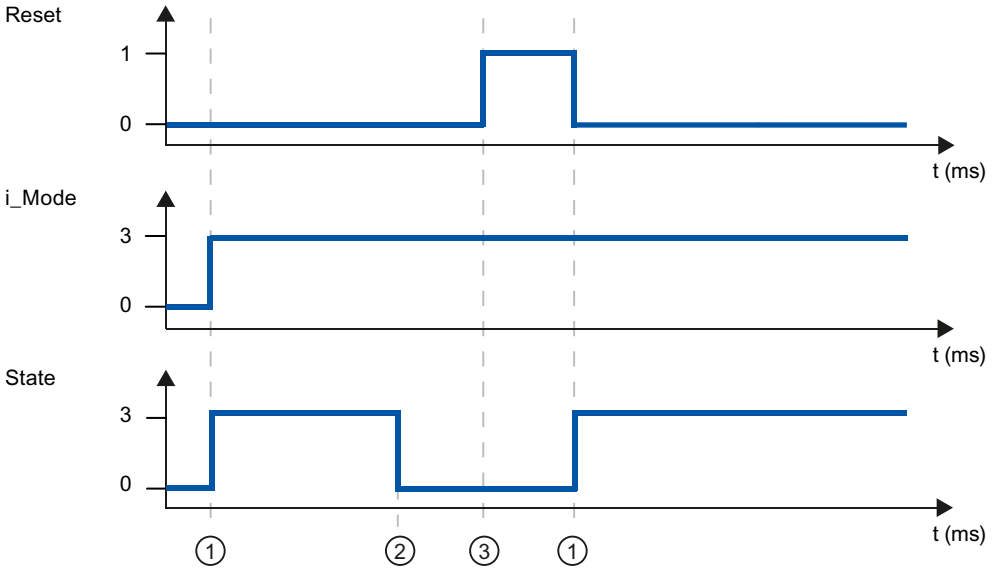
Output parameters of PID_Compact V1 (Page 3474)

Parameter Reset V1

The response to Reset = TRUE depends on the version of the PID_Compact instruction.

Reset response PID_Compact V.1.1 or higher

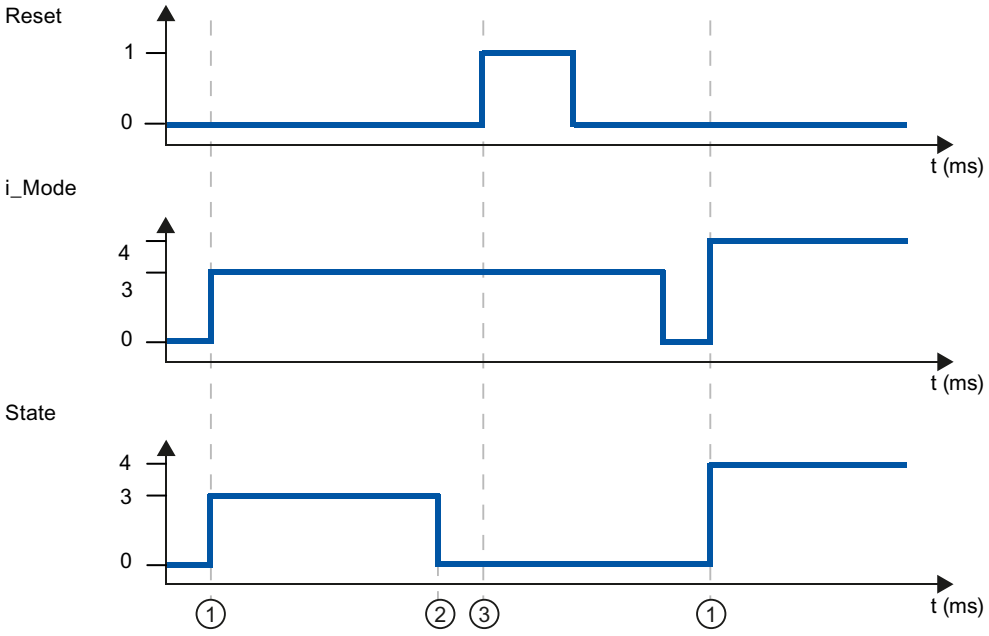
A rising edge at Reset resets the errors and warnings and clears the integral action. A falling edge at Reset triggers a change to the most recently active operating mode.



① Activation
② Error
③ Reset

Reset response PID_Compact V.1.0

A rising edge at Reset resets the errors and warnings and clears the integral action. The controller is not reactivated until the next edge at i_Mode.



① Activation
② Error
③ Reset

Tag sd_warning V1

If several warnings are pending, the values of variable sd_warning are displayed by means of binary addition. The display of warning 0003, for example, indicates that the warnings 0001 and 0002 are also pending.

sd_warning (DW#16#....)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0002	Oscillation increased during fine tuning.
0004	The setpoint was outside the set limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. The PID parameters were instead calculated using the "i_CtrlTypeTIR = 3" method.
0010	The operating mode could not be changed because ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.

The following warnings are deleted as soon as the cause is dealt with:

- 0004
- 0020
- 0040

All other warnings are cleared with a rising edge at Reset.

Tag i_Event_SUT V1

i_Event_SUT	Name	Description
0	SUT_INIT	Initialize pretuning
100	SUT_STDABW	Calculate the standard deviation
200	SUT_GET_POI	Find the point of inflection
9900	SUT_IO	Pretuning successful
1	SUT_NIO	Pretuning not successful

See also

Static tags of PID_Compact V1 (Page 3475)

Parameters State and sRet.i_Mode V1 (Page 3479)

Tag i_Event_TIR V1

i_Event_TIR	Name	Description
-100	TIR_FIRST_SUT	Fine tuning is not possible. Pretuning will be executed first.
0	TIR_INIT	Initialize fine tuning
200	TIR_STDABW	Calculate the standard deviation
300	TIR_RUN_IN	Attempt to reach the setpoint
400	TIR_CTRLN	Attempt to reach the setpoint with the existing PID parameters (if pretuning has been successful)
500	TIR_OSZIL	Determine oscillation and calculate parameters
9900	TIR_IO	Fine tuning successful
1	TIR_NIO	Fine tuning not successful

See also

Static tags of PID_Compact V1 (Page 3475)

Parameters State and sRet.i_Mode V1 (Page 3479)

PID_3Step

New features of PID_3Step

PID_3Step V2.2

- **Use with S7-1200**
  As of PID_3Step V2.2, the instruction with V2 functionality can also be used on S7-1200 with firmware version 4.0 or higher.

PID_3Step V2.0

- **Reaction to error**
  The reaction to ActivateRecoverMode = TRUE has been completely overhauled. PID_3Step reacts in a more fault tolerant manner in the default setting.

  Notice

  Your system may be damaged.

  If you use the default setting, PID_3Step remains in automatic mode even if the process value limits are exceeded. This may damage your system.

  It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

  You use the ErrorAck input parameter to acknowledge the errors and warnings without restarting the controller or clearing the integral action.
  Switching operating modes does not acknowledge errors that are no longer pending.

- **Switching the operating mode**
  You specify the operating mode at the Mode in/out parameter and use a rising edge at ModeActivate to start the operating mode. The Retain.Mode tag has been omitted.
  The transition time measurement can no longer be started with GetTransitTime.Start, but only with Mode = 6 and a rising edge at ModeActivate.

- **Multi-instance capability**
  You can call up PID_3Step as multi-instance DB. No technology object is created in this case and no parameter assignment interface or commissioning interface is available. You must assign parameters for PID_3Step directly in the multi-instance DB and commission it via a watch table.

- **Startup characteristics**
  The operating mode specified at the Mode parameter is also started on a falling edge at Reset and during a CPU cold restart, if RunModeByStartup = TRUE.

- **ENO characteristics**
  ENO is set depending on the operating mode.
  If State = 0, then ENO = FALSE.
  If State ≠ 0, then ENO = TRUE.

- **Manual mode**
  The Manual_UP and Manual_DN input parameters no longer function as edge-triggered parameters. Edge-triggered manual mode continues to be possible using the ManualUpInternal and ManualDnInternal tags.
  In "Manual mode without endstop signals" (Mode = 10), the endstop signals Actuator_H and Actuator_L are ignored even though they are activated.
- **Default value of PID parameters**
  The following default settings have been changed:
  - Proportional action weighting (PWeighting) from 0.0 to 1.0
  - Derivative action weighting (DWeighting) from 0.0 to 1.0
  - Coefficient for derivative delay (TdFiltRatio) from 0.0 to 0.2
- **Limiting of motor transition time**
  You configure the maximum percentage of the motor transition time that the actuator will travel in one direction in the Config.VirtualActuatorLimit tag.
- **Setpoint value specification during tuning**
  You configure the permitted fluctuation of the setpoint during tuning at the CancelTuningLevel tag.
- **Switching a disturbance variable on**
  You can switch a disturbance variable on at the Disturbance parameter.
- **Troubleshooting**
  If the endstop signals are not activated (ActuatorEndStopOn = FALSE), ScaledFeedback is determined without Actuator_H or Actuator_L.

PID_3Step V1.1

- **Manual mode on CPU startup**
  If ManualEnable = TRUE when the CPU starts, PID_3Step starts in manual mode. A rising edge at ManualEnable is not necessary.
- **Reaction to error**
  The ActivateRecoverMode tag is no longer effective in manual mode.
- **Troubleshooting**
  The Progress tag is reset following successful tuning or transition time measurement.

Compatibility with CPU and FW

The following table shows which version of PID_3Step can be used on which CPU.

CPU	FW	PID_3Step
S7-1200	≥ V4.X	V2.2 V1.1
S7-1200	≥ V3.X	V1.1 V1.0
S7-1200	≥ V2.X	V1.1 V1.0
S7-1200	≥ V1.X	-

CPU	FW	PID_3Step
S7-1500	≥ V1.5	V2.2 V2.1 V2.0
S7-1500	≥ V1.1	V2.1 V2.0
S7-1500	≥ V1.0	V2.0

CPU processing time and memory requirement PID_3Step V2.x

CPU processing time

Typical CPU processing times of the PID_3Step technology object as of Version V2.0, depending on CPU type.

CPU	Typ. CPU processing time PID_3Step V2.x
CPU 1211C ≥ V4.0	410 µs
CPU 1215C ≥ V4.0	410 µs
CPU 1217C ≥ V4.0	410 µs
CPU 1505S ≥ V1.0	50 µs
CPU 1510SP-1 PN ≥ V1.6	120 µs
CPU 1511-1 PN ≥ V1.5	120 µs
CPU 1512SP-1 PN ≥ V1.6	120 µs
CPU 1516-3 PN/DP ≥ V1.5	65 µs
CPU 1518-4 PN/DP ≥ V1.5	5 µs

Memory requirement

Memory requirement of an instance DB of the PID_3Step technology object as of Version V2.0.

	Memory requirement of the instance DB of PID_3Step V2.x
Load memory requirement	Approx. 15000 bytes
Total work memory requirement	1040 bytes
Retentive work memory requirement	60 bytes

PID_3Step V2

Description of PID_3Step V2

Description

You use the PID_3Step instruction to configure a PID controller with self tuning for valves or actuators with integrating behavior.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Approach substitute output value
- Transition time measurement
- Error monitoring
- Approach substitute output value with error monitoring
- Manual mode without endstop signals

For a more detailed description of the operating modes, see the State parameter.

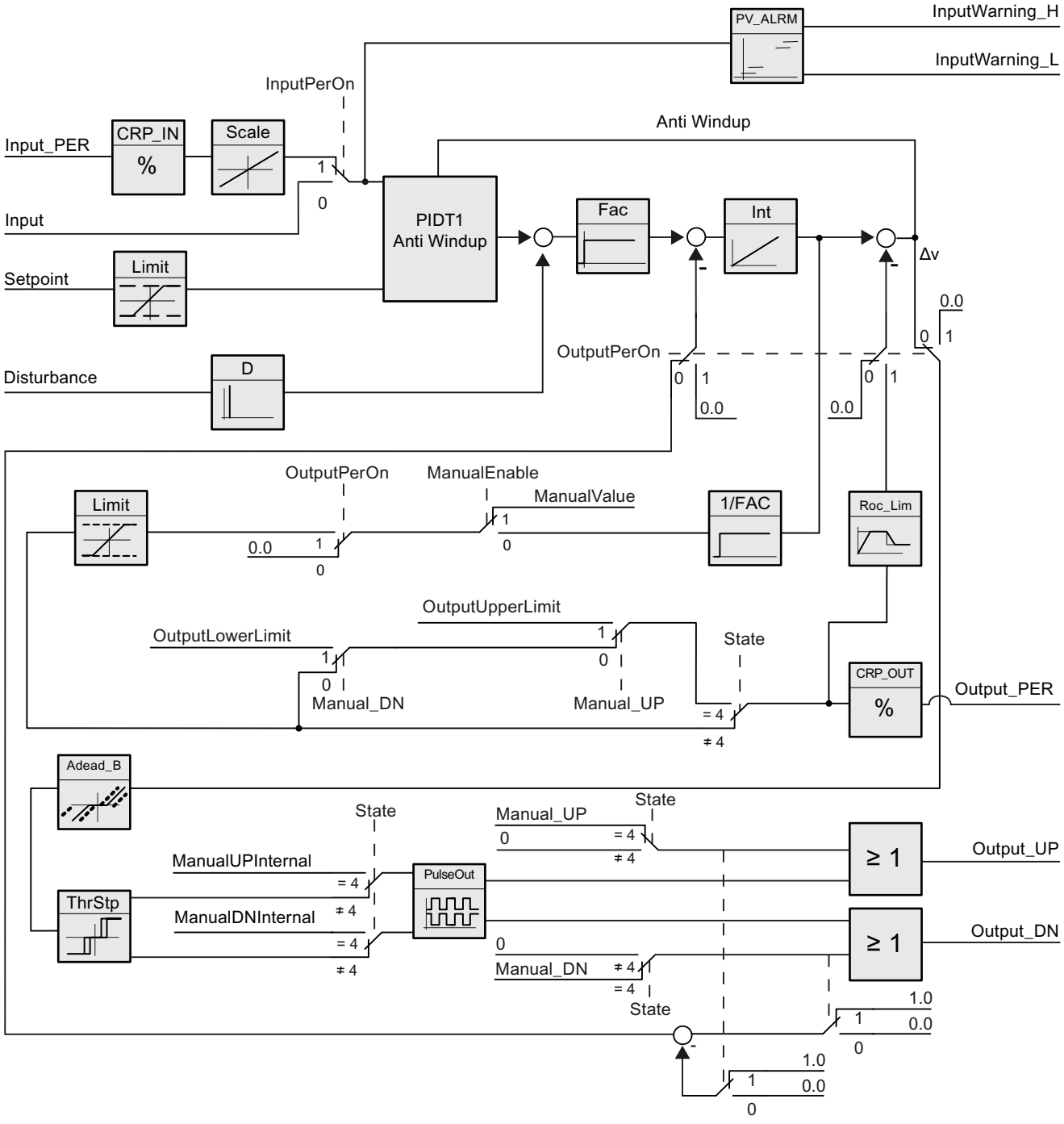
PID algorithm

PID_3Step is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation:

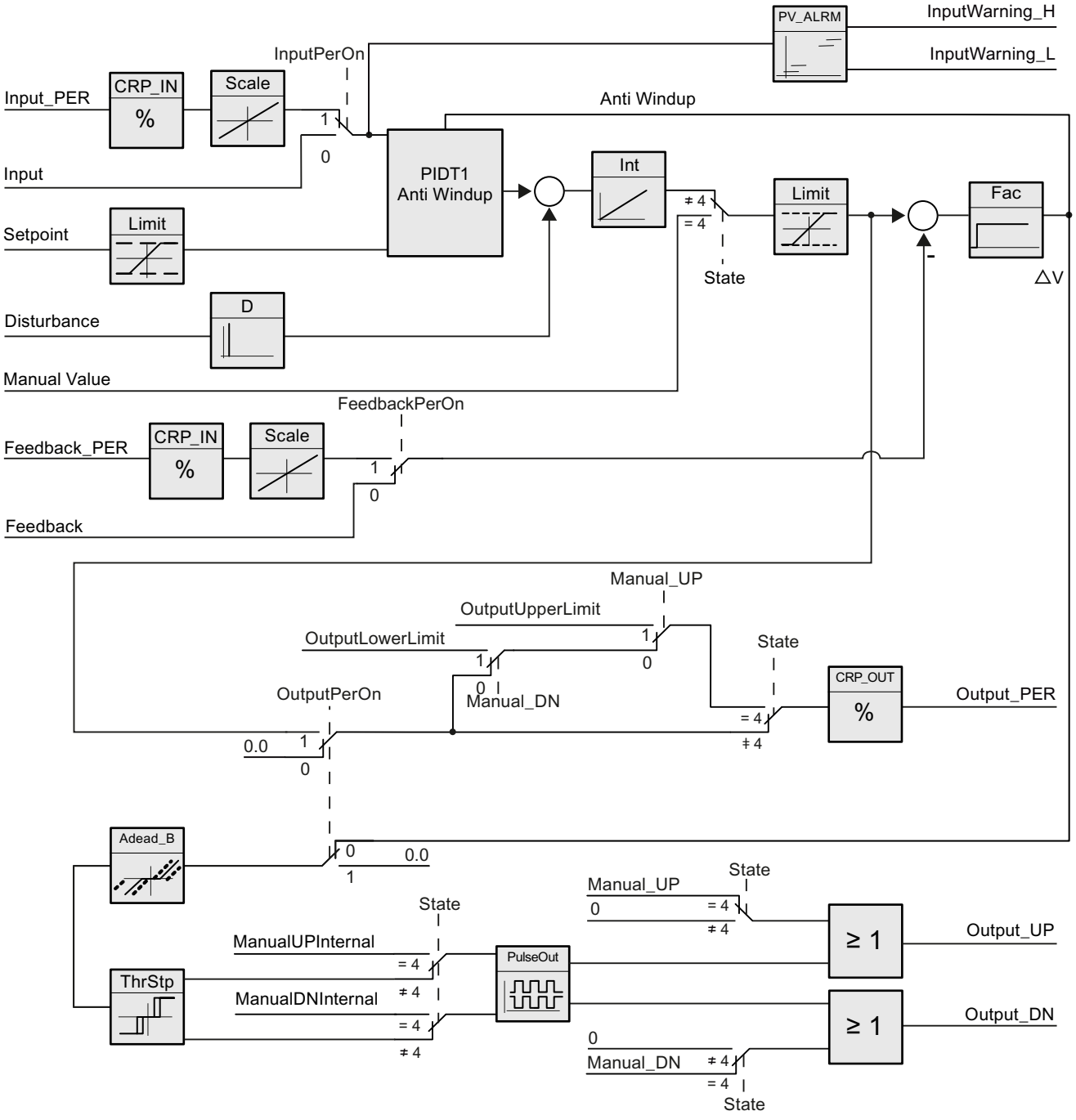
$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
Δy	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
T_D	Derivative action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
c	Derivative action weighting

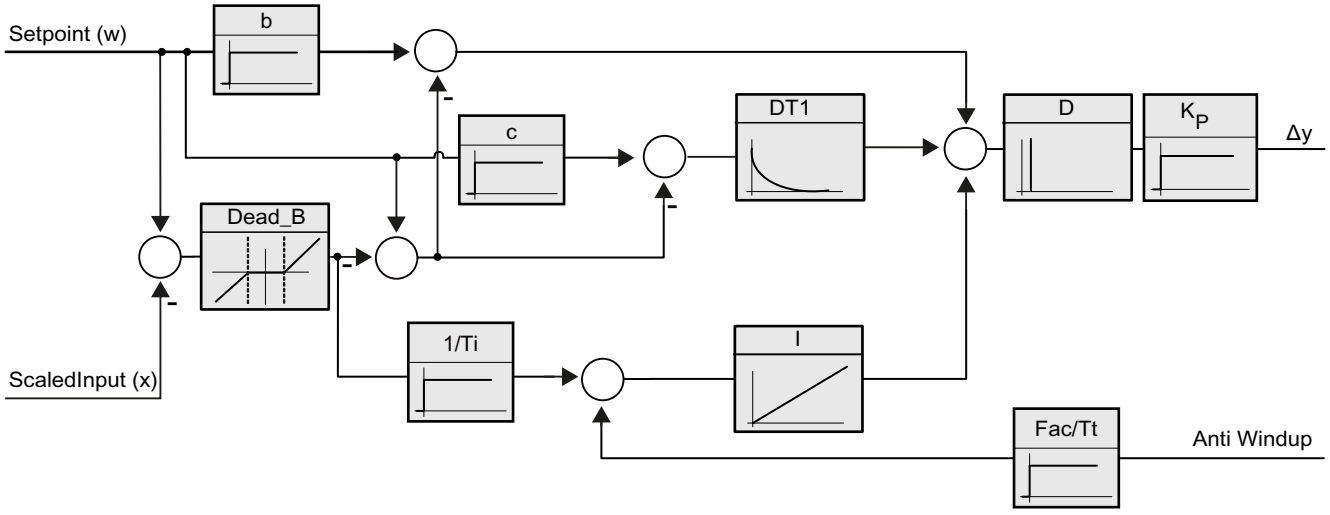
Block diagram without position feedback



Block diagram with position feedback



Block diagram of PIDT1 with anti-windup



Call

PID_3Step is called in the constant time scale of a cycle interrupt OB.

If you call PID_3Step as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_3Step directly in the multi-instance DB and commission it via a watch table.

Download to device

The actual values of retentive tags are only updated when you download PID_3Step completely.

Downloading technology objects to device (Page 7204)

Startup

When the CPU starts up, PID_3Step starts in the operating mode that is saved in the Mode in/out parameter. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

Reaction to error

In automatic mode and during commissioning, the reaction to error depends on the ErrorBehaviour and ActivateRecoverMode tags. In manual mode, the reaction is independent of ErrorBehaviour and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the reaction additionally depends on the error that occurred.

ErrorBehaviour	ActivateRecoverMode	Configuration editor > actuator setting > Set Output to	Reaction
FALSE	FALSE	Current output value	Switch to "Inactive" mode (State = 0) The actuator remains in the current position.
FALSE	TRUE	Current output value while error is pending	Switch to "Error monitoring" mode (State = 7) The actuator remains in the current position while the error is pending.
TRUE	FALSE	Substitute output value	Switch to "Approach substitute output value" mode (State = 5) The actuator moves to the configured substitute output value. Switch to "Inactive" mode (State = 0) The actuator remains in the current position.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Approach substitute output value with error monitoring" mode (State = 8) The actuator moves to the configured substitute output value. Switch to "Error monitoring" mode (State = 7)

In manual mode, PID_3Step uses ManualValue as output value, unless the following errors occur:

- 2000h: Invalid value at Feedback_PER parameter.
- 4000h: Invalid value at Feedback parameter.
- 8000h: Error during digital position feedback.

You can only change the position of the actuator with Manual_UP and Manual_DN, not with ManualValue.

The Error parameter indicates whether an error has occurred in this cycle. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.

See also

Parameters State and Mode V2 (Page 3509)

Parameter ErrorBits V2 (Page 3514)

Configuring PID_3Step V2 (Page 7259)

Mode of operation of PID_3Step V2

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit variables. If the process value is outside these limits, an error occurs (ErrorBits = 0001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning variables. If the process value is outside these warning limits, a warning occurs (Warning = 0040h), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit variables. PID_3Step automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID_3Step checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit variables. The output value limits must be within "Low endstop" and "High endstop".

- High endstop: Config.FeedbackScaling.UpperPointOut
- Low endstop: Config.FeedbackScaling.LowerPointOut

Rule:

UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit ≥ LowerPointOut

The valid values for "High endstop" and "Low endstop" depend upon:

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
FALSE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
FALSE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
TRUE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%

If OutputPerOn = FALSE and FeedbackOn = FALSE, you cannot limit the output value. Output_UP and Output_DN are then reset at Actuator_H = TRUE or Actuator_L = TRUE. If endstop signals are also not present, Output_UP and Output_DN are reset after a travel time of Config.VirtualActuatorLimit × Retain.TransitTime/100.

The output value is 27648 at 100% and -27648 at -100%. PID_3Step must be able to close the valve completely.

Substitute output value

If an error has occurred, PID_3Step can output a substitute output value and move the actuator to a safe position that is specified in the SavePosition tag. The substitute output value must be within the output value limits.

Monitoring signal validity

The values of the following parameters are monitored for validity when used:

- Setpoint
- Input
- Input_PER
- Input_PER
- Feedback
- Feedback_PER
- Disturbance
- ManualValue
- SavePosition
- Output_PER

Monitoring the PID_3Step sampling time

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_3Step instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (ErrorBits = 0800h).

The error occurs during tuning if:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

The error occurs in automatic mode if:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID_3Step in OB1. You must then accept a lower control quality due to the deviating sampling time.

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_3Step are executed at every call.

Measuring the motor transition time

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The actuator is moved in one direction for a maximum time of Config.VirtualActuatorLimit × Retain.TransitTime/100. PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ. You can measure the motor transition time during commissioning. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_3Step does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

See also

Configuring PID_3Step V1 (Page 7276)

Changing the PID_3Step V2 interface

The following table shows what has changed in the PID_3Step instruction interface.

PID_3Step V1	PID_3Step V2	Change
Input_PER	Input_PER	Data type from Word to Int
Feedback_PER	Feedback_PER	Data type from Word to Int
	Disturbance	New
Manual_UP	Manual_UP	Function
Manual_DN	Manual_DN	Function
	ErrorAck	New
	ModeActivate	New

PID_3Step V1	PID_3Step V2	Change
Output_PER	Output_PER	Data type from Word to Int
	ManualUPInternal	New
	ManualDNInternal	New
	CancelTuningLevel	New
	VirtualActuatorLImit	New
Config.Loadbackup	Loadbackup	Renamed
Config.TransitTime	Retain.TransitTime	Renamed and retentivity added
GetTransitTime.Start		Replaced by Mode and ModeActivate
SUT.CalculateSUTParams	SUT.CalculateParams	Renamed
SUT.TuneRuleSUT	SUT.TuneRule	Renamed
TIR.CalculateTIRParams	TIR.CalculateParams	Renamed
TIR.TuneRuleTIR	TIR.TuneRule	Renamed
Retain.Mode	Mode	Function Declaration of static for in-out parameters

Input parameters of PID_3Step V2

Table 11-88

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A tag of the user program is used as the source of the process value. If you are using the Input parameter, then Config.InputPerOn = FALSE must be set.
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using the Input_PER parameter, then Config.InputPerOn = TRUE must be set.
Actuator_H	BOOL	FALSE	Digital position feedback of the valve for the high endstop If Actuator_H = TRUE, the valve is at the high endstop and is no longer moved towards this direction.
Actuator_L	BOOL	FALSE	Digital position feedback of the valve for the low endstop If Actuator_L = TRUE, the valve is at the low endstop and is no longer moved towards this direction.
Feedback	REAL	0.0	Position feedback of the valve If you are using the Feedback parameter, then Config.FeedbackPerOn = FALSE must be set.

Parameter	Data type	Default	Description
Feedback_PER	INT	0	Analog position feedback of a valve If you are using the Feedback_PER parameter, then Config.FeedbackPerOn = TRUE must be set. Feedback_PER is scaled based on the tags: • Config.FeedbackScaling.LowerPointIn • Config.FeedbackScaling.UpperPointIn • Config.FeedbackScaling.LowerPointOut • Config.FeedbackScaling.UpperPointOut
Disturbance	REAL	0.0	Disturbance tag or precontrol value
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge activates "manual mode", while State = 4, Mode remain unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog. • A TRUE -> FALSE edge activates the operating mode that is specified by Mode. We recommend that you change the operating mode using ModeActivate only.
ManualValue	REAL	0.0	In manual mode, the absolute position of the valve is specified. ManualValue is only evaluated if you are using Output_PER, **or** if position feedback is available.
Manual_UP	BOOL	FALSE	• Manual_UP = TRUE The valve is opened even if you are using Output_PER or a position feedback. The valve is no longer moved if the high endstop has been reached. See also Config.VirtualActuatorLimit • Manual_UP = FALSE If you are using Output_PER or a position feedback, the valve is moved to ManualValue. Otherwise, the valve is no longer moved. If Manual_UP and Manual_DN are set to TRUE simultaneously, the valve is not moved.
Manual_DN	BOOL	FALSE	• Manual_DN = TRUE The valve is closed even if you are using Output_PER or a position feedback. The valve is no longer moved if the low endstop has been reached. See also Config.VirtualActuatorLimit • Manual_DN = FALSE If you are using Output_PER or a position feedback, the valve is moved to ManualValue. Otherwise, the valve is no longer moved.
ErrorAck	BOOL	FALSE	• FALSE -> TRUE edge ErrorBits and Warning are reset.

Parameter	Data type	Default	Description
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Switch to "Inactive" mode – ErrorBits and Warning are reset. – Integral action is cleared (PID parameters are retained) • As long as Reset = TRUE, PID_3Step remains in "Inactive" mode (State = 0). • TRUE -> FALSE edge PID_3Step switches to the operating mode that is saved in the Mode parameter.
ModeActivate	BOOL	FALSE	• FALSE -> TRUE edge PID_3Step switches to the operating mode that is saved in the Mode parameter.

Output parameters of PID_3Step V2

Table 11-89

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
ScaledFeedback	REAL	0.0	Scaled position feedback For an actuator without position feedback, the position of the actuator indicated by ScaledFeedback is very imprecise. ScaledFeedback may only be used for rough estimation of the current position in this case.
Output_UP	BOOL	FALSE	Digital output value for opening the valve If Config.OutputPerOn = FALSE, the Output_UP parameter is used.
Output_DN	BOOL	FALSE	Digital output value for closing the valve If Config.OutputPerOn = FALSE, the Output_DN parameter is used.
Output_PER	INT	0	Analog output value If Config.OutputPerOn = TRUE, Output_PER is used.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit). The setpoint is limited to Config.SetpointUpperLimit .
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached (Setpoint ≤ Config.SetpointLowerLimit). The setpoint is limited to Config.SetpointLowerLimit .
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.

Parameter	Data type	Default	Description
State	INT	0	The State parameter (Page 3509) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Approach substitute output value • State = 6: Transition time measurement • State = 7: Error monitoring • State = 8: Approach substitute output value with error monitoring • State = 10: Manual mode without end stop signals
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 3514) shows which error messages are pending. ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck.

See also

Parameters State and Mode V2 (Page 3509)

Parameter ErrorBits V2 (Page 3514)

In-out parameters of PID_3Step V2

Table 11-90

Parameter	Data type	Default	Description
Mode	INT	4	At the Mode parameter, you specify the operating mode to which PID_3Step is to switch. Options are: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode • Mode = 6: Transition time measurement • Mode = 10: Manual mode without endstop signals The operating mode is activated by: • Rising edge at ModeActivate • Falling edge at Reset • Falling edge at ManualEnable • Cold restart of CPU if RunModeByStartup = TRUE Mode is retentive. A detailed description of the operating modes can be found in Parameters State and Mode V2 (Page 3509).

Static tags of PID_3Step V2

You must not change tags that are not listed. These are used for internal purposes only.

Tag	Data type	Default	Description
ManualUpInternal	BOOL	FALSE	In manual mode, each rising edge opens the valve by 5% of the total control range or for the duration of the minimum motor transition time. ManualUpInternal is only evaluated if you are not using Output_PER or a position feedback. This tag is used in the commissioning dialog.
ManualDnInternal	BOOL	FALSE	In manual mode, every rising edge closes the valve by 5% of the total control range or for the duration of the minimum motor transition time. ManualDnInternal is only evaluated if you are not using Output_PER or position feedback. This tag is used in the commissioning dialog.
ActivateRecoverMode	BOOL	TRUE	The ActivateRecoverMode V2 (Page 3517) tag determines the reaction to error.
RunModeByStartup	BOOL	TRUE	Activate operating mode at Mode parameter after CPU restart If RunModeByStartup = TRUE, PID_3Step starts in the operating mode saved in the Mode parameter after CPU startup. If RunModeByStartup = FALSE, PID_3Step remains in "Inactive" mode after CPU startup.
LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE.

Tag	Data type	Default	Description
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., °C, or °F.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature
ErrorBehaviour	BOOL	FALSE	If ErrorBehaviour = FALSE and an error has occurred, the valve stays at its current position and the controller switches directly to "Inactive" or "Error monitoring" mode. If ErrorBehaviour = TRUE and an error occurs, the actuator moves to the substitute output value and only then switches to "Inactive" or "Error monitoring" mode. If the following errors occur, you can no longer move the valve to a configured substitute output value. • 2000h: Invalid value at Feedback_PER parameter. • 4000h: Invalid value at Feedback parameter. • 8000h: Error during digital position feedback. • 20000h: Invalid value at SavePosition tag.
Warning	DWORD	DW#16#0	The Warning tag (Page 3509) shows the warnings since Reset = TRUE or ErrorAck =TRUE. Warning is retentive. Cyclic warnings (for example, process value warning) are shown until the cause of the warning is removed. They are automatically deleted once their cause has gone. Non-cyclic warnings (for example, point of inflection not found) remain and are deleted like errors.
SavePosition	REAL	0.0	Substitute output value If ErrorBehaviour = TRUE, the actuator is moved to a position that is safe for the plant when an error occurs. As soon as the substitute output value has been reached, PID_3Step switches the operating mode according to ActivateRecoverMode.
CurrentSetpoint	REAL	0.0	Currently active setpoint. This value is frozen at the start of tuning.
CancelTuningLevel	REAL	10.0	Permissible fluctuation of setpoint during tuning. Tuning is not canceled until: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)
Config.InputPerOn	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.OutputPerOn	BOOL	FALSE	If OutputPerOn = TRUE, the Output_PER parameter is used. If OutputPerOn = FALSE, the Ouput_UP and Output_DN parameters are used.
Config.InvertControl	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.FeedbackOn	BOOL	FALSE	If FeedbackOn = FALSE, a position feedback is simulated. Position feedback is generally activated when FeedbackOn = TRUE.

Tag	Data type	Default	Description
Config.FeedbackPerOn	BOOL	FALSE	FeedbackPerOn is only effective when FeedbackOn = TRUE. If FeedbackPerOn = TRUE, the analog input is used for the position feedback (Feedback_PER parameter). If FeedbackPerOn = FALSE, the Feedback parameter is used for the position feedback.
Config.ActuatorEndStopOn	BOOL	FALSE	If ActuatorEndStopOn = TRUE, the digital position feedback Actuator_L and Actuator_H are taken into consideration.
Config.InputUpperLimit	REAL	120.0	High limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	+3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.402822e+38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	High limit of output value For details, see OutputLowerLimit
Config.OutputLowerLimit	REAL	0.0	Low limit of output value If OutputPerOn = TRUE or FeedbackOn = TRUE, the range of values from -100% to +100%, including zero, is valid. At -100%, Output = -27648; at +100% Output = 27648 If OutputPerOn = FALSE, the range of values from 0% to 100% is valid. The valve is completely closed at 0% and completely opened at 100%.
Config.SetpointUpperLimit	REAL	+3.402822e+38	High limit of setpoint If you set SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is preassigned as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.

Tag	Data type	Default	Description
Config.SetpointLowerLimit	REAL	-3.402822e+38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured absolute process value low limit is preassigned as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime	REAL	0.0	Minimum ON time Minimum time in seconds for which the servo drive must be switched on.
Config.MinimumOffTime	REAL	0.0	Minimum OFF time Minimum time in seconds for which the servo drive must be switched off.
Config.VirtualActuatorLimit	REAL	150.0	If all the following conditions have been satisfied, the actuator is moved in one direction for the maximum period of VirtualActuatorLimit × Retain.TransitTime/100 and the warning 2000h is output: • Config.OutputPerOn = FALSE • Config.ActuatorEndStopOn = FALSE • Config.FeedbackOn = FALSE If Config.OutputPerOn = FALSE and Config.ActuatorEndStopOn = TRUE or Config.FeedbackOn = TRUE, only the warning 2000h is output. If Config.OutputPerOn = TRUE, VirtualActuatorLimit is not taken into consideration.
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.FeedbackScaling.UpperPointIn	REAL	27648.0	Scaling Feedback_PER high Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointIn	REAL	0.0	Scaling Feedback_PER low Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.

Tag	Data type	Default	Description
Config.FeedbackScaling.UpperPointOut	REAL	100.0	High endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointOut	REAL	0.0	Low endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
GetTransitTime.InvertDirection	BOOL	FALSE	If InvertDirection = FALSE, the valve is fully opened, closed, and then reopened in order to determine the valve transition time. If InvertDirection = TRUE, the valve is fully closed, opened, and then closed again.
GetTransitTime.SelectFeedback	BOOL	FALSE	If SelectFeedback = TRUE, then Feedback_PER, or Feedback is taken into consideration in the transition time measurement. If SelectFeedback = FALSE, then Actuator_H and Actuator_L are taken into consideration in the transition time measurement.
GetTransitTime.State	INT	0	Current phase of the transition time measurement • State = 0: Inactive • State = 1: Open valve completely • State = 2: Close valve completely • State = 3: Move valve to target position (NewOutput) • State = 4: Transition time measurement successfully completed • State = 5: Transition time measurement canceled
GetTransitTime.NewOutput	REAL	0.0	Target position for transition time measurement with position feedback The target position must be between "High endstop" and "Low endstop". The difference between NewOutput and ScaledFeedback must be at least 50% of the permissible control range.
CycleTime.StartEstimation	BOOL	TRUE	If StartEstimation = TRUE, the measurement of the PID_3Step sampling time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If EnEstimation = TRUE, the PID_3Step sampling time is calculated. If CycleTime.EnEstimation = FALSE, the PID_3Step sampling time is not calculated and you need to correct the configuration of CycleTime.Value manually.
CycleTime.EnMonitoring	BOOL	TRUE	If EnMonitoring = TRUE, the PID_3Step sampling time is monitored. If it is not possible to execute PID_3Step within the sampling time, the error 0800h is output and the operating mode is switched. ActivateRecoverMode and ErrorBehaviour determine which operating mode is switched to. If EnMonitoring = FALSE, the PID_3Step sampling time is not monitored, the error 0800h is not output, and the operating mode is not switched.
CycleTime.Value	REAL	0.1	PID_3Step sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.

Tag	Data type	Default	Description
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time in seconds
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time in seconds
CtrlParamsBackUp.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	1.0	Saved proportional action weighting
CtrlParamsBackUp.DWeighting	REAL	1.0	Saved derivative action weighting
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm in seconds
CtrlParamsBackUp.InputDeadBand	REAL	0.0	Saved deadband width of the control deviation
PIDSelfTune.SUT.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRule. CalculateParams is set to FALSE following calculation.
PIDSelfTune.SUT.TuneRule	INT	1	Methods used to calculate parameters during pretuning: • SUT.TuneRule = 0: PID rapid I • SUT.TuneRule = 1: PID slow I • SUT.TuneRule = 2: Chien, Hrones and Reswick PID • SUT.TuneRule = 3: Chien, Hrones, Reswick PI • SUT.TuneRule = 4: PID rapid II • SUT.TuneRule = 5: PID slow II
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning: • State = 0: Initialize pretuning • State = 50: Determine start position without position feedback • State = 100: Calculate standard deviation • State = 200: Determine point of inflection • State = 300: Determine rise time • State = 9900: Pretuning successful • State = 1: Pretuning not successful

Tag	Data type	Default	Description
PIDSelfTune.TIR.RunIn	BOOL	FALSE	With the RunIn tag, you can specify that fine tuning can also be performed without pretuning. • RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_3Step switches to the mode from which tuning was started. • RunIn = TRUE The pretuning is skipped. PID_3Step attempts to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start. RunIn is set to FALSE after fine tuning.
PIDSelfTune.TIR.Calculate-Params	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRule. CalculateParams is set to FALSE following calculation.
PIDSelfTune.TIR.TuneRule	INT	0	Methods used to calculate parameters during fine tuning: • TIR.TuneRule = 0: PID automatic • TIR.TuneRule = 1: PID rapid • TIR.TuneRule = 2: PID slow • TIR.TuneRule = 3: Ziegler-Nichols PID • TIR.TuneRule = 4: Ziegler-Nichols PI • TIR.TuneRule = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of fine tuning: • State = -100 Fine tuning is not possible. Pretuning will be performed first. • State = 0: Initialize fine tuning • State = 200: Calculate standard deviation • State = 300: Attempt to reach the setpoint with the maximum or minimum output value • State = 400: Attempt to reach the setpoint with existing PID parameters (if pretuning was successful) • State = 500: Determine oscillation and calculate parameters • State = 9900: Fine tuning successful • State = 1: Fine tuning not successful
Retain.TransitTime	REAL	30.0	Motor transition time in seconds Time in seconds the actuating drive requires to move the valve from the closed to the opened state. TransitTime is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.SetByUser	BOOL	FALSE	If SetByUser = FALSE, the PID parameters are determined automatically and PID_3Step operates with a deadband at the output value. The deadband width is calculated during tuning on the basis of the standard deviation of the output value and saved in Retain.CtrlParams.OutputDeadBand. If SetByUser = TRUE, the PID parameters are entered manually and PID_3 Step operates without a deadband at the output value. Retain.CtrlParams.OutputDeadBand = 0.0 SetByUser is retentive.
Retain.CtrlParams.Gain	REAL	1.0	Active proportional gain To invert the control logic, use the Config.InvertControl tag. Negative values at Gain also invert the control logic. We recommend you use only InvertControl to set the control logic. The control logic is also inverted if InvertControl = TRUE and Gain < 0.0. Gain is retentive.
Retain.CtrlParams.Ti	REAL	20.0	• Ti > 0.0: Active integral action time in seconds • Ti = 0.0: Integral action is deactivated Ti is retentive.
Retain.CtrlParams.Td	REAL	0.0	• Td > 0.0: Active derivative action time in seconds • Td = 0.0: Derivative action is deactivated Td is retentive.
Retain.CtrlParams.TdFiltRatio	REAL	0.2	Active derivative delay coefficient The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient • 0.0: Derivative action is effective for one cycle only and therefore almost not effective. • 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant. • > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed. TdFiltRatio is retentive.
Retain.CtrlParams.PWeighting	REAL	1.0	Active proportional action weighting The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Proportional action for setpoint change is fully effective • 0.0: Proportional action for setpoint change is not effective The proportional action is always fully effective when the process value is changed. PWeighting is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.DWeighting	REAL	1.0	Active derivative action weighting The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Derivative action is fully effective upon setpoint change • 0.0: Derivative action is not effective upon setpoint change The derivative action is always fully effective when the process value is changed. DWeighting is retentive.
Retain.CtrlParams.Cycle	REAL	1.0	Active sampling time of PID algorithm in seconds, rounded to an integer multiple of the cycle time of the calling OB. Cycle is retentive.
Retain.CtrlParams.InputDeadBand	REAL	0.0	Deadband width of the control deviation InputDeadBand is retentive.

Note

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller.

See also

Parameters State and Mode V2 (Page 3509)

Tag ActivateRecoverMode V2 (Page 3517)

Downloading technology objects to device (Page 7204)

Parameters State and Mode V2

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID_3Step switches to the operating mode saved in the Mode in-out parameter.

When the CPU is switched on or switches from Stop to RUN mode, PID_3Step starts in the operating mode that is saved in the Mode parameter. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

Meaning of values

State	Description of operating mode
0	Inactive The controller is switched off and no longer changes the valve position.
1	Pretuning The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning. Pretuning requirements: • The motor transition time has been configured or measured. • Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) • ManualEnable = FALSE • Reset = FALSE • The setpoint and the process value lie within the configured limits. The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher as compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp. The controller switches to automatic mode following successful pretuning. If pretuning is unsuccessful, the switchover of operating mode is dependent on ActivateRecoverMode and ErrorBehaviour. The pretuning phase is indicated with the SUT.State tag.

State	Description of operating mode
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are recalculated based on the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning. PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel The PID parameters are backed up before fine tuning. They can be reactivated with LoadBackUp. Requirements for fine tuning: • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. • ManualEnable = FALSE • Reset = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started from: • Automatic mode (State = 3) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual mode (State = 4) If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. If PIDSelfTune.TIR.RunIn = TRUE, pretuning is skipped and an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. The controller switches to automatic mode following successful fine tuning. If fine tuning is unsuccessful, the switchover of operating mode is dependent on ActivateRecoverMode and ErrorBehaviour. The fine tuning phase is indicated with the TIR.State tag.
3	Automatic mode In automatic mode, PID_3Step controls the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate. The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor. The ActivateRecoverMode tag is taken into consideration in automatic mode.

State	Description of operating mode
4	Manual mode In manual mode, you specify manual output values in the Manual_UP and Manual_DN parameters or ManualValue parameter. Whether or not the actuator can be moved to the output value in the event of an error is described in the ErrorBits parameter. You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only. The switchover from manual mode to automatic mode is bumpless. Manual mode is also possible when an error is pending.
5	Approach substitute output value This operating mode is activated in the event of an error, if Errorbehaviour = TRUE and ActivateRecoverMode = FALSE.. PID_3Step moves the actuator to the substitute output value and then switches to "Inactive" mode.
6	Transition time measurement The time that the motor needs to completely open the valve from the closed condition is determined. This operating mode is activated when Mode = 6 and ModeActivate = TRUE is set. If endstop signals are used to measure the transition time, the valve will be opened completely from its current position, closed completely, and opened completely again. If GetTransitTime.InvertDirection = TRUE, this behavior is inverted. If position feedback is used to measure the transition time, the actuator will be moved from its current position to a target position. The output value limits are not taken into consideration during the transition time measurement. The actuator can travel to the high or the low endstop.
7	Error monitoring The control algorithm is switched off and no longer changes the valve position. This operating mode is activated instead of "Inactive" mode in the event of an error. All the following conditions must be met: • Automatic mode (Mode = 3) • Errorbehaviour = FALSE • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 3517) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.
8	Approach substitute output value with error monitoring This operating mode is activated instead of "approach substitute output value" mode when an error occurs. PID_3Step moves the actuator to the substitute output value and then switches to "error monitoring" mode. All the following conditions must be met: • Automatic mode (Mode = 3) • Errorbehaviour = TRUE • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 3517) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.
10	Manual mode without endstop signals The endstop signals are **not** taken into consideration, even though Config.ActuatorEndStopOn = TRUE. The output value limits are not taken into consideration. Otherwise, PID_3Step behaves the same as in manual mode.

ENO characteristics

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

Automatic switchover of operating mode during commissioning

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID_3Step automatically switches the operating mode in the event of an error. The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. At the start of transition time measurement, pretuning, or fine tuning, PID_3Step saved the value of State in the Mode in/out parameter. PID_3Step therefore switches to the operating mode from which transition time measurement or tuning was started.

If ActivateRecoverMode = FALSE, "Inactive" or "Approach substitute output value" mode is activated.

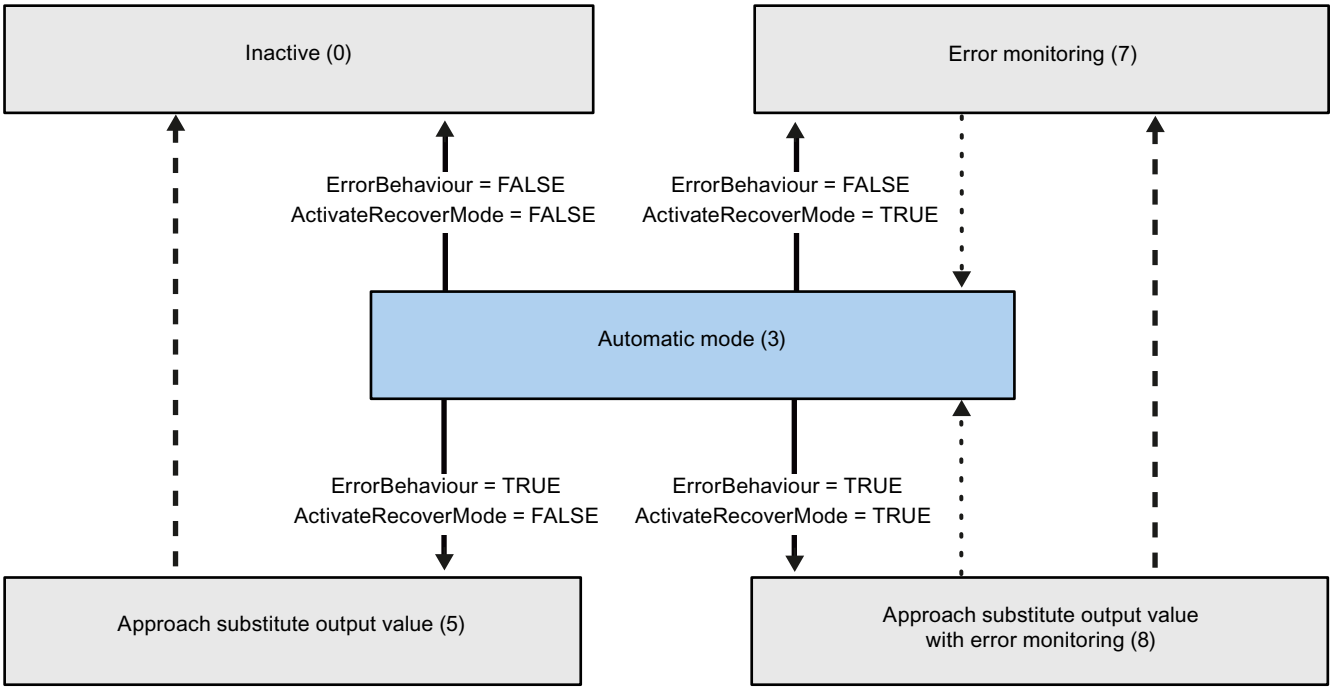
Automatic switchover of operating mode after transition time measurement

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated after successful transition time measurement.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode after successful transition time measurement.

Automatic switchover of operating mode in automatic mode

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.



Automatic switchover of operating mode in the event of an error

Automatic switchover of operating mode once the current operation has been completed.

Automatic switchover of operating mode when error is no longer pending.

See also

Tag ActivateRecoverMode V2 (Page 3517)

Parameter ErrorBits V2 (Page 3514)

Parameter ErrorBits V2

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0003h, for example, indicates that the errors 0001h and 0002h are pending simultaneously.

If there is a position feedback, PID_3Step uses ManualValue as output value in manual mode. The exception is Errorbits = 10000h.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
0004	Error during fine tuning. Oscillation of the process value could not be maintained. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0010	The setpoint was changed during tuning. You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0020	Pretuning is not permitted during fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step remains in fine tuning mode.
0080	Error during pretuning. Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0100	Error during fine tuning resulted in invalid parameters. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
0400	Calculation of output value failed. Check the PID parameters. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
0800	Sampling time error: PID_3Step is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
2000	Invalid value at Feedback_PER parameter. Check whether an error is pending at the analog input. The actuator cannot be moved to the substitute output value and remains in its current position. In manual mode, you can change the position of the actuator only with Manual_UP and Manual_DN, and not with ManualValue. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
4000	Invalid value at Feedback parameter. Value has an invalid number format. The actuator cannot be moved to the substitute output value and remains in its current position. In manual mode, you can change the position of the actuator only with Manual_UP and Manual_DN, and not with ManualValue. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
8000	Error during digital position feedback. Actuator_H = TRUE and Actuator_L = TRUE. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. In order to move the actuator from this state, you must deactivate the "Actuator endstop" (Config.ActuatorEndStopOn = FALSE) or switch to manual mode without endstop signals (Mode = 10). If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
10000	Invalid value at ManualValue parameter. Value has an invalid number format. The actuator cannot be moved to the manual value and remains in its current position. Specify a valid value in ManualValue or move the actuator in manual mode with Manual_UP and Manual_DN.
20000	Invalid value at SavePosition tag. Value has an invalid number format. The actuator cannot be moved to the substitute output value and remains in its current position.
40000	Invalid value at Disturbance parameter. Value has an invalid number format. If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_3Step remains in automatic mode. If pretuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_3Step switches to the operating mode saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled. The error has no effect during transition time measurement.

Tag ActivateRecoverMode V2

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

Notice

Your system may be damaged.

If ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode even if the process limit values are exceeded. This may damage your system.

It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

Automatic mode

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" mode. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_3Step switches between the calculated output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more of the following errors occur, PID_3Step stays in automatic mode: • 0001h: The "Input" parameter is outside the process value limits. • 0800h: Sampling time error • 40000h: Invalid value at Disturbance parameter. If one or more of the following errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at Input_PER parameter. • 0200h: Invalid value at Input parameter. • 0400h: Calculation of output value failed. • 1000h: Invalid value at Setpoint parameter. If one or more of the following errors occur, PID_3Step can no longer move the actuator: • 2000h: Invalid value at Feedback_PER parameter. • 4000h: Invalid value at Feedback parameter. • 8000h: Error during digital position feedback. • 20000h: Invalid value at SavePosition tag. Value has an invalid number format. The characteristics are independent of ErrorBehaviour. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

Pretuning, fine tuning, and transition time measurement

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" mode. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate. The controller changes to "Inactive" mode after successful transition time measurement.
TRUE	If the following error occurs, PID_3Step remains in the active mode: • 0020h: Pretuning is not permitted during fine tuning. The following errors are ignored: • 10000h: Invalid value at ManualValue parameter. • 20000h: Invalid value at SavePosition tag. When any other error occurs, PID_3Step cancels the tuning and switches to the mode from which tuning was started.

Manual mode

ActivateRecoverMode is not effective in manual mode.

See also

Static tags of PID_3Step V2 (Page 3501)

Parameters State and Mode V2 (Page 3509)

Tag Warning V2

If several warnings are pending simultaneously, their values are displayed with binary addition. The display of warning 0005h, for example, indicates that the warnings 0001h and 0004h are pending simultaneously.

Warning (DW#16#...)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRule = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
0400	The transition time cannot be measured because the actuator settings do not match the selected measuring method.
0800	The difference between the current position and the new output value is too small for transition time measurement. This can produce incorrect results. The difference between the current output value and new output value must be at least 50% of the entire control range.
1000	The substitute output value cannot be reached because it is outside the output value limits.
2000	The actuator was moved in one direction for longer than Config.VirtualActuatorLimit × Retain.TransitTime. Check whether the actuator has reached an endstop signal.

The following warnings are deleted as soon as the cause is eliminated:

- 0001h
- 0004h
- 0008h
- 0040h
- 0100h
- 2000h

All other warnings are cleared with a rising edge at Reset or ErrorAck.

PID_3Step V1

Description PID_3Step V1

Description

You use the PID_3Step instruction to configure a PID controller with self tuning for valves or actuators with integrating behavior.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Approach substitute output value
- Transition time measurement
- Approach substitute output value with error monitoring
- Error monitoring

For a more detailed description of the operating modes, see the State parameter.

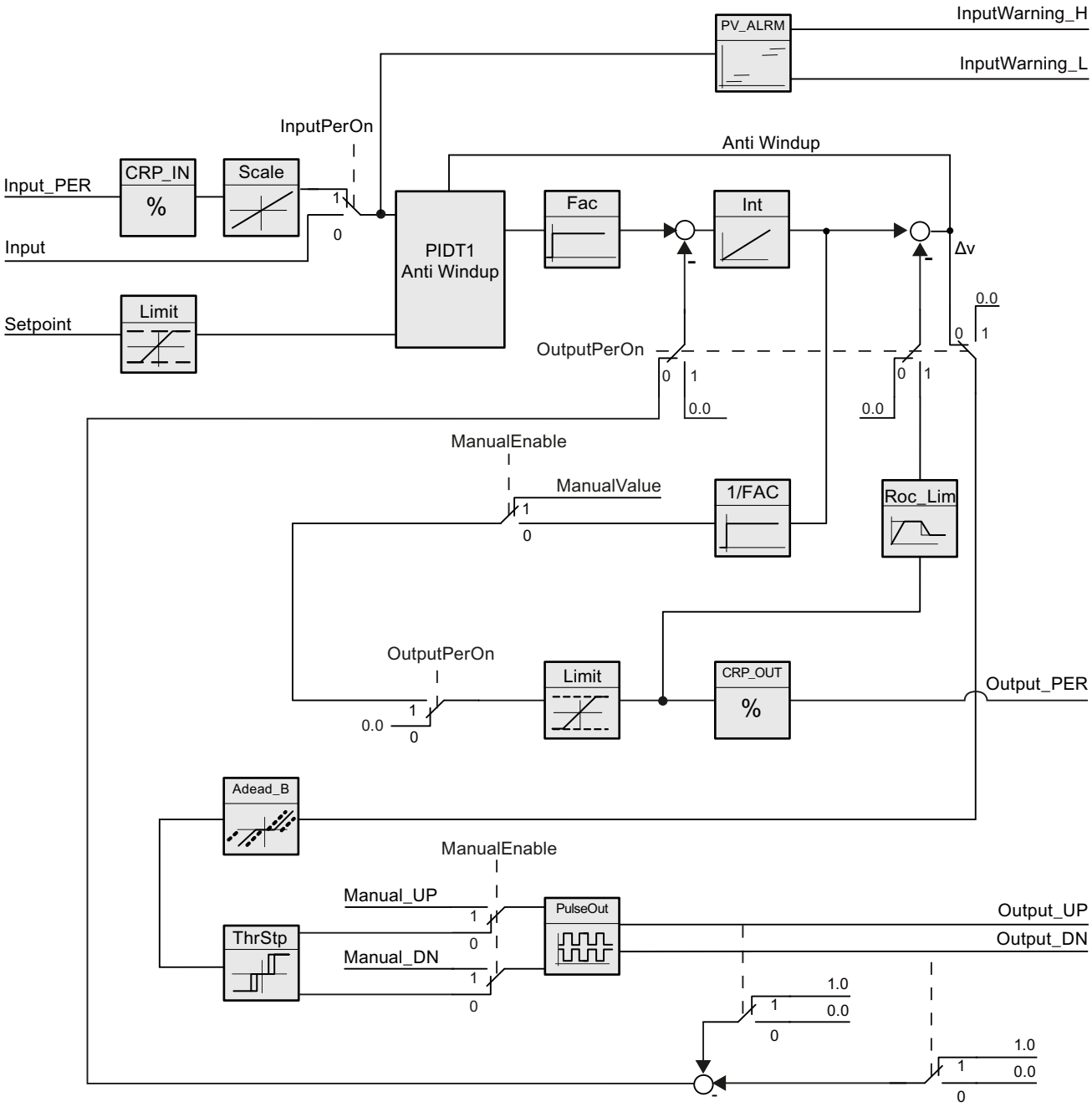
PID algorithm

PID_3Step is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The following equation is used to calculate the output value.

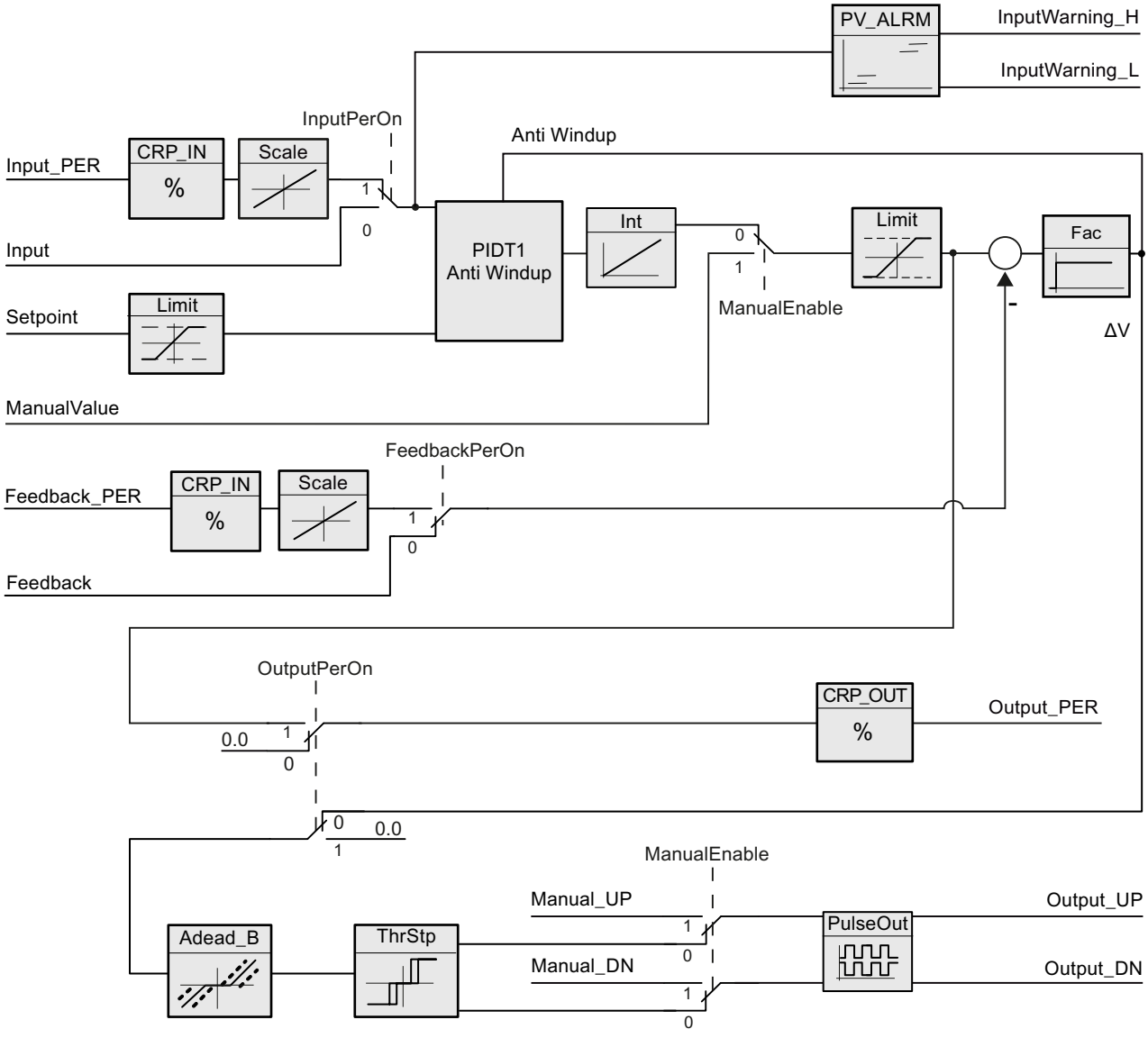
$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

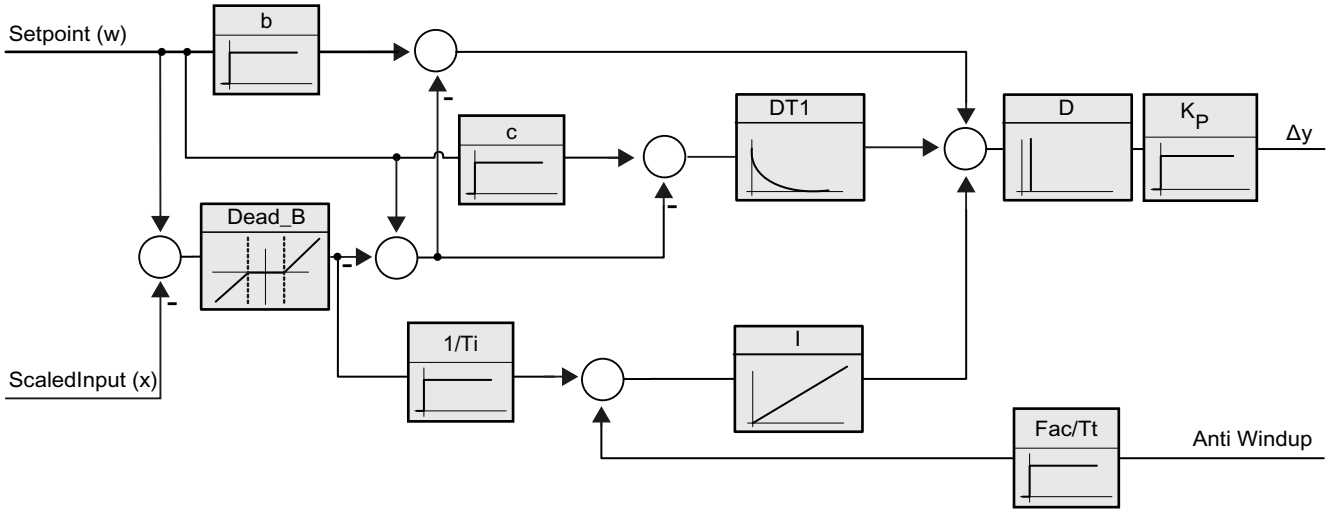
Block diagram without position feedback



Block diagram with position feedback



Block diagram of PIDT1 with anti-windup



Call

PID_3Step is called in a constant time interval of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

Download to device

The actual values of retentive tags are only updated when you download PID_3Step completely.

Downloading technology objects to device (Page 7204)

Startup

At the startup of the CPU, PID_3Step starts in the operating mode that was last active. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

Reaction to error

If errors occur, these are output in the Error parameter. You configure the reaction of PID_3Step using the ErrorBehaviour and ActivateRecoverMode tags.

ErrorBehaviour	ActivateRecoverMode	Actuator setting configuration Set Output to	Reaction
0	FALSE	Current output value	Switch to "Inactive" mode (Mode = 0)
0	TRUE	Current output value while error is pending	Switch to "Error monitoring" mode (Mode = 7)

ErrorBehaviour	ActivateRecoverMode	Actuator setting configuration Set Output to	Reaction
1	FALSE	Substitute output value	Switch to "Approach substitute output value" mode (Mode = 5) Switch to "Inactive" mode (Mode = 0)
1	TRUE	Substitute output value while error is pending	Switch to "Approach substitute output value with error monitoring" mode (Mode = 8) Switch to "Error monitoring" mode (Mode = 7)

The ErrorBits parameter shows which errors have occurred.

See also

Parameter State and Retain.Mode V1 (Page 3537)

Parameter ErrorBits V1 (Page 3544)

Configuring PID_3Step V1 (Page 7276)

Operating principle PID_3Step V1

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0001hex).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warnings = 0040hex), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID_3Step automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID_3Step checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. The output value limits must be within "Low endstop" and "High endstop".

- High endstop: Config.FeedbackScaling.UpperPointOut
- Low endstop: Config.FeedbackScaling.LowerPointOut

Rule:

UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit ≥ LowerPointOut

The valid values for "High endstop" and "Low endstop" depend upon:

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
FALSE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
FALSE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	FALSE	FALSE	Cannot be set (100.0%)	Cannot be set (100.0%)
TRUE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%

If OutputPerOn = FALSE and FeedbackOn = FALSE, you cannot limit the output value. The digital outputs are reset with Actuator_H = TRUE or Actuator_L = TRUE, or after a travel time amounting to 110% of the motor transition time.

The output value is 27648 at 100% and -27648 at -100%. PID_3Step must be able to close the valve completely. Therefore, zero must be included in the output value limits.

Substitute output value

If an error has occurred, PID_3Step can output a substitute output value and move the actuator to a safe position that is specified in the SavePosition tag. The substitute output value must be within the output value limits.

Monitoring signal validity

The values of the following parameters are monitored for validity:

- Setpoint
- Input
- Input_PER
- Feedback
- Feedback_PER
- Output

Monitoring the PID_3Step sampling time

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_3Step instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (ErrorBits = 0800 hex).

PID_3Step is set to "Inactive" mode during tuning under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

In automatic mode, PID_3Step is set to "Inactive" mode under the following conditions:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_3Step are executed at every call.

Measuring the motor transition time

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The maximum time that the actuator is moved in one direction is 110% of the motor transition time. PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ. You can measure the motor transition time during commissioning. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_3Step does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

See also

Configuring PID_3Step V1 (Page 7276)

PID_3Step V1 input parameters

Table 11-91

Parameters	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A variable of the user program is used as source for the process value. If you are using parameter Input, then Config.InputPer-On = FALSE must be set.
Input_PER	WORD	W#16#0	An analog input is used as source for the process value. If you are using parameter Input_PER, then Config.InputPer-On = TRUE must be set.
Actuator_H	BOOL	FALSE	Digital position feedback of the valve for the high endstop If Actuator_H = TRUE, the valve is at the high endstop and is no longer moved towards this direction.
Actuator_L	BOOL	FALSE	Digital position feedback of the valve for the low endstop If Actuator_L = TRUE, the valve is at the low endstop and is no longer moved towards this direction.
Feedback	REAL	0.0	Position feedback of the valve If you are using parameter Feedback, then Config.Feedback-PerOn = FALSE must be set.
Feedback_PER	WORD	W#16#0	Analog feedback of the valve position If you are using parameter Feedback_PER, then Config.FeedbackPerOn = TRUE must be set. Feedback_PER is scaled based on the variables: • Config.FeedbackScaling.LowerPointIn • Config.FeedbackScaling.UpperPointIn • Config.FeedbackScaling.LowerPointOut • Config.FeedbackScaling.UpperPointOut
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge selects "Manual mode", while State = 4, Retain.Mode remains unchanged. • A TRUE -> FALSE edge selects the most recently active operating mode A change of Retain.Mode will not take effect during ManualEnable = TRUE. The change of Retain.Mode will only be considered upon a TRUE -> FALSE edge at ManualEnable . **PID_3Step V1.1**If at start of the CPU ManualEnable = TRUE, PID_3Step starts in manual mode. A rising edge (FALSE > TRUE) at ManualEnable is not necessary. **PID_3Step V1.0** At the start of the CPU, PID_3Step only switches to manual mode with a rising edge (FALSE->TRUE) at ManualEnable . Without rising edge, PID_3Step starts in the last operating mode in which ManualEnable was FALSE.
ManualValue	REAL	0.0	In manual mode, you specify the absolute position of the valve. ManualValue will only be evaluated if you are using OutputPer, **or** if position feedback is available.

Parameters	Data type	Default	Description
Manual_UP	BOOL	FALSE	In manual mode, every rising edge opens the valve by 5% of the total control range, or for the duration of the minimum motor transition time. Manual_UP is evaluated only if you are not using Output_PER and there is no position feedback available.
Manual_DN	BOOL	FALSE	In manual mode, every rising edge closes the valve by 5% of the total control range, or for the duration of the minimum motor transition time. Manual_DN is evaluated only if you are not using Output_PER and there is no position feedback available.
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Change to "Inactive" mode – Intermediate controller values are reset (PID parameters are retained) • TRUE -> FALSE edge Change in most recent active mode

PID_3Step V1 output parameters

Table 11-92

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
ScaledFeedback	REAL	0.0	Scaled position feedback For an actuator without position feedback, the position of the actuator indicated by ScaledFeedback is very imprecise. ScaledFeedback may only be used for rough estimation of the current position in this case.
Output_UP	BOOL	FALSE	Digital output value for opening the valve If Config.OutputPerOn = FALSE, the Output_UP parameter is used.
Output_DN	BOOL	FALSE	Digital output value for closing the valve If Config.OutputPerOn = FALSE, the Output_DN parameter is used.
Output_PER	WORD	W#16#0	Analog output value If Config.OutputPerOn = TRUE, Output_PER is used.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached. In the CPU, the setpoint is limited to the configured absolute setpoint high limit. The configured absolute process value high limit is the default for the setpoint high limit. If you configure Config.SetpointUpperLimit to a value within the process value limits, this value is used as the setpoint high limit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached. In the CPU, the setpoint is limited to the configured absolute setpoint low limit. The configured absolute process value low limit is the default setting for the setpoint low limit. If you configure Config.SetpointLowerLimit to a value within the process value limits, this value is used as the setpoint low limit.

Parameter	Data type	Default	Description
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 3537) shows the current operating mode of the PID controller. You change the operating mode with the Retain.Mode tag. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Approach substitute output value • State = 6: Transition time measurement • State = 7: Error monitoring • State = 8: Approach substitute output value with error monitoring
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 3544) indicates the error messages.

See also

Parameter State and Retain.Mode V1 (Page 3537)

Parameter ErrorBits V1 (Page 3544)

PID_3Step V1 static tags

You must not change tags that are not listed. These are used for internal purposes only.

Table 11-93

Tag	Data type	Default	Description
ActivateRecoverMode	BOOL	TRUE	The ActivateRecoverMode tag (Page 3545) determines the reaction to error.
RunModeByStartup	BOOL	TRUE	Activate Mode after CPU restart If RunModeByStartup = TRUE, the controller returns to the last active operating mode after a CPU restart. If RunModeByStartup = FALSE, the controller remains inactive after a CPU restart.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., °C, or °F.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.

Tag	Data type	Default	Description
ErrorBehaviour	INT	0	If ErrorBehaviour = 0 and an error has occurred, the valve stays at its current position and the controller switches directly to "Inactive" or "Error monitoring" mode. If ErrorBehaviour = 1 and an error occurs, the actuator moves to the substitute output value and only then switches to "Inactive" or "Error monitoring" mode. If the following errors occur, you can no longer move the valve to a configured substitute output value. • 2000h: Invalid value at Feedback_PER parameter. • 4000h: Invalid value at Feedback parameter. • 8000h: Error during digital position feedback.
Warning	DWORD	DW#16#0	The Warning tag (Page 3537) displays the warnings generated since a Reset or since the last switchover of operating mode. Cyclic warnings (for example, process value warning) are shown until the cause of the warning is removed. They are automatically deleted once their cause has gone. Non-cyclic warnings (for example, point of inflection not found) remain and are deleted like errors.
SavePosition	REAL	0.0	Substitute output value If ErrorBehaviour = 1 and an error occurs, the actuator moves to a safe position for the plant and only then switches to "Inactive" mode.
CurrentSetpoint	REAL	0.0	Currently active setpoint. This value is frozen at the start of tuning.
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)
Config.InputPerOn	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.OutputPerOn	BOOL	FALSE	If OutputPerOn = TRUE, the Output_PER parameter is used. If OutputPerOn = FALSE, the Ouput_UP and Output_DN parameters are used.
Config.LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. This set was saved prior to the last tuning operation. LoadBackUp is automatically reset to FALSE.
Config.InvertControl	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.FeedbackOn	BOOL	FALSE	If FeedbackOn = FALSE, a position feedback is simulated. Position feedback is generally activated when FeedbackOn = TRUE.
Config.FeedbackPerOn	BOOL	FALSE	FeedbackPerOn is only effective when FeedbackOn = TRUE. If FeedbackPerOn = TRUE, the analog input is used for the position feedback (Feedback_PER parameter). If FeedbackPerOn = FALSE, the Feedback parameter is used for the position feedback.
Config.ActuatorEndStopOn	BOOL	FALSE	If ActuatorEndStopOn = TRUE, the digital position feedback Actuator_L and Actuator_H are taken into consideration.

Tag	Data type	Default	Description
Config.InputUpperLimit	REAL	120.0	High limit of the process value At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	+3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.402822e+38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	High limit of output value For details, see OutputLowerLimit
Config.OutputLowerLimit	REAL	0.0	Low limit of output value If OutputPerOn = TRUE or FeedbackOn = TRUE, the range of values from -100% to +100%, including zero, is valid. At -100%, Output = -27648; at +100%, Output = 27648 If OutputPerOn = FALSE, the range of values from 0% to 100% is valid. The valve is completely closed at 0% and completely opened at 100%.
Config.SetpointUpperLimit	REAL	+3.402822e+38	High limit of setpoint If you set SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is preassigned as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit	REAL	-3.402822e+38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured absolute process value low limit is preassigned as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.

Tag	Data type	Default	Description
Config.MinimumOnTime	REAL	0.0	Minimum ON time Minimum time in seconds for which the servo drive must be switched on.
Config.MinimumOffTime	REAL	0.0	Minimum OFF time Minimum time in seconds for which the servo drive must be switched off.
Config.TransitTime	REAL	30.0	Motor transition time Time in seconds the actuating drive requires to move the valve from the closed to the opened state.
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.FeedbackScaling.UpperPointIn	REAL	27648.0	Scaling Feedback_PER high Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointIn	REAL	0.0	Scaling Feedback_PER low Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.UpperPointOut	REAL	100.0	High endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointOut	REAL	0.0	Low endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
GetTransitTime.InvertDirection	BOOL	FALSE	If InvertDirection = FALSE, the valve is fully opened, closed, and then reopened in order to determine the valve transition time. If InvertDirection = TRUE, the valve is fully closed, opened, and then closed again.

Tag	Data type	Default	Description
GetTransitTime.SelectFeedback	BOOL	FALSE	If SelectFeedback = TRUE, then Feedback_PER, or Feedback is taken into consideration in the transition time measurement. If SelectFeedback = FALSE, then Actuator_H and Actuator_L are taken into consideration in the transition time measurement.
GetTransitTime.Start	BOOL	FALSE	If Start = TRUE, the transition time measurement is started.
GetTransitTime.State	INT	0	Current phase of the transition time measurement • State = 0: Inactive • State = 1: Open valve completely • State = 2: Close valve completely • State = 3: Move valve to target position (NewOutput) • State = 4: Transition time measurement successfully completed • State = 5: Transition time measurement canceled
GetTransitTime.NewOutput	REAL	0.0	Target position for transition time measurement with position feedback The target position must be between "High endstop" and "Low endstop". The difference between NewOutput and ScaledFeedback must be at least 50% of the permissible control range.
CycleTime.StartEstimation	BOOL	TRUE	If StartEstimation = TRUE, the measurement of the PID_3Step sampling time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If EnEstimation = TRUE, the PID_3Step sampling time is calculated.
CycleTime.EnMonitoring	BOOL	TRUE	If EnMonitoring = TRUE, the PID_3Step sampling time is monitored. If it is not possible to execute PID_3Step within the sampling time, the error 0800h is output and the operating mode is switched. ActivateRecoverMode and ErrorBehaviour determine which operating mode is switched to. If EnMonitoring = FALSE, the PID_3Step sampling time is not monitored, the error 0800h is not output, and the operating mode is not switched.
CycleTime.Value	REAL	0.1	PID_3Step sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser. You can reload values from the CtrlParamsBackUp structure with Config.LoadBackUp = TRUE.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time
CtrlParamsBackUp.TdFiltRatio	REAL	0.0	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	0.0	Saved proportional action weighting
CtrlParamsBackUp.DWeighting	REAL	0.0	Saved derivative action weighting
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm
CtrlParamsBackUp.InputDeadBand	REAL	0.0	Saved dead band width of the control deviation

Tag	Data type	Default	Description
PIDSelfTune.SUT.Calculate-SUTParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateSUTParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRuleSUT. CalculateSUTParams is set to FALSE following calculation.
PIDSelfTune.SUT.TuneRule-SUT	INT	1	Methods used to calculate parameters during pretuning: • TuneRuleSUT = 0: PID rapid I • TuneRuleSUT = 1: PID slow I • TuneRuleSUT = 2: Chien, Hrones and Reswick PID • TuneRuleSUT = 3TuneRuleSUT = 3: Chien, Hrones, Reswick PI • TuneRuleSUT = 4: PID rapid II • TuneRuleSUT = 5: PID slow II
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning:
PIDSelfTune.TIR.RunIn	BOOL	FALSE	• RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_3Step switches to "Inactive" mode. • RunIn = TRUE The pretuning is skipped. PID_3Step attempts to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start. RunIn is set to FALSE after fine tuning.
PIDSelfTune.TIR.Calculate-TIRParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateTIRParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRuleTIR. CalculateTIRParams is set to FALSE following calculation.
PIDSelfTune.TIR.TuneRule-TIR	INT	0	Methods used to calculate parameters during fine tuning: • TuneRuleTIR = 0: PID automatic • TuneRuleTIR = 1: PID rapid • TuneRuleTIR = 2: PID slow • TuneRuleTIR = 3: Ziegler-Nichols PID • TuneRuleTIR = 4: Ziegler-Nichols PI • TuneRuleTIR = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of "fine tuning":

Tag	Data type	Default	Description
Retain.Mode	INT	0	A change to the value of Retain.Mode initiates a switch to another operating mode. The following operating mode is enabled upon a change of Mode to: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode • Mode = 5: Approach substitute output value • Mode = 6: Transition time measurement • Mode = 7Mode = 7: Error monitoring • Mode = 8: Approach substitute output value with error monitoring Mode is retentive.
Retain.CtrlParams.SetByUser	BOOL	FALSE	If SetByUser = FALSE, the PID parameters are determined automatically and PID_3Step operates with a dead band at the output value. The dead band width is calculated during tuning on the basis of the standard deviation of the output value and saved in Retain.CtrlParams.OutputDeadBand. If SetByUser = TRUE, the PID parameters are entered manually and PID_3 Step operates without a dead band at the output value. Retain.CtrlParams.OutputDeadBand = 0.0 SetByUser is retentive.
Retain.CtrlParams.Gain	REAL	1.0	Active proportional gain Gain is retentive.
Retain.CtrlParams.Ti	REAL	20.0	• Ti > 0.0: Active integral action time • Ti = 0.0: Integral action is deactivated Ti is retentive.
Retain.CtrlParams.Td	REAL	0.0	• Td > 0.0: Active derivative action time • Td = 0.0: Derivative action is deactivated Td is retentive.
Retain.CtrlParams.TdFiltRatio	REAL	0.0	Active derivative delay coefficient TdFiltRatio is retentive.
Retain.CtrlParams.PWeighting	REAL	0.0	Active proportional action weighting PWeighting is retentive.
Retain.CtrlParams.DWeighting	REAL	0.0	Active derivative action weighting DWeighting is retentive.
Retain.CtrlParams.Cycle	REAL	1.0	Active sampling time of PID algorithm in seconds, rounded to an integer multiple of the cycle time of the calling OB. Cycle is retentive.
Retain.CtrlParams.InputDeadBand	REAL	0.0	Dead band width of the control deviation InputDeadBand is retentive.

Note

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller. "Inactive" mode is forced by setting the "Retain.Mode" tag to "0".

See also

Parameter State and Retain.Mode V1 (Page 3537)

Tag ActivateRecoverMode V1 (Page 3545)

Downloading technology objects to device (Page 7204)

Parameter State and Retain.Mode V1

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

To switch from one operating mode to another, you must change the Retain.Mode tag. This also applies when the value for the new operating mode is already in Retain.Mode. For example, set Retain.Mode = 0 first and then Retain.Mode = 3. Provided the current operating mode of the controller permits this switchover, State will be set to the value of Retain.Mode.

When PID_3Step automatically switches from one operating mode to another, the following applies: State != Retain.Mode.

Examples:

- After successful pretuning
  State = 3 and Retain.Mode = 1
- In the event of an error
  State = 0 and Retain.Mode remain at the previous value, for example, Retain.Mode = 3
- ManualEnalbe = TRUE
  State = 4 and Retain.Mode remain at the previous value, e.g., Retain.Mode = 3

Note

You want, for example, to repeat successful fine tuning without exiting automatic mode with Mode = 0.

Setting Retain.Mode to an invalid value such as 9999 for one cycle has no effect on State. Set Mode = 2 in the next cycle. In this way, you can generate a change to Retain.Mode without first switching to "Inactive" mode.

Meaning of values

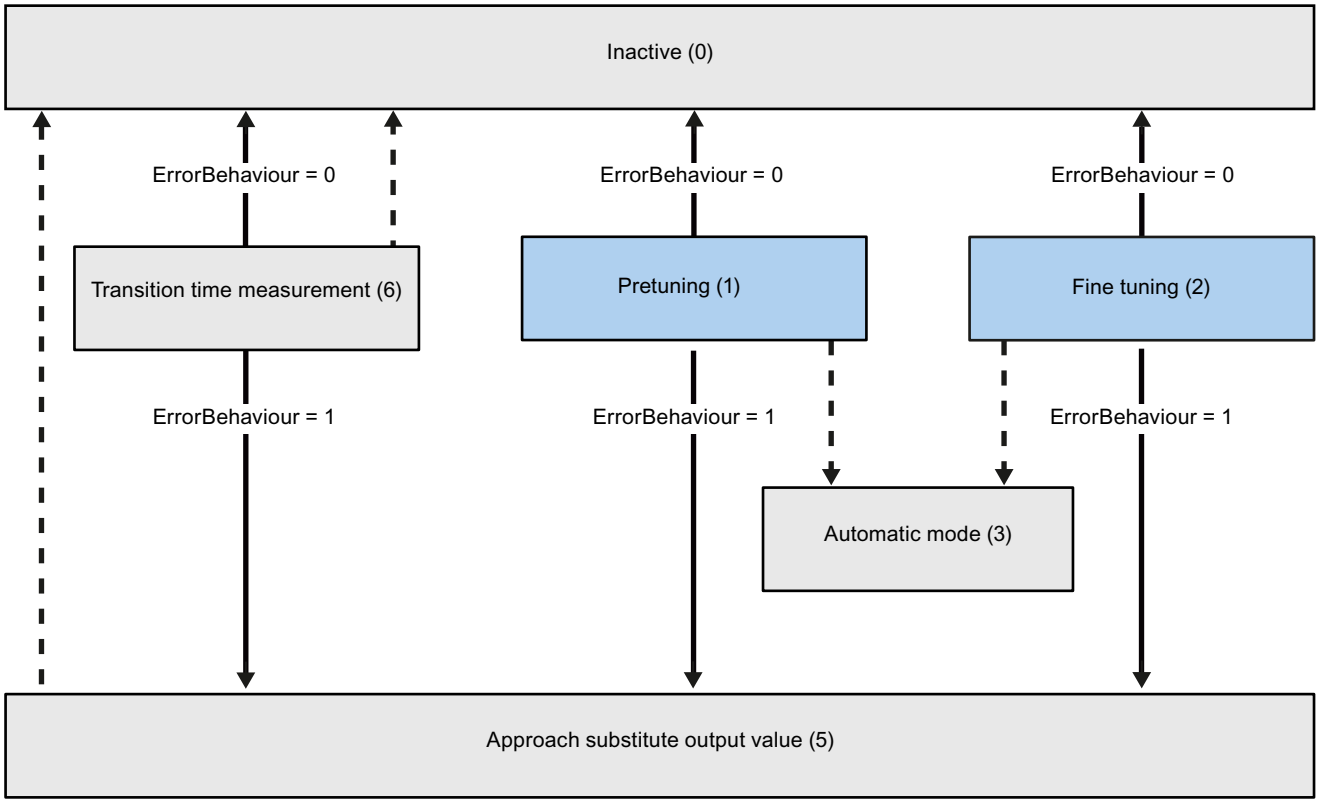
State / Retain.Mode	Description
0	Inactive The controller is switched off and no longer changes the valve position.
1	Pretuning The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The optimized PID parameters are calculated as a function of the maximum rate of rise and dead time of the controlled system. Pretuning requirements: • State = 0 or State = 4 • ManualEnable = FALSE • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher as compared to the noise. Before the PID parameters are recalculated, they are backed up and can be reactivated with Config.Load-BackUp. The setpoint is frozen in the CurrentSetpoint tag. The controller switches to automatic mode following successful pretuning and to "Inactive" mode following unsuccessful pretuning. The pretuning phase is indicated with the SUT.State tag.

State / Retain.Mode	Description
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned based on the amplitude and frequency of this oscillation. The differences between the process response during pretuning and fine tuning are analyzed. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before fine tuning. They can be reactivated with Config.LoadBackUp. The setpoint is frozen in the CurrentSetpoint tag. Requirements for fine tuning: • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. • ManualEnable = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started from: • Automatic mode (State = 3) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual mode (State = 4) Pretuning is always started first. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. If PIDSelfTune.TIR.RunIn = TRUE, pretuning is skipped and an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. The controller switches to automatic mode following successful fine tuning. If fine tuning was not successful, the controller switches to "Inactive" mode. The fine tuning phase is indicated with the TIR.State tag.
3	Automatic mode In automatic mode, PID_3Step controls the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Changing the Retain.Mode tag to the value 3. When the CPU is switched on or switches from Stop to RUN mode, PID_3Step starts in the most recently active operating mode. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE. The ActivateRecoverMode tag is taken into consideration in automatic mode.

State / Retain.Mode	Description
4	Manual mode In manual mode, you specify manual output values in the Manual_UP and Manual_DN parameters or ManualValue parameter. Whether or not the actuator can be moved to the output value in the event of an error is described in the ErrorBits parameter. This operating mode is enabled if Retain.Mode = 4, or on a rising edge at ManualEnable. If ManualEnable changes to TRUE, only State changes. Retain.Mode retains its current value. On a falling edge at ManualEnable, PID_3Step returns to the previous operating mode. The switchover to automatic mode is bumpless. **PID_3Step V1.1** Manual mode is always possible in the event of an error. **PID_3Step V1.0** Manual mode is dependent on the ActivateRecoverMode tag in the event of an error.
5	Approach substitute output value This operating mode is activated in the event of an error or when Reset = TRUE if Errorbehaviour = 1 and ActivateRecoverMode = FALSE.. PID_3Step moves the actuator to the substitute output value and then switches to "Inactive" mode.
6	Transition time measurement The time that the motor needs to completely open the valve from the closed condition is determined. This operating mode is activated when GetTransitTime.Start = TRUE is set. If endstop signals are used to measure the transition time, the valve will be opened completely from its current position, closed completely, and opened completely again. If GetTransitTime.InvertDirection = TRUE, this behavior is inverted. If position feedback is used to measure the transition time, the actuator will be moved from its current position to a target position. The output value limits are not taken into consideration during the transition time measurement. The actuator can travel to the high or the low endstop.
7	Error monitoring The control algorithm is switched off and no longer changes the valve position. This operating mode is activated instead of "Inactive" mode in the event of an error. All the following conditions must be met: • Mode = 3 (automatic mode) • Errorbehaviour = 0 • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 3545) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.
8	Approach substitute output value with error monitoring This operating mode is activated instead of "Approach substitute output value" mode in the event of an error. PID_3Step moves the actuator to the substitute output value and then switches to "Error monitoring" mode. All the following conditions must be met: • Mode = 3 (automatic mode) • Errorbehaviour = 1 • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 3545) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

Automatic switchover of operating mode during commissioning

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour on the switchover of operating mode from transition time measurement, pretuning, and fine tuning modes.

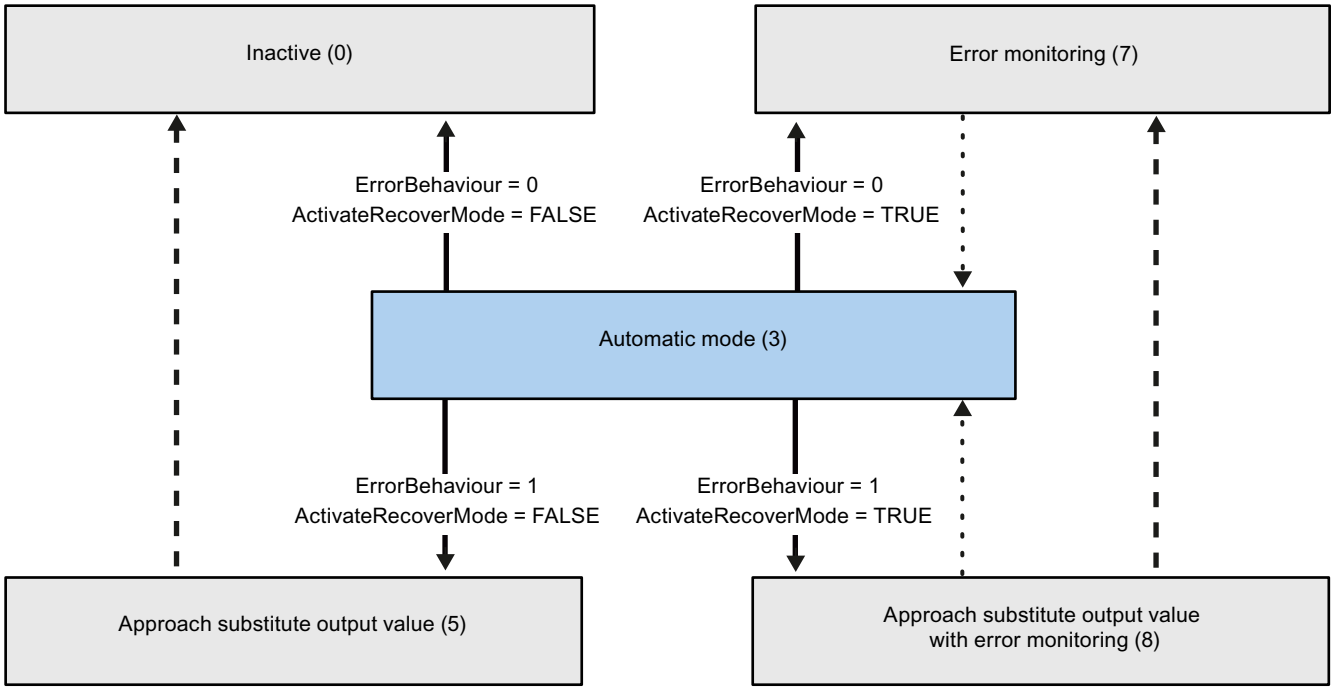


Automatic switchover of operating mode in the event of an error

Automatic switchover of operating mode once the current operation has been completed.

Automatic switchover of operating mode in automatic mode (PID_3Step V1.1)

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.

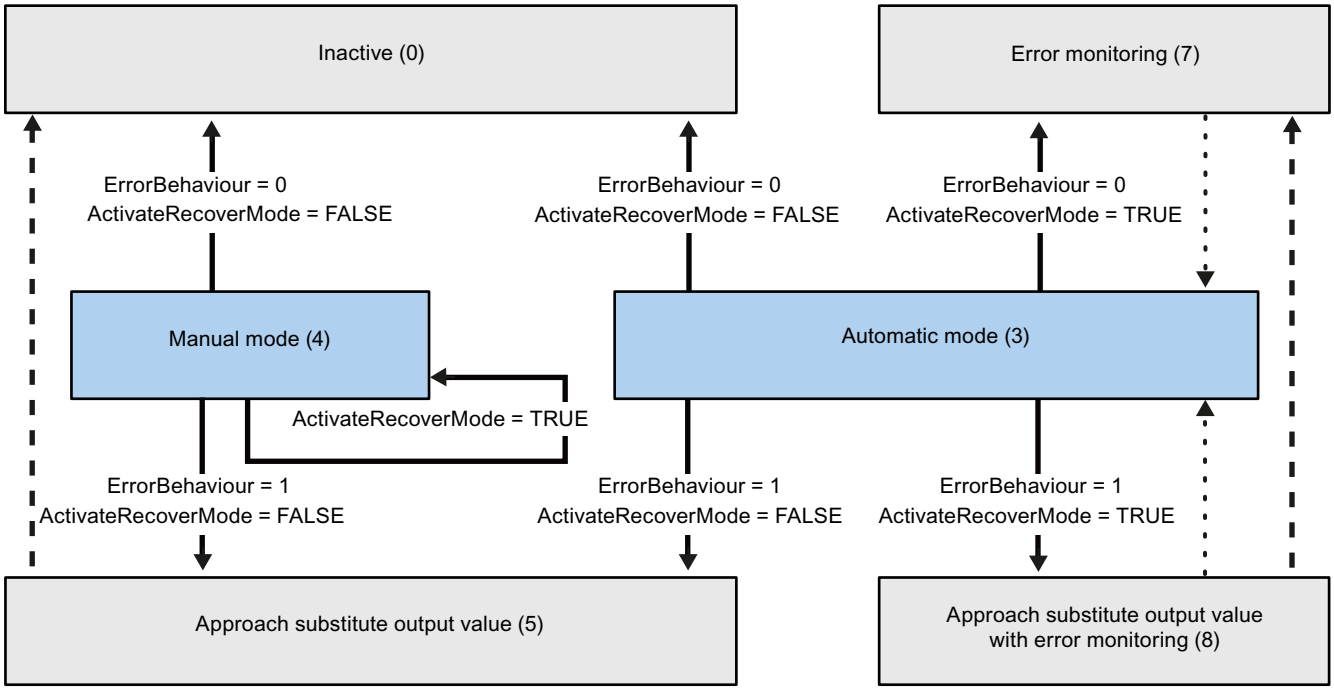


Automatic switchover of operating mode in the event of an error

Automatic switchover of operating mode once the current operation has been completed.

Automatic switchover of operating mode in automatic and manual modes (PID_3Step V1.0)

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.



Automatic switchover of operating mode in the event of an error

Automatic switchover of operating mode once the current operation has been completed.

Automatic switchover of operating mode when error is no longer pending.

See also

Tag ActivateRecoverMode V1 (Page 3545)

Parameter ErrorBits V1 (Page 3544)

Parameter ErrorBits V1

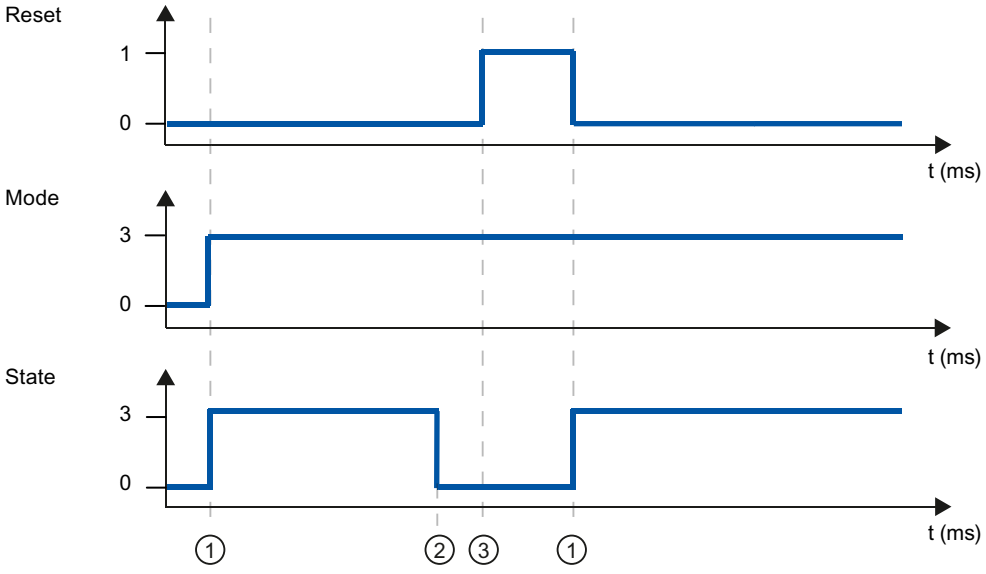
If several errors are pending simultaneously, the values of the error codes are displayed with binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending simultaneously.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If ActivateRecoverMode = TRUE and ErrorBehaviour = 1, the actuator moves to the substitute output value. If ActivateRecoverMode = TRUE and ErrorBehaviour = 0, the actuator stops in its current position. If ActivateRecoverMode = FALSE, the actuator stops in its current position. PID_3Step V1.1 You can move the actuator in manual mode. PID_3Step V1.0 Manual mode is not possible in this state. You cannot move the actuator again until you eliminate the error.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
0004	Error during fine tuning. Oscillation of the process value could not be maintained.
0020	Pretuning is not permitted in automatic mode or during fine tuning.
0080	Error during pretuning. Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic.
0100	Error during fine tuning resulted in invalid parameters.
0200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
0400	Calculation of output value failed. Check the PID parameters.
0800	Sampling time error: PID_3Step is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
2000	Invalid value at Feedback_PER parameter. Check whether an error is pending at the analog input. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. You must deactivate position feedback (Config. FeedbackOn = FALSE) to move the actuator from this state. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.

ErrorBits (DW#16#...)	Description
4000	Invalid value at Feedback parameter. Value has an invalid number format. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. You must deactivate position feedback (Config. FeedbackOn = FALSE) to move the actuator from this state. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
8000	Error during digital position feedback. Actuator_H = TRUE and Actuator_L = TRUE. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. In order to move the actuator from this state, you must deactivate the "Actuator endstop" (Config.ActuatorEndStopOn = FALSE). If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.

Parameter Reset V1

A rising edge at Reset resets the errors and warnings and clears the integral action. A falling edge at Reset triggers a change to the most recently active operating mode.



① Activation
② Error
③ Reset

Tag ActivateRecoverMode V1

The effect of the ActivateRecoverMode variable depends on the version of the PID_3Step.

Behavior in version 1.1

The ActivateRecoverMode variable determines the behavior in the event of an error in automatic mode. ActivateRecoverMode is not effective during pretuning, fine tuning and transition time measurement.

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" operating mode. The controller is activated by a reset or a change in Retain.Mode.
TRUE	If errors occur frequently in automatic mode, this setting has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, PID_3Step **cannot** approach the configured substitute output value. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

Behavior in version 1.0

The ActivateRecoverMode variable determines the behavior in the event of an error in automatic and manual mode. ActivateRecoverMode is not effective during pretuning, fine tuning and transition time measurement.

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" operating mode. The controller is activated by a reset or a change in Retain.Mode.
TRUE	**Errors in automatic mode** If errors occur frequently in automatic mode, this setting has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, PID_3Step **cannot** approach the configured substitute output value. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode. **Errors in manual mode** If one or more of the following errors occur, PID_3Step stays in manual mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, you **cannot** move the valve to a suitable position.

See also

PID_3Step V1 static tags (Page 3530)

Parameter State and Retain.Mode V1 (Page 3537)

Tag Warning V1

If several warnings are pending simultaneously, their values are displayed with binary addition. The display of warning 0003, for example, indicates that the warnings 0001 and 0002 are pending simultaneously.

Warning (DW#16#...)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0002	Oscillation increased during fine tuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. The PID parameters were instead calculated using the TuneRuleTIR = 3 method.
0010	The operating mode could not be changed because ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Retain.Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The rule used for tuning produces an incorrect result, or is not supported.
0400	Method selected for transition time measurement not suitable for actuator. The transition time cannot be measured because the actuator settings do not match the selected measuring method.
0800	The difference between the current position and the new output value is too small for transition time measurement. This can produce incorrect results. The difference between the current output value and new output value must be at least 50% of the entire control range.
1000	The substitute output value cannot be reached because it is outside the output value limits.

The following warnings are deleted as soon as the cause is eliminated:

- 0004
- 0020
- 0040
- 0100

All other warnings are cleared with a rising edge at Reset.

Tag SUT.State V1

SUT.State	Name	Description
0	SUT_INIT	Initialize pretuning
50	SUT_TPDN	Determine start position without position feedback
100	SUT_STDABW	Calculate the standard deviation
200	SUT_GET_POI	Find the point of inflection
300	SUT_GET_RISETM	Determine the rise time

SUT.State	Name	Description
9900	SUT_IO	Pretuning successful
1	SUT_NIO	Pretuning not successful

Tag TIR.State V1

TIR.State	Name	Description
-100	TIR_FIRST_SUT	Fine tuning is not possible. Pretuning will be executed first.
0	TIR_INIT	Initialize fine tuning
200	TIR_STDABW	Calculate the standard deviation
300	TIR_RUN_IN	Attempt to reach the setpoint with the maximum or minimum output value
400	TIR_CTRLN	Attempt to reach the setpoint with the existing PID parameters (if pretuning has been successful)
500	TIR_OSZIL	Determine oscillation and calculate parameters
9900	TIR_IO	Fine tuning successful
1	TIR_NIO	Fine tuning not successful

PID_Temp

Compatibility with CPU and FW

The following table shows which version of PID_Temp can be used on which CPU.

CPU	FW	PID_Temp
S7-1200	≥ V4.1	V1.0
S7-1500	≥ V1.7	V1.0

CPU processing time and memory requirement PID_Temp V1

CPU processing time

Typical CPU processing times of the PID_Temp technology object as of Version 1.0, depending on CPU type.

CPU	Typ. CPU processing time PID_Temp V1
CPU 1211C ≥ V4.1	580 µs
CPU 1215C ≥ V4.1	580 µs
CPU 1217C ≥ V4.1	580 µs
CPU 1505S ≥ V1.0	50 µs
CPU 1510SP-1 PN ≥ V1.7	130 µs
CPU 1511-1 PN ≥ V1.7	130 µs
CPU 1512SP-1 PN ≥ V1.7	130 µs

CPU 1516-3 PN/DP ≥ V1.7	75 µs
CPU 1518-4 PN/DP ≥ V1.7	6 µs

Memory requirement

Memory requirement of an instance DB of the PID_Temp technology object as of Version V1.0.

	Memory requirement of the instance DB of PID_Temp V1
Load memory requirement	Approx. 17000 bytes
Total work memory requirement	1280 bytes
Retentive work memory requirement	100 bytes

PID_Temp

Description of PID_Temp

Description

The PID_Temp instruction provides a PID controller with integrated tuning for temperature processes. PID_Temp can be used for pure heating or heating/cooling applications.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Substitute output value with error monitoring

For a more detailed description of the operating modes, see the State parameter.

PID algorithm

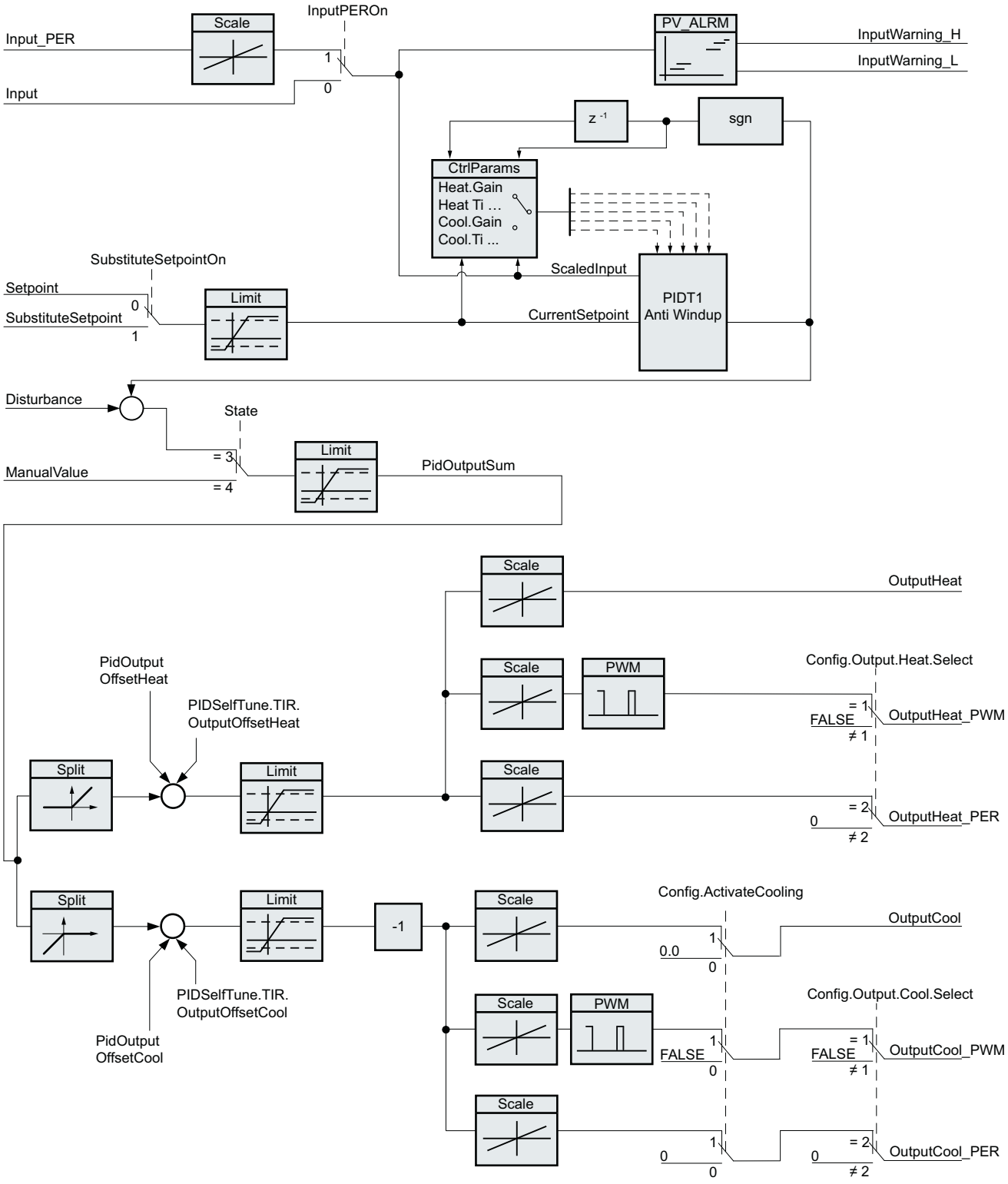
PID_Temp is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation (control zone and deadband deactivated):

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

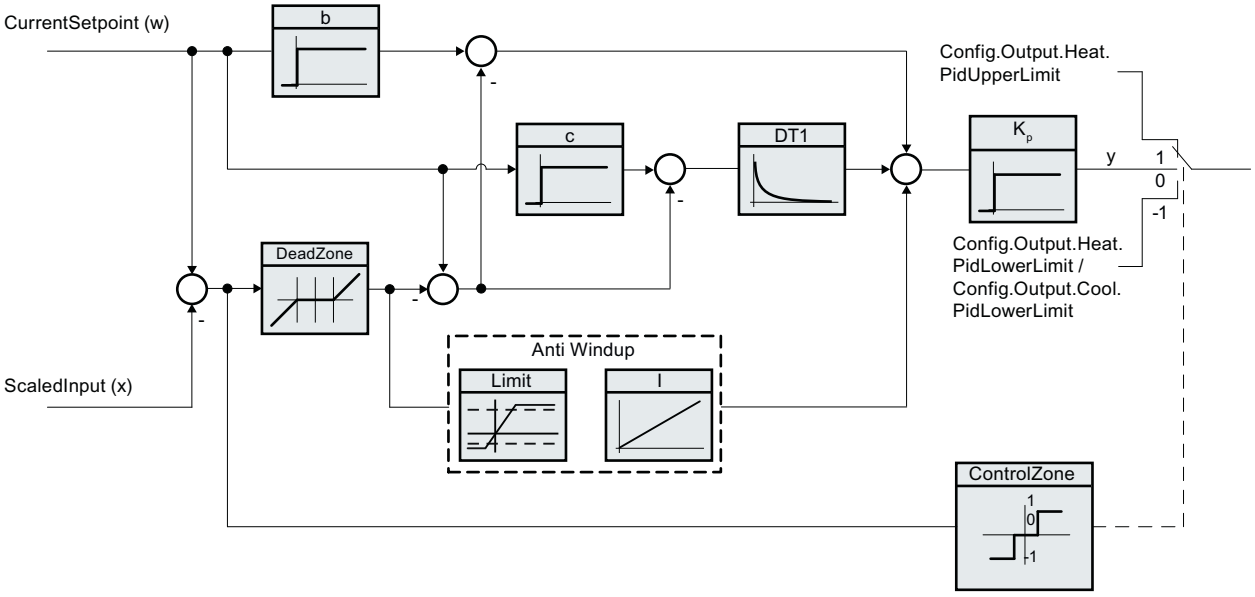
The table below shows the meaning of the icons used in the equation and in the subsequent figures.

Icon	Description	Associated parameters of the PID_Temp instruction
y	Output value of the PID algorithm	-
K_p	Proportional gain	Retain.CtrlParams.Heat.Gain Retain.CtrlParams.Cool.Gain CoolFactor
s	Laplace operator	-
b	Proportional action weighting	Retain.CtrlParams.Heat.PWeighting Retain.CtrlParams.Cool.PWeighting
w	Setpoint	CurrentSetpoint
x	Process value	ScaledInput
T_I	Integral action time	Retain.CtrlParams.Heat.Ti Retain.CtrlParams.Cool.Ti
T_D	Derivative action time	Retain.CtrlParams.Heat.Td Retain.CtrlParams.Cool.Td
a	Derivative delay coefficient (derivative delay T1 = a × T_D)	Retain.CtrlParams.Heat.TdFiltRatio Retain.CtrlParams.Cool.TdFiltRatio
c	Derivative action weighting	Retain.CtrlParams.Heat.DWeighting Retain.CtrlParams.Cool.DWeighting
DeadZone	Deadband width	Retain.CtrlParams.Heat.DeadZone Retain.CtrlParams.Cool.DeadZone
ControlZone	Control zone width	Retain.CtrlParams.Heat.ControlZone Retain.CtrlParams.Cool.ControlZone

PID_Temp block diagram



Block diagram of PIDT1 with anti-windup



Call

PID_Temp is called in the constant time scale of a cyclic interrupt OB.

If you call PID_Temp as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_Temp directly in the multi-instance DB and commission it via a watch table.

Download to device

The process values of retentive variables are only updated when you download PID_Temp completely.

Download technology object to device (Page 7204)

Startup

When the CPU starts up, PID_Temp starts in the operating mode that is saved in the Mode in/out parameter. To switch to "Inactive" operating mode during startup, set RunModeByStartup = FALSE.

Reaction to error

The behavior in the case of an error is determined by the tags SetSubstituteOutput and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the behavior also depends on the error that occurred.

SetSubstituteOutput	ActivateRecoverMode	Configuration editor > Basic settings of output > Set PidOutputSum to	Reaction
Not relevant	FALSE	Zero (Inactive)	Switch to "Inactive" (State = 0) mode The output value of the PID algorithm and all outputs for heating and cooling are set to 0. The scaling of the outputs for heating and cooling is not active.
FALSE	TRUE	Current value for error while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The current output value is transferred to the actuator while the error is pending.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The value at SubstituteOutput is transferred to the actuator while the error is pending.

In manual mode, PID_Temp uses ManualValue as output value, unless ManualValue is invalid.

- If ManualValue is invalid, SubstituteOutput is used.
- If ManualValue and SubstituteOutput are invalid, Config.Output.Heat.PidLowerLimit is used.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.

Functional description of PID_Temp

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0000001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warning = 0000040h), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID_Temp automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller area. PID_Temp checks whether this area is within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Substitute setpoint

You can specify a substitute setpoint at the SubstituteSetpoint tag and activate it with SubstituteSetpointOn = TRUE. In this way, you can temporarily specify the setpoint directly, for example for a slave controller in a cascade, without having to change the user program. The limits set for the setpoint also apply to the substitute setpoint.

Heating and cooling

With the default setting, PID_Temp only uses the outputs for heating (OutputHeat, OutputHeat_PWM, OutputHeat_PER). The output value of the PID algorithm (PidOutputSum) is scaled and output at the outputs for heating. You specify with Config.Output.Heat.Select if OutputHeat_PWM or OutputHeat_PER is calculated. OutputHeat is always calculated.

With Config.ActivateCooling = TRUE, you can also activate the outputs for cooling (OutputCool, OutputCool_PWM, OutputCool_PER). Positive output values of the PID algorithm (PidOutputSum) are scaled and output at the outputs for heating. Negative output values of the PID algorithm are scaled and output at the outputs for cooling. You specify with Config.Output.Cool.Select if OutputCool_PWM or OutputCool_PER is calculated. OutputCool is always calculated.

Two methods are available to calculate the PID output value with activated cooling:

- Cooling factor (Config.AdvancedCooling = FALSE):
  The output value calculation for cooling takes place with the PID parameters for heating, taking into consideration the configurable cooling factor Config.CoolFactor. This method is suitable if the heating and cooling actuator have similar time responses but different gains. When you select this method, pretuning and fine tuning for cooling as well as the PID parameter set for cooling are not available. You can only execute the tuning for heating.
- PID parameter switching (Config.AdvancedCooling = TRUE):
  The output value calculation for cooling takes place by means of a separate PID parameter set. Based on the calculated output value and the control deviation, the PID algorithm decides whether the PID parameter for heating or cooling is used. This method is suitable if the heating and cooling actuator have different time responses and different gains. Pretuning and fine tuning for cooling are only available when you select this method.

Output value limits and scaling

Depending on the operating mode, the PID output value (PidOutputSum) is calculated automatically by the PID algorithm or defined by the manual value (ManualValue) or the configured substitute output value (SubstituteOutput).

The PID output value is limited according to the configuration:

- If cooling is deactivated (Config.ActivateCooling = FALSE), Config.Output.Heat.PidUpperLimit is the high limit and Config.Output.Heat.PidLowerLimit the low limit.
- If cooling is activated (Config.ActivateCooling = TRUE), Config.Output.Heat.PidUpperLimit is the high limit and Config.Output.Cool.PidLowerLimit the low limit.

The PID output value is scaled and output at the outputs for heating and cooling. Scaling can be defined separately for each output and is specified in the structures Config.Output.Heat or Config.Output.Cool with 2 value pairs each:

Output	Value pair	Parameter
OutputHeat	Value pair 1	High limit PID output value (heating) Config.Output.Heat.PidUpperLimit, Scaled high output value (heating) Config.Output.Heat.UpperScaling
	Value pair 2	Low limit PID output value (heating) Config.Output.Heat.PidLowerLimit, Scaled low output value (heating) Config.Output.Heat.LowerScaling
OutputHeat_PWM	Value pair 1	High limit PID output value (heating) Config.Output.Heat.PidUpperLimit, Scaled high PWM output value (heating) Config.Output.Heat.PwmUpperScaling
	Value pair 2	Low limit PID output value (heating) Config.Output.Heat.PidLowerLimit, Scaled low PWM output value (heating) Config.Output.Heat.PwmLowerScaling
OutputHeat_PER	Value pair 1	High limit PID output value (heating) Config.Output.Heat.PidUpperLimit, Scaled high analog output value (heating) Config.Output.Heat.PerUpperScaling
	Value pair 2	Low limit PID output value (heating) Config.Output.Heat.PidLowerLimit, Scaled low analog output value (heating) Config.Output.Heat.PerLowerScaling
OutputCool	Value pair 1	Low limit PID output value (cooling) Config.Output.Cool.PidLowerLimit, Scaled high output value (cooling) Config.Output.Cool.UpperScaling
	Value pair 2	High limit PID output value (cooling) Config.Output.Cool.PidUpperLimit, Scaled low output value (cooling) Config.Output.Cool.LowerScaling

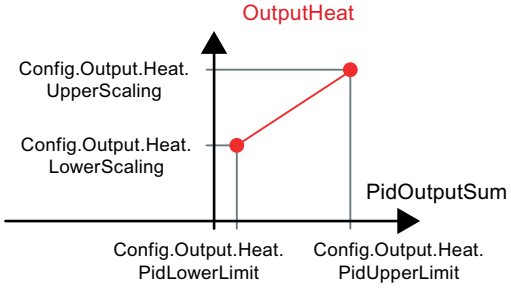
Output	Value pair	Parameter
OutputCool_PWM	Value pair 1	Low limit PID output value (cooling) Config.Output.Cool.PidLowerLimit, Scaled high PWM output value (cooling) Config.Output.Cool.PwmUpperScaling
	Value pair 2	High limit PID output value (cooling) Config.Output.Cool.PidUpperLimit, Scaled low PWM output value (cooling) Config.Output.Cool.PwmLowerScaling
OutputCool_PER	Value pair 1	Low limit PID output value (cooling) Config.Output.Cool.PidLowerLimit, Scaled high analog output value (cooling) Config.Output.Cool.PerUpperScaling
	Value pair 2	High limit PID output value (cooling) Config.Output.Cool.PidUpperLimit, Scaled low analog output value (cooling) Config.Output.Cool.PerLowerScaling

If cooling is activated (Config.ActivateCooling = TRUE), Config.Output.Heat.PidLowerLimit must have the value 0.0.

Config.Output.Cool.PidUpperLimit must always have the value 0.0.

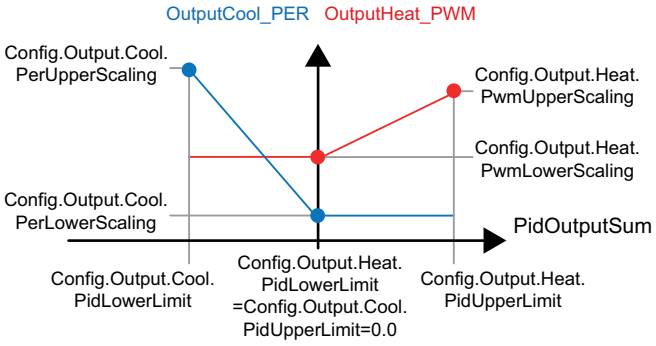
Example:

Output scaling when using output OutputHeat (cooling deactivated; Config.Output.Heat.PidLowerLimit may be unequal to 0.0):



Example:

Output scaling when using output OutputHeat_PWM and OutputCool_PER (cooling activated; Config.Output.Heat.PidLowerLimit must be 0.0):



With the exception of the "Inactive" operating mode, the value at an output is always located between its scaled high output value and scaled low output value, for example, for OutputHeat always between Config.Output.Heat.UpperScaling and Config.Output.Heat.LowerScaling.

If you want to limit the value at the associated output, you must also adjust these scaling values.

Cascading

PID_Temp supports you when you use cascade control (see: Program creation (Page 7323)).

Substitute output value

In the event of an error, PID_Temp can output a substitute output value that you define at the SubstituteOutput tag. The substitute output value must be within the limits for the PID output value. The values at the outputs for heating and cooling resulting from the substitute output value are the result of the configured output scaling.

Monitoring signal validity

The values of the following parameters are monitored for validity when used:

- Setpoint
- SubstituteSetpoint
- Input
- Input_PER
- Disturbance
- ManualValue
- SubstituteOutput
- PID parameters in the structures Retain.CtrlParams.Heat and Retain.CtrlParams.Cool.

Monitoring the sampling time PID_Temp

Ideally, the sampling time is equivalent to the cycle time of the cyclic interrupt OB. The PID_Temp instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (Error = 0000800h).

The error occurs during tuning if:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

The error occurs in automatic mode if:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID_Temp in OB1. You must then accept a lower control quality due to the deviating sampling time.

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time of the cyclic interrupt OB (sampling time PID_Temp). All other functions of the PID_Temp are executed at every call.

If cooling and PID parameter switching are activated, PID_Temp uses a separate sampling time of the PID algorithm for heating and cooling. In all other configurations, only the sampling time of the PID algorithm for heating is used.

If you use OutputHeat_PWM or OutputCool_PWM, the sampling time of the PID algorithm is used as time period of the pulse width modulation. The accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The cycle time should be no more than a tenth of the PID algorithm sampling time.

If the PID algorithm sampling time and thus the time period of the pulse width modulation is very high when you use OutputHeat_PWM or OutputCool_PWM, you can define a deviating shorter time period at the Config.Output.Heat.PwmPeriode or Config.Output.Cool.PwmPeriode parameters to improve the smoothness of the process value.

Control logic

PID_Temp can be used for heating or heating/cooling applications and always works with normal control logic.

An increase of the PID output value (PidOutputSum) is intended to increase the process value. The values at the outputs for heating and cooling resulting from the PID output value are the result of the configured output scaling.

An inverted control logic or negative proportional gain are not supported.

If you only need an output value for your application in which an increase is to reduce the process value (for example, discharge control), you can use PID_Compact with inverted control logic.

Input parameters of PID_Temp

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode Valid range of values: Config.SetpointUpperLimit ≥ Setpoint ≥ Config.SetpointLowerLimit Config.InputUpperLimit ≥ Setpoint ≥ Config.InputLowerLimit
Input	REAL	0.0	A tag of the user program is used as the source of the process value. If you are using the Input parameter, Config.InputPerOn = FALSE must be set.
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using the Input_PER parameter, Config.InputPerOn = TRUE must be set.
Disturbance	REAL	0.0	Disturbance variable or precontrol value
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge activates "Manual mode", State = 4, Mode remains unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog. • A TRUE -> FALSE edge activates the operating mode that is specified by Mode. We recommend that you change the operating mode using Mode and ModeActivate only.
ManualValue	REAL	0.0	Manual value This value is used in manual mode as PID output value (PidOutputSum). The values at the outputs for heating and cooling resulting from this manual value are the result of the configured output scaling (structures Config.Output.Heat and Config.Output.Cool). For controllers with activated cooling output (Config.ActivateCooling = TRUE), define: • a positive manual value to output the value at the outputs for heating • a negative manual value to output the value at the outputs for cooling The permitted value range is determined by the configuration. • Cooling output deactivated (Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ ManualValue ≥ Config.Output.Heat.PidLowerLimit • Cooling output activated (Config.ActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ ManualValue ≥ Config.Output.Cool.PidLowerLimit
ErrorAck	BOOL	FALSE	• FALSE -> TRUE edge ErrorBits and Warning are reset.

Parameter	Data type	Default	Description
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Switch to "Inactive" mode – ErrorBits and Warning are reset. – Integral action is cleared (PID parameters are retained) • As long as Reset = TRUE, – PID_Tempremains in "Inactive" mode (State = 0). – you cannot change the operating mode with Mode and ModeActivate or ManualEnable – you cannot use the commissioning dialog. • TRUE -> FALSE edge PID_Temp switches to the operating mode that is saved in the Mode parameter.
ModeActivate	BOOL	FALSE	• FALSE -> TRUE edge PID_Temp switches to the operating mode that is saved at the Mode input.

Output parameters of PID_Temp

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
OutputHeat	REAL	0.0	Output value (heating) in REAL format The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Heat.PidUpperLimit, Config.Output.Heat.UpperScaling and Config.Output.Heat.PidLowerLimit, Config.Output.Heat.LowerScaling and output in REAL format at OutputHeat. OutputHeat is always calculated.
OutputCool	REAL	0.0	Output value (cooling) in REAL format The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Cool.PidUpperLimit, Config.Output.Cool.LowerScaling and Config.Output.Cool.PidLowerLimit, Config.Output.Cool.UpperScaling and output in REAL format at OutputCool. OutputCool is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE).
OutputHeat_PER	INT	0	Analog output value (heating) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Heat.PidUpperLimit, Config.Output.Heat.PerUpperScaling and Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PerLowerScaling and output as analog value at OutputHeat_PER. OutputHeat_PER is only calculated if Config.Output.Heat.Select = 2.

Parameter	Data type	Default	Description
Output-Cool_PER	INT	0	Analog output value (cooling) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Cool.PidUpperLimit, Config.Output.Cool.PerLowerScaling and Config.Output.Cool.PidLowerLimit, Config.Output.Cool.PerUpperScaling and output as analog value at OutputCool_PER. OutputCool_PER is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE) and Config.Output.Cool.Select = 2.
OutputHeat_PWM	BOOL	FALSE	Pulse-width modulated output value (heating) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Heat.PidUpperLimit, Config.Output.Heat.PwmUpperScaling and Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PwmLowerScaling and output as pulse-width modulated value (variable switch on and switch off times) at OutputHeat_PWM. OutputHeat_PWM is only calculated if Config.Output.Heat.Select = 1.
Output-Cool_PWM	BOOL	FALSE	Pulse-width modulated output value (cooling) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Cool.PidUpperLimit, Config.Output.Cool.PwmLowerScaling and Config.Output.Cool.PidLowerLimit, Config.Output.Cool.PwmUpperScaling and output as pulse-width modulated value (variable switch on and switch off times) at OutputCool_PWM. OutputCool_PWM is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE) and Config.Output.Cool.Select = 1.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit) or Setpoint ≥ Config.InputUpperLimit. The setpoint high limit is the minimum of Config.SetpointUpperLimit and Config.InputUpperLimit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit is reached (Setpoint ≤ Config.SetpointLowerLimit) or Setpoint ≤ Config.InputLowerLimit. The setpoint low limit is the maximum of Config.SetpointLowerLimit and Config.InputLowerLimit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit (ScaledInput ≥ Config.InputUpperWarning).
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit (ScaledInput ≤ Config.InputLowerWarning).
State	INT	0	The PID_Temp state and mode parameters (Page 3591) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. For pretuning and fine tuning, you specify with Heat.EnableTuning and Cool.EnableTuning whether tuning takes place for heating or cooling. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Substitute output value with error monitoring

Parameter	Data type	Default	Description
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The PID_Temp ErrorBits parameter (Page 3599) shows the pending error messages. ErrorBits is retentive and is reset with a rising edge at Reset or ErrorAck.

PID_Temp in/out parameters

Parameter	Data type	Default	Description
Mode	INT	4	At Mode, specify the operating mode to which PID_Temp is to switch. Options are: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode The operating mode is activated by: • Rising edge at ModeActivate • Falling edge at Reset • Falling edge at ManualEnable • Cold restart of CPU if RunModeByStartup = TRUE For pretuning and fine tuning, you specify with Heat.EnableTuning and Cool.EnableTuning whether tuning takes place for heating or cooling. Mode is retentive. A detailed description of the operating modes can be found in State and Mode parameters (Page 3591).
Master	DWORD	DW#16#0	Interface for cascade control If this PID_Temp instance is used as slave controller in a cascade (Config.Cascade.IsSlave = TRUE), assign the Master parameter at the instruction call with the Slave parameter of the master controller. Example: Call of a slave controller "PID_Temp_2" with master controller "PID_Temp_1" in SCL: --- `"PID_Temp_2"(Master := "PID_Temp_1".Slave, Setpoint := "PID_Temp_1".OutputHeat);` --- You use this interface to exchange slave controller information about operating mode, limit and substitute setpoint with your master controller. Keep in mind that the call of the master controller has to take place before the call of the slave controller in the same cyclic interrupt OB. Assignment: • Bits 0 to 15: Unassigned • Bits 16 to 23 – Limit counter: A slave controller whose output value is limited increments this counter. Depending on the configured number of slaves (Config.Cascade.CountSlaves) and of the anti-windup mode (Config.Cascade.AntiWindUpMode), the master controller reacts accordingly. • Bit 24 – Automatic mode of the slave controllers: TRUE, if all slave controllers are in automatic mode • Bit 25 – Substitute setpoint of the slave controllers: TRUE, if a slave controller has activated the substitute setpoint (SubstituteSetpointOn = TRUE)

Parameter	Data type	Default	Description
Slave	DWORD	DW#16#0	Interface for cascade control You use this interface to exchange slave controller information about operating mode, limit and substitute setpoint with your master controller. See description of Master parameter

See also

PID_Temp state and mode parameters (Page 3591)

Program creation (Page 7323)

Cascade control with PID_Temp (Page 7321)

PID_Temp static tags

You must not change tags that are not listed. These are used for internal purposes only.

Tag	Data type	Default	Description
IntegralResetMode	Int	1	The IntegralResetMode tag determines the default setting of the integral action PIDCtrl.IOutputOld when you change the operating mode from "Inactive" to "Automatic mode". This setting only works for one cycle. • IntegralResetMode = 0: Smoothing The value is assigned in such a way that the switchover is bumpless. • IntegralResetMode = 1: Deleting The value is cleared. Any control deviation will cause a jump change of the output value. • IntegralResetMode = 2: Holding The value is not changed. You can define a new value using the user program. • IntegralResetMode = 3: Pre-assigning The value is automatically pre-assigned as if PidOutputSum = OverwriteInitialOutputValue in the last cycle. This setting is useful, for example, for an override controller.
OverwriteInitialOutputValue	REAL	0.0	If IntegralResetMode = 3, the value of PIDCtrl.IOutputOld is pre-assigned as if "PidOutputSum" = "OverwriteInitialOutputValue" in the last cycle.
RunModeByStartup	BOOL	TRUE	Activate operating mode at Mode parameter after CPU restart • If RunModeByStartup = TRUE, PID_Temp starts in the operating mode saved in the Mode parameter after CPU startup. • If RunModeByStartup = FALSE, PID_Temp remains in "Inactive" mode after CPU startup.
LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded from the CtrlParamsBackUp structure. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE. The acceptance is bumpless.

Tag	Data type	Default	Description
SetSubstitu-teOutput	BOOL	TRUE	Selection of the output value while an error is pending (State = 5): • If SetSubstituteOutput = TRUE and ActivateRecoverMode = TRUE, the configured substitute output value SubstituteOutput is output as PID output value as long as an error is pending. • If SetSubstituteOutput = FALSE and ActivateRecoverMode = TRUE, the actuator remains at the current PID output value as long as an error is pending. • If ActivateRecoverMode = FALSE, SetSubstituteOutput is not effective. • If SubstituteOutput is invalid (ErrorBits = 0020000h), the substitute output value cannot be output. In this case, the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit) is used as PID output value.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF. This parameter is used for display in the editors and does not influence the control algorithm.
Physical-Quantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature. This parameter is used for display in the editors and does not influence the control algorithm.
ActivateRe-coverMode	BOOL	TRUE	The ActivateRecoverMode tag determines the reaction to error.
Warning	DWORD	0	The Warning tag shows the warnings since Reset = TRUE or ErrorAck =TRUE. Warning is retentive.
Progress	REAL	0.0	Progress of current tuning phase as a percentage (0.0 - 100.0)
CurrentSet-point	REAL	0.0	CurrentSetpoint always displays the currently effective setpoint. This value is frozen during tuning.
CancelTu-ningLevel	REAL	10.0	Permissible fluctuation of setpoint during tuning. Tuning is not canceled until: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel

Tag	Data type	Default	Description
SubstituteOutput	REAL	0.0	The substitute output value is used as PID output value as long as the following conditions are met: • One or more errors are pending in automatic mode for which ActivateRecoverMode is in effect • SetSubstituteOutput = TRUE • ActivateRecoverMode = TRUE The values at the outputs for heating and cooling resulting from the substitute output value are the result of the configured output scaling (structures Config.Output.Heat and Config.Output.Cool). For controllers with activated cooling output (Config.ActivateCooling = TRUE), define: • a positive substitute output value to output the value at the outputs for heating • a negative substitute output value to output the value at the outputs for cooling The permitted value range is determined by the configuration. • Cooling output deactivated (Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ SubstituteOutput ≥ Config.Output.Heat.PidLowerLimit • Cooling output activated (Config.ActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ SubstituteOutput ≥ Config.Output.Cool.PidLowerLimit
PidOutputSum	REAL	0.0	PID output value PidOutputSum displays the output value of the PID algorithm. Depending on the operating mode, it is either calculated automatically or defined by the manual value or the configured substitute output value. The values at the outputs for heating and cooling resulting from the PID output value are the result of the configured output scaling (structures Config.Output.Heat and Config.Output.Cool). The PidOutputSum is limited as defined in the configuration. • Cooling output deactivated (Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ PidOutputSum ≥ Config.Output.Heat.PidLowerLimit • Cooling output activated (ConfigActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ PidOutputSum ≥ Config.Output.Cool.PidLowerLimit
PidOutputOffsetHeat	REAL	0.0	Offset of the PID output value heating PidOutputOffsetHeat is added to the value that results from PidOutputSum for the heating branch. Enter a positive value for PidOutputOffsetHeat to receive a positive offset at the outputs for heating. The resulting values at the outputs for heating are the result of the configured output scaling (Config.Output.Heat structure). This offset can be used for actuators which need a fixed minimum value, for example, fans with minimum speed.

Tag	Data type	Default	Description
PidOutputOffsetCool	REAL	0.0	Offset of the PID output value cooling PidOutputOffsetCool is added to the value that results from PidOutputSum for the cooling branch. Enter a negative value for PidOutputOffsetCool to receive a positive offset at the outputs for cooling. The resulting values at the outputs for cooling are the result of the configured output scaling (Config.Output.Cool structure). This offset can be used for actuators which need a fixed minimum value, for example, fans with minimum speed.
SubstituteSetpointOn	BOOL	FALSE	Activates the substitute setpoint as controller setpoint. • FALSE = the Setpoint parameter is used. • TRUE = the SubstituteSetpoint parameter is used as setpoint SubstituteSetpointOn can be used to specify the setpoint of a slave controller in a cascade directly without having to change the user program.
SubstituteSetpoint	REAL	0.0	Substitute setpoint If SubstituteSetpointOn = TRUE, the SubstituteSetpoint parameter is used as setpoint. Valid range of values: Config.SetpointUpperLimit ≥ SubstituteSetpoint ≥ Config.SetpointLowerLimit, Config.InputUpperLimit ≥ SubstituteSetpoint ≥ Config.InputLowerLimit
DisableCooling	BOOL	FALSE	DisableCooling = TRUE deactivates the cooling branch for heating/cooling controllers (Config.ActivateCooling = TRUE) in Automatic mode by setting PidOutputSum to 0.0 as low limit. PidOutputOffsetCool and the output scaling for the cooling outputs remain active. DisableCooling can be used for tuning of multi-zone applications to temporarily deactivate the cooling branch as long as all controllers have not completed their tuning yet. This parameter is set/reset by the user manually and is not automatically reset by the PID_Temp instruction.
AllSlaveAutomaticState	BOOL	FALSE	If this PID_Temp instance is used as master controller in a cascade (Config.Cascade.IsMaster = TRUE), AllSlaveAutomaticState = TRUE indicates that all slave controllers are in automatic mode. Tuning, manual mode or automatic mode of the master controller can only be executed accurately if all slave controllers are in automatic mode. AllSlaveAutomaticState is only determined if you interconnect the master controller and slave controller with the master and slave parameters. For details, see the Master parameter.
NoSlaveSubstituteSetpoint	BOOL	FALSE	If this PID_Temp instance is used as master controller in a cascade (Config.Cascade.IsMaster = TRUE), NoSlaveSubstituteSetpoint = TRUE indicates that no slave controller has activated its substitute setpoint. Tuning, manual mode or automatic mode of the master controller can only be executed accurately if no slave controller has activated its substitute setpoint. NoSlaveSubstituteSetpoint is only determined if you interconnect the master controller and slave controller with the master and slave parameters. For details, see the Master parameter.

Tag	Data type	Default	Description
Heat.EnableTuning	BOOL	TRUE	Enabling of tuning for heating Heat.EnableTuning must be set for the following tunings (at the same time or prior to the start with Mode and ModeActivate): • Pretuning heating • Pretuning heating and cooling • Fine tuning heating This parameter is not automatically reset by the PID_Temp instruction.
Cool.EnableTuning	BOOL	FALSE	Enabling of tuning for cooling Cool.EnableTuning must be set for the following tunings (simultaneously with or prior to the start with Mode and ModeActivate): • Pretuning cooling • Pretuning heating and cooling • Fine tuning cooling Only effective if the cooling output and PID parameter switching are activated ("Config.ActivateCooling" = TRUE and "Config.AdvancedCooling" = TRUE). This parameter is not automatically reset by the PID_Temp instruction.
Config.InputPerOn	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used for detecting the process value. If InputPerOn = FALSE, the Input parameter is used.
Config.InputUpperLimit	REAL	120.0	High limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. If the limit is exceeded, an error is output and the reaction is determined by ActivateRecoverMode. At the I/O input, the process value can be a maximum of 18% higher than the nominal range (overrange). This means the limit cannot be exceeded when you use an I/O input with the pre-setting for high limit and process value scaling. When pretuning is started, the difference between high and low limit of the process value is checked to determine whether the distance between setpoint and process value meets the necessary requirements. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. If the limit is undershot, an error is output and the reaction is determined by ActivateRecoverMode. InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	3.402822e+38	Warning high limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. If the limit is exceeded, a warning is output at the Warning parameter. • If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. • If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning

Tag	Data type	Default	Description
Config.InputLowerWarning	REAL	-3.402822e+38	Warning low limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. If the limit is undershot, a warning is output at the Warning parameter. • If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. • If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning
Config.SetpointUpperLimit	REAL	3.402822e+38	High limit of setpoint Setpoint and SubstituteSetpoint are monitored to ensure adherence to this limit. If the limit is exceeded, a warning is output at the Warning parameter. • If you configure SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is used as the setpoint high limit. • If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit. SetpointUpperLimit > SetpointLowerLimit
Config.SetpointLowerLimit	REAL	-3.402822e+38	Low limit of the setpoint Setpoint and SubstituteSetpoint are monitored to ensure adherence to this limit. If the limit is undershot, a warning is output at the Warning parameter. • If you set SetpointLowerLimit outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. • If you configure SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit. SetpointLowerLimit < SetpointUpperLimit
Config.ActivateCooling	BOOL	FALSE	Activate cooling output • Config.ActivateCooling = FALSE Only the outputs for heating are used. • Config.ActivateCooling = TRUE The outputs for heating and cooling are used. If you are using the cooling output, the controller must not be configured as master controller (Config.Cascade.IsMaster must be FALSE) .

Tag	Data type	Default	Description
Config.AdvancedCooling	BOOL	TRUE	Method for heating/cooling • Cooling factor (Config.AdvancedCooling = FALSE) The output value calculation for cooling takes place with the PID parameters for heating (Retain.CtrlParams.Heat structure) taking into consideration the configurable cooling factor Config.CoolFactor. This method is suitable if the heating and cooling actuator have similar time responses but different gains. Pretuning and fine tuning for cooling are not available when you select this method. You can only execute the tuning for heating. • PID parameter switching (Config.AdvancedCooling = TRUE) The output value calculation for cooling takes place by means of a separate PID parameter set (Retain.CtrlParams.Cool structure). This method is suitable if the heating and cooling actuator have different time responses and different gains. Pretuning and fine tuning for cooling are only available when you select this method (Mode = 1 or 2, Cool.EnableTuning = TRUE). Config.AdvancedCooling is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE).
Config.CoolFactor	REAL	1.0	Cooling factor If Config.AdvancedCooling = FALSE, Config.CoolFactor is considered as factor in the calculation of the output value for cooling. Different gains of the heating and cooling actuator can be considered in this way. Config.CoolFactor is not set automatically or adjusted during tuning. You must correctly configure Config.CoolFactor manually with the ratio "heating actuator gain/cooling actuator gain". Example: Config.CoolFactor = 2.0 means that the gain of the heating actuator is twice as high as the gain of the cooling actuator. Config.CoolFactor is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and cooling factor is selected as method for heating/cooling (Config.AdvancedCooling = FALSE). Config.CoolFactor > 0.0
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is scaled based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). UpperPointIn > LowerPointIn
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is scaled based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). LowerPointIn < UpperPointIn

Tag	Data type	Default	Description
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is scaled based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). UpperPointOut > LowerPointOut
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is scaled based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). LowerPointOut < UpperPointOut
Config.Output.Heat.Select	INT	1	Selecting the output value for heating Config.Output.Heat.Select specifies which outputs are used for heating: • Heat.Select = 0 - OutputHeat is used • Heat.Select = 1 - OutputHeat and OutputHeat_PWM are used • Heat.Select = 2 -OutputHeat and OutputHeat_PER are used Outputs that are not used are not calculated and remain at their default value.
Config.Output.Heat.PwmPeriode	REAL	0.0	Time period of the pulse width modulation (PWM) for heating (OutputHeat_PWM output) in seconds: • Heat.PwmPeriode = 0.0 The sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as time period of the PWM. • Heat.PwmPeriode > 0.0 The value is rounded off to an integer multiple of the PID_Temp sampling time (CycleTime.Value) and used as time period of the PWM. This setting can be used to improve the smoothing of the process value with a long sampling time of the PID algorithm. The value must meet the following conditions: – Heat.PwmPeriode ≤ Retain.CtrlParams.Heat.Cycle, – Heat.PwmPeriode > Config.Output.Heat.MinimumOnTime – Heat.PwmPeriode > Config.Output.Heat.MinimumOffTime
Config.Output.Heat.PidUpperLimit	REAL	100.0	High limit of the PID output value for heating The PID output value (PidOutputSum) is limited to the high limit. Heat.PidUpperLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for heating: • Heat.UpperScaling for OutputHeat • Heat.PwmUpperScaling for OutputHeat_PWM • Heat.PerUpperScaling for OutputHeat_PER If you want to limit the value at the associated output, you must also adjust these scaling values. Heat.PidUpperLimit > Heat.PidLowerLimit

Tag	Data type	Default	Description
Config.Output.Heat.PidLowerLimit	REAL	0.0	Low limit of the PID output value for heating For controllers with **deactivated** cooling output (Config.ActivateCooling = FALSE), the PID output value (PidOutputSum) is limited to this low limit. For controllers with **activated** cooling output (Config.ActivateCooling = TRUE), the value must be 0.0. Heat.PidLowerLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for heating: • Heat.LowerScaling for OutputHeat • Heat.PwmLowerScaling for OutputHeat_PWM • Heat.PerLowerScaling for OutputHeat_PER If you want to limit the value at the associated output, you must also adjust these scaling values. The permitted value range is determined by the configuration. • Cooling output deactivated (Config.ActivateCooling = FALSE): Heat.PidLowerLimit < Heat.PidUpperLimit • Cooling output activated (Config.ActivateCooling = TRUE): Heat.PidLowerLimit = 0.0
Config.Output.Heat.UpperScaling	REAL	100.0	Scaled high output value for heating Heat.UpperScaling and Heat.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for heating (OutputHeat). The OutputHeat value is always located between Heat.UpperScaling and Heat.LowerScaling. Heat.UpperScaling ≠ Heat.LowerScaling
Config.Output.Heat.LowerScaling	REAL	0.0	Scaled low output value for heating Heat.LowerScaling and Heat.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for heating (OutputHeat). The OutputHeat value is always located between Heat.UpperScaling and Heat.LowerScaling. Heat.UpperScaling ≠ Heat.LowerScaling
Config.Output.Heat.PwmUpperScaling	REAL	100.0	Scaled high PWM output value for heating Heat.PwmUpperScaling and Heat.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for heating (OutputHeat_PWM). The OutputHeat_PWM value is always located between Heat.PwmUpperScaling and Heat.PWMLowerScaling. Heat.PwmUpperScaling is only effective if OutputHeat_PWM is selected as output for heating (Heat.Select = 1) 100.0 ≥ Heat.PwmUpperScaling ≥ 0.0 Heat.PwmUpperScaling ≠ Heat.PwmLowerScaling

Tag	Data type	Default	Description
Config.Output.Heat.PwmLowerScaling	REAL	0.0	Scaled low PWM output value for heating Heat.PwmLowerScaling and Heat.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for heating (OutputHeat_PWM). The OutputHeat_PWM value is always located between Heat.PwmUpperScaling and Heat.PwmLowerScaling. Heat.PwmLowerScaling is only effective if OutputHeat_PWM is selected as output for heating (Heat.Select = 1) 100.0 ≥ Heat.PwmLowerScaling ≥ 0.0 Heat.PwmUpperScaling ≠ Heat.PwmLowerScaling
Config.Output.Heat.PerUpperScaling	REAL	27648.0	Scaled high analog output value for heating Heat.PerUpperScaling and Heat.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for heating (OutputHeat_PER). The OutputHeat_PER value is always located between Heat.PerUpperScaling and Heat.PerLowerScaling. Heat.PerUpperScaling is only effective if OutputHeat_PER is selected as output for heating (Heat.Select = 2) 32511.0 ≥ Heat.PerUpperScaling ≥ -32512.0 Heat.PerUpperScaling ≠ Heat.PerLowerScaling
Config.Output.Heat.PerLowerScaling	REAL	0.0	Scaled low analog output value for heating Heat.PerLowerScaling and Heat.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for heating (OutputHeat_PER). The OutputHeat_PER value is always located between Heat.PerUpperScaling and Heat.PerLowerScaling. Heat.PerLowerScaling is only effective if OutputHeat_PER is selected as output for heating (Heat.Select = 2) 32511.0 ≥ Heat.PerLowerScaling ≥ -32512.0 Heat.PerUpperScaling ≠ Heat.PerLowerScaling
Config.Output.Heat.MinimumOnTime	REAL	0.0	Minimum on time of the pulse width modulation for heating (OutputHeat_PWM output) A PWM pulse is never shorter than this value. The value is rounded off to: Heat.MinimumOnTime = n × CycleTime.Value Heat.MinimumOnTime is only effective if the output for heating OutputHeat_PWM is selected (Heat.Select = 1)". 100000.0 ≥ Heat.MinimumOnTime ≥ 0.0
Config.Output.Heat.MinimumOffTime	REAL	0.0	Minimum off time of the pulse width modulation for heating (OutputHeat_PWM output) A PWM pause is never shorter than this value. The value is rounded off to: Heat.MinimumOffTime = n × CycleTime.Value Heat.MinimumOffTime is only effective if the output for heating OutputHeat_PWM is selected (Heat.Select = 1)". 100000.0 ≥ Heat.MinimumOffTime ≥ 0.0

Tag	Data type	Default	Description
Config.Output.Cool.Select	INT	1	Selecting the output value for cooling Config.Output.Cool.Select specifies which outputs are used for cooling: • Cool.Select = 0 - OutputCool is used • Cool.Select = 1 -OutputCool and OutputCool_PWM are used • Cool.Select = 2 - OutputCool and OutputCool_PER are used Outputs that are not used are not calculated and remain at their default value. Only effective if the cooling output is activated (Config.ActivateCooling = TRUE).
Config.Output.Cool.PwmPeriode	REAL	0.0	Time period of the pulse width modulation for cooling (OutputCool_PWM output) in seconds: • Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = FALSE: sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as time period of the PWM. • Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = TRUE: The sampling time of the PID algorithm for cooling (Retain.CtrlParams.Cool.Cycle) is used as time period of the PWM. • Cool.PwmPeriode > 0.0: The value is rounded off to an integer multiple of the PID_Temp sampling time (CycleTime.Value) and used as time period of the PWM. This setting can be used to improve the smoothing of the process value with a long sampling time of the PID algorithm. The value must meet the following conditions: – Cool.PwmPeriode ≤ Retain.CtrlParams.Cool.Cycle or Retain.CtrlParams.Heat.Cycle – Cool.PwmPeriode > Config.Output.Cool.MinimumOnTime – Cool.PwmPeriode > Config.Output.Cool.MinimumOffTime Only effective if the cooling output is activated (Config.ActivateCooling = TRUE).
Config.Output.Cool.PidUpperLimit	REAL	0.0	High limit of the PID output value for cooling The value must be 0.0. Cool.PidUpperLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for cooling: • Cool.LowerScaling for OutputCool • Cool.PwmLowerScaling for OutputCool_PWM • Cool.PerLowerScaling for OutputCool_PER If you want to limit the value at the associated output, you must also adjust these scaling values. Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). Cool.PidUpperLimit = 0.0

Tag	Data type	Default	Description
Config.Output.Cool.PidLowerLimit	REAL	-100.0	Low limit of the PID output value for cooling For controllers with activated cooling output (Config.ActivateCooling = TRUE), the PID output value (PidOutputSum) is limited to this low limit. Cool.PidLowerLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for cooling: • Cool.UpperScaling for OutputCool • Cool.PwmUpperScaling for OutputCool_PWM • Cool.PerUpperScaling for OutputCool_PER If you want to limit the value at the associated output, you must also adjust these scaling values. Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). Cool.PidLowerLimit < Cool.PidUpperLimit
Config.Output.Cool.UpperScaling	REAL	100.0	Scaled high output value for cooling Cool.UpperScaling and Cool.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for cooling (OutputCool). The OutputCool value is always located between Cool.UpperScaling and Cool.LowerScaling. Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). Cool.UpperScaling ≠ Cool.LowerScaling
Config.Output.Cool.LowerScaling	REAL	0.0	Scaled low output value for cooling Cool.LowerScaling and Cool.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for cooling (OutputCool). The OutputCool value is always located between Cool.UpperScaling and Cool.LowerScaling. Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). Cool.UpperScaling ≠ Cool.LowerScaling
Config.Output.Cool.PwmUpperScaling	REAL	100.0	Scaled high PWM output value for cooling Cool.PwmUpperScaling and Cool.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for cooling (OutputCool_PWM). The OutputCool_PWM value is always located between Cool.PwmUpperScaling and Cool.PwmLowerScaling. Cool.PwmUpperScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PWM is selected as output for cooling (Cool.Select = 1). 100.0 ≥ Cool.PwmUpperScaling ≥ 0.0 Cool.PwmUpperScaling ≠ Cool.PwmLowerScaling

Tag	Data type	Default	Description
Config.Output.Cool.PwmLowerScaling	REAL	0.0	Scaled low PWM output value for cooling Cool.PwmLowerScaling and Cool.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for cooling (OutputCool_PWM). The OutputCool_PWM value is always located between Cool.PwmUpperScaling and CoolPwm.LowerScaling. Cool.PwmLowerScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PWM is selected as output for cooling (Cool.Select = 1). 100.0 ≥ Cool.PwmLowerScaling ≥ 0.0 Cool.PwmUpperScaling ≠ Cool.PwmLowerScaling
Config.Output.Cool.PerUpperScaling	REAL	27648.0	Scaled high analog output value for cooling Cool.PerUpperScaling and Cool.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for cooling (OutputCool_PER). The OutputCool_PER value is always located between Cool.PerUpperScaling and Cool.PerLowerScaling. Cool.PerUpperScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PER is selected as output for cooling (Cool.Select = 2). 32511.0 ≥ Cool.PerUpperScaling ≥ -32512.0 Cool.PerUpperScaling ≠ Cool.PerLowerScaling
Config.Output.Cool.PerLowerScaling	REAL	0.0	Scaled low analog output value for cooling Cool.PerLowerScaling and Cool.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for cooling (OutputCool_PER). The OutputCool_PER value is always located between Cool.PerUpperScaling and Cool.PerLowerScaling. Cool.PerLowerScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PER is selected as output for cooling (Cool.Select = 2). 32511.0 ≥ Cool.PerLowerScaling ≥ -32512.0 Cool.PerUpperScaling ≠ Cool.PerLowerScaling
Config.Output.Cool.MinimumOnTime	REAL	0.0	Minimum on time of the pulse width modulation for cooling (OutputCool_PWM output) A PWM pulse is never shorter than this value. The value is rounded off to: Cool.MinimumOnTime = n × CycleTime.Value Cool.MinimumOnTime is only effective if the output for cooling OutputCool_PWM is selected (Cool.Select = 1). Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). 100000.0 ≥ Cool.MinimumOnTime ≥ 0.0

Tag	Data type	Default	Description
Config.Output.Cool.MinimumOffTime	REAL	0.0	Minimum off time of the pulse width modulation for cooling (OutputCool_PWM output) A PWM pause is never shorter than this value. The value is rounded off to: Cool.MinimumOffTime = n × CycleTime.Value Cool.MinimumOffTime is only effective if the output for cooling OutputCool_PWM is selected (Cool.Select = 1). Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). 100000.0 ≥ Cool.MinimumOffTime ≥ 0.0
If you are using PID_Temp in a cascade, the master controller and slave controller exchange information via the master and slave parameters. You need to make the interconnection. For details, see the Master parameter.			
Config.Cascade.IsMaster	BOOL	FALSE	The controller is master in a cascade and provides the slave setpoint. Set IsMaster = TRUE if you are using this PID_Temp instance as master controller in a cascade. A master controller defines the setpoint of a slave controller with its output. A PID_Temp instance can be master controller and slave controller at the same time. If the controller is used as master controller, the cooling output must be deactivated (Config.ActivateCooling = FALSE).
Config.Cascade.IsSlave	BOOL	FALSE	The controller is slave in a cascade and provides the master setpoint. Set IsSlave = TRUE if you are using this PID_Temp instance as slave controller in a cascade. A slave controller receives its setpoint (Setpoint parameter) from the output of its master controller (OutputHeat parameter). A PID_Temp instance can be master controller and slave controller at the same time.
Config.Cascade.AntiWindUpMode	INT	1	Anti-windup behavior in the cascade Options are: • Anti-windup = 0 The AntiWindUp functionality is deactivated. The master controller does not respond to the limit of its slave controllers. • Anti-windup = 1 The integral action of the master controller is reduced in the ratio "Slaves in limit" to "Number of slaves" ("CountSlaves" parameter). This means the effects of the limit are reduced to the control response. • Anti-windup = 2 The integral action of the master controller is held as soon as a slave controller is in the limit. Only effective if the controller is configured as master controller (Config.Cascade.IsMaster = TRUE).
Config.Cascade.CountSlaves	INT	1	Number of subordinate slaves Here you enter the number of directly subordinate slave controllers which receive their setpoint from this master controller. Only effective if the controller is configured as master controller (Config.Cascade.IsMaster = TRUE). 255 ≥ CountSlaves ≥ 1

Tag	Data type	Default	Description
CycleTime.StartEstimation	BOOL	TRUE	If CycleTime.EnEstimation = TRUE, CycleTime.StartEstimation = TRUE starts automatic determination of the PID_Temp sampling time (cycle time of the calling OB). CycleTime.StartEstimation = FALSE is set once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If CycleTime.EnEstimation = TRUE, the PID_Temp sampling time is determined automatically. If CycleTime.EnEstimation = FALSE, the sampling time PID_Temp is not determined automatically and must be configured correctly manually with CycleTime.Value.
CycleTime.EnMonitoring	BOOL	TRUE	If CycleTime.EnMonitoring = FALSE, the PID_Temp sampling time is not monitored. If PID_Temp cannot be executed within the sampling time, no error (ErrorBits=0000800h) is output and PID_Temp does not respond as configured with ActivateRecoverMode.
CycleTime.Value	REAL	0.1	PID_Temp sampling time (cycle time of the calling OB) in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.			
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser
CtrlParamsBackUp.Heat.Gain	REAL	1.0	Saved proportional gain for heating
CtrlParamsBackUp.Heat.Ti	REAL	20.0	Saved integral action time for heating in seconds
CtrlParamsBackUp.Heat.Td	REAL	0.0	Saved derivative action time for heating in seconds
CtrlParamsBackUp.Heat.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient for heating
CtrlParamsBackUp.Heat.PWeighting	REAL	1.0	Saved weighting of the proportional action for heating
CtrlParamsBackUp.Heat.DWeighting	REAL	1.0	Saved weighting of the derivative action for heating
CtrlParamsBackUp.Heat.Cycle	REAL	1.0	Saved sampling time of the PID algorithm for heating in seconds
CtrlParamsBackUp.Heat.ControlZone	REAL	3.402822e+38	Saved control zone width for heating

Tag	Data type	Default	Description
CtrlParams-Back-Up.Heat.DeadZone	REAL	0.0	Saved deadband width for heating
CtrlParams-Back-Up.Cool.Gain	REAL	1.0	Saved proportional gain for cooling
CtrlParams-Back-Up.Cool.Ti	REAL	20.0	Saved integral action time for cooling in seconds
CtrlParams-Back-Up.Cool.Td	REAL	0.0	Saved derivative action time for cooling in seconds
CtrlParams-Back-Up.Cool.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient for cooling
CtrlParams-Back-Up.Cool.PWeighting	REAL	1.0	Saved proportional action weighting factor for cooling
CtrlParams-Back-Up.Cool.DWeighting	REAL	1.0	Saved derivative action weighting factor for cooling
CtrlParams-Back-Up.Cool.Cycle	REAL	1.0	Saved sampling time of the PID algorithm for cooling in seconds
CtrlParams-Back-Up.Cool.ControlZone	REAL	3.402822e+38	Saved control zone width for cooling
CtrlParams-Back-Up.Cool.DeadZone	REAL	0.0	Saved deadband width for cooling
PIDSelf-Tune.SUT.Calculate-Param-sHeat	BOOL	FALSE	The properties of the heating branch of the controlled system are saved during pretuning for heating. If SUT.CalculateParamsHeat = TRUE, the PID parameters for heating are recalculated on the basis of these properties (Retain.CtrlParams.Heat structure). This enables you to change the parameter calculation method (PIDSelfTune.SUT.TuneRuleHeat parameter) without having to repeat the tuning. SUT.CalculateParamsHeat is set to FALSE after the calculation. Only possible if the pretuning was successful (SUT.ProcParHeatOk = TRUE).

Tag	Data type	Default	Description
PIDSelf-Tune.SUT.Calculate-ParamsCool	BOOL	FALSE	The properties of the cooling branch of the controlled system are saved during tuning for cooling. If SUT.CalculateParamsCool = TRUE, the PID parameters for cooling are recalculated on the basis of these properties (Retain.CtrlPar-ams.Cool structure). This enables you to change the parameter calculation method (PIDSelfTune.SUT.TuneRuleCool parameter) without having to repeat the tuning. SUT.CalculateParamsCool is set to FALSE after the calculation. Only possible if the pretuning was successful (SUT.ProcParCoolOk = TRUE). Only effective if Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE.
PIDSelf-Tune.SUT.T uneRule-Heat	INT	2	Method for PID parameter calculation with pretuning for heating Options are: • SUT.TuneRuleHeat = 0: PID to CHR • SUT.TuneRuleHeat = 1: PI to CHR • SUT.TuneRuleHeat = 2: PID for temperature processes to CHR (results in a longer and rather asymptomatic control response with fewer overshoots than SUT.TuneRuleHeat = 0) (CHR = Chien, Hrones and Reswick) Only with SUT.TuneRuleHeat = 2 is the control zone Retain.CtrlPar-ams.Heat.ControlZone automatically set during pretuning for heating.
PIDSelf-Tune.SUT.T uneRule-Cool	INT	2	Method for PID parameter calculation with pretuning for cooling Options are: • SUT.TuneRuleCool = 0: PID to CHR • SUT.TuneRuleCool = 1: PI to CHR • SUT.TuneRuleCool = 2: PID for temperature processes to CHR (results in a longer and rather asymptomatic control response with fewer overshoots than SUT.TuneRuleCool = 0) (CHR = Chien, Hrones and Reswick) Only with SUT.TuneRuleCool = 2 is the control zone Retain.CtrlPar-ams.Cool.ControlZone automatically set during pretuning for cooling. SUT.TuneRuleCool is only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE, Config.AdvancedCool-ing = TRUE).

Tag	Data type	Default	Description
PIDSelf-Tune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning: • State = 0: Initialize pretuning • State = 100: Calculate standard deviation for heating • State = 200: Calculate standard deviation for cooling • State = 300: Determine point of inflection for heating • State = 400: Determine point of inflection for cooling • State = 500: Set heating to setpoint after reaching point of inflection • State = 600: Set cooling to setpoint after reaching point of inflection • State = 700: Compare efficiency of the heating actuator and cooling actuator • State = 800: Heating and cooling activated • State = 900: Cooling activated • State = 1000: Determine delay time after switching off heating • State = 9900: Pretuning successful • State = 1: Pretuning not successful
PIDSelf-Tune.SUT.ProcParHeatOk	BOOL	FALSE	TRUE: The calculation of the process parameters for pretuning heating was successful. This tag is set during tuning. It must be TRUE for calculation of the PID parameters for heating.
PIDSelf-Tune.SUT.ProcParCoolOk	BOOL	FALSE	TRUE: The calculation of the process parameters for pretuning cooling was successful. This tag is set during tuning. It must be TRUE for calculation of the PID parameters for cooling.
PIDSelf-Tune.SUT.AdaptDelay-Time	INT	0	The AdaptDelayTime tag determines the adaptation of the delay time for heating at the operating point (for "Pretuning heating" and "Pretuning heating and cooling"). Options are: • SUT.AdaptDelayTime = 0: No adaptation of delay time. The SUT.State = 1000 phase is skipped. This option results in a shorter tuning time than with SUT.AdaptDelayTime = 1. • SUT.AdaptDelayTime = 1: Adaptation of the delay time to the setpoint in SUT.State = 1000 phase by switching off heating temporarily. This option results in a longer tuning time than with SUT.AdaptDelayTime = 0. It can improve the control response if the process behavior depends significantly on the operating point (non-linearity). This option should not be used for multi-zone applications with strong thermal connections.

Tag	Data type	Default	Description
PIDSelf-Tune.SUT.Cooling-Mode	INT	0	The CoolingMode tag determines the manipulated variable output to determine the cooling parameters (for pretuning heating and cooling). Options are: • SUT.CoolingMode = 0: Switch off heating and switch on cooling after reaching the setpoint. The SUT.State = 700 phase is skipped. Phase SUT.State = 500 is followed by phase SUT.State = 900. This option can improve the control response if the gain of the cooling actuator is low compared to the gain of the heating actuator. It results in a shorter tuning time than with SUT.CoolingMode = 1 or 2. • SUT.CoolingMode = 1: Switch on cooling in addition to heating after reaching the setpoint. The SUT.State = 700 phase is skipped. Phase SUT.State = 500 is followed by phase SUT.State = 800. This option can improve the control response if the gain of the cooling actuator is high compared to the gain of the heating actuator. • SUT.CoolingMode = 2: After heating up to the setpoint, a decision is automatically made in phase SUT.State = 700 as to whether heating is switched off. Phase SUT.State = 500 is followed by phase SUT.State = 700 and then SUT.State = 800 or SUT.State = 900. This option requires more time than options 0 and 1.
PIDSelf-Tune.TIR.RunIn	BOOL	FALSE	Use the RunIn tag to specify the sequence of fine tuning during start from automatic mode. • RunIn = FALSE If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint (TIR.State = 500 or 600). Only then will fine tuning start. • RunIn = TRUE PID_Temp tries to reach the setpoint with minimum or maximum output value (TIR.State = 300 or 400). This can produce increased overshoot. Fine tuning then starts automatically. RunIn is set to FALSE after fine tuning. During start of fine tuning from Inactive or Manual mode, PID_Temp reacts as described under RunIn = TRUE.
PIDSelf-Tune.TIR.CalculateParamsHeat	BOOL	FALSE	The properties of the heating branch of the controlled system are saved during fine tuning for heating. If TIR.CalculateParamsHeat= TRUE, the PID parameters for heating are recalculated on the basis of these properties (Retain.CtrlParams.Heat structure). This enables you to change the parameter calculation method (PIDSelfTune.TIR.TuneRuleHeat parameter) without having to repeat the tuning. TIR.CalculateParamsHeat is set to FALSE after the calculation. Only possible if fine tuning heating was successful beforehand (TIR.ProcParHeatOk = TRUE).

Tag	Data type	Default	Description
PIDSelf-Tune.TIR.CalculateParamsCool	BOOL	FALSE	The properties of the cooling branch of the controlled system are saved during fine tuning for cooling. If TIR.CalculateParamsCool= TRUE, the PID parameters for cooling are recalculated on the basis of these properties (Retain.CtrlParams.Cool structure). This enables you to change the parameter calculation method (PIDSelfTune.TIR.TuneRuleCool parameter) without having to repeat the tuning. TIR.CalculateParamsCool is set to FALSE after the calculation. Only possible if fine tuning cooling was successful beforehand (TIR.ProcParCoolOk = TRUE). Only effective if Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE
PIDSelf-Tune.TIR.TuneRule-Heat	INT	0	Method for parameter calculation during fine tuning for heating Options are: • TIR.TuneRuleHeat = 0: PID automatic • TIR.TuneRuleHeat = 1: PID fast (faster control response with higher amplitudes of the output value than with TIR.TuneRuleHeat = 2) • TIR.TuneRuleHeat = 2: PID slow (slower control response with lower amplitudes of the output value than with TIR.TuneRuleHeat = 1) • TIR.TuneRuleHeat = 3: ZN PID • TIR.TuneRuleHeat = 4: ZN PI • TIR.TuneRuleHeat = 5: ZN P (ZN=Ziegler-Nichols) To be able to repeat the calculation of the PID parameters for heating with TIR.CalculateParamsHeat and TIR.TuneRuleHeat = 0, 1 or 2, the previous fine tuning also has to have been executed with TIR.TuneRuleHeat = 0, 1 or 2. If this is not the case, TIR.TuneRuleHeat = 3 is used. The recalculation of the PID parameters for heating with TIR.CalculateParamsHeat and TIR.TuneRuleHeat = 3, 4 or 5 is always possible.
PIDSelf-Tune.TIR.TuneRule-Cool	INT	0	Method for parameter calculation during fine tuning for cooling Options are: • TIR.TuneRuleCool = 0: PID automatic • TIR.TuneRuleCool = 1: PID fast (faster control response with higher amplitudes of the output value than with TIR.TuneRuleCool = 2) • TIR.TuneRuleCool = 2: PID slow (slower control response with lower amplitudes of the output value than with TIR.TuneRuleCool = 1) • TIR.TuneRuleCool = 3: ZN PID • TIR.TuneRuleCool = 4: ZN PI • TIR.TuneRuleCool = 5: ZN P (ZN=Ziegler-Nichols) To be able to repeat the calculation of the PID parameters for cooling with TIR.CalculateParamsCool and TIR.TuneRuleCool = 0, 1 or 2, the previous fine tuning also has to have been executed with TIR.TuneRuleCool = 0, 1 or 2. If this is not the case, TIR.TuneRuleCool = 3 is used. The recalculation of the PID parameters for cooling with TIR.CalculateParamsCool and TIR.TuneRuleCool = 3, 4 or 5 is always possible. Only effective if the cooling output and PID parameter switching are activated (ConfigActivateCooling = TRUE and Config.AdvancedCooling = TRUE).

Tag	Data type	Default	Description
PIDSelf-Tune.TIR.State	INT	0	The TIR.State tag indicates the current phase of "fine tuning": • State = 0: Initialize fine tuning • State = 100: Calculate standard deviation for heating • State = 200: Calculate standard deviation for cooling • State = 300: Attempting to reach setpoint for heating with two-step control • State = 400: Attempting to reach setpoint for cooling with two-step control • State = 500: Attempting to reach setpoint for heating with PID control • State = 600: Attempting to reach setpoint for cooling with PID control • State = 700: Calculate standard deviation for heating • State = 800: Calculate standard deviation for cooling • State = 900: Determine oscillation and calculate parameters for heating • State = 1000: Determine oscillation and calculate parameters for cooling • State = 9900: Fine tuning successful • State = 1: Fine tuning not successful
PIDSelf-Tune.TIR.ProcParHeatOk	BOOL	FALSE	TRUE: The calculation of the process parameters for fine tuning heating was successful. This tag is set during tuning. It must be met for calculation of the PID parameters for heating.
PIDSelf-Tune.TIR.ProcParCoolOk	BOOL	FALSE	TRUE: The calculation of the process parameters for fine tuning cooling was successful. This tag is set during tuning. It must be met for calculation of the PID parameters for cooling.
PIDSelf-Tune.TIR.OutputOffsetHeat	REAL	0.0	Tuning offset heating of the PID output value TIR.OutputOffsetHeat is added to the value that results from PidOutputSum for the heating branch. To receive a positive offset at the outputs for heating, define a positive value for TIR.OutputOffsetHeat. The resulting values at the outputs for heating are the result of the configured output scaling (Struktur Config.Output.Heat). This tuning offset can be used in controllers with activated cooling output and PID parameter switching (Config.ActivateCooling = TRUE, Config.Advanced-Cooling = TRUE) for fine tuning cooling. If the outputs for cooling are not active at the setpoint that is to be tuned (PidOutputSum > 0.0), fine tuning cooling is not possible. In this case, define a positive tuning offset heating which is greater than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning. This step increases the values at the outputs for heating and activates the outputs for cooling (PidOutputSum < 0.0). Fine tuning cooling is now possible. When fine tuning is complete, TIR.OutputOffsetHeat is reset to 0.0. Major changes at TIR.OutputOffsetHeat in one step can result in temporary overshoots. Config.Output.Heat.PidUpperLimit ≥ PIDSelfTune.TIR.OutputOffsetHeat ≥ Config.Output.Heat.PidLowerLimit

Tag	Data type	Default	Description
PIDSelf-Tune.TIR.OutputOffset-Cool	REAL	0.0	Tuning offset cooling of the PID output value TIR.OutputOffsetCool is added to the value that results from PidOutputSum for the cooling branch. To receive a positive offset at the outputs for cooling, define a negative value for TIR.OutputOffsetCool. The resulting values at the outputs for cooling are the result of the configured output scaling (Struktur Config.Output.Coool). This tuning offset can be used in controllers with activated cooling output (Config.ActivateCooling = TRUE) for fine tuning heating. If the outputs for heating are not active at the setpoint that is to be tuned (PidOutputSum < 0.0), fine tuning heating is not possible. In this case, define a negative tuning offset cooling which is less than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning. This step increases the values at the outputs for cooling and activates the outputs for heating (PidOutputSum > 0.0). Fine tuning heating is now possible. When fine tuning is complete, TIR.OutputOffsetCool is reset to 0.0. Major changes at TIR.OutputOffsetCool in one step can result in temporary overshoots. Config.Output.Cool.PidUpperLimit ≥ PIDSelfTune.TIR.OutputOffsetCool ≥ Config.Output.Cool.PidLowerLimit
PIDSelf-Tune.TIR.WaitForControlIn	BOOL	FALSE	Waiting with fine tuning after reaching the setpoint If TIR.WaitForControlIn = TRUE, fine tuning waits in between reaching the setpoint (TIR.State = 500 or 600) and calculation of the standard deviation (TIR.State = 700 or 800) until a FALSE -> TRUE edge is given at TIR.FinishControlIn. TIR.WaitForControlIn can be used for simultaneous fine tuning of several controllers in multi-zone applications to synchronize tuning of the individual zones. It ensures that all zones have reached their setpoints before the actual tuning starts. The influence of thermal connections between the zones on tuning can be reduced in this way. TIR.WaitForControlIn is only effective if fine tuning is started from automatic mode with PIDSelfTune.TIR.RunIn = FALSE.
PIDSelf-Tune.TIR.ControlIn-Ready	BOOL	FALSE	If TIR.WaitForControlIn = TRUE, PID_Temp sets TIR.ControlInReady = TRUE as soon as the setpoint has been reached and waits with additional tuning steps until a FALSE -> TRUE edge is given at TIR.FinishControlIn.
PIDSelf-Tune.TIR.FinishControlIn	BOOL	FALSE	If TIR.ControlInReady = TRUE, a FALSE -> TRUE edge at TIR.FinishControlIn stops the wait and fine tuning resumes.
PIDCtr.IOutputOld	REAL	0.0	Integral action in last cycle
Retain.CtrlParams.SetByUser	BOOL	FALSE	If the PID parameters are entered manually in the configuration editor, SetByUser = TRUE. This parameter is used for display in the editors and does not influence the control algorithm. SetByUser is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.Heat.Gain	REAL	1.0	Active proportional gain for heating Heat.Gain is retentive. Heat.Gain ≥ 0.0
Retain..CtrlParams.Heat.Ti	REAL	20.0	Active integral action time for heating in seconds The integral action for heating is switched off with Heat.CtrlParams.Ti = 0.0. Heat.Ti is retentive. 100000.0 ≥ Heat.Ti ≥ 0.0
Retain.CtrlParams.Heat.Td	REAL	0.0	Active derivative action time for heating in seconds The derivative action for heating is switched off with Heat.CtrlParams.Td = 0.0. Heat.Td is retentive. 100000.0 ≥ Heat.Td ≥ 0.0
Retain.CtrlParams.Heat.TdFiltRatio	REAL	0.2	Active derivative delay coefficient for heating The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient • 0.0: Derivative action is effective for one cycle only and therefore almost not effective. • 0.5: This value has proved useful in practice for controlled systems with one dominant time constant. • > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed. Heat.TdFiltRatio is retentive. Heat.TdFiltRatio ≥ 0.0
Retain.CtrlParams.Heat.PWeighting	REAL	1.0	Active weighting of the proportional action for heating The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Proportional action for setpoint change is fully effective • 0.0: Proportional action for setpoint change is not effective The proportional action is always fully effective when the process value is changed. Heat.PWeighting is retentive. 1.0 ≥ Heat.PWeighting ≥ 0.0
Retain.CtrlParams.Heat.DWeighting	REAL	1.0	Active weighting of the derivative action for heating The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Derivative action is fully effective upon setpoint change • 0.0: Derivative action is not effective upon setpoint change The derivative action is always fully effective when the process value is changed. Heat.DWeighting is retentive. 1.0 ≥ Heat.DWeighting ≥ 0.0

Tag	Data type	Default	Description
Retain.CtrlParams.Heat.Cycle	REAL	1.0	Active sampling time of the PID algorithm for heating in seconds CtrlParams.Heat.Cycle is calculated during tuning and rounded to an integer multiple of CycleTime.Value. If Config.Output.Heat.PwmPeriode = 0.0, Heat.Cycle is used as time period of the pulse width modulation for heating. If Config.Output.Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = FALSE, Heat.Cycle is used as time period of the pulse width modulation for cooling. Heat.Cycle is retentive. 100000.0 ≥ Heat.Cycle > 0.0
Retain.CtrlParams.Heat.ControlZone	REAL	3.402822e+38	Active control zone width for heating The control zone for heating is switched off with Heat.ControlZone = 3.402822e+38. Heat.ControlZone is only set automatically during pretuning heating or pretuning heating and cooling if PIDSelfTune.SUT.TuneRuleHeat = 2 is selected as method of the parameter calculation. For controllers with deactivated cooling output (Config.ActivateCooling = FALSE) or controllers with activated cooling output and cooling factor (Config.AdvancedCooling = FALSE), the control zone is symmetrically located between Setpoint – Heat.ControlZone and Setpoint + Heat.ControlZone. For controllers with activated cooling output and PID parameter switching (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE), the control zone is located between Setpoint – Heat.ControlZone and Setpoint + Cool.ControlZone. Heat.ControlZone is retentive. Heat.ControlZone > 0.0
Retain.CtrlParams.Heat.DeadZone	REAL	0.0	Active deadband width for heating (see PID parameters (Page 7307)) The deadband for heating is switched off with Heat.DeadZone = 0.0. Heat.DeadZone is not set automatically or adjusted during tuning. You must correctly configure Heat.DeadZone manually. For controllers with deactivated cooling output (Config.ActivateCooling = FALSE) or controllers with activated cooling output and cooling factor (Config.AdvancedCooling = FALSE), the deadband is symmetrically located between Setpoint – Heat.DeadZone and Setpoint + Heat.DeadZone. For controllers with activated cooling output and PID parameter switching (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE), the deadband is located between Setpoint – Heat.DeadZone and Setpoint + Cool.DeadZone. Heat.DeadZone is retentive. Heat.DeadZone ≥ 0.0
Retain.CtrlParams.Cool.Gain	REAL	1.0	Active proportional gain for cooling Cool.Gain is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). Cool.Gain ≥ 0.0

Tag	Data type	Default	Description
Retain.CtrlParams.Cool.Ti	REAL	20.0	Active integral action time for cooling in seconds The integral action for cooling is switched off with Cool.CtrlParams.Ti = 0.0. Cool.Ti is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). 100000.0 ≥ Cool.Ti ≥ 0.0
Retain.CtrlParams.Cool.Td	REAL	0.0	Active derivative action time for cooling in seconds The derivative action for cooling is switched off with Cool.CtrlParams.Td = 0.0. Cool.Td is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). 100000.0 ≥ Cool.Td ≥ 0.0
Retain.CtrlParams.Cool.TdFiltRatio	REAL	0.2	Active derivative delay coefficient for cooling The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient • 0.0: Derivative action is effective for one cycle only and therefore almost not effective. • 0.5: This value has proved useful in practice for controlled systems with one dominant time constant. • > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed. Cool.TdFiltRatio is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). Cool.TdFiltRatio ≥ 0.0
Retain.CtrlParams.Cool.PWeighting	REAL	1.0	Active weighting of the proportional action for cooling The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Proportional action for setpoint change is fully effective • 0.0: Proportional action for setpoint change is not effective The proportional action is always fully effective when the process value is changed. Cool.PWeighting is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). 1.0 ≥ Cool.PWeighting ≥ 0.0

Tag	Data type	Default	Description
Retain.CtrlParams.Cool.DWeighting	REAL	1.0	Active weighting of the derivative action for cooling The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Derivative action is fully effective upon setpoint change • 0.0: Derivative action is not effective upon setpoint change The derivative action is always fully effective when the process value is changed. Cool.DWeighting is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). 1.0 ≥ Cool.DWeighting ≥ 0.0
Retain.CtrlParams.Cool.Cycle	REAL	1.0	Active sampling time of the PID algorithm for cooling in seconds CtrlParams.Cool.Cycle is calculated during tuning and rounded off to an integer multiple of CycleTime.. If Config.Output.Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = TRUE, Cool.Cycle is used as time period of the pulse width modulation for cooling. If Config.Output.Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = FALSE, Heat.Cycle is used as time period of the pulse width modulation for cooling. Cool.Cycle is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). 100000.0 ≥ Cool.Cycle > 0.0
Retain.CtrlParams.Cool.ControlZone	REAL	3.402822e+38	Active control zone width for cooling The control zone for cooling is switched off with Cool.ControlZone = 3.402822e+38. Cool.ControlZone is only set automatically during pretuning cooling or pretuning heating and cooling if PIDSelfTune.SUT.TuneRuleCool = 2 is selected as method of the parameter calculation. Cool.ControlZone is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). Cool.ControlZone > 0.0
Retain.CtrlParams.Cool.DeadZone	REAL	0.0	Active deadband width for cooling (see PID parameters (Page 7307)) The deadband for cooling is switched off with Cool.DeadZone = 0.0. Cool.DeadZone is not set automatically or adjusted during tuning. You must correctly configure Cool.DeadZone manually. Cool.DeadZone is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). Cool.DeadZone ≥ 0.0

Note

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller.

See also

PID_Temp ActivateRecoverMode tag (Page 3601)

PID_Temp Warning tag (Page 3604)

Multi-zone controlling with PID_Temp (Page 7327)

PID_Temp state and mode parameters

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID_Temp switches to the operating mode saved in the Mode in-out parameter.

Heat.EnableTuning and Cool.EnableTuning specify for pretuning and fine tuning, if tuning takes place for heating or cooling.

If the CPU is switched on or switches from Stop to RUN mode, PID_Temp starts in the operating mode that is saved in the Mode parameter. To leave PID_Temp in "Inactive" mode, set RunModeByStartup = FALSE.

Meaning of values

State / Mode	Description of operating mode
0	**Inactive** The following output values are output in "Inactive" mode: • 0.0 as PID output value (PidOutputSum) • 0.0 as output value for heating (OutputHeat) and output value for cooling (OutputCool) • 0 as analog output value for heating (OutputHeat_PER) and analog output value for cooling (OutputCool_PER) • FALSE as PWM output value for heating (OutputHeat_PWM) and PWM output value for cooling (OutputCool_PWM) This does not depend on the configured output value limits and scaling in the structures Config.Output.Heat and Config.Output.Cool.

State / Mode	Description of operating mode
1	**Pretuning** The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning. PID_Temp offers different pretuning types, depending on the configuration: • Pretuning heating: A jump change is output at the output value heating, the PID parameters for heating are calculated (Retain.CtrlParams.Heat structure), and control to the setpoint then takes place in automatic mode. If the process behavior strongly depends on the operating point, an adaptation of the delay time can be activated at the setpoint with PIDSelfTune.SUT.AdaptDelayTime. • Pretuning heating and cooling: A jump change is output at the output value heating. As soon as the process value is close to the setpoint, a jump change is output at the output value cooling. The PID parameters for heating (Retain.CtrlParams.Heat structure) and cooling (Retain.CtrlParams.Cool structure) are calculated. Then, control to the setpoint takes place in automatic mode. If the process behavior strongly depends on the operating point, an adaptation of the delay time can be activated at the setpoint with PIDSelfTune.SUT.AdaptDelayTime. Depending on the effect of the cooling actuator compared to the heating actuator, the quality of tuning can be influenced by whether or not the heating and cooling outputs are operated simultaneously during tuning. You can specify this with PIDSelfTune.SUT.CoolingMode. • Pretuning cooling: A jump change is output at the output value cooling and the PID parameters for cooling are calculated (Struktur Retain.CtrlParams.Cool). Then, control to the setpoint takes place in automatic mode. If you want to tune the PID parameters for heating and cooling, you can expect a better control response with "pretuning heating" followed by "pretuning cooling" rather than with "pretuning heating and cooling". However, pretuning in two steps takes longer. **General requirements for pretuning:** • The PID_Temp instruction is called in a cyclic interrupt OB. • Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) • ManualEnable = FALSE • Reset = FALSE • The setpoint and the process value lie within the configured limits. **Requirements for pretuning heating:** • Heat.EnableTuning = TRUE • Cool.EnableTuning = FALSE • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * |Config.InputUpperLimit - Config.InputLowerLimit| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint is greater than the process value.

State / Mode	Description of operating mode
	Setpoint > Input **Requirements for pretuning heating and cooling:** • Heat.EnableTuning = TRUE· • Cool.EnableTuning = TRUE • The cooling output is activated (Config.ActivateCooling = TRUE). • The PID parameter switching is activated (Config.AdvancedCooling = TRUE). • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * |Config.InputUpperLimit - Config.InputLowerLimit| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint is greater than the process value. Setpoint > Input **Requirements for pretuning cooling:** • Heat.EnableTuning = FALSE· • Cool.EnableTuning = TRUE· • The cooling output is activated (Config.ActivateCooling = TRUE). • The PID parameter switching is activated (Config.AdvancedCooling = TRUE). • A "pretuning heating" or "pretuning heating and cooling" has been successful (PIDSelfTune.SUT.ProcParHeatOk = TRUE), if possible at the same setpoint. • The process value must be close to the setpoint. |Setpoint - Input| < 0.05 * |Config.InputUpperLimit - Config.InputLowerLimit| The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" or "Manual mode". The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel The method for calculation of the PID parameters can be specified separately for heating and cooling with PIDSelfTune.SUT.TuneRuleHeat and PIDSelfTune.SUT.TuneRuleCool. Before the PID parameters are recalculated, they are backed up in the CtrlParamsBackUp structure and can be reactivated with LoadBackUp. After successful pretuning, the switch is made to automatic mode. After unsuccessful pretuning, the switch to the mode is determined by ActivateRecoverMode. The phase of pretuning is indicated with PIDSelfTune.SUT.State.

State / Mode	Description of operating mode
2	**Fine tuning** Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning. PID_Temp automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. PID_Temp offers different fine tuning types, depending on the configuration: • Fine tuning heating: PID_Temp generates an oscillation of the process value with periodic changes at the output value heating and calculates the PID parameters for heating (Struktur Retain.CtrlParams.Heat). • Fine tuning cooling: PID_Temp generates an oscillation of the process value with periodic changes at the output value cooling and calculates the PID parameters for cooling (Struktur Retain.CtrlParams.Cool). **Temporary tuning offset for heating/cooling controllers** If PID_Temp is used as heating/cooling controller (Config.ActivateCooling = TRUE), the PID output value (PidOutputSum) at the setpoint must meet the following requirements for a process value oscillation to be generated and fine tuning to be successful: • Positive PID output value for fine tuning heating • Negative PID output value for fine tuning cooling If this requirement is not met, you can define a temporary offset for fine tuning which is output at the output with the opposite effect: • Offset for cooling output (PIDSelfTune.TIR.OutputOffsetCool) with fine tuning heating. Define a negative tuning offset cooling which is less than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning. • Offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) with fine tuning cooling. Define a positive tuning offset heating which is greater than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning. The defined offset is balanced by the PID algorithm so that the process value remains at the setpoint. This means the size of the offset can be adapted accordingly with the PID output value so that it meets the requirements listed above. To avoid larger overshoots of the process value when defining the offset, it can also be increased in several steps. If PID_Temp exits the fine tuning mode, the tuning offset is reset. Example for definition of an offset for fine tuning cooling: • Without offset: – Setpoint = Process value (ScaledInput) = 80°C – PID output value (PidOutputSum) = 30.0 – Output value heating (OutputHeat) = 30.0 – Output value cooling (OutputCool) = 0.0 An oscillation of the process value around the setpoint cannot be created with the cooling output alone. Fine tuning would fail here. • With definition of an offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) = 80.0 – Setpoint = process value (ScaledInput) = 80°C – PID output value (PidOutputSum) = -50.0

State / Mode	Description of operating mode
	– Output value heating (OutputHeat) = 80.0 – Output value cooling (OutputCool) = -50.0 By defining an offset for the heating output, the cooling output can now create an oscillation of the process value around the setpoint. This means fine tuning can take place successfully. **General requirements for fine tuning:** • The PID_Temp instruction is called in a cyclic interrupt OB. • No disturbances are expected. • The setpoint and the process value lie within the configured limits. • The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint. • ManualEnable = FALSE • Reset = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode **Requirements for fine tuning heating:** • Heat.EnableTuning = TRUE • Cool.EnableTuning = FALSE • If PID_Temp is configured as heating/cooling controller (Config.ActivateCooling = TRUE), the heating output must be active at the operating point at which tuning is to take place (PidOutputSum > 0.0 (see tuning offset)). **Requirements for fine tuning cooling:** • Heat.EnableTuning = FALSE • Cool.EnableTuning = TRUE • The cooling output is activated (Config.ActivateCooling = TRUE). • The PID parameter switching is activated (Config.AdvancedCooling = TRUE) • The cooling output must be active at the operating point at which tuning is to take place (PidOutputSum < 0.0 (see tuning offset)). **The course of fine tuning is determined by the mode from which it is started:** • Automatic mode (State = 3) with PIDSelfTune.TIR.RunIn = FALSE (default) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_Temp controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) with PIDSelfTune.TIR.RunIn = TRUE Attempts are made to reach the setpoint with the minimum or maximum output value: – with minimum or maximum output value heating for fine tuning heating – with minimum or maximum output value cooling for fine tuning cooling. This can produce increased overshoot. Fine tuning starts when the setpoint is reached.

State / Mode	Description of operating mode
	If the setpoint cannot be reached, PID_Temp does not automatically abort tuning. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel The method for calculation of the PID parameters can be specified separately for heating and cooling with PIDSelfTune.TIR.TuneRuleHeat and PIDSelfTune.TIR.TuneRuleCool. Before the PID parameters are recalculated, they are backed up in the CtrlParamsBackUp structure and can be reactivated with LoadBackUp. The controller changes to automatic mode after successful fine tuning. After unsuccessful fine tuning, the switch to the mode is determined by ActivateRecoverMode. The "Fine tuning" phase is indicated with PIDSelfTune.TIR.State.
3	**Automatic mode** In automatic mode, PID_Temp corrects the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is met: • Pretuning successfully completed • Fine tuning successfully completed • Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate. The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor. The ActivateRecoverMode tag is taken into consideration in automatic mode.
4	**Manual mode** In manual mode, you specify a manual PID output value in the ManualValue parameter. The values at the outputs for heating and cooling resulting from this manual value are the result of the configured output scaling. You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only. The switchover from manual mode to automatic mode is bumpless. The ActivateRecoverMode tag is taken into consideration in manual mode.
5	**Substitute output value with error monitoring** The control algorithm is deactivated. The SetSubstituteOutput tag determines which PID output value (PidOutputSum) is output in this operating mode. • SetSubstituteOutput = FALSE: Last valid PID output value • SetSubstituteOutput = TRUE: Substitute output value (SubstituteOutput) You cannot activate this operating mode using Mode = 5. In the event of an error, it is activated instead of "Inactive" operating mode if all the following conditions are met: • Automatic mode (State = 3) • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode is effective. As soon as the errors are no longer pending, PID_Temp switches back to automatic mode.

ENO characteristics

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

Automatic switchover of operating mode during commissioning

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID_Temp automatically switches the operating mode in the event of an error.

The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. When you start pretuning or fine tuning, PID_Temp has saved the value of State in the Mode in-out parameter. This means PID_Temp switches to the mode from which tuning was started.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode.

See also

Output parameters of PID_Temp (Page 3561)

PID_Temp in/out parameters (Page 3564)

PID_Temp ErrorBits parameter

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0000003h, for example, indicates that the errors 0000001h and 0000002h are pending simultaneously.

ErrorBits (DW#16#...)	Description
0000000	There is no error.
0000001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode. If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0000002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode. If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0000004	Error during fine tuning. Oscillation of the process value could not be maintained. If PID_Temp is used as heating-cooling controller (Config.ActivateCooling = TRUE), the PID output value (PidOutputSum) at the setpoint must be positive for fine tuning heating and negative • for fine tuning cooling to be able • to generate actual value oscillation If this requirement is not met, use the tuning offsets (PIDSelfTune.TIR.OutputOffsetCool and PIDSelf-Tune.TIR.OutputOffsetHeat tags), see Fine tuning (Page 7316). If ActivateRecoverMode was = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0000008	Error at start of pretuning. The process value is too close to the setpoint or greater than the setpoint. Start fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0000010	The setpoint was changed during tuning. You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0000020	Pretuning is not permitted during fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp remains in fine tuning mode.
0000040	Error during pretuning. Cooling could not reduce the process value. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
0000100	Error during fine tuning resulted in invalid parameters. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0000200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode. If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0000400	Calculation of output value failed. Check the PID parameters. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0000800	Sampling time error: PID_Temp is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode. If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0001000	Invalid value at "Setpoint" parameter or "SubstituteSetpoint": Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode. If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0010000	Invalid value at ManualValue parameter. Value has an invalid number format. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp remains in manual mode and uses SubstituteOutput as PID output value. As soon as you specify a valid value in ManualValue, PID_Temp uses it as the PID output value.
0020000	Invalid value at SubstituteOutput tag. Value has an invalid number format. PID_Temp remains in the "Substitute output value with error monitoring" mode or manual mode and uses the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit) as PID output value. As soon as you specify a valid value in SubstituteOutput, PID_Temp uses it as the PID output value.
0040000	Invalid value at Disturbance parameter. Value has an invalid number format. If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_Temp remains in automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled.

ErrorBits (DW#16#...)	Description
0200000	Error in Master in the cascade: Slaves are not in automatic mode or have activated substitute setpoint and prevent tuning of the master. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0400000	Fine tuning heating is not permitted while cooling is active. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0800000	The process value must be close to the setpoint to start pretuning cooling. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
1000000	Error at start of tuning: Heat.EnableTuning and Cool.EnableTuning are not set or do not match the configuration. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
2000000	Pretuning cooling requires successful pretuning heating. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
4000000	Error at start of fine tuning: Heat.EnableTuning and Cool.EnableTuning must not be set simultaneously. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
8000000	Error during calculation of the PID parameters resulted in invalid parameters. The invalid parameters are discarded and the original PID parameters are retained unchanged. We can distinguish between the following cases: • If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode. • If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. • If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.

PID_Temp ActivateRecoverMode tag

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

Automatic mode and manual mode

Notice

Your system may be damaged.

If ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode or in manual mode even if there is an error and the process limit values are exceeded.

This may damage your system.

It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

ActivateRecoverMode	Description
FALSE	PID_Temp switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	**Automatic mode** If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_Temp switches between the calculated PID output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or several of the following errors occur and automatic mode was active before the error occurred, PID_Temp remains in automatic mode: • 0000001h: The "Input" parameter is outside the process value limits. • 0000800h: Sampling time error • 0040000h: Invalid value at Disturbance parameter. • 8000000h: Error during calculation of the PID parameters If one or several of the following errors occur and automatic mode was active before the error occurred, PID_Temp switches to "Substitute output value with error monitoring" mode: • 0000002h: Invalid value at Input_PER parameter. • 0000200h: Invalid value at Input parameter. • 0000400h: Calculation of output value failed. • 0001000h: Invalid value at Setpoint parameter or SubstituteSetpoint. As soon as the errors are no longer pending, PID_Temp switches back to automatic mode. If the following error occurs in "Substitute output value with error monitoring" mode, PID_Temp sets the PID output value to Config.Output.Heat.PidLowerLimit as long as this error is pending: • 0020000h: Invalid value at SubstituteOutput tag. Value has an invalid number format. This behavior is independent of SetSubstituteOutput.
	Manual mode If one or several errors occur and manual mode was active before the error occurred, PID_Temp remains in manual mode. If the following error occurs in manual mode, as long as this error is pending, PID_Temp sets the PID output value to SubstituteOutput: • 0010000h: Invalid value at ManualValue parameter. Value has an invalid number format. If the error 0010000h is pending in manual mode and the following error occurs, PID_Temp sets the PID output value to Config.Output.Heat.PidLowerLimit as long as this error is pending: • 0020000h: Invalid value at SubstituteOutput tag. Value has an invalid number format. This behavior is independent of SetSubstituteOutput.

Pretuning and fine tuning

ActivateRecoverMode	Description
FALSE	PID_Temp switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If the following error occurs, PID_Temp remains in the active mode: • 0000020h: Pretuning is not permitted during fine tuning. The following errors are ignored: • 0010000h: Invalid value at ManualValue parameter. • 0020000h: Invalid value at SubstituteOutput tag. When any other error occurs, PID_Temp cancels the tuning and switches to the mode from which tuning was started.

PID_Temp Warning tag

If several warnings are pending simultaneously, the values of the Warning tag are displayed with binary addition. If the warning 0000003h is displayed, for example, the warnings 0000001h and 0000002h are pending simultaneously.

Warning (DW#16#....)	Description
0000000	No warning pending.
0000001	The point of inflection was not found during pretuning.
0000004	The setpoint was limited to the configured limits.
0000008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRuleHeat method or TIR.TuneRuleCool = 3.
0000010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0000020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0000040	The process value exceeded one of its warning limits.
0000080	Invalid value at Mode. The operating mode is not switched.
0000100	The manual value was limited to the limits of the PID output value.
0000200	The specified rule for tuning is not supported. No PID parameters are calculated.
0001000	The substitute output value cannot be reached because it is outside the output value limits.
0004000	The specified number of the output value for heating and/or cooling is not supported. Only the output OutputHeat or OutputCool is used.
0008000	Invalid value at PIDSelfTune.SUT.AdaptDelayTime. The default value 0 is used.
0010000	Invalid value at PIDSelfTune.SUT.CoolingMode. The default value 0 is used.
0020000	The activation of cooling (Config.ActivateCooling tag) is not supported by the controller that is used as master (Config.Cascade.IsMaster tag). PID_Temp works as heating controller. Set the Config.ActivateCooling tag to FALSE.
0040000	Invalid value at Retain.CtrlParams.Heat.Gain, Retain.CtrlParams.Cool.Gain or Config.CoolFactor. PID_Temp supports only positive values for proportional gain (heating and cooling) and cooling factor. Automatic mode remains active with PID output value 0.0. The integral component is stopped.

The following warnings are deleted as soon as the cause has been remedied or you repeat the action with valid parameters:

- 0000001h
- 0000004h
- 0000008h
- 0000040h
- 0000100h

All other warnings are cleared with a rising edge at Reset or ErrorAck.

PwmPeriode tag

If the PID algorithm sampling time (Retain.CtrlParams.Heat.Cycle or Retain.CtrlParams.Heat.Cycle) and thus the time period of the pulse width modulation is very high when you use OutputHeat_PWM or OutputCool_PWM, you can define a deviating shorter time period at the Config.Output.Heat.PwmPeriode or Config.Output.Cool.PwmPeriode parameters to improve the smoothness of the process value.

Time period of the pulse width modulation at OutputHeat_PWM

Time period of the PWM at output OutputHeat_PWM depending on Config.Output.Heat.PwmPeriode:

- Heat.PwmPeriode = 0.0 (default)
  The sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as time period of the PWM.
- Heat.PwmPeriode > 0.0
  The value is rounded off to an integer multiple of the PID_Temp sampling time (CycleTime.Value) and used as time period of the PWM.
  The value must meet the following conditions:
  - Heat.PwmPeriode ≤ Retain.CtrlParams.Heat.Cycle
  - Heat.PwmPeriode > Config.Output.Heat.MinimumOnTime
  - Heat.PwmPeriode > Config.Output.Heat.MinimumOffTime

Time period of the pulse width modulation at OutputCool_PWM

Time period of the PWM at output OutputCool_PWM depending on Config.Output.Cool.PwmPeriode and the method for heating/cooling:

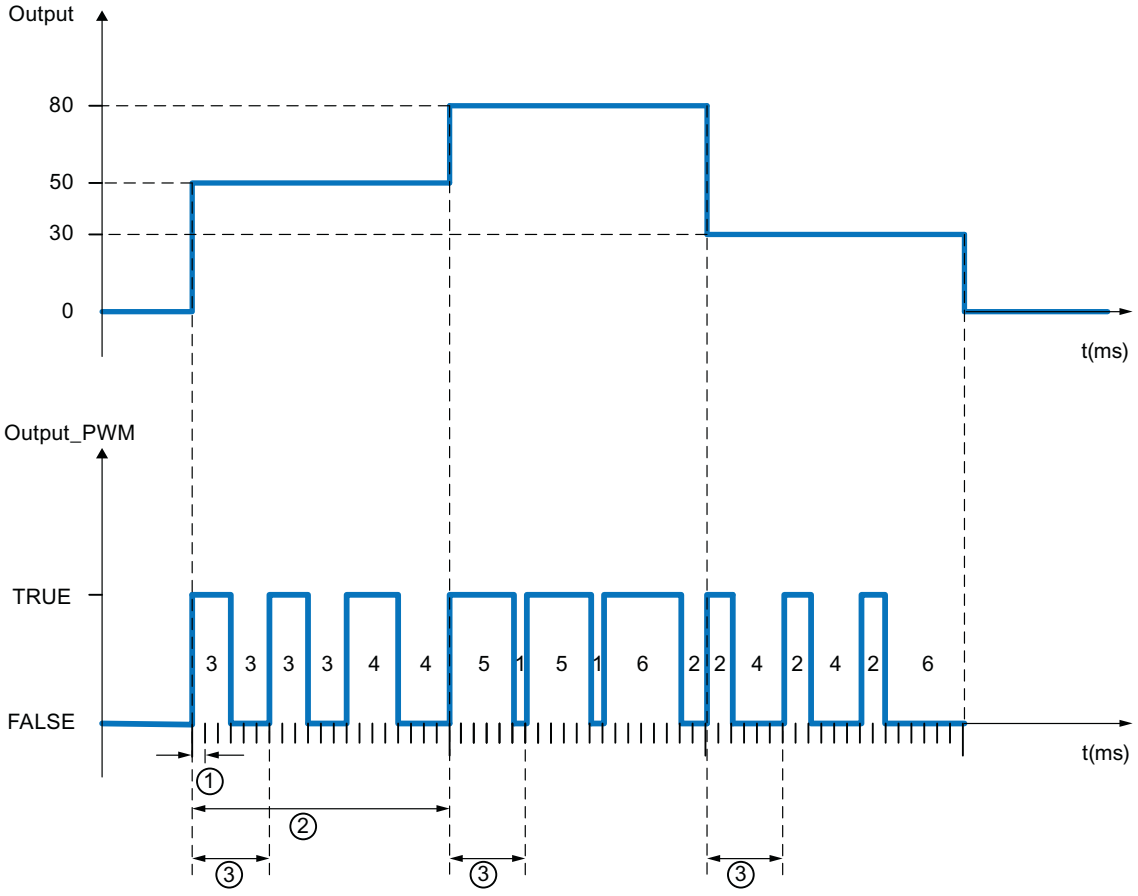
- Cool.PwmPeriode = 0.0 and cooling factor (Config.AdvancedCooling = FALSE):
  The sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as time period of the PWM.
- Cool.PwmPeriode = 0.0 and PID parameter switching (Config.AdvancedCooling = TRUE):
  The sampling time of the PID algorithm for cooling (Retain.CtrlParams.Cool.Cycle) is used as time period of the PWM.
- Cool.PwmPeriode > 0.0:
  The value is rounded off to an integer multiple of the PID_Temp sampling time (CycleTime.Value) and used as time period of the PWM.
  The value must meet the following conditions:
  - Cool.PwmPeriode ≤ Retain.CtrlParams.Cool.Cycle or Retain.CtrlParams.Heat.Cycle
  - Cool.PwmPeriode > Config.Output.Cool.MinimumOnTime
  - Cool.PwmPeriode > Config.Output.Cool.MinimumOffTime

Config.Output.Cool.PwmPeriode is only effective if the cooling output is activated (Config.ActivateCooling =TRUE).

When you use PwmPeriode, the accuracy of the PWM output signal is determined by the relationship of PwmPeriode to the PID_Temp sampling time (cycle time of the OB). PwmPeriode should be at least 10 times the PID_Temp sampling time.

If the sampling time of the PID algorithm is not an integer multiple of PwmPeriode, each last period of the PWM within the sampling time of the PID algorithm is extended accordingly.

Example for OutputHeat_PWM



① PID_Temp sampling time = 100.0 ms (cycle time of the calling cyclic interrupt OB, CycleTime.Value tag)

② PID algorithm sampling time = 2000.0 ms (Retain.CtrlParams.Heat.Cycle tag)

③ Time period of the PWM for heating = 600.0 ms (Config.Output.Heat.PwmPeriode tag)

11.6.5 Communication

11.6.5.1 S7 communication

Data consistency

Definition

A data block which cannot be modified by concurrent processes is called consistent data area. A data block which is larger than the consistent data area can thus be falsified as a whole during transmission. This means that a data block which belongs together and which is larger than consistent data area can consist in part of new and of old consistent data at the same time.

Example

An inconsistency can occur when an instruction for communication is interrupted, for example, by a hardware interrupt OB with higher priority. If the user program in this OB now changes the data that has already been partly processed by the instruction, the transferred data originates:

- In part from the time before the hardware interrupt was processed
- And in part from the time after the hardware interrupt was processed

This means that these data are inconsistent (not coherent).

Ensuring data consistency

If the communication process can be interrupted by an interrupt OB, you must ensure that the data is transferred consistently. Make sure that only an image of the data and not the data to be transferred is not changed directly by the interrupt OB. Copy the image of the data to the transfer area of the communication instruction prior to the next data transfer.

- If the user program contains a communication instruction that accesses common data, access to this data area can be coordinated, for example, by means of the DONE parameter. The data consistency of the communication areas which are transferred locally with a communication instruction can therefore be ensured in the user program.
- In the case of S7 communication instructions "PUT (Page 3615)"/"GET (Page 3612)", the size of the consistent data areas must already be taken into consideration during programming or configuration, because a communication block is not available in the user program of the target device (server) to synchronize communication data to the user program.

- For the S7-300 and C7-300 (exception: CPU 318-2 DP) the communication data are copied consistently into the user memory in blocks of 32 bytes in the cycle control point of the operating system. Data consistency is not guaranteed for larger data areas. If a defined data consistency is required, the communication data in the user program may not exceed 32 bytes (maximum of 8 bytes, depending on the version).
- In the S7-400 and S7-1500, on the other hand, the communication data in 462-byte blocks is not processed at the cycle control point, but in fixed time slices during the program cycle. The consistency of a tag is ensured by the system. These communication areas can then be accessed consistently using the "PUT (Page 3615)" / "GET (Page 3612)" instructions or when reading/writing tags, for example by an OP or an OS.

Note

Additional information on data consistency is provided in the description of the specific instructions.

Effect on interrupt reaction times

The interrupt reaction times of the CPU are slightly extended by copying the data. The greater the quantity of data which must be transferred with absolute consistency, the longer the interrupt reaction time of a system.

Common parameters of instructions for S7 communication

Classification

The parameters of the instructions for S7 communication can be divided into the following five categories according to their functions:

1. Control parameters are used to activate an instruction.
2. Addressing parameters are used to address the remote communication partner.
3. Send parameters point to the data areas that are to be sent to the remote partner.
4. Receive parameters point to the data areas where the data received from remote partners will be entered.
5. Status parameters are used to monitor whether the instruction has completed its task without error or for the analysis of any errors that have occurred.

Control parameter

Data exchange will only be activated if the appropriate control parameters have a defined value (for example, are set) when the instruction is called or when the value has undergone a specific change since the previous call (for example, a positive edge).

Addressing parameters

Parameter	Description
ID	Reference to the local connection description (specified by the configuration of the connection).
R_ID	With the R_ID parameter, you specify that a send and a receive instruction belong together: The R_ID parameter must match in the instruction at the sending end and the instruction at the receiving end. This allows the communication of several instruction pairs via the same logic connection. • R_ID must be specified in the form DW#16#wxyzWXYZ. • The instruction pairs of a logical connection specified in R_ID must be unique for this connection.

Note

Addressing parameters ID and R_ID

You can reassign the addressing parameters ID and R_ID during runtime. The new parameters are validated with each new job after the previous job has been closed.

You can use the following options to reduce the number of instance DBs and therefore the work memory required:

1. With tag IDs, you can use several connections via one data instance block.
2. With tag R_IDs, you can define several pairs of sending and receiving instructions for one job with a single instance.
3. You can combine cases 1 and 2.

Please note that the new parameters are valid after the last job is executed. When you activate the sending operation, the R_ID parameter in the instruction at the sending end must match its counterpart at the receiving end.

Status parameter

With the status parameters, you monitor whether the instruction has completed its task correctly or whether it is still active. The status parameters also indicate errors.

Note

The status parameters are valid for one cycle only, namely from the first command following the call until the next call. As a result, you must evaluate these parameters after each instruction cycle.

Send and receive parameters

For communication instructions configured at both ends

- The number of the SD_i and RD_i parameters used must match at the send and receive end.
- The data types of the SD_i and RD_i parameters that belong together must match at the send and receive end.
- The amount of data to be sent according to the SD_i parameter must not exceed the range made available by the corresponding RD_i (does not apply to "BSEND (Page 3621)" / "BRCV (Page 3623)"). The RD_i parameters must (with exception of "BSEND"/"BRCV") have the identical data size.

If you do not keep to the rules above, this is indicated by ERROR = 1 and STATUS = 4.

Note

Supplying the send and receive parameters

With the VARIANT data type, send and receive parameters must always be supplied any time a communication instruction is called. It is not possible, for example, to supply the send parameters of the communication instructions at startup and to only trigger the send job in cyclic operation.

User data size

With the "USEND (Page 3618)", "URCV (Page 3619)", "GET (Page 3612)" and "PUT (Page 3615)" instructions, the amount of data to be transferred must not exceed a defined user data size. The maximum user data size depends on:

- The instruction used
- The communication partner.

The guaranteed minimum size of the user data for an instruction with 1-4 tags is listed in the following table:

Instruction	Partner: S7-300	Partner: S7-400	Partner: S7-1200	Partner: S7-1500
PUT / GET	160 bytes	400 bytes	160 bytes	880 bytes
USEND / URCV	160 bytes	440 bytes	-	920 bytes
BSEND / BRCV	32768/65534 bytes	65534 bytes	-	• 65534 bytes with standard access • 65535 bytes with optimized access

Further information on the user data size can be found in the technical data of the respective CPU.

Exact user data size

If the user data size specified above is insufficient you can determine the maximum byte length of the user data as follows:

First read the data block size valid for communication from the following table:

Local CPU	Remote CPU	Data block size in bytes
S7-1200	Any	240
S7-1500	S7-300	240
	S7-400	480
	S7-1200	240
	S7-1500	960

Use this value in the following table to read the maximum possible user data length in bytes as total of the parameters used. It applies for even lengths of the areas SD_i, RD_i, and ADDR_i.

For each range of uneven length the maximum possible user data length is reduced by one byte.

		Number of SD_i, RD_i, ADDR_i parameters used			
Data block size	Instruction	1	2	3	4
240 (S7-300)	PUT/GET/ USEND	160	-	-	-
240 (S7-300 via integrated interface)	PUT	212	-	-	-
	GET	222	-	-	-
	USEND	212	-	-	-
240 (S7-400)	PUT	212	196	180	164
	GET	222	218	214	210
	USEND	212	-	-	-
480 (S7-400)	PUT	452	436	420	404
	GET	462	458	454	450
	USEND	452	448	444	440
240 (S7-1200)	PUT	212	196	180	164
	GET	222	218	214	210
960 (S7-1500)	PUT	932	916	900	884
	GET	942	938	934	930
	USEND	932	928	924	920

GET: Read data from a remote CPU

Description

With the instruction "GET", you can read data from a remote CPU.

The instruction is started on a positive edge at control input REQ:

- The relevant pointers to the areas to be read out (ADDR_i) are then sent to the partner CPU. The partner CPU can be in RUN or STOP mode.
- The partner CPU returns the data:
  - If the reply exceeds the maximum user data length, this is displayed with error code "2" at the STATUS parameter.
  - The received data is copied to the configured receive areas (RD_i) at the next call.
- Completion of this action is indicated by the status parameter NDR having the value "1".

Reading can only be activated again after the previous reading process has been completed. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being read or if the data type check results in an error.

Changes in the data areas addressed on the partner CPU are not registered by the "GET" instruction.

Requirements for using the instruction

- The "Permit access with PUT/GET communication from remote partner" function was activated in the properties of the partner CPU under "Protection".
- The blocks which you access with the "GET" instruction were created with the access type "standard".
- Make sure that the areas defined with the parameters ADDR_i and SD_i match in terms of number, length and data type.
- The area to be read (ADDR_i parameter) cannot be larger than the area for data storage (RD_i parameter).

Parameters

The following table shows the parameters of the "GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange on a positive edge.
ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameters for specifying the connection to the partner CPU.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: – 0000H: Neither warning nor error – <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
ADDR_1	InOut	REMOTE	I, Q, M, D	Pointers to the areas on the partner CPU that are to be read. When the REMOTE pointer accesses a DB, the DB must always be specified. Example: P#DB10.DBX5.0 Byte 10.
ADDR_2	InOut	REMOTE		
ADDR_3	InOut	REMOTE		
ADDR_4	InOut	REMOTE		
RD_1	InOut	VARIANT	I, Q, M, D, L	Pointers to the areas on the local CPU in which the read data is entered.
RD_2	InOut	VARIANT		
RD_3	InOut	VARIANT		
RD_4	InOut	VARIANT		

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "GET" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	2	• Negative acknowledgment from the partner device. The function cannot be executed. • Response from the remote station exceeds the maximum user data length (see: Common parameters of instructions for S7 communication (Page 3609)). • Access protection is activated on the partner CPU. Deactivate access protection in the CPU settings.
1	4	Error in the pointers to the data storage RD_i: • Data types of the parameters RD_i and ADDR_i are not compatible with each other. • The length of the area RD_i is smaller than the length of the data of the ADDR_i parameter that is to be read.
1	8	Access error on the partner CPU.

ERROR	STATUS (decimal)	Explanation
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).

Note

Data consistency

Data is received consistently if you read the part of the receive area RD_i currently being used completely before initiating another job.

PUT: Write data to a remote CPU

Description

You can write data to a remote CPU with the instruction "PUT".

The instruction is started on a positive edge at control input REQ:

- The pointers to the areas to be written (ADDR_i) and the data (SD_i) are then sent to the partner CPU. The partner CPU can be in RUN or STOP mode.
- The data to be sent is copied from the configured send areas ((SD_i). The partner CPU saves the sent data under the addresses supplied with the data and returns an execution acknowledgment.
- If no errors occur, this is indicated at the next instruction call with status parameter DONE = "1". The writing process can only be activated again after the last job is complete.

Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being written or if the execution check results in an error.

Requirements for using the instruction

- The "Permit access with PUT/GET communication from remote partner" function was activated in the properties of the partner CPU under "Protection".
- The blocks which you access with the "PUT" instruction were created with the access type "standard".
- Make sure that the areas defined with the parameters ADDR_i and SD_i match in terms of number, length and data type.
- The area to be written (ADDR_i parameter) must be as large as the send area (SD_i parameter).

Parameters

The following table shows the parameters of the "PUT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange on a positive edge.
ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameters for specifying the connection to the partner CPU.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: – 0000H: Neither warning nor error – <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
ADDR_1	InOut	REMOTE	I, Q, M, D	Pointers to the areas on the partner CPU to which the data will be written. When the REMOTE pointer accesses a DB, the DB must always be specified. Example: P#DB10.DBX5.0 Byte 10. When transferring data structures (e.g., Struct) the data type CHAR must be used at the ADDR_i parameters.
ADDR_2	InOut	REMOTE		
ADDR_3	InOut	REMOTE		
ADDR_4	InOut	REMOTE		
SD_1	InOut	VARIANT	I, Q, M, D, L	Pointers to the areas on the local CPU which contain the data to be sent. Only the data types BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL are permitted. When transferring data structures (e.g., Struct) the data type CHAR must be used at the SD_i parameters.
SD_2	InOut	VARIANT		
SD_3	InOut	VARIANT		
SD_4	InOut	VARIANT		

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "PUT" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example, cable, CPU off, CP in STOP mode). • Connection to partner not yet established.
1	2	• Negative acknowledgment of partner CPU. The function cannot be executed. • Access on the partner CPU was not granted. Activate the access in the CPU settings.
1	4	Error in the pointers to the data storage: • Data types of the parameters SD_i and ADDR_i are not compatible with each other. • The length of the area SD_i is greater than the length of the data of the ADDR_i parameter that is to be written. • Not possible to access SD_i. • Maximum user data size exceeded. • Number of the parameters SD_i and ADDR_i does not match.
1	8	Access error with the partner CPU (e.g. DB not loaded or write-protected).
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).

Data consistency

When a send operation is activated (positive edge at REQ), the data to be sent from the send area SD_i is copied from the user program. After the block call, you can write to these areas without corrupting the current send data.

Note

The send operation is only complete when the DONE status parameter has the value "1".

Other

USEND: Send data uncoordinated

Description

The "USEND" instruction sends data to a remote partner instruction of type "URCV (Page 3619)". The sending process is carried out without coordination with the partner instruction. This means that the data transfer is carried out without acknowledgment by the partner instruction.

When a send operation is activated (positive edge at REQ), the data to be sent from the send areas SD_i are copied from the user program. After the instruction call, you can write to these send areas again without corrupting the current send data.

Successful completion of the send operation is indicated by the value of the DONE status parameter being set to "1".

Parameters

The following table shows the parameters of the "USEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange on a positive edge.
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter R_ID for defining the instruction pairs "USEND" and "URCV". See also: Common parameters of instructions for S7 communication (Page 3609)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred. STATUS supplies detailed information on the type of error.

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.
SD_i (1≤ i ≤4)	InOut	VARIANT	I, Q, M, D	Pointer on the i-th send area. Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL or STRUCT. The maximum user data size for SD_i parameters depends on the partner CPU ("URCV" instruction) and on the number of parameters used. Additional information is available in: Common parameters of instructions for S7 communication (Page 3609)

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameters ERROR and STATUS

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	4	• Errors in the send area pointers SD_i relating to the data length or the data type. • Maximum user data length was exceeded.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	Value at R_ID parameter already exists in the connection specified at the ID parameter (R_ID value must be unique for the connection).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).

URCV: Receive data uncoordinated

Description

The "URCV" instruction receives data asynchronously from a remote partner instruction of type "USEND (Page 3618)" and copies this data into the configured receive areas.

The instruction is ready to receive if there is a logical 1 at the EN_R input. An active job can be canceled with EN_R=0.

The receive data areas are referenced by the parameters RD_1 to RD_4. Make sure that the areas defined by the parameters RD_i / RD_1 and SD_i / SD_1 (with the associated partner instruction "USEND (Page 3618)") match in terms of number and length.

Successful completion of the copy operation is indicated when the value of the NDR status parameter is set to logical "1". Once the status parameter NDR has changed to "1", there will be new receive data in your receive areas (RD_i). A new block call may cause these data to be overwritten with new receive data. If you want to prevent this, call "URCV" (for example, during cyclic block processing) with the value 0 at EN_R until you have finished processing the received data.

Parameters

The following table shows the parameters of the "URCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter for defining the instruction pairs "USEND" and "URCV". See also: Common parameters of instructions for S7 communication (Page 3609)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error • 1: An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.
RD_i (1≤ i ≤4)	InOut	VARIANT	I, Q, M, D	Pointer to the i-th receive area: Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL or STRUCT. Additional information is available in: Common parameters of instructions for S7 communication (Page 3609)

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameters ERROR and STATUS

ERROR	STATUS (decimal)	Explanation
0	9	Warning: older received data were overwritten by newer received data.
0	11	Warning: The received data is already being processed in a priority class with lower priority (error can occur when copying data to the receive area).
0	25	Communication has started. The job is being processed.

ERROR	STATUS (decimal)	Explanation
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	4	Errors in the receive area pointers RD_i relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	Value at R_ID parameter already exists in the connection specified at the ID parameter (R_ID value must be unique for the connection).
1	19	The associated "USEND (Page 3618)" instruction sends data faster than it can be copied to the receive areas by "URCV".
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).

BSEND: Send data in segments

Description

The "BSEND" instruction sends data to a remote partner instruction of type "BRCV (Page 3623)". With this type of data transfer, more data can be transported between the communication partners than is possible with all other communication instructions for configured S7 connections. In each case, the maximum data volume amounts to 65534 bytes (standard access) or 65535 bytes (optimized access) for the integrated interface and for SIMATIC Net CP.

Functional description

You specify the instruction pair "BSEND" and "BRCV" with the input parameter R_ID. The R_ID parameter must be identical in the related instructions.

The send job is activated after the instruction is called and a positive edge is detected at control input REQ. After the call, "BSEND" is not processed in the background, which means the data can only be read within the user program.

The data area to be transmitted is segmented. Each segment is sent individually to the partner. Each segment is acknowledged by the partner after the application of this segment by "BRCV (Page 3623)". If the data is segmented, "BSEND" must be called multiple times until all segments have been transferred.

The data area of the data to be sent is specified by SD_1. To ensure data consistency, you may only write to the part of the currently used sending area SD_1 after the current send operation is complete. This is the case when the value of the status parameter DONE changes to "1".

The length of the send data is specified by LEN based on the job. With LEN = "0", all data addressed with the SD_1 parameter is sent.

If there is a positive edge at control input R, the current sending process is canceled.

Due to the asynchronous data transfer, a new data transfer can only be initiated if the previous data has been fetched by a partner instruction call. When the data has been collected, the status parameter "NDR" is set in the partner instruction "BRCV".

Note

Migration of S7-400 user programs

An S7-400 CPU interprets the parameter SD_1 as pointer and not as data area.

With S7-1500, LEN may not exceed the area of SD1. This was permitted with S7-400. Recommendation: Use the maximum size for the LEN parameter (65534 bytes for the integrated interface) as size of the data area at the SD_1 parameter.

Parameters

The following table shows the parameters of the "BSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange at a positive edge
R	Input	BOOL	I, Q, M, D, L or constant	Control parameter reset, activates an abort of the current data exchange at a positive edge
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter for defining the instruction pairs "BSEND" and "BRCV (Page 3623)". See also: Common parameters of instructions for S7 communication (Page 3609)
SD_1	InOut	VARIANT	I, Q, M, D	Pointer to send area
LEN	InOut	WORD	I, Q, M, D, L	Length of the block of data to be sent in bytes. With LEN = "0", all data is sent by SD_1.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameters ERROR and STATUS

The following table contains all specific error information for "BSEND" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	2	Negative acknowledgment of partner instruction. The instruction cannot be executed.
1	3	R_ID is unknown on the connection specified by the ID or the receive block has not yet been called.
1	4	• Error in the data length or data type in the send area pointer SD_1. • Value for LEN is greater than area SD_1.
1	5	Reset request was executed.
1	6	Partner instruction is in DISABLED status (EN_R has the value "0"). Also check the input parameters of "BRCV (Page 3623)" for consistency with "BSEND".
1	7	"BRCV (Page 3623)" partner instruction not called since the last data transfer.
1	8	Access to remote object in the user memory was rejected: The destination area for the associated "BRCV (Page 3623)" is too small. ERROR = 1, STATUS = 4 or ERROR = 1, STATUS = 10 reported at the output parameters of "BRCV (Page 3623)".
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	R_ID already exists in the connection.
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).

BRCV: Receive data in segments

Description

The "BRCV" instruction receives data from a remote partner instruction of type "BSEND (Page 3621)". The R_ID parameter must be identical in the related instructions.

The instruction is ready to receive data (STATUS = 25) after it is called with the value "1" at control input EN_R. An active job can be canceled with EN_R=0.

The maximum receive area is specified by RD_1. Data will be received consistently if you fully evaluate the part of the RD_1 receive area currently in use before calling the block again with value 1 at control input EN_R.

After each received data segment, an acknowledgment is sent to the partner instruction. If there are multiple segments it will be necessary to call "BRCV" several times until all of the segments have been received. Asynchronous data receipt is indicated by STATUS = 17. The

amount of data currently received is indicated at the parameter LEN. The RD_1 parameter must remain constant during the operation.

Value "1" at the NDR status parameter indicates that all data segments have been received without error. The received data remains unchanged until the next call with EN_R=1.

Parameters

The following table shows the parameters of the "BRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter for defining the instruction pairs "BSEND (Page 3621)" and "BRCV". See also: Common parameters of instructions for S7 communication (Page 3609)
RD_1	InOut	VARIANT	I, Q, M, D	Pointer to the receive area.
LEN	InOut	WORD	I, Q, M, D, L	Length of the data already received in bytes.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error • 1: An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameters ERROR and STATUS

The following table contains all specific error information for "BRCV" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	17	Warning: Instruction receives data asynchronously. The LEN parameter shows the amount of data already received in bytes.
0	25	Communication has started. The job is being processed.
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established

ERROR	STATUS (decimal)	Explanation
1	2	Function cannot be executed (protocol error)
1	4	Error in the receive area pointer RD_1 relating to the data length or data type. The block of data sent is longer than the receive area.
1	5	Reset request received, incomplete transfer.
1	8	Access error in associated "BSEND (Page 3621)": After the last valid data segment has been sent, ERROR = 1 and STATUS = 4 or ERROR = 1 and STATUS = 10 is reported.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	Value at R_ID parameter already exists in the connection specified at the ID parameter (R_ID value must be unique for the connection).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).

11.6.5.2 Open User Communication

TSEND_C: Send data via Ethernet

TSEND_C: Send data via Ethernet

Description

The "TSEND_C" instruction sets up and establishes a TCP or ISO-on-TCP communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU. The connection description specified at the CONNECT parameter is used to set up the communication connection.

The instruction is executed asynchronously and has the following functions:

- A communication connection is set up and established:
  The communication connection is set up and established with CONT=1. Once the connection is successfully established, the DONE parameter is set to "1" for one cycle. An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TSEND_C" again. For information on the number of possible communication connections, refer to the technical specifications for your CPU.
- Sending data via an existing communication connection:
  You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. Do not use a data area with the data type BOOL or Array of BOOL at the DATA parameter. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".

- The send job is executed when a rising edge is detected at the REQ parameter. With the LEN parameter, you specify the maximum number of bytes sent with a send job. When sending data (rising edge at the REQ parameter), the CONT parameter must have the value "1" in order to establish or maintain a connection. The data to be sent must not be edited until the send job is completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter is not confirmation that the data sent has already been read by the communication partner.
- Terminating the communication connection:
  The communication connection is terminated when the CONT parameter is set to "0" even if an ongoing data transmission is not complete yet. This does not apply if you are using an already configured connection for "TSEND_C".

When setting the COM_RST parameter to "1", the currently established connection or a current data transmission can be reset at any time. This terminates the existing communication connection and a new connection is established. If data is being transferred when it executes again, this can lead to a loss of data.

To enable "TSEND_C" again after the execution (DONE = 1), call the instruction once with REQ = 0.

Parameters

The following table shows the parameters of the "TSEND_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Starts the send job on a rising edge.
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection • 1: Establish and maintain the communication connection When sending data (rising edge at the REQ parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum number of bytes to be sent with the job. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Auto-Hotspot
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the send area containing the address and the length of the data to be sent (maximum length: 8192 bytes).
COM_RST	InOut	BOOL	I, Q, M, D, L	Restarts the instruction: • 0: Irrelevant • 1: Complete restart of the instruction causing an existing connection to be terminated and a new connection to be established.

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameters BUSY, DONE and ERROR

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Parameters ERROR and STATUS

ERROR	STATUS* (W#16#...)	Description
0	0000	Job completed without error.
0	0001	The connection establishment is complete.
0	0003	The connection termination is complete.
0	7000	No job processing active.
0	7001	• Start execution of the job • Establish connection • Wait for connection partner
0	7002	Data was sent.
0	7003	Connection is being terminated.
0	7004	Connection established and monitored, no job processing active.

ERROR	STATUS* (W#16#...)	Description
1	80A0	Group error for error codes 80A1 and 80A2.
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.
1	80A3	Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80AA	A connection is currently being established with the same connection ID by another block. Repeat the job with a new rising edge at the REQ parameter.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B3	Inconsistent parameter assignment: Group error for error codes 80A0 to 80A2, 80A4, 80B4 to 80B9.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): "local_tsap_id_len >= B#16#02", and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	8085	The LEN parameter is larger than the highest permitted value.
1	8086	The ID parameter within the CONNECT parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	The ID of the local device in the connection description does not correspond to the CPU.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST= 1.
1	80C6	Remote network error. Remote partner cannot be reached.

ERROR	STATUS* (W#16#...)	Description
1	8722	Parameter CONNECT: The source area is invalid. The area does not exist in the DB.
1	873A	Parameter CONNECT: Access to the connection description is not possible (for example, DB does not exist).
1	877F	Parameter CONNECT: Internal error.
1	8822	Parameter DATA: Invalid source area, the area does not exist in the DB.
1	8824	Parameter DATA: Area error in the VARIANT pointer.
1	8832	Parameter DATA: The DB number is too high.
1	883A	Parameter CONNECT: Access to specified connection data not possible (e.g. because the DB does not exist).
1	887F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	893A	Parameter DATA: Access to send area not possible (e.g. because the DB does not exist).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Note

Error messages of the instructions "TCON", "TSEND", "T_DIAG" and "TDISCON"

Internally, the "TSEND_C" instruction uses the instructions "TCON (Page 3662)", "TSEND (Page 3674)", "T_DIAG (Page 3694)", "T_RESET (Page 3693)" and "TDISCON (Page 3669)". The error messages of these instructions can also be output at the STATUS parameter. The meaning of the error codes is described in the corresponding instructions. In the event of identical error codes for internally used instructions with different meanings, the instance data block of "TSEND_C" can be used to determine which instruction output the error.

TSEND_C: Send data via Ethernet

Description

The "TSEND_C" instruction sets up and establishes a communications connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU.

The instruction is executed asynchronously and has the following functions:

- Setting up and establishing a communication connection
- Sending data via an existing communication connection
- Terminating or resetting the communication connection

Internally, the instruction "TSEND_C" uses the communication instructions TCON", "TSEND", "T_DIAG", "T_RESET" and "TDISCON".

Setting up and establishing a communication connection

The communication connection is set up and established with CONT=1. For information on the number of possible communication connections, refer to the technical specifications for your CPU. The connection description specified at the CONNECT parameter is used to set up the communication connection. The following connection types can be used:

- Programmed connections (structure of connection via "TCON"):
  - TCP / UDP: Connection description via the TCON_IP_v4 system data type
  - ISO-on-TCP: Connection description via the TCON_IP_RFC system data type
  - ISO: Connection description via the TCON_ISOnative system data type (for CP1543-1 only)
- Configured connections
  - Specify an existing connection in the TCON_Configured system data type.

An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TSEND_C" again.

Sending data via an existing communication connection

The send job is executed when a rising edge is detected at the REQ parameter. As described above, the communication connection is established first.

You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. Do not use a data area with the data type BOOL or Array of BOOL at the DATA parameter. With the LEN parameter, you specify the maximum number of bytes sent with a send job. If you use a send area with optimized access at the DATA parameter, the LEN parameter must have the value "0".

The data to be sent must not be edited until the send job is completed.

Terminating and resetting the communication connection

The communication connection is terminated when the CONT parameter is set to "0" even if an ongoing data transmission is not complete yet. This does not apply if you are using a configured connection for "TSEND_C".

The connection can be reset at any time by setting the parameter COM_RST to "1". This terminates the existing communication connection and a new connection is established. If data is being transferred at this time, this can lead to data loss.

Parameters

The following table shows the parameters of the "TSEND_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	Starts the send job on a rising edge.
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection. • 1: Establish and maintain the communication connection.
LEN	Input	UDINT	I, Q, M, D, L or constant	Optional parameter (hidden) Maximum number of bytes to be sent with the job. If you use a send area with optimized access at the DATA parameter, the value "0" must be used at the LEN parameter.
CONNECT	InOut	VARIANT	D	Pointer to the structure of the connection description: • Programmed connection: – For TCP or UDP, use the TCON_IP_v4 system data type For description, refer to:Auto-Hotspot – For ISO-on-TCP, use the TCON_IP_RFC system data type For description, refer to: Auto-Hotspot – For ISO, use the TCON_ISOnative system data type (only for CP1543-1) For description, refer to instruction "TCON (Page 3664)". • Configured connection: – For existing connections, use the TCON_Configured system data type For a description see "System data type for configured connections" below
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the send area containing the address and the length of the data to be sent.
ADDR	InOut	VARIANT	D	Optional parameter (hidden) Pointer to the address of the recipient.
COM_RST	InOut	BOOL	I, Q, M, D, L	Optional parameter (hidden) Resets the connection: • 0: Irrelevant • 1: The existing connection is reset. The COM_RST parameter is reset after evaluation by the "TSEND_C" instruction and should not, therefore, be interconnected statically.

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Send job not yet started or still in progress. • 1: Send job executed without error. This state is only displayed for one cycle. The output parameter DONE is set if an intermediate step was completed successfully during processing (connection establishment, sending, connection termination) and if the execution of "TSEND_C" was completed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Send job not yet started or already completed. • 1: Send job not yet completed. A new send job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred during connection establishment, data transmission or connection termination. The output parameter ERROR can be set due to an error in the "TSEND_C" instruction or the communication instructions used internally.
STATUS	Output	WORD	I, Q, M, D, L	Status of instruction (see the "ERROR and STATUS" parameters" description).

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

REQ, CONT and COM_RST parameters

The parameter CONT controls the connection establishment of the "TSEND_C" instruction regardless of the REQ parameter. The behavior of the CONT parameter partially depends on whether a programmed or a configured connection is used:

- With CONT = "0": No data is sent (regardless of whether a programmed or a configured connection is used).
- When changing CONT = "0" to "1":
  - With a programmed connection, it is established with "TCON".
  - With a configured connection, it is checked with "T_DIAG".
- With CONT = "1":
  - As long as no data is sent (REQ="0"), the connection is checked with "T_DIAG".
  - If the internally used communication instructions signal that no connection end point exists, the connection is automatically reestablished with "TCON".
- When changing CONT = "1" to "0":
  - With a programmed connection, it is terminated with "TDISCON".
  - With a configured connection, it is reset with "T_RESET".

The parameter COM_RST resets the connection when changing from "0" to "1":

- If a connection is established, it is reset with "T_RESET" (regardless of whether a programmed or configured connection is used).
- If no connection is established, the setting of the parameter has no effect.

The REQ and COM_RST parameters only have an effect if CONT has been set to "1". The following table shows the relationship between the REQ, CONT and COM_RST parameters:

REQ	CONT	COM_RST	Status of the instruction	Description
irrelevant	0	irrelevant	Not yet executed	No job active (STATUS = 7000).
irrelevant	0	irrelevant	Initialization	Connection is being terminated. The instruction is being reset.
irrelevant	0 > 1	irrelevant	Connection establishment	Connection is being established. Data is not being transferred yet.
0	1	0	Connection established	The connection is established and is monitored with the instruction "T_DIAG".
irrelevant	1	0 > 1	Connection established	The connection is interrupted by "T_RESET" briefly and reset.
0 > 1	1	0	Connection established	Instruction starts sending.
irrelevant	1	0 > 1	Data is being sent	Data transmission is interrupted. The connection is being reset.

System data type for configured connections

For configured connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured:

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to the connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	-	Connection type Select 254 (decimal) for a configured connection.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a send job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

DONE	BUSY	ERROR	Description
0	0	0	The instruction has not been executed yet (no rising edge at REQ parameter).
0	1	0	The instruction is being executed and calls the internally used communication instructions.
1	0	0	The send job was completed successfully. "0000" is output at the STATUS parameter. DONE = "1" is only displayed for one cycle.
0	0	1	The execution of the instruction or an intermediate step during processing was terminated with an error. If there is a subsequent error due to an internally used communication instruction, the error that occurred first during processing is displayed. This state is only displayed for one cycle.

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Send job was executed without error.
0	0001	Communication connection established.
0	0003	Communication connection closed.
0	7000	No active send job execution; no communication connection established.
0	7001	Initial call for establishing a connection.
0	7002	Second call for establishing a connection
0	7003	Communication connection is being terminated.
0	7004	Communication connection has been established and is being monitored. No send job execution active.
0	7005	Data transmission in progress.
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A3	The nested "T_DIAG" instruction has reported that the connection has closed.
1	80A4	IP address of the remote endpoint of the connection is invalid or it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80AA	A connection is currently being established with the same connection ID by another block. Repeat the job with a new rising edge at the REQ parameter.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.

ERROR	STATUS* (W#16#...)	Description
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	8085	The LEN parameter is larger than the highest permitted value.
1	8086	The ID parameter within the CONNECT parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	InterfaceID is invalid. It is either zero or it does not point to a local CPU interface or a CP.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON (Page 3669)" instruction. • The connection used is being terminated by a call with COM_RST = 1.
1	80C6	Remote network error. Remote partner cannot be reached.
1	8722	Parameter CONNECT: The source area is invalid. The area does not exist in the DB.
1	873A	Parameter CONNECT: Access to the connection description is not possible (for example, because the DB is not available).
1	877F	Parameter CONNECT: Internal error.
1	8822	Parameter DATA: Invalid source area, the area does not exist in the DB.
1	8824	Parameter DATA: Area error in the VARIANT pointer.
1	8832	Parameter DATA: The DB number is too high.
1	883A	Parameter CONNECT: Access to specified connection data not possible (e.g. because the DB does not exist).
1	887F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	893A	Parameter DATA: Access to send area not possible (e.g. because the DB does not exist).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on switching display formats, refer to "See also".		

Note

Error messages of the instructions "TCON", "TSEND", "T_DIAG", "T_RESET" and "TDISCON"

Internally, the "TSEND_C" instruction uses the instructions "TCON (Page 3664)", "TSEND (Page 3674)", "T_DIAG (Page 3694)", "T_RESET (Page 3693)" and "TDISCON (Page 3669)". The error messages of these instructions can also be output at the STATUS parameter. The meaning of the error codes is described in the corresponding instructions. In the event of identical error codes for internally used instructions with different meanings, the instance data block of "TSEND_C" can be used to determine which instruction output the error.

TRCV_C: Receive data via Ethernet

TRCV_C: Receive data via Ethernet

Description

The "TRCV_C" instruction is executed asynchronously and has the following functions:

1. **Setting up and establishing a communication connection:**
   "TRCV_C" sets up and establishes a TCP or ISO-on-TCP communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU.
   The connection description specified at the CONNECT parameter is used to set up the communication connection. To establish a connection, the CONT parameter must be set to the value "1". Once the connection is successfully established, the DONE parameter is set to "1".
   An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TRCV_C" again.
   For information on the number of possible communication connections, refer to the technical specifications for your CPU.
2. **Receiving data via an existing communication connection:**
   If the EN_R parameter is set to the value "1", receipt of data is enabled. When receiving data (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
   The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0) in accordance with the protocol variant used. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
   After data has been received successfully, the signal state at the DONE parameter is "1". If errors occur in the data transfer, the DONE parameter is set to "0".
3. **Terminating the communication connection:**
   The communication connection is terminated immediately when the CONT parameter is set to "0".

TRCV_C is executed again when the COM_RST parameter is set. This terminates the existing communication connection and a new connection is established. If data is being received when it executes again, this can lead to a loss of data.

Receive modes of TRCV_C

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type parameter of the connection description	Parameter LEN
TCP (Ad-hoc mode)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	0
TCP (Receipt of data with specified length)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192
ISO on TCP (Message-oriented data transfer)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	• 1 to 1452, if a CP is used. • 1 to 8192, if no CP is used.

TCP (Ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV" instruction.

You set ad-hoc mode by assigning the value "0" to the LEN parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). The data length actually received is output at the RCVD_LEN parameter.

TCP (Receipt of data with specified length)

You use the value of the LEN parameter to specify the length for the data receipt. The data receipt is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

ISO on TCP (Message-oriented data transfer)

Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. When using ISO on TCP, "TRCV_C" signals data receipt as soon as the message block has been completely received. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV_C" signals an error. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

Parameters

The following table shows the parameters of the "TRCV_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Receive enable
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection • 1: Establish and maintain the communication connection When receiving data (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the data to be received (maximum value: 8192 bytes). If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Auto-Hotspot
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area
COM_RST	InOut	BOOL	I, Q, M, D, L	Restarts the instruction: • 0: Irrelevant • 1: Complete restart of the instruction causing an existing connection to be terminated
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameters BUSY, DONE and ERROR

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TRCV_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Parameters ERROR and STATUS

ERROR	STATUS* (W#16#...)	Description
0	0000	Job completed without error.
0	0001	The connection establishment is complete.
0	0003	The connection termination is complete.
0	7000	No job processing active.
0	7001	• Start execution of the job • Establish connection • Wait for connection partner
0	7002	Data is being received.
0	7003	Connection is being terminated.
0	7004	• Connection established and monitored • No job processing active
0	7006	Data is currently being received.
1	8085	• The LEN parameter is larger than the highest permitted value. • The value at the LEN or DATA parameter was changed after the first call.
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	The ID of the local device (local_device_id) in the connection description does not correspond to the CPU.
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.

ERROR	STATUS* (W#16#...)	Description
1	80A3	• Attempt being made to re-establish an existing connection. • Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B3	Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): "local_tsap_id_len >= B#16#02", and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST= 1.
1	80C6	The remote partner cannot be reached (network error).
1	8722	Error in the CONNECT parameter: Invalid source area (area not declared in data block).
1	873A	Error in the CONNECT parameter: Access to connection description is not possible (no access to data block).
1	877F	Error in the CONNECT parameter: Internal error
1	8922	Parameter DATA: Invalid target area; the area does not exist in the DB.
1	8924	Parameter DATA: Area error in the VARIANT pointer.
1	8932	Parameter DATA: The DB number is too high.
1	893A	Parameter CONNECT: Access to specified connection data not possible (e.g. because the DB does not exist).
1	897F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	8A3A	Parameter DATA: No access to the data area, for example because the data block does not exist.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Note

Error messages of the instructions "TCON", "TRCV" and "TDISCON"

Internally, the TRV_C instruction uses the "TCON (Page 3662)", "TRCV (Page 3677)" and "TDISCON (Page 3669)" instructions. The error messages of these instructions are contained in the respective descriptions.

TRCV_C: Receive data via Ethernet

Description

The "TRCV_C" instruction is executed asynchronously and implements the following functions in sequence:

- Setting up and establishing a communication connection
- Receiving data via an existing communication connection
- Terminating or resetting the communication connection

Internally, the instruction "TRCV_C" uses the communication instructions TCON", "TRCV", "T_DIAG", "T_RESET" and "TDISCON".

Setting up and establishing a communication connection

The communication connection is set up and established with CONT=1. For information on the number of possible communication connections, refer to the technical specifications for your CPU. The connection description specified at the CONNECT parameter is used to set up the communication connection. The following connection types can be used:

- Programmed connections (structure of connection via "TCON"):
  - TCP / UDP: Connection description via the TCON_IP_v4 system data type
  - ISO-on-TCP: Connection description via the TCON_IP_RFC system data type
  - ISO: Connection description via the TCON_ISOnative system data type (for CP1543-1 only)
- Configured connections
  - Specify an existing connection in the TCON_Configured system data type.

An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TRCV_C" again.

Receiving data via an existing communication connection

Receipt of data is enabled when the EN_R parameter is set to the value "1". The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0), depending on the protocol variant being used. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".

Receive modes of TRCV_C:

- **TCP (Ad-hoc mode)**
  The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV_C" instruction.
  You set ad-hoc mode by assigning the value "1" to the ADHOC parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). The data length actually received is output at the RCVD_LEN parameter.
- **TCP (Receipt of data with specified length)**
  Assign the value "0" to the ADHOC parameter for receipt of data with specified length. If ad-hoc mode is disabled, the data reception is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.
- **ISO on TCP (Message-oriented data transfer)**
  Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV_C" signals an error. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type parameter of the connection description	LEN parameter
TCP (Ad-hoc mode)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	1 up to maximum length (depending on the CPU)
TCP (Receipt of data with specified length)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192
ISO on TCP (Message-oriented data transfer)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	1 to 8192

Terminating the communication connection

The communication connection is terminated when the CONT parameter is set to "0" even if an ongoing data transmission is not complete yet. This does not apply if you are using a configured connection, however.

The connection can be reset at any time by setting the parameter COM_RST to "1". This terminates the existing communication connection and a new connection is established. If data is being transferred at this time, this can lead to data loss.

Parameters

The following table shows the parameters of the "TRCV_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L, T, C or constant	Receive enable
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection. • 1: Establish communication connection and maintain after receipt of data.
LEN	Input	UDINT	I, Q, M, D, L or constant	Maximum length of the data to be received. If you use a receive area with optimized access at the DATA parameter, the value "0" must be used at the LEN parameter.
ADHOC	Input	BOOL	I, Q, M, D, L or constant	Optional parameter (hidden) Use ad-hoc mode for the TCP protocol variant.
CONNECT	InOut	VARIANT	D	Pointer to the connection description • Programmed connection: – For TCP or UDP, use the structure TCON_IP_v4 For description, refer to:Auto-Hotspot – For ISO-on-TCP, use the structure TCON_IP_RFC For description, refer to: Auto-Hotspot – For ISO, use the structure TCON_ISOnative (only for CP1543-1) For description, refer to instruction "TCON (Page 3664)": "Structure of the connection description according to TCON_ISOnative" • Configured connection: – For existing connections, use the TCON_Configured system data type. For a description, see "System data type for configured connections" below.
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area.
ADDR	InOut	VARIANT	D	Optional parameter (hidden) Pointer to the address of the sender with connection type UDP.

Parameter	Declaration	Data type	Memory area	Description
COM_RST	InOut	BOOL	I, Q, M, D, L	Optional parameter (hidden) Resets the connection: • 0: Irrelevant • 1: The existing connection is reset. The COM_RST parameter is reset after evaluation by the "TRCV_C" instruction and should not, therefore, be interconnected statically.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Receipt not yet started or still in progress. • 1: Receipt executed without error. This state is only displayed for one cycle. The output parameter DONE is set if an intermediate step was completed successfully during processing (connection establishment, receipt, connection termination) and if the execution of "TRCV_C" was completed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Receipt not yet started or already completed. • 1: Receipt not yet completed. A new send job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred during connection establishment, data reception or connection termination. The output parameter ERROR can be set due to an error in the "TRCV_C" instruction or the communication instructions used internally.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

EN_R, CONT and COM_RST parameters

The parameter CONT controls the connection establishment of the "TRCV_C" instruction regardless of the EN_R parameter. The behavior of the CONT parameter partially depends on whether a programmed or a configured connection is used:

- With CONT = "0": No data is received (regardless of whether a programmed or a configured connection is used).
- When changing CONT = "0" to "1":
  - With a programmed connection, it is established with "TCON".
  - With a configured connection, it is checked with "T_DIAG".

- With CONT = "1":
  - As long as no data is received (EN_R="0"), the connection is checked with "T_DIAG".
  - If the internally used communication instructions signal that no connection end point exists, the connection is automatically reestablished with "TCON".
- When changing CONT = "1" to "0":
  - With a programmed connection, it is terminated with "TDISCON".
  - With a configured connection, it is reset with "T_RESET".

The parameter COM_RST resets the connection when changing from "0" to "1":

- If a connection is established, it is reset with "T_RESET" (regardless of whether a programmed or configured connection is used).
- If no connection is established, the setting of the parameter has no effect.

The EN_R and COM_RST parameters only have an effect if CONT has been set to "1". The following table shows the relationship between the EN_R, CONT and COM_RST parameters:

EN_R	CONT	COM_RST	Status of the instruction	Description
irrelevant	0	irrelevant	Not yet executed	No job active (STATUS = 7000).
irrelevant	0	irrelevant	Initialization	Connection is being terminated. The instruction is being reset.
irrelevant	0 > 1	irrelevant	Connection establishment	Connection is being established. Data is not being transferred yet.
0	1	0	Connection established	The connection is established and is monitored with the instruction "T_DIAG".
irrelevant	1	0 > 1	Connection established	The connection is interrupted by "T_RESET" briefly and reset.
0 > 1	1	0	Connection established	Instruction starts receiving.
irrelevant	1	0 > 1	Data is being received	Data transmission is interrupted. The connection is being reset.

System data type for configured connections

For configured connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured:

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to the connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	-	Connection type Select 254 (decimal) for a configured connection.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TRCV_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

DONE	BUSY	ERROR	Description
0	0	0	The instruction has not been executed yet (no rising edge at EN_R parameter).
0	1	0	The instruction is being executed and calls the internally used communication instructions.
1	0	0	The receipt has been completed successfully. "0000" is output at the STATUS parameter. DONE = "1" is only displayed for one cycle.
0	0	1	The execution of the instruction or an intermediate step during processing was terminated with an error. If there is a subsequent error due to an internally used communication instruction, the error that occurred first during processing is displayed. This state is only displayed for one cycle.

ERROR and STATUS parameters

ERROR	STATUS (W#16#...)	Description
0	0000	Receive job executed without error.
0	0001	Communication connection established.
0	0003	Communication connection closed.
0	7000	No job processing active.
0	7001	Initial call for establishing a connection.
0	7002	Second call for establishing a connection.
0	7003	Communication connection is being terminated.
0	7004	Communication connection has been established and is being monitored. No receive job processing active.
0	7006	Data is currently being received.
1	8085	• The LEN parameter is larger than the highest permitted value. • The value at the LEN or DATA parameter was changed after the first call.
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	InterfaceID is invalid. It is either zero or it does not point to a local CPU interface or a CP.

ERROR	STATUS (W#16#...)	Description
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A3	The nested "T_DIAG" instruction has reported that the connection has closed.
1	80A4	IP address of the remote endpoint of the connection is invalid or it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80AA	A connection is currently being established with the same connection ID by another block. Repeat the job with a new rising edge at the REQ parameter.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): "local_tsap_id_len >= B#16#02", and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST = 1.
1	80C6	The remote partner cannot be reached (network error).
1	8722	Error in the CONNECT parameter: Invalid source area (area not declared in data block).
1	873A	Error in the CONNECT parameter: Access to connection description is not possible (no access to data block).
1	877F	Error in the CONNECT parameter: Internal error
1	8922	Parameter DATA: Invalid target area; the area does not exist in the DB.
1	8924	Parameter DATA: Area error in the VARIANT pointer.
1	8932	Parameter DATA: The DB number is too high.
1	893A	Parameter CONNECT: Access to specified connection data not possible (e.g. because the DB does not exist).
1	897F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	8A3A	Parameter DATA: No access to the data area, for example because the data block does not exist.

Note

Error messages of the instructions "TCON", "TRCV" and "TDISCON"

Internally, the TRV_C instruction uses the "TCON (Page 3664)", "TRCV (Page 3677)" and "TDISCON (Page 3669)" instructions. The error messages of these instructions are contained in the respective descriptions.

See also

TSEND_C: Send data via Ethernet (Page 3629)

TMAIL_C: Transfer email

Description of TMAIL_C

Description

You can use the "TMAIL_C" instruction to send an e-mail via the Ethernet interface of the S7-1500 CPU or S7-1200 > V4.0, a communication module (CM), or a communication processor (CP).

The instruction can only be used once the hardware has been configured and if the network infrastructure allows for a communication connection to the mail server.

You define the content of the e-mail, and the connection data, using the following parameters:

- You define the recipient addresses with the parameters TO_S and CC.
- You define the content of the e-mail with the parameters SUBJECT and TEXT.
- You can define an attachment using VARIANT pointers at the ATTACHMENT and ATTACHMENT_NAME parameters.
- The connection data is defined, and addressing and authentication for the mail server executed, using the system data type TMail_V4, TMail_V6 or TMail_FQDN at the MAIL_ADDR_PARAM parameter.
  - If you are using the interface of the S7-1500 CPU, the system data type TMail_V4 must be used. In this case, the e-mail can only be sent via SMTP.
  - Any of the system data types can be used if you are using the interface of a CM/CP. The e-mail can then also be sent via SMTPS.
- You start the sending of an e-mail with an edge change from "0" to "1" for the REQ parameter.
- The job status is indicated by the output parameters "BUSY", "DONE", "ERROR" and "STATUS".

You cannot send an SMS directly with the "TMAIL_C" instruction. Whether or not the e-mail can be forwarded by the mail server as an SMS depends on your telecommunications provider.

Note

Number of e-mails to be sent

You can send more than one e-mail simultaneously using a PLC. When a CP 1243-8 or CP 1543-1 is used, you can only send one e-mail for each CP. When two CPs are used, parallel sending of two e-mails is possible.

Operation of the instruction

The "TMAIL_C" instruction works asynchronously, which means its execution extends over multiple calls. You must specify an instance when you call the instruction "TMAIL_C".

In the following cases, the connection to the mail server will be lost:

- If the CPU switches to STOP while "TMAIL_C" is active.
- If communication problems occur at the Industrial Ethernet bus.

In this case, the transfer of the e-mail will be interrupted and it will not reach its recipient. The connection is also canceled once the instruction has been successfully executed and the e-mail sent.

Notice

Changing user programs

You can change the parts of your user program that directly affect calls of "TMAIL_C" only:

- The CPU is in "STOP" mode.
- No e-mail is being sent (REQ = 0 and BUSY = 0).

This relates, in particular, to deleting and replacing program blocks that contain "TMAIL_C" calls or calls for the instance of "TMAIL_C".

Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status with the TCP/IP communication functions via Industrial Ethernet.

A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The TO_S, CC, SUBJECT, TEXT, ATTACHMENT and MAIL_ADDR_PARAM parameters are applied by the "TMAIL_C" instruction while it is running, which means that they may only be changed after the job has been completed (BUSY = 0).

SMTP authentication

Authentication refers here to a procedure for verifying identity, for example, with a password query.

If you are using the S7-1500 CPU interface, the instruction "TMAIL_C" supports the SMTP authentication procedure AUTH-LOGIN which is required by most mail servers. For information about the authentication procedure of your mail server, please refer to your mail server manual or the website of your Internet service provider.

- Before you can use the AUTH-LOGIN authentication procedure, the "TMAIL_C" instruction requires the user name with which it is to log on to the mail server. This user name corresponds to the user name with which you set up a mail account on your mail server. It is transferred via the UserName parameter to the structure at parameter MAIL_ADDR_PARAM.
  If no user name is specified at the MAIL_ADDR_PARAM parameter, the AUTH-LOGIN authentication procedure is not used. The e-mail is then sent without authentication.
- To log on, the "TMAIL_C" instruction also requires the associated password. This password corresponds to the password you specified when you set up your mail account. It is transferred via the PassWord parameter to the structure at parameter MAIL_ADDR_PARAM.

Parameters

The following table shows the parameters of the "TMAIL_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	Control parameter REQUEST: Activates the sending of an e-mail upon a rising edge.
TO_S (Page 3652)	Input	STRING	D, L or constant	Recipient addresses STRING with a maximum length of 240 characters (bytes). For the e-mail address format, see the example in the parameter description.
CC (Page 3652)	Input	STRING	D, L or constant	CC recipient addresses (optional) STRING with a maximum length of 240 characters (bytes). Same e-mail address format as for the TO_S parameter. If an empty string is assigned here, the e-mail is not sent to a CC recipient.
SUBJECT	Input	STRING	D, L or constant	Subject of the e-mail STRING with a maximum length of 240 characters (bytes).
TEXT	Input	STRING	D, L or constant	Text of the e-mail (optional) STRING with a maximum length of 240 characters (bytes). If an empty string is assigned at this parameter, the e-mail is sent without text.
ATTACHMENT	Input	VARIANT	D	E-mail attachment (optional) Reference to a byte/word/double word field (ArrayOfByte, ArrayOfWord or ArrayOfDWord) with a maximum length of 64 Kb. If no value is assigned, the e-mail is sent without an attachment.

Parameter	Declaration	Data type	Memory area	Description
ATTACHMENT_NAME	Input	STRING	D, L or constant	E-mail attachment name (optional) Reference to a character string with a maximum length of 50 characters (bytes) to define the file name of the attachment. If an empty string is assigned at this parameter, the e-mail attachment will be sent with the file name "attachment.bin".
MAIL_ADDR_PARAM (Page 3652)	Input	VARIANT	D	Connection parameter and address of the e-mail server To define the connection parameters, use the structure TMail_V4, TMail_V6 or TMail_FQDN (see parameter description).
DONE (Page 3655)	Output	BOOL	I, Q, M, D, L	Status parameter • DONE = 0: Job not yet started or still executing. • DONE = 1: Job was executed without errors.
BUSY (Page 3655)	Output	BOOL	I, Q, M, D, L	Status parameter • BUSY=0: The processing of "TMAIL_C" was stopped. • BUSY = 1: E-mail transmission is not yet complete.
ERROR (Page 3655)	Output	BOOL	I, Q, M, D, L	Status parameter • ERROR = 0: No error has occurred. • ERROR = 1: An error occurred during processing. STATUS supplies detailed information on the type of error.
STATUS (Page 3656)	Output	WORD	I, Q, M, D, L	Status parameter Return value or error information of the "TMAIL_C" instruction (see parameter description).

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Note

Optional parameters

The optional parameters CC, TEXT, and ATTACHMENT are only sent with the e-mail if the corresponding parameters contain a string of length > 0.

Example

You can find an example for sending e-mails with the TMAIL_C instruction under the following link: Example: Sending an e-mail with TMAIL_C (Page 3659)

TO_S and CC parameters

Description

The TO_S and CC parameters are strings, for example, with the following content:

- <wenna@mydomain.com>, <ruby@mydomain.com>
- <admin@mydomain.com>, <judy@mydomain.com>

Note the following rules when entering the parameters:

- A space and an opening pointed bracket "<" must be entered before each address.
- A closing pointed bracket ">" must be entered after each address.
- A comma must be entered between the addresses in TO and CC.

For runtime and memory space reasons, the "TMAIL_C" instruction does not perform a syntax check of parameter TO_S or CC.

MAIL_ADDR_PARAM parameter

Description

At the MAIL_ADDR_PARAM parameter, you define the connection for sending the e-mail in the structure TMail_V4, TMail_V6 or Tmail_FQDN, and save the e-mail server address and login details.

The structure you use at the MAIL_ADDR_PARAM parameter depends on the format in which the e-mail server is to be addressed:

- TMail_V4: Addressing by IP address to IPv4.
- TMail_V6: Addressing by IP address to IPv6.
- TMail_FQDN: Addressing by fully qualified domain name (FQDN).

Which structure you can use depends on the interface addressed at the InterfaceId parameter:

- If you want to use the "TMAIL_C" instruction with the internal interface, the TMail_V4 structure must be used at the MAIL_ADDR_PARAM parameter.
- If you are using a communication processor (CP) or communication module (CM), you can use all three addressing options (IPv4, IPv6 and FQDN).

Table 11-94 **TMail_V4: Addressing the mail server by IP address (IPv4)**

Parameter	Data type	Description
TMail_V4	Struct	
InterfaceId	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#20 as the connection type for IPv4.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	• =0: SMTP used (**S**imple **M**ail **T**ransfer **P**rotocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1500 CPU. • ≠0: SMTPS used to secure the connection before it is established (with CPs/CMs). You use the CertIndex parameter to specify the certificate to be used (see "Project navigation" > "Global security settings" > "Certificate manager").
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	IP_V4	IP address of the mail server. IPv4 in the following format: XXX.XXX.XXX.XXX (decimal). Example: 192.142.131.237.
UserName	STRING[254]	Mail server login name
PassWord	STRING[254]	Mail server password
From	EMAIL_ADDR	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@".
FullQualifiedDomainName	STRING[254]	Fully Qualified Domain Name (FQDN for short) of the mail server. Example: "mymailserver.com".

Table 11-95 **TMail_V6: Addressing the mail server by IP address (IPv6)**

Parameter	Data type	Description
TMail_V6	Struct	
InterfaceId	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#21 as the connection type for IPv6.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	• =0: SMTP used (**S**imple **M**ail **T**ransfer **P**rotocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1500 CPU. • ≠0: SMTPS used to secure the connection before it is established (with CPs/CMs). You use the CertIndex parameter to specify the certificate to be used (see "Project navigation" > "Global security settings" > "Certificate manager").
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	IP_V6	IP address of the mail server (IPv6) in the following format: XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX (hexadecimal). The address is divided into 8 blocks of 2 bytes each (16 bytes in total). Example: 2001:db8:1f11:08d3:290:27ff:0370:2093
UserName	STRING[254]	Mail server login name
PassWord	STRING[254]	Mail server password
From	EMAIL_ADDR	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@".
FullQualifiedDomainName	STRING[254]	Fully Qualified Domain Name (FQDN for short) of the mail server. Example: "mymailserver.com".

Table 11-96 **TMail_FQDN: Addressing the mail server by FQDN**

Parameter	Data type	Description
TMail_V6	Struct	
TMail_FQDN	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#22 as the connection type for FQDN.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	• =0: SMTP used (**S**imple **M**ail **T**ransfer **P**rotocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1500 CPU. • ≠0: SMTPS used to secure the connection before it is established (with CPs/CMs). You use the CertIndex parameter to specify the certificate to be used (see "Project navigation" > "Global security settings" > "Certificate manager").
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	STRING[254]	FQDN (**F**ully **Q**ualified **D**omain **N**ame) of the mail server. The mail server is addressed using the fully qualified domain name. Example: "www.mymailserver.com.".
UserName	STRING[254]	Mail server login name
PassWord	STRING[254]	Mail server password
From	Struct	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@".
FullQualifiedDomainName	STRING[254]	Fully Qualified Domain Name (FQDN for short) of the mail server. Example: "mymailserver.com".

DONE, BUSY and ERROR parameters

Description

The output parameters DONE, BUSY and ERROR are each displayed for only one cycle if the status of the BUSY output parameter changes from "1" to "0".

The following table shows the relationship between DONE, BUSY, and ERROR. Using this table, you can determine the current status of the instruction "TMAIL_C and when the sending of the e-mail is complete.

DONE	BUSY	ER-ROR	Description
0	1	0	The job is being processed.
1	0	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS (Page 3656) parameter.
0	0	0	The "TMAIL_C" instruction was not assigned a (new) job.

STATUS parameter

Description

The following table shows the return values of "TMAIL_C" at the STATUS parameter:

Return value STATUS* (W#16#...):	Explanation	Notes
0000	The processing of "TMAIL_C" was completed without errors.	Error-free completion of "TMAIL_C" does not mean that the e-mail sent will necessarily arrive. Incorrectly entering the recipient addresses does not generate a status error of the "TMAIL_C" instruction. In this case, there is no guarantee that the e-mail will be sent to other recipients, even if these were entered correctly.
7001	"TMAIL_C" is active (BUSY = 1).	First call: Job triggered.
7002	"TMAIL_C" is active (BUSY = 1).	Intermediate call: Job already active.
8xxx	The processing of "TMAIL_C" was completed with an error code of the communication instructions called internally.	For detailed information, please refer to the STATUS parameter descriptions for the "TCON (Page 3664)", "TDISCON (Page 3669)", "TSEND (Page 3674)" and "TRCV (Page 3677)" communication instructions.
8010	Error during connection establishment	You can find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the "TCON (Page 3664)" instruction.
8011	Error sending the data	You can find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the "TSEND (Page 3674)" instruction.

Return value STATUS* (W#16#...):	Explanation	Notes
8012	Error receiving the data	You can find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the "TRCV (Page 3677)" instruction.
8013	Error during connection establishment	You can find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the TCON (Page 3664)" and "TDISCON (Page 3669)" instructions.
8014	Establishment of a connection is not possible.	You may have entered an incorrect mail server IP address (MailServerAddress (Page 3652)) or too short a time span (WatchDogTime (Page 3652)) for connection establishment. It is also possible that the CPU has no connection to the network or that the CPU configuration is incorrect.
8015	Incorrect data type for MAIL_ADDR_PARAM	The only valid data types are the system data types (structures) TMail_V4, TMail_V6 and TMail_FQDN.
8016	Incorrect data type for the ATTACHMENT parameter	The only valid data types are ArrayOfByte, ArrayOfWord and ArrayOfDWord.
8017	Incorrect data length for the ATTACHMENT parameter	Data length must be <= 65534 bytes.
8401	No channels available. Possible cause: There is already an e-mail connection via the CP. A second connection cannot be established in parallel.	Specific error for CP 1543
8403	A TCP/IP connection to the mail server was not possible.	Specific error for CP 1543
8405	Server has denied login request.	Specific error for CP 1543
8406	SMTP client has detected an internal SSL error or a problem with the certificate structure.	Specific error for CP 1543
8407	Request to use SSL denied.	Specific error for CP 1543
8408	Client unable to identify socket for establishing a TCP/IP connection to the mail server.	Specific error for CP 1543
8409	Writing via the connection not possible. Possible cause: The communication partner has reset the connection or the connection has been lost.	Specific error for CP 1543
8410	Reading via the connection not possible. Possible cause: The communication partner has reset the connection or the connection has been lost.	Specific error for CP 1543

Return value STATUS* (W#16#...):	Explanation	Notes
8411	E-mail could not be sent. Cause: Insufficient memory space to execute the send operation.	Specific error for CP 1543
8412	DNS server configured was unable to resolve the specified domain name.	Specific error for CP 1543
8413	An internal error in the DNS subsystem prevented domain name resolution.	Specific error for CP 1543
8414	Empty character string entered as domain name.	Specific error for CP 1543
8415	An internal error has occurred in the cURL module. Execution was stopped.	Specific error for CP 1543
8416	An internal error has occurred in the SMTP module. Execution was stopped.	Specific error for CP 1543
8417	Request to SMTP on a channel already in use or invalid channel ID. Execution was stopped.	Specific error for CP 1543
8418	E-mail send job aborted. Possible causes: Execution time exceeded (WatchDogTime parameter) or CP transition from start to stop.	Specific error for CP 1543
8419	Channel interrupted and cannot be used before the connection is closed.	Specific error for CP 1543
8420	Server certificate string could not be verified with CP root certificate.	Specific error for CP 1543
8421	Internal error occurred. Execution was stopped.	Specific error for CP 1543
82xx, 84xx, or 85xx	The error message originates from the mail server and corresponds, except for the "8", to the error number of the SMTP protocol. The following lines list several error codes that can occur.	You can find more detailed information on the SMTP error code and other error codes in the SMTP protocol on the Internet or in the error documentation of the mail server. You can also view the most recent error message from the mail server in your instance DB in the BUFFER1 parameter. You can find the last data sent by the "TMAIL_C" instruction under DATEN in the instance DB.
8450	Action not executed: Mailbox not available/cannot be reached	Try again later.
8451	Action aborted: Local processing error	Try again later.
8500	Syntax error: Error not recognized. This also includes the error when a command string is too long. This can also occur when the e-mail server does not support the LOGIN authentication procedure.	Check the parameters of "TMAIL_C". Try to send an e-mail without authentication. To do this, replace the content of the UserName parameter with an empty string. If no user name is specified, the LOGIN authentication procedure is not used.

Return value STATUS* (W#16#...):	Explanation	Notes
8501	Syntax error: Incorrect input at a parameter	Possible cause: Incorrect address at the TO_S or CC parameter (see also: TO_S and CC parameters (Page 3652)).
8502	Command unknown or not implemented	Check your entries, in particular the FROM parameter. It may be incomplete and you may have forgotten the "@" or "." (see also: TO_S and CC parameters (Page 3652)).
8504	Command parameters not implemented	Check your entries, in particular the FROM parameter. It may be incomplete and you may have forgotten the "@" (see also: TO_S and CC parameters (Page 3652)).
8535	SMTP authentication incomplete	You have possibly entered an incorrect user name or incorrect password.
8550	Mail server cannot be reached. You have no access rights.	You may have entered an incorrect user name or password, or the mail server may not support your login. Another cause of error could be a mistake in the domain name after the "@" or a missing "." at the TO_S, CC and FROM parameters (see also: TO_S and CC parameters (Page 3652)).
8552	Action aborted: Assigned memory size has been exceeded	Try again later.
8554	Transfer failed	Try again later.
8555	Error due to incorrect entry of the e-mail address.	If multiple addresses were entered, these need to be separated by a comma.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Example: Sending an e-mail with TMAIL_C

Introduction

The following call example shows you how the TMAIL_C instruction works.

Requirements

Hardware must be configured beforehand for sending e-mail and the infrastructure of the network must allow a communication connection to the mail server. In addition, you need to create two structures to supply the input parameters of TMAIL_C: One for the e-mail attachment, the other for the connection information and address data.

E-mail attachment

The e-mail should be sent with an attachment. Create an Array of Byte with the name "Data" in a global data block as a source for the attachment. The array will be interconnected with the ATTACHMENT input parameter at a later time.

MyDBMailAttachment

	Name	Datentyp	Startwert
1	Static		
2	Data	Array[0..99] of Byte	
3	<Hinzufügen>		

Connection information and address data

You enter the connection information of the CPU and address data of the mail server in the TMail_V4 system data type:

- Create a "Par1" tag with the "TMail_V4" data type in a global data block.
- In the following structure, enter the parameters of your CPU based on the configuration and the connection data of your mail server. For more information, refer to the description of the MAIL_ADDR_PARAM (Page 3652) input parameter.

The "Par1" tag will be interconnected with the MAIL_ADDR_PARAM input parameter later.

MyDBSendMail

	Name	Datentyp	Startwert
1	Static		
2	Par1	TMail_V4	
3	InterfaceId	HW_ANY	64
4	ID	CONN_OUC	100
5	ConnectionType	Byte	16#20
6	ActiveEstablished	Bool	false
7	CertIndex	Byte	16#0
8	WatchDogTime	Time	T#5ms
9	MailServerAddress	IP_V4	
10	ADDR	Array[1..4] of Byte	
11	ADDR[1]	Byte	192
12	ADDR[2]	Byte	168
13	ADDR[3]	Byte	100
14	ADDR[4]	Byte	10
15	UserName	String[254]	'myusername'
16	PassWord	String[254]	'mypassword'
17	From	EMAIL_ADDR	
18	LocalPartPlusAt...	String[64]	'station1@'
19	FullQualifiedD...	String[254]	'mycpu.com'

Calling the instruction

Create a new function block in the SCL language with the following interfaces:

- InOut
  - Name: SendEMail
  - Data type: Bool
- Static
  - Name: STATUS
  - Data type: Word

Copy the following source code with the call for TMAIL_C to the programming window:

Call TMAIL_C

```
BEGIN
    IF #SendEMail = true THEN
     "TMAIL_C_DB"(REQ := NOT "TMAIL_C_DB".BUSY,
                  TEXT := 'The cpu switched to run',
                  ATTACHMENT := "MyDBMailAttachment".Data,
                  MAIL_ADDR_PARAM := "MyDBSendMail".Par1);
     IF ("TMAIL_C_DB".BUSY = false) AND ("TMAIL_C_DB".DONE = false)
AND
     ("TMAIL_C_DB".ERROR = false) THEN
         #SendEMail := false;
     END_IF;
     IF ("TMAIL_C_DB".DONE = false) OR ("TMAIL_C_DB".ERROR = true) THEN
         #STATUS := "TMAIL_C_DB".STATUS;
       END_IF;
    END_IF;
```

Result

This instruction must be called once with SendEMail = true. In the following calls, SendEMail should not be allocated when calling. SendEMail is reset by the instruction shown, as soon as an e-mail has been sent or an error has occurred during sending.

If the email was not sent successfully, the STATUS parameter contains the corresponding value of the STATUS parameter from the TMAIL_C instruction.

See also

Description of TMAIL_C (Page 3648)

Other

TCON: Establishing a communication connection

TCON: Establish communication connection

Description

You use the "TCON" instruction to set up and establish a communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU. "TCON" is executed asynchronously.

The connection data specified for the CONNECT and ID parameters are used to set up the communication connection. To establish the connection, a rising edge must be detected at the REQ parameter. Once the connection is successfully established, the DONE parameter is set to "1".

Number of possible connections

For information on the number of possible communication connections, refer to the technical specifications for your CPU.

Connection with TCP and ISO-on-TCP

Both communication partners call the "TCON" instruction to set up and establish the communication connection. During parameter assignment, you specify which partner is the active communication end point and which is the passive one.

If the connection aborts, for example due to a line break or due to the remote communication partner, the active partner attempts to reestablish the configured connection. You do not have to call "TCON" again. This applies only if "TCON" has been executed successfully once (DONE = 1).

An existing connection is terminated and the connection set up is removed when the "TDISCON (Page 3669)" instruction is executed or when the CPU changes to STOP mode. To set up and establish the connection again, you will need to execute "TCON" again.

Parameters

The following table shows the parameters of the "TCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the job to establish the connection specified in the ID upon a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the assigned connection. Range of values: W#16#0001 to W#16#0FFF
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Auto-Hotspot

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not a job has been executed successfully. The ERROR parameter is set when errors occurred during execution of "TCON". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Connection successfully established.
0	7000	No job processing active
0	7001	Start job execution, establish connection.
0	7002	Connection is being established (REQ irrelevant).
1	8085	Connection ID (ID parameter) is already being used by a configured connection.
1	8086	The ID parameter is outside the valid range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8089	The CONNECT parameter does not point to a data block.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.

ERROR	STATUS* (W#16#...)	Explanation
1	809B	The element InterfaceId within the TCON_xxx structure does not reference a hardware identifier of a CPU or CM/CP interface or has the value "0".
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.
1	80A2	Local or remote port is being used by the system.
1	80A3	Attempt being made to re-establish an existing connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A5	Connection ID is already in use.
1	80A7	Communication error: You executed "TDISCON (Page 3669)" before "TCON" had completed.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B4	You have violated one or more of the following conditions for passive connection establishment with the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0 • With local_tsap_id_len >= B#16#03, local_tsap_id[1] is an ASCII character. • local_tsap_id[1] is an ASCII character and local_tsap_id_len >= B#16#03.
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error at the connection_type parameter of the SDT TCON_Param.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80B8	Connection description of the structure element ID and the block parameter ID are not identical.
1	80C3	All connection resources are in use.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is currently receiving new parameters. • The configured connection is currently being removed by a "TDISCON (Page 3669)" instruction.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TCON: Establish communication connection

Description

You use the "TCON" instruction to set up and establish a communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU. "TCON" is executed asynchronously.

The connection data specified for the CONNECT and ID parameters are used to set up the communication connection.

- At the CONNECT parameter, if possible, only use the predefined structures as created by the connection parameter assignment in the Inspector window of the program editor.
- If the CONNECT parameter is not interconnected with a structure listed in the parameter table or if the structure includes an error, the error code 8089 is output at the STATUS parameter.

To establish the connection, a rising edge must be detected at the REQ parameter. Once the connection is successfully established, the DONE parameter is set to "1".

The "TCON" ≥ V3.0 instruction can also be used with CPU S7-1200 version ≥ 4.0.

Number of possible connections

For information on the number of possible communication connections, refer to the technical specifications for your CPU.

Connection with TCP and ISO-on-TCP

Both communication partners call the "TCON" instruction to set up and establish the communication connection. During parameter assignment, you specify which partner is the active communication end point and which is the passive one.

If the connection aborts, for example due to a line break or due to the remote communication partner, the active partner attempts to reestablish the configured connection. You do not have to call "TCON" again. This applies only if "TCON" has been executed successfully once (DONE = 1).

An existing connection is terminated and the connection set up is removed when the "TDISCON (Page 3669)" instruction is executed or when the CPU changes to STOP mode. To set up and establish the connection again, you will need to execute "TCON" again.

Parameters

The following table shows the parameters of the "TCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the job to establish the connection specified on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the assigned connection. Value range: W#16#0001 to W#16#0FFF

Parameter	Declaration	Data type	Memory area	Description
CONNECT	InOut	VARIANT	D	Pointer to the connection description • For TCP or UDP, use the structure TCON_IP_v4 For description, refer to: Auto-Hotspot • For ISO-on-TCP, use the structure TCON_IP_RFC For description, refer to: Auto-Hotspot • For ISO, use the structure TCON_ISOnative (only for CP1543-1) For description, refer to: "Structure of the connection description according to TCON_ISOnative"
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not a job has been executed successfully. The ERROR parameter is set when errors occurred during execution of "TCON". The error information is output at the STATUS parameter.

The instruction "TCON" generates an error message in version 3.0 if an active connection establishment to a remote partner fails. Create a rising edge at the REQ parameter to establish a connection once again.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Structure of the connection description according to TCON_ISOnative

A connection description DB with a structure according to TCON_ISOnative is used to assign the communication connection parameters for ISO. The fixed data structure of the TCON_ISOnative contains all parameters that are required to establish the connection.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceId	HW_ANY	64	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	1	Reference to this connection (unique ID in the value range: 1 to 4095).
4	ConnectionType	BYTE	16#16	Connection type: ISO
5	ActiveEstablished	BOOL	TRUE	ID for the manner in which the connection is established: • FALSE: Passive connection establishment • TRUE: Active connection establishment
8 … 13	RemoteMacAddress	ARRAY [1..6] of BYTE	-	MAC address of the partner end point, for example, for 00-74-41-FD-AE-84: • MacAddr[1] = 00 • MacAddr[2] = 74 • MacAddr[3] = 41 • MacAddr[4] = FD • MacAddr[5] = AE • MacAddr[6] = 84
14 … 19	LocalMacAddress	ARRAY [1..6] of BYTE	-	MAC address of the local end point, for example, for 00-74-41-FD-AE-84: • MacAddr[1] = 00 • MacAddr[2] = 74 • MacAddr[3] = 41 • MacAddr[4] = FD • MacAddr[5] = AE • MacAddr[6] = 84
20 … 53	RemoteTSelector	Struct	-	TSelector of the remote connection partner: • Bytes 20 to 21 = TSelLength • Bytes 22 to 53: = TSel[1-32]
	TSelLength	UINT	-	Value range from 0 to 32
	TSel	ARRAY [1..32] of BYTE	-	Value range in each case 0 to 255 in bytes
54 … 87	LocalTSelector	Struct	-	TSelector of the remote connection partner: • Bytes 20 to 21 = TSelLength • Bytes 22 to 53: = TSel[1-32]
	TSelLength	UINT		Value range from 0 to 32
	TSel	ARRAY [1..32] of BYTE	-	Value range in each case 0 to 255 in bytes
88 … 89	CrRetransmitionTime	UINT	-	Duration until connection attempt is repeated in seconds.

Byte	Parameter	Data type	Start value	Description
90 .. . 91	DataRetransmition-Time	UINT	100 ms	Duration until data transfer is repeated in milliseconds.
92 .. . 93	MaxRetransmition-Count	UINT	-	Maximum number of repetitions.
94 .. . 95	InactivityTime	UINT	-	in seconds
96 .. . 97	WindowTime	UINT		in seconds

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Connection successfully established.
0	7000	No job processing active
0	7001	Start job execution, establish connection.
0	7002	Connection is being established (REQ irrelevant).
1	8085	Connection ID (ID parameter) is already being used by a configured connection.
1	8086	The ID parameter is outside the valid range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8089	The CONNECT parameter does not point to a connection description or the connection description was created manually.
1	809A	The structure at the CONNECT parameter is not supported or the length is invalid.
1	809B	The element InterfaceId within the TCON_xxx structure does not reference a hardware identifier of a CPU or CM/CP interface or has the value "0".
1	80A2	Local or remote port is being used by the system. The following ports are reserved locally: 20, 21, 80, 102, 135, 161, 162, 443, 34962, 34963, 34964 as well as the area 49152 to 65535.
1	80A3	ID is used by a connection created by the user program, which uses the same connection description at the CONNECT parameter.
1	80A4	IP address of the remote endpoint of the connection is invalid or it corresponds to the IP address of the local partner.
1	80A7	Communication error: You executed "TDISCON (Page 3669)" before "TCON" had completed.
1	80B4	Only with TCON_IP_RFC: The local T selector was not specified or the first byte does not contain the value 0x0E (only with a length of T selector = 2) or the local T selector starts with "SIMATIC-".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP (parameter ActiveEstablished of the structure TCON_IP_v4 / TCON_PARAM has the value TRUE).
1	80B6	Parameter assignment error in the ConnectionType parameter of the data block for connection description. • Only valid with TCON_IP_v4: 0x11, 0x0B and 0x13. • Only valid with TCON_IP_RFC: 0x0C and 0x12

ERROR	STATUS* (W#16#...)	Explanation
1	80B7	With TCON_IP_v4: • TCP (active connection establishment): Remote port is "0". • TCP (passive connection establishment): Local port is "0". • UDP: Local port is "0". With TCON_IP_RFC: • Local (LocalTSelector) or remote (RemoteTSelector) T selector was specified with a length of more than 32 bytes. • For TSelLength of the T selector (local or remote), a length greater than 32 was entered. • Error in the length of the IP address of the specific connection partner.
1	80B8	Parameter ID in the local connection description (structure at CONNECT parameter) and parameter ID of the instruction are different.
1	80C3	All connection resources are in use.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is currently receiving new parameters. • The configured connection is currently being removed by a "TDISCON (Page 3669)" instruction.
1	80C5	The connection partner refuses to establish the connection, has terminated the connection or actively ended it.
1	80C6	The connection partner cannot be reached (network error).
1	80C7	Execution timeout.
1	80C8	Value at the ID parameter is already being used by a connection that was created using the user program. The connection uses the identical ID, but different connection settings at the parameter CONNECT.
1	80C9	Validation of the connection partner failed. The connection partner that wants to establish the connection does not match the defined partner of the structure at the CONNECT parameter.
1	80CE	The IP address of the local interface is 0.0.0.0.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on switching display formats, refer to "See also".		

TDISCON: Terminate communication connection

Description

The "TDISCON" instruction terminates a communications connection from the CPU to a connection partner.

Functional description

"The TDISCON" instruction works asynchronously, which means its job processing extends over multiple calls. You start the job for terminating a connection by calling the "TDISCON" instruction with REQ = 1.

After "TDISCON" has been successfully executed, the ID specified for "TCON" is no longer valid and cannot be used for sending or receiving.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions (see also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193)).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can recognize the current status of "TDISCON" or when the establishment of the connection is completed.

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS parameter.
0	0	0	The instruction was not assigned a (new) job.

Parameters

The following table shows the parameters of the instruction "TDISCON":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
REQ	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Control parameter REQUEST starts the job for terminating the connection specified by ID. The job starts on a rising edge.
ID	Input	CONN_OUC (WORD)	D, L or constant	I, Q, M, D, L or constant	Reference to the connection to be terminated (connection ID) Range of values: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status parameter: • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status parameter: • BUSY = 1: The job is not yet completed. • BUSY = 0: The job is completed or has not started yet.
ERROR	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status parameter: • ERROR=0: No error. • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
STATUS	Output	WORD	I, Q, M, D, L	I, Q, M, D, L	Status parameter: Error information (see "Parameters ERROR and STATUS")

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameters ERROR and STATUS

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Connection terminated successfully.
0	7000	No job processing active.
0	7001	Start of job processing, connection being terminated.
0	7002	Intermediate call (REQ irrelevant), connection being terminated.
1	8086	The ID parameter is not in the permitted value range.
1	80A3	Attempt is being made to terminate a non-existent connection or the connection is already terminated.
1	80C4	Temporary communication error: New parameters are being assigned to the interface or the connection is currently being established.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TSEND: Send data via communication connection

TSEND: Send data via communication connection

Description

The following description of the "TSEND" instruction is valid for the CPU S7-1200 to version 3.0.

You use the "TSEND" instruction to send data over an existing communication connection. "TSEND" is executed asynchronously.

You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. All data types except BOOL and Array of BOOL can be used for the data to be sent.

The send job is executed when a rising edge is detected at the REQ parameter.

With the LEN parameter, you specify the maximum number of bytes sent with a send job.

- When data is transferred with TCP (streaming protocol), the "TSEND" instruction provides no information about the length of the data sent to "TRCV (Page 3677)".
- When data is transferred with ISO-on-TCP (message-oriented protocol), the length of the data sent is communicated to "TRCV (Page 3677)". The amount of data sent with "TSEND" as packet must also be received again at the receiving end ("TRCV (Page 3677)"):
  - If the receive buffer is too small for the sent data, an error occurs at the receiving end.
  - If the receive buffer is sufficiently large, "TRCV" returns with DONE=1 as soon as the data packet is received.

The data to be sent must not be edited until the send job is completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter is not confirmation that the data sent has already been read by the communication partner.

Parameters

The following table shows the parameters of the "TSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the send job on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON". Range of values: W#16#0001 to W#16#0FFF
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum number of bytes to be sent with the job.
DATA	InOut	VARIANT	I, Q, M, D	Pointer to the send area containing the address and the length of the data to be sent. The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

LEN and DATA parameters

- With LEN = 0 , all the data specified with the DATA parameter is sent.
- If the number of bytes at the LEN parameter is larger than the length of the data to be sent that was defined with the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS parameter below).
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.

- With data types STRING and WSTRING, all data is transferred if the parameter LEN = 0. When LEN > 0, the length must be at least the maximum number of bytes plus two additional bytes which contain the length information. You can find more detailed information on the structure of the data types in: "Overview of the valid data types (Page 1908)".
- The maximum number of bytes that can be transmitted is 65534.
- When you use structured tags from optimized DBs, the address of the structured tag should be interconnected at the DATA parameter and the parameter LEN = 0. This guarantees type-safe transmission of the entire structure if the same structure is used at the receiving end.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TSEND" is executed asynchronously, you must keep the data in the send area consistent until the DONE parameter or the ERROR parameter changes to the value "1".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Send job completed without error.
0	7000	No job processing active.
0	7001	Start of job execution, data is being sent. When processing this job, the operating system accesses the data in the DATA send area.
0	7002	Job executing (REQ irrelevant). When processing this job, the operating system accesses the data in the DATA send area.
1	8085	• The LEN parameter is greater than the highest permitted value (65536). • DATA and LEN parameters both have the value "0".
1	8086	The ID parameter is outside the permitted address range (1..0xFFF).
1	8088	The LEN parameter is greater than the area specified in DATA.

ERROR	STATUS* (W#16#...)	Description
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Transfer via this connection is not possible. • The interface is being re-initialized.
1	80B3	The protocol variant (ConnectionType parameter in the connection description) is set to UDP. Use the instruction "TUSEND" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.
1	80C5	Connection terminated by the communication partner.
1	80C6	Network error. Communication partner cannot be reached.
1	80C7	Execution timeout.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TSEND: Send data via communication connection

Description

The following description of the "TSEND" instruction is valid for the CPU S7-1500 and S7-1200 ≥ V4.0.

You use the "TSEND" instruction to send data over an existing communication connection. "TSEND" is executed asynchronously.

You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. All data types except BOOL and Array of BOOL can be used for the data to be sent.

The send job is executed when a rising edge is detected at the REQ parameter.

With the LEN parameter, you specify the maximum number of bytes sent with a send job.

- When data is transferred with TCP (streaming protocol), the "TSEND" instruction provides no information about the length of the data sent to "TRCV (Page 3677)".
- When data is transferred with ISO-on-TCP (message-oriented protocol), the length of the data sent is communicated to "TRCV (Page 3677)". The amount of data sent with "TSEND" as packet must also be received again at the receiving end ("TRCV (Page 3677)"):
  - If the receive buffer is too small for the sent data, an error occurs at the receiving end.
  - If the receive buffer is sufficiently large, "TRCV" returns with DONE=1 as soon as the data packet is received.

The data to be sent must not be edited until the send job is completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter is not confirmation that the data sent has already been read by the communication partner.

Parameters

The following table shows the parameters of the "TSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the send job on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON". Value range: W#16#0001 to W#16#0FFF
LEN	Input	UDINT	I, Q, M, D, L or constant	Maximum number of bytes that are sent with the job (maximum permissible value for S7-1200: 8192, maximum permissible value for S7-1500: 65536).
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the send area containing the address and the length of the data to be sent. The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block • Local data
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameters LEN and DATA

- With LEN = 0 , all the data specified with the DATA parameter is sent.
- If the number of bytes at the LEN parameter is larger than the length of the data to be sent that was defined with the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS parameter below).
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- With data types STRING and WSTRING, all data is transferred if the parameter LEN = 0. When LEN > 0, the length must be at least the maximum number of bytes plus two additional bytes which contain the length information. You can find more detailed information on the structure of the data types in: "Overview of the valid data types (Page 1908)".

- The maximum number of bytes that can be transmitted depends on the device.
- When you use structured tags from optimized DBs, the address of the structured tag should be interconnected at the DATA parameter and the parameter LEN = 0. This guarantees type-safe transmission of the entire structure if the same structure is used at the receiving end.

Parameters BUSY, DONE and ERROR

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TSEND" is executed asynchronously, you must keep the data in the send area consistent until the DONE parameter or the ERROR parameter changes to the value "1".

Parameters ERROR and STATUS

ERROR	STATUS* (W#16#...)	Description
0	0000	Send job completed without error.
0	7000	No job processing active.
0	7001	Start of job execution, data is being sent. When processing this job, the operating system accesses the data in the DATA send area.
0	7002	Job executing (REQ irrelevant). When processing this job, the operating system accesses the data in the DATA send area.
1	8085	• The LEN parameter is greater than the maximum permissible value (for S7-1200: 8192, for S7-1500: 65536). • DATA and LEN parameters both have the value "0".
1	8086	The ID parameter is outside the permitted address range (1..0xFFF).
1	8088	The LEN parameter is greater than the area specified in DATA.
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Transfer via this connection is not possible. • The interface is being re-initialized.

ERROR	STATUS* (W#16#...)	Description
1	80B1	You changed the DATA parameter before the current job finished.
1	80B3	The protocol variant (ConnectionType parameter in the connection description) is set to UDP. Use the instruction "TUSEND" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.
1	80C5	Connection terminated by the communication partner.
1	80C6	Network error. Communication partner cannot be reached.
1	80C7	Execution timeout.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

TRCV: Receive data via communication connection

TRCV: Receive data via communication connection

Description

The following description of the "TRCV" instruction is valid for the CPU S7-1200 up to version 3.0.

You use the "TRCV" instruction to receive data over an existing communication connection. "TRCV" is executed asynchronously.

Receipt of data is enabled when the EN_R parameter is set to the value "1". The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0), depending on the protocol variant being used.

While data is being received, you cannot make changes to the DATA parameter or the defined receive area to ensure consistency of the received data.

After successful receipt of data, the NDR parameter is set to the value "1". You can query the amount of data actually received at the RCVD_LEN parameter.

Receive modes of "TRCV"

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type* parameter of the connection description	LEN parameter
TCP (Ad-hoc mode)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	0
TCP (Receipt of data with specified length)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192
ISO on TCP (Message-oriented data transfer)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	• 1 to 1452, if a CP is used. • 1 to 8192, if no CP is used.
* See "Auto-Hotspot".			

TCP (Ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV" instruction.

You set ad-hoc mode by assigning the value "0" to the LEN parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). When ad-hoc mode is active, receipt of data is already signaled after a received byte at the NDR parameter.

TCP (Receipt of data with specified length)

For receipt of data with a specified length, enter the length of the data at the LEN parameter. The data receipt is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

ISO on TCP (Message-oriented data transfer)

Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. When using ISO on TCP, "TRCV" signals data receipt as soon as the message block has been completely received. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV" signals an error. The successful receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

Parameters

The following table shows the parameters of the "TRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Receive enable
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON (Page 3662)". Range of values: W#16#0001 (1) to W#16#0FFF (4095)
LEN	Input	UDINT	I, Q, M, D, L or constant	Length of the receive area in bytes (hidden). If you use a memory area with optimized access at the DATA parameter, the LEN parameter must have the value "0".
DATA	InOut	VARIANT	I, Q, M, D	Pointer to the receive area
NDR	Output	BOOL	I, Q, M, D, L	Status parameter (New Data Received): • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter: Output of status and error information.
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

LEN, DATA and RCVD_LEN parameters

- If LEN = 0, the received data is saved in the receive area specified at the DATA parameter. The number of bytes received is indicated at the RCVD_LEN parameter.
- If the length specified at the LEN parameter is greater than the length of the data received at the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS) parameter in the following.
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- If the DATA parameter references a data block with optimized access, the LEN parameter must be set to "0".

- If a STRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=2 (LEN = 1 is not permitted).
- If a WSTRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=5.

BUSY, NDR and ERROR parameters

You can check the status of the job with the BUSY, NDR, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the NDR parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of TRCV. The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, NDR and ERROR parameters:

BUSY	NDR	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TRCV" is executed asynchronously, the data in the receive area is only consistent when the NDR parameter is set to the value "1".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Job completed successfully. The current length of the received data is output at the RCVD_LEN parameter.
0	7000	Block not ready to receive.
0	7001	Block is ready to receive, receive job was activated.
0	7002	Interim call, the receive job is executing. Note: While the job is being processed, data is written to the receive area. Access to the receive area during this time may provide inconsistent data.
1	8085	• The LEN parameter is larger than the highest permitted value. • The value of the LEN or DATA parameter was changed after the first call. • Both LEN parameters and the DATA parameter have the value "0" or LEN is longer than the maximum permitted value (65536).
1	8086	The ID parameter is outside the permitted address range (1 .. 0x0FFF).
1	8088	• Receive area is too small. • The value at the LEN parameter is larger than the receive area set at the DATA parameter.

ERROR	STATUS* (W#16#...)	Explanation
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Receive job over this connection is not possible. • The connection is being re-initialized.
1	80B3	The protocol variant (connection_type parameter in the connection description) is set to UDP. Use the instruction "TURCV" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.
1	80C5	The remote partner has terminated the connection.
1	80C6	The remote partner cannot be reached (network error).
1	80C7	Execution timeout.
1	80C9	The length of the receive area is smaller than the length of the sent data.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TRCV: Receive data via communication connection

Description

The following description of the "TRCV" instruction is valid for the CPU S7-1500 and S7-1200 ≥ V4.0.

You use the "TRCV" instruction to receive data over an existing communication connection. "TRCV" is executed asynchronously.

Receipt of data is enabled when the EN_R parameter is set to the value "1". The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0), depending on the protocol variant being used.

While data is being received, you cannot make changes to the DATA parameter or the defined receive area to ensure consistency of the received data.

After successful receipt of data, the NDR parameter is set to the value "1". You can query the amount of data actually received at the RCVD_LEN parameter.

Receive modes of "TRCV"

The following table shows how the received data is entered in the receive area.

Protocol variant	ADHOC parameter	Availability of data in the receive area	connection_type parameter of the connection description	Parameter LEN
TCP (Ad-hoc mode)	1 (ad-hoc activated)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	1 up to maximum length (depending on the CPU)
TCP (Receipt of data with specified length)	0 (ad-hoc deactivated)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192
ISO on TCP (Message-oriented data transfer)	-	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	• 1 to 1452, if a CP is used. • 1 to 8192, if no CP is used.

TCP (Ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV" instruction.

You set ad-hoc mode by assigning the value "1" to the ADHOC parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). When ad-hoc mode is active, receipt of data is already signaled after a received byte at the NDR parameter.

TCP (Receipt of data with specified length)

Assign the value "0" to the ADHOC parameter for receipt of data with specified length. If ad-hoc mode is disabled, the data reception is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The successful receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

ISO on TCP (Message-oriented data transfer)

Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. When using ISO on TCP, "TRCV" signals data reception as soon as the message block has been completely received. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV" signals an error. The successful receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

Parameters

The following table shows the parameters of the "TRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Receive enable
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON". Value range: W#16#0001 to W#16#0FFF
LEN	Input	UDINT	I, Q, M, D, L or constant	Length of the receive area in bytes (hidden) (maximum value for S7-1200: 8192, maximum value for S7-1500: 65536). If you use a receive area with optimized access at the DATA parameter, the LEN parameter must have the value "0".
ADHOC	Input	BOOL	I, Q, M, D, L or constant	Use ad-hoc mode for the TCP protocol variant (hidden).
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area
NDR	Output	BOOL	I, Q, M, D, L	Status parameter (New Data Received): • 0: Job not yet started or still in progress • 1: New data received
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameters LEN, DATA and RCVD_LEN

- If LEN = 0, the received data is saved in the receive area specified at the DATA parameter. The number of bytes received is indicated at the RCVD_LEN parameter.
- If the length specified at the LEN parameter is greater than the length of the data received at the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS) parameter in the following.
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- If the DATA parameter references a data block with optimized access, the LEN parameter must be set to "0". If the length of the data does not match for elementary data types, the data will not be received and the error code 8088 is output at the STATUS parameter.

- If a STRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=2 (LEN = 1 is not permitted).
- If a WSTRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=5.

Parameters BUSY, NDR and ERROR

You can check the status of the job with the BUSY, NDR, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the NDR parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of TRCV. The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, NDR and ERROR parameters:

BUSY	NDR	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TRCV" is executed asynchronously, the data in the receive area is only consistent when the NDR parameter is set to the value "1".

Parameters ERROR and STATUS

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Job completed successfully. The current length of the received data is output at the RCVD_LEN parameter.
0	7000	Block not ready to receive.
0	7001	Block is ready to receive, receive job was activated.
0	7002	Interim call, the receive job is executing. Note: While the job is being processed, data is written to the receive area. Access to the receive area during this time may provide inconsistent data.
1	8085	• The LEN parameter is greater than the maximum permissible value (for S7-1200: 8192 bytes, for S7-1500: 65536 bytes). • The value of the LEN or DATA parameter was changed after the first call. • Both LEN parameters and the DATA parameter have the value "0" or LEN is longer than the maximum permissible value (for S7-1200): 8192 bytes, for S7-1500: 65536 bytes).
1	8086	The ID parameter is outside the permitted value range (1 .. 0x0FFF).
1	8088	• Receive area is too small. • The value at the LEN parameter is larger than the receive area set at the DATA parameter.

ERROR	STATUS* (W#16#...)	Explanation
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Receive job over this connection is not possible. • The connection is being re-initialized.
1	80B1	You changed the DATA parameter before the current job finished.
1	80B3	The protocol variant (connection_type parameter in the connection description) is set to UDP. Use the instruction "TURCV" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.
1	80C5	The remote partner has terminated the connection.
1	80C6	The remote partner cannot be reached (network error).
1	80C7	Execution timeout.
1	80C9	The length of the receive area is smaller than the length of the sent data.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

See also

TRCV: Receive data via communication connection (Page 3677)

TCON: Establish communication connection (Page 3664)

Structure of the address information for the remote partner with UDP

Overview

When using a UDP connection, the address information for the remote partner is stored in the system data type TADDR_Param:

- By using the instruction "TUSEND (Page 3687)" the address information of the recipient is assigned to the parameter ADDR via TADDR_Param.
  The stored address data of the remote partner are read from the system data type by the instruction.
- By using the instruction "TURCV (Page 3690)" the address of the sender is received at the parameter ADDR via TADDR_Param.
  The address data is written by the instruction in the system data type.

Structure of the address information according to TADDR_Param

The system data type TADDR_Param contains the address information of the remote partner and consists of the IP address and the port number.

The system data type TADDR_Param has the following structure:

Byte	Parameter	Data type	Start value	Description
0 to 3	rem_ip_addr	ARRAY [1..4] of USINT	B#16#00 ...	IP address of the remote partner, e.g. 192.168.002.003: • rem_ip_addr[1] = B#16#C0 (192) • rem_ip_addr[2] = B#16#A8 (168) • rem_ip_addr[3] = B#16#02 (002) • rem_ip_addr[4] = B#16#03 (003) The IP address can be taken from the Devices & networks view in the interface properties of the remote partner. As an alternative these are also displayed in the properties of the UDP connection under Address details.
4 to 5	rem_port_nr	UINT	B#16#00 ...	Remote port-number (possible values see: Auto-Hotspot): • rem_port_nr[1] = high byte of the port no. in hexadecimal notation • rem_port_nr[2] = low byte of port no. in hexadecimal notation The port number can be taken from the Devices & networks view in the UDP connection properties. The port number is indicated under Address details as a decimal value. Example: Port number = 2000 (decimal) / W#16#07D0 (hexadecimal) • rem_port_nr[1] = 07 (high byte) • rem_port_nr[2] = D0 (low byte)
6 to 7	reserved	WORD	B#16#00 ...	Not used. Leave the value "0" at this parameter.

Creating TADDR_Param in a data block

You have the following options for creating TADDR_Param:

- Create a new data block and select TADDR_Param as type in the dialog "Add new data block".
- Open an existing data block create a new tag and enter in the column Data type TADDR_Param.

A data block can contain several system data types TADDR_Param.

TUSEND: Send data via Ethernet (UDP)

Description

The "TUSEND" instruction sends data to the remote partner addressed by the ADDR parameter using UDP.

Warning

Data transfer via UDP

Data transferred via UDP to RFC 768 to the remote partner are sent without acknowledgement and are therefore unsecured. This means the data can be lost without any indication at the block.

Note

For sequential send operations to different partners, you only need to adjust the ADDR parameter when calling "TUSEND". You do not need to call the "TCON (Page 3662)" and "TDISCON (Page 3669)" instructions again. The UDP port must be set up at the specific remote partner in order for this partner to receive data.

Functional description

"The TUSEND" instruction works asynchronously, which means its job processing extends over multiple calls. Create a rising edge at the REQ parameter to establish a connection once again.

The output parameters BUSY, DONE, ERROR and STATUS indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current status of "TUSEND" or when the send process is concluded.

BUSY	DONE	ERROR	Description
TRUE	FALSE	FALSE	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TUSEND", make sure the data in the send area remains consistent until the DONE parameter or the ERROR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TUSEND" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter REQUEST starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the associated parameter ID in the local connection description with the instruction "TCON". Value range: W#16#0001 to W#16#0FFF
LEN	Input	UINT	I, Q, M, D, L or constant	Number of bytes to be sent with the job Value range: 1 to 1472
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without errors. This value is only displayed for one cycle.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: An error has occurred during processing, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Error information
DATA	InOut	VARIANT	I, Q, M, D	Send area, contains address and length The address refers to: • The process image of the inputs • The process image of the outputs • A bit memory • A data block
ADDR	InOut	VARIANT	D	Pointer to the system data type TADDR_Param. Store the address information of the remote partner (IP address and port number) in a data block with the system data type TADDR_Param. See also: Structure of the address information for the remote partner with UDP (Page 3685)

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Send job completed without error
0	7000	No job processing active
0	7001	Start of job processing, data being sent Note: During this processing phase, the operating system accesses the data in the DATA send area.
0	7002	Intermediate call (REQ irrelevant), job is being processed Note: During this processing phase, the operating system accesses the data in the DATA send area.
1	8085	The LEN parameter has the value "0" or is greater than the highest permitted value.
1	8086	The ID parameter is not in the permitted value range.
0	8088	The LEN parameter is greater than the memory area specified in DATA.
1	8089	The ADDR parameter does not point to a data block with the TADDR_Param structure.
1	80A1	Communication error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. Transmission over this connection is not possible. • The interface is being reinitialized.
1	80B1	You changed the DATA parameter before the current job finished.
1	80A4	IP address (at the ADDR parameter) of the remote connection end point is invalid; it may correspond to the local partner's own IP address.
1	80B3	• The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TSEND (Page 3674)". • ADDR parameter: Invalid information for port no.
1	80B7	The length of the structure referenced by the parameter ADDR is not 8 bytes.
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection between the user program and the communication layer of the operating system cannot be established at this time. • New parameter settings are being assigned to the interface.
1	80C5	The remote partner has terminated the connection.
1	80C6	The remote partner cannot be reached (network error).
1	80C7	Execution timeout.
-	General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on switching display formats, refer to "See also".		

TURCV: Receive data via Ethernet (UDP)

Description

The "TURCV" instruction receives data via UDP. After successful completion of "TURCV", the ADDR parameter will show you the address of the remote partner (the sender).

Warning

Unsecured data transfer

Data transferred via UDP to RFC 768 to the remote partner are sent without acknowledgement and are therefore unsecured. This means the data can be lost without any indication at the block.

Functional description

"The TURCV" instruction works asynchronously, which means its job processing extends over multiple calls. You start the receive job by calling "TURCV" with EN_R = 1.

The output parameters BUSY, DONE, ERROR and STATUS indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 2193).

The following table shows the relationship between BUSY, NDR and ERROR. Using this table, you can determine the current status of TURCV or when the receive process is concluded.

BUSY	NDR	ERROR	Description
TRUE	FALSE	FALSE	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed. New data was received.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TURCV", the data in the receive area is only consistent when the NDR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TURCV" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Control parameter enabled to receive When EN_R = 1, "TURCV" is ready to receive. The receive job is being processed.
ID	Input	WORD	I, Q, M, D, L or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the corresponding parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	UINT	I, Q, M, D, L or constant	Length of the receive area in bytes: 0 (recommended) or 1 to 1472
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • NDR = 0: Job not yet started or still running • NDR = 1: Job successfully completed
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Error information
RCVD_LEN	Output	UINT	I, Q, M, D, L	Amount of data actually received in bytes
DATA	InOut	VARIANT	I, Q, M, D, L	Receive area The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block
ADDR	InOut	VARIANT	D	Pointer to the system data type TADDR_Param. The address information of the remote partner (IP address and port number) is written in a data block with the system data type TADDR_Param. See also: Structure of the address information for the remote partner with UDP (Page 3685)

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	New data was accepted. The current length of the received data is shown in RCVD_LEN.
0	7000	Block not ready to receive
0	7001	Block is ready to receive, receive job was activated
0	7002	Intermediate call, receive job being processed Note: During this processing phase, "TURCV" writes data to the receive area. For this reason, an error could result in inconsistent data in the receive area.
1	8085	The LEN parameter is greater than the largest permitted value, or you changed the value of the LEN or DATA parameter since the first call
1	8086	The ID parameter is not in the permitted value range
1	8088	• Receive area is too small • Value in LEN is higher than the receive area specified by DATA
1	8089	The ADDR parameter does not point to a data block with the TADDR_Param structure.
1	80A1	Communication error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. A receive job over this connection is not possible. • New parameter settings are being assigned to the interface.
1	80B1	You changed the DATA parameter before the current job finished.
1	80B3	The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TRCV (Page 3677)".
1	80B7	The length at ADDR parameter does not correspond to 8 bytes.
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: New parameter settings are being assigned to the interface.
1	80C5	The remote partner has terminated the connection.
1	80C7	Execution timeout.
1	80C9	With RFC1006 / UDP: The received data is longer than expected (size of receive buffer exceeded).
-	General error information	See also: GET_ERR_ID: Get error ID locally (Page 2417)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on switching display formats, refer to "See also".		

See also

TCON: Establish communication connection (Page 3662)

TDISCON: Terminate communication connection (Page 3669)

T_RESET: Resetting the connection

Description

The "T_RESET" instruction terminates and then reestablishes an existing connection.

The local end points of the connection are retained. They are generated automatically:

- If a connection has been configured and loaded to the CPU.
- If a connection has been generated by the user program, for example, by calling the instruction "TCON (Page 3662)".

The instruction "T_RESET" can be executed for all connection types (TCP, UDP, ISO-on-TCP, etc.). It does not matter whether the local interface of the CPU or the interface of a CM/CP was used for the connection.

Once the instruction "T_RESET" has been called using the REQ parameter, the connection specified with the ID parameter is terminated and, if necessary, the data send and receive buffer cleared. Canceling the connection also cancels any data transfer in progress. There is therefore a risk of losing data if data transfer is in progress. The CPU defined as the active connection partner will then automatically attempt to restore the interrupted communication connection. You therefore do not need to call the "TCON (Page 3662)" instruction to reestablish the communication connection.

The output parameters DONE, BUSY and STATUS indicate the status of the job.

Parameters

The following table shows the parameters of the "T_RESET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter REQUEST starts the job for terminating the connection specified by ID. The job starts on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection to the passive partner ID being interrupted must be identical to the corresponding parameter ID in the local connection description. Range of values: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter BUSY • 0: Job is complete. • 1: The job is not yet completed.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR • 0: No error occurred. • 1: Error occurred during processing. The STATUS parameter supplies detailed information on the type of error
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS Error information (see "STATUS parameter" table).

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter STATUS

STATUS* (W#16#...)	Explanation
0000	No error.
0001	Connection has not been established.
7001	Connection termination launched.
7002	Connection being terminated.
8081	Unknown connection specified at the ID parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

T_DIAG: Checking the connection

Description

You use the "T_DIAG" instruction to check the status of a connection and read further information on the local end point of this connection.

- The connection is referenced by the ID parameter. You can read both connection end points configured in the connection editor and programmed connection end points (e.g. with the "TCON" instruction).
  Temporary connection end points (for example end points created when you connect to an engineering station) cannot be diagnosed, as no connection ID is generated in this process.
- The connection information read is stored in a structure referenced by the RESULT parameter.
- The output parameter STATUS indicates whether it was possible to read the connection information. The connection information in the structure at the RESULT parameter is only valid if the "T_DIAG" instruction has been completed with STATUS = W#16#0000 and ERROR = FALSE.
  Connection information cannot be evaluated if an error occurs.

Possible connection information

Two different structures can be used to read the connection information at the RESULT parameter:

- The "TDiag_Status" structure only contains the most important information about a connection end point, for example the protocol used, the connection status and the number of data bytes sent or received.
- The structure "TDiag_StatusExt" supplies not just the most important information, but also the number of connection attempts, the reason for a connection abort, etc.

The structure and parameters of the two structures are described below (see the "TDIAG_Status and TDIAG_StatusExt structures" table).

Parameters

The following table shows the parameters of the "T_DIAG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the instruction to check the connection specified in the ID parameter when there is a positive edge.
ID	Input	CONN_OUC (WORD)	I, Q, M, D, L or constant	Reference to the assigned connection. Value range: W#16#0001 to W#16#0FFF
RESULT	InOut	VARIANT	D	Pointer to the structure in which the connection information is stored. The structure TDiag_Status or TDiag_StatusExt can be used at the RESULT parameter (for a description, see the "TDIAG_Status and TDIAG_StatusExt structures" table).
DONE	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Instruction not yet started or still in progress. • 1: Instruction executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Instruction not yet started or already completed. • 1: Instruction not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error. • 1: Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Parameters BUSY, DONE and ERROR

You can check the status of "T_DIAG" instruction execution with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not an instruction has been executed successfully. The ERROR parameter is set if errors occur during execution of "T_DIAG".

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ER-ROR	Description
1	-	-	The instruction is being processed.
0	1	0	The instruction has been executed successfully. The data in the structure referenced by RESULT are only valid if this is the case.
0	0	1	Instruction completed with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new instruction has been assigned.

Parameter STATUS

The following table shows the meaning of the values at the STATUS parameter:

STA-TUS* (W#16#...)	Explanation
0000	The instruction "T_DIAG" has been executed successfully. The data in the structure referenced at the RESULT parameter can be evaluated.
7000	No instruction processing active.
7001	Instruction processing launched.
7002	Connection information is being read (REQ parameter irrelevant).
8086	The value at the ID parameter is outside the valid range (W#16#0001 ... W#16#0FFF).
8089	The RESULT parameter points to an invalid data type (structures TDIAG_Status and TDIAG_StatusExt only).
80A3	The ID parameter references a connection end point which does not exist. With programmed connections, this error can also occur after the "TDISCON" instruction is called.
80C4	Internal error. Access to the connection end point is temporarily unavailable.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Structures TDIAG_Status and TDIAG_StatusExt

The table below details the form of the TDIAG_Status and TDIAG_StatusExt structures:

- From the InterfaceID to the ReceivedBytes parameter, the "TDIAG_StatusExt" and "TDIAG_Status" structures are identical.
- The structure TDIAG_StatusExt also includes the parameters ConnTrials to LastDisconnTimeStamp.

The value of each element is only valid if the instruction has been executed without errors. If an error occurs, the content of the parameters will not change.

Name	Data type	Description
The following parameters are in both the TDIAG_Status and the TDIAG_StatusExt structure:		
InterfaceID	HW_ANY	Interface ID (LADDR) of the CPU or the CM/CP.
ID	CONN_OUC	ID of the connection diagnosed. Following a successful call, the value of this element is identical to the parameter ID of the "T_DIAG" instruction.
ConnectionType	BYTE	Protocol type used for connection: • 0x01: Not used. • ... • 0x0B: TCP protocol (IP_v4) • 0x0C: ISO-on-TCP protocol (RFC1006) • 0x0D: TCP protocol (DNS) • 0x0E: Dial-in protocol • 0x0F: WDC protocol • 0x10: SMTP protocol • 0x11: TCP protocol • 0x12: TCP and ISO-on-TCP protocol (RFC1006) • 0x13: UDP protocol • 0x14: Reserved • 0x15: PROFIBUS bus access protocol (FDL) • 0x16: ISO 8073 transport protocol (ISO native) • ... • 0x20: SMTP or SMTPS protocol - based on IPv4 • 0x21: SMTP or SMTPS protocol - based on IPv6 • 0x22: SMTP or SMTPS protocol - based on FQDN (**F**ully **Q**ualified **D**omain **N**ame) • ... • 0x70: S7 connection • Other: Reserved
ActiveEstablished	BOOL	• FALSE: Locally, the passive connection end point • TRUE: Locally, the active connection end point

Name	Data type	Description
State	BYTE	Current status of the connection end point • 0x00: Not used. • 0x01: Connection terminated. Temporary status, for example, after the "T_RESET" instruction is called. The system then automatically attempts to reestablish the connection. • 0x02: The active connection end point is attempting to establish a connection to the remote communication partner. • 0x03: The passive connection end point is waiting for establishment of the connection to the remote communication partner. • 0x04: Connection established. • 0x05: The connection is being terminated. This may be because the "T_RESET" or "T_DISCON" instruction has been called. Other possible reasons are protocol errors and line breaks. • 0x06..0xFF: Not used.
Kind	BYTE	Mode of the connection end point: • 0x00: Not used. • 0x01: Configured, static connection which has been configured and loaded to the CPU. • 0x02: Configured, dynamic connection which has been configured and loaded to the CPU (not currently supported). • 0x03: Programmed connection generated in the user program with the instruction "TCON". A call of the instruction "TDISCON" or a transition to CPU STOP status has destroyed the connection end point. • 0x04: Temporary, dynamic connection established by the engineering station (ES) or operator station (OS), for example. (this connection type cannot currently by diagnosed as there is no ID). • 0x05..0xFF: Not used.
SentBytes	UDINT	Number of data bytes sent.
ReceivedBytes	UDINT	Number of data bytes received.
The following parameters only occur in the TDiag_StatusExt structure:		
ConnTrials	UDINT	Number of connection attempts. Once a connection has been established, ConnTrials has the value 0. If this element is not equal to 0, there are connection problems. Note: With a passive connection end point, this value is never greater than 1.
ConnTrialsSuccess	UDINT	Number of successful connection attempts. This element is never reset during the life cycle of a connection end point, and returns to 0 after reaching 0xFFFF FFFF. Note: This parameter is 1 if there has never been a problem in this connection.

Name	Data type	Description
LastConnErrReason	UDINT	Error ID output during the last connection attempt with errors (the error messages are identical to those at the LastDisconnReason parameter): • 0x4F01: Remote connection end point cannot be reached (this error usually occurs during connection establishment). • 0x4F02: Connection terminated locally. • 0x4F03: Connection terminated by remote communication partner. • 0x4F04: Connection terminated by a protocol error. • 0x4F05: Connection terminated by a network problem detected locally. • 0x4F06: Connection terminated by a network problem detected remotely. • 0x4F07: Connected terminated due to timeout in protocol. • 0x4F08: Incorrect parameter assignment: Connection to be established to local partner's own address. • 0x4F09: Connection temporarily reset by a call of the instruction "T_RESET". • 0x4F0A: Insufficient connection resources available (quantity exceeded) • 0x4F0B: Internal error: Incorrect addressing parameters • 0x4F0C: Internal CPU communication error • 0x4F0D: Internal AS communication error between CPU and CM/CP • 0x4F0E: The local TCP/UDP port (or RFC1006-T selector) specified is already in use.
LastConnErrTimeStamp	LDT	Time of the last connection attempt with errors.
LastDisconnReason	UDINT	Error ID which led to the last connection termination (the error messages are identical to those at the LastConnErrReason parameter): • 0x4F01: Remote connection end point cannot be reached (this error usually occurs during connection establishment). • 0x4F02: Connection terminated locally. • 0x4F03: Connection terminated by remote communication partner. • 0x4F04: Connection terminated by a protocol error. • 0x4F05: Connection terminated by a network problem detected locally. • 0x4F06: Connection terminated by a network problem detected remotely. • 0x4F07: Connected terminated due to timeout in protocol. • 0x4F08: Incorrect parameter assignment: Connection to be established to local partner's own address. • 0x4F09: Connection temporarily reset by a call of the instruction "T_RESET". • 0x4F0A: Insufficient connection resources available (quantity exceeded) • 0x4F0B: Internal error: Incorrect addressing parameters • 0x4F0C: Internal CPU communication error • 0x4F0D: Internal AS communication error between CPU and CM/CP • 0x4F0E: The local TCP/UDP port (or RFC1006-T selector) specified is already in use.
LastDisconnTimeStamp	LDT	Time of the last connection termination.

T_CONFIG: Configure interface

Description T_CONFIG

Description

The "T_CONFIG" instruction is used for the program-controlled configuration of the integrated PROFINET interfaces of the CPU or the interface of a CP/CM.

You can change the Ethernet address of the PROFINET device name from the user program via the instruction. The existing configuration data is overwritten.

You can make the following changes via the instruction "T_CONFIG":

- Settings for the IP protocol
  - IP address
  - Subnet mask
  - Router address
- Settings for PROFINET
  - Assignment of the PROFINET device name

The settings correspond with the configuration options under "IP protocol" and "PROFINET" in the dialog "Ethernet addresses". This is displayed in the view "devices & networks" under the properties of the PROFINET interface.

Warning

Restart of CPU after execution of the "T_CONFIG" instruction

The CPU is restarted after you have executed the instruction to change an IP parameter. The CPU goes to STOP mode, a warm restart is carried out and the CPU starts up again (RUN mode).

Make sure that the control process is in a secure operating mode after the CPU has been restarted following execution of the instruction. Uncontrolled operation can result in serious material damage or personal injury due to malfunctions or programming errors, for example. Non-retentive data could be lost.

Requirement

In order to use the instruction must be explicitly specified in the hardware configuration that the assignment of IP address parameters and device name must be made by the user program.

- For this purpose open the properties of the PROFINET interface in the device view. Activate the following options in the dialog "Ethernet addresses":
  - In order to change the IP address parameters with "T_CONFIG": Select the setting "IP address is set directly at the device" under "IP protocol".
  - In order to change the PROFINET device name with "T_CONFIG": Select the setting "PROFINET device name is set directly at the device" under "PROFINET".
- The configuration data must be stored in the following system data types and assigned at the parameter CONF_DATA (Page 3703):
  - Store the IP address, subnet mask, and the router address in the system data type IF_CONF_V4.
  - Store the device name in the system type IF_CONF_NOS. Observe the restrictions that exist for assigning the name of the device (see parameter CONF_DATA (Page 3703)).

Functional description

The "T_CONFIG" instruction works asynchronously, which means its execution extends over multiple calls. The configuration process is started by calling "T_CONFIG" with REQ = 1. Only one job can be active at any time.

The block is edge triggered which means that after BUSY= FALSE the block must be re-called with REQ=FALSE to enable the instance.

Parameters

The following table shows the parameters of the "T_CONFIG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Calling the instruction with REQ = 1 starts processing of the instruction.
INTERFACE	Input	HW_INTERFACE	I, Q, M, D, L or constant	Hardware identifier of the interface The hardware ID is displayed in the properties of the interface in the device view, and in the system constants of the PLC tags.
CONF_DATA (Page 3703)	Input	VARIANT	D, L	Pointer to the parent structure that contains the system data types IF_CONF_HEADER, IF_CONF_V4 and IF_CONF_NOS (see description of the parameters CONF_DATA).
DONE (Page 3707)	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Processing not yet complete. • 1: Processing of instruction finished successfully.
BUSY (Page 3707)	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Processing of the instruction has not started, completed or canceled yet. • 1: Processing of instruction is in progress
ERROR (Page 3707)	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error • 1: Error

Parameter	Declaration	Data type	Memory area	Description
STATUS (Page 3707)	Output	DWORD	I, Q, M, D, L	Detailed status information: Detailed error and status information in the form of an error code are output at the parameter STATUS.
ERR_LOC (Page 3707)	Output	DWORD	I, Q, M, D, L	Error location: • 0: Error during execution of the instruction or when assigning parameters. • > 0: Errors in the structure or content of the configuration data at the parameter CONF_DATA.

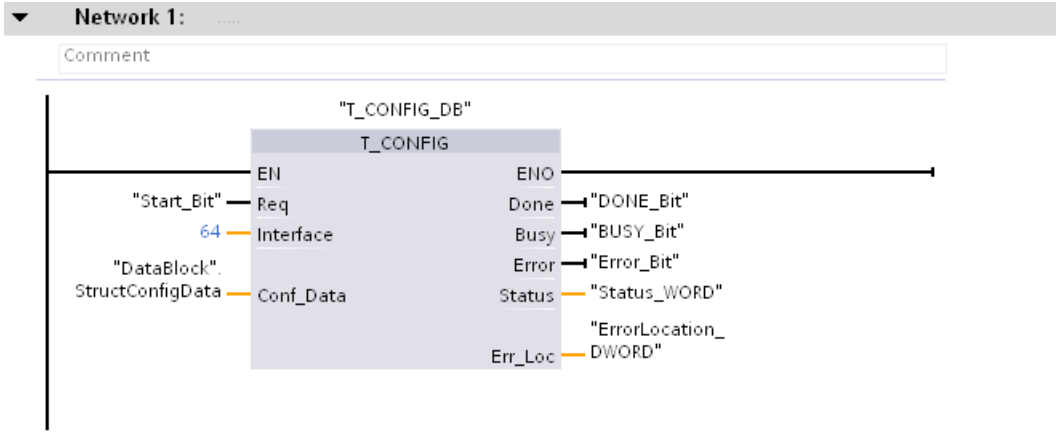
You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Example

In the following example the device name of an IO device is changed with the instruction "T_CONFIG".

Call of instruction

- The execution of the instruction is started with REQ=1.
- The hardware ID of the PROFINET interface is specified at the parameter Interface.
- The structure "StructConfigData" is referenced at the parameter CONF_DATA. The structure was created in the global data block "DataBlock".



Parameters CONF_DATA

The structure "StructConfigData" at the parameter CONF_DATA contains the following information:

- In the Header (IF_CONF_HEADER).
  SubfieldCount = 1: Indicates that only one additional structure (nos) is used below.
- In the structure "nos" (system data type IF_CONF_NOS)
  - Lenght = 11: Information on overall length of the structure NOS (5 bytes for the device name "myplc" + 6 bytes for the parameters Id, Length and Mode)
    Note: Instead of an absolute length, you can use the default start value (Lenght = 0) for a dynamic length.
  - Mode = 1: Permanent change of the device name in "myplc".
  - NOS[1] ... NOS[5]: Device name (1 character / byte)

DataBlock

	Name	Data type	Start value
1	Static		
2	StructConfigData	Struct	
3	Header	IF_CONF_Header	
4	FieldType	UInt	0
5	FieldId	UInt	0
6	SubfieldCount	UInt	1
7	nos	IF_CONF_NOS	
8	Id	UInt	40
9	Length	UInt	11
10	Mode	UInt	1
11	NOS	array [1..240] of Byte	
12	NOS[1]	Byte	'm'
13	NOS[2]	Byte	'y'
14	NOS[3]	Byte	'p'
15	NOS[4]	Byte	'l'
16	NOS[5]	Byte	'c'
17	NOS[6]	Byte	16#0
18	NOS[7]	Byte	16#0
19	NOS[8]	Byte	16#0
20	NOS[9]	Byte	16#0
21	NOS[10]	Byte	16#0
22	NOS[11]	Byte	16#0

See also

Evaluating errors with output parameter RET_VAL (Page 2195)

Parameter CONF_DATA

Structure of the configuration data

The configuration data at the parameter CONF_DATA can be stored in a global data block or in the section "Static" of the block interface.

The configuration data must be stored according to the following structure:

Name		Data type	Description
ConfData		Struct	Parent structure that is assigned at the parameter CONF_DATA.
	Header	IF_CONF_HEADER	The header is used to define the number of these system data types. The system data type IF_CONF_HEADER must always be included.
	IPData	IF_CONF_V4	The IP address, subnet mask and router address are stored in this system data type. Only create IF_CONF_V4 if you also want to change the Ethernet address with "T_CONFIG".
	NoS	IF_CONF_NOS	The PROFINET device names are stored in this system data type. Only create IF_CONF_NOS if you also want to change the device names with "T_CONFIG".

The system data types IF_CONF_HEADER, IF_CONF_V4 and IF_CONF_NOS are created by entering the name of the system data type in the column "data type" of the data block or the block interface. The name for the system data types can be freely assigned.

System data type IF_CONF_Header

Use the system data type IF_CONF_Header to specify the number of system data types IF_CONF_V4 and IF_CONF_NOS that are used during the execution of "T_CONFIG".

Byte	Parameter	Data type	Start value	Description
0 ... 1	FieldType	UINT	0	Field type: Must always have the value "0".
2 ... 3	FieldId	UINT	0	Error ID: Must always have the value "0".
4 ... 5	SubfieldCount	UINT	0	Number of system data types IF_CONF_V4 and IF_CONF_NOS: • 1: Only one of the system data types is used. • 2: Both system data types are used

System data type IF_CONF_V4

The IP address, subnet mask and router address are defined with the system data type IF_CONF_V4.

Byte	Parameter	Data type	Start value	Description
0 ... 1	Id	UINT	30	System data type ID The start value of this parameter may not be changed.
2 ... 3	Length	UINT	18	Length of the system data type IF_CONF_V4 The start value must be used for the length specification since the parameters of IF_CONF_V4 have a fixed length and structure.
4 ... 5	Mode	UINT	0	Validity of addressing: • 1: Permanent validity of the configuration data • 2: Temporary validity of the configuration data including the deletion of existing permanent configuration data
6 ... 9	InterfaceAddress	IP_V4 *	0.0.0.0	IP address
10 ... 12	SubnetMask	IP_V4 *	0.0.0.0	Subnet mask

Byte	Parameter	Data type	Start value	Description
14 ... 16	DefaultRouter	IP_V4 *	0.0.0.0	Router address
* The data type IP_V4 is a structure of 4 BYTE, which includes the respective address of the respective parameter (e.g. at parameter SubnetMask the four-digit address of the subnet mask of the IP protocol).				

Subfield IF_CONF_NOS

Use the subfield IF_CONF_NOS to specify the station name to be assigned during execution of the instruction "T_CONFIG".

Byte	Parameter	Data type	Start value	Description
0 ... 1	Id	UINT	40	System data type ID The start value of this parameter may not be changed.
2 ... 3	Length	UINT	246	Length of the system data type IF_CONF_NOS in bytes. • For an absolute length, the value for the Lenght parameter consists of: – 6 bytes for the parameter Id, Length and Mode. – up to 240 bytes for the device name (parameter NOS). Example: An overall length of 10 is the result for the device name "plc1" with a length of 4 characters (=4 bytes). • Use the default start value 246 at the Lenght parameter for a dynamic length. Make sure that you enter the value "0" after the name (see description for NOS parameter).

Byte	Parameter	Data type	Start value	Description
4 ... 5	Mode	UINT	0	Validity of the device name change: • 1: Permanent validity of the device name. • 2: Temporary validity of the device name.
6 ... 244	NOS	ARRAY [1...240] of Byte	0	Device name (Name of Station) • You must occupy the ARRAY from the first byte. The station name is deleted if "0" is assigned as the first byte. • The minimum length for the name is 2 bytes. The maximum length for a name is 240 bytes. • If the device name is shorter than specified at the parameter Length then a zero byte (16#0 hex) must be entered after the actual station name (according to IEC 61185-6-10). Otherwise NOS will be rejected and the instruction "T_CONFIG" outputs the error codeDW#16#C0809400 at the parameter STATUS. • If the device name is longer than specified at the parameter Length then the device name will only be written up to to specified length. The following restrictions apply for the device name: • The name must be specified in ASCII code. • Only lowercase letters, numbers, hyphen or dots may be used for the name. – The name may not begin or end with a hyphen. – The name may not begin with numbers. – The name may not have the form n.n.n.n (n = 0, ... 999). – The name may not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9). • A name component between two points may be up to 63 characters long. • No special characters such as umlauts, brackets, underscore, slash, blank etc. The error code C080_9400 is output at the parameter STATUS if an invalid character is used.

DONE, BUSY and ERROR parameters

Description

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current status of the instruction and when the transfer of configuration data is concluded.

BUSY	DONE	ERROR	Description
TRUE	FALSE	FALSE	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the parameter STATUS (Page 3707).
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Parameter STATUS and ERR_LOC

Description

The status and error messages of the instruction "T_CONFIG" are output at the parameters STATUS and ERR_LOC.

- The cause of the error is output at the parameter STATUS.
- The location of the error that occurred is output at the parameter ERR_LOC. The following options are available here:
  - 16#0000_0000: Error when calling the instruction (e.g. errors when assigning parameters to the instruction or in communication with the PROFINET interface).
  - 16#0001_0000: Error with the configuration data in the parameters of the system data type IF_CONF_HEADER.
  - 16#0001_0001: Error with the configuration data in the parameters of the system types IF_CONF_V4 or IF_CONF_NOS.

The following table shows the possible values for the parameters STATUS and ERR_LOC:

STATUS*	ERR_LOC*	Explanation
0000_0000	0000_0000	Order processing completed without errors.
0070_0000	0000_0000	No job processing active.
0070_0100	0000_0000	Start of the order processing.
0070_0200	0000_0000	Intermediate call (REQ irrelevant).
C08x_yy00	0000_0000	General error information. See also: GET_ERR_ID: Get error ID locally (Page 2417)
C080_8000	0000_0000	Error at call of the instruction: The hardware ID at the parameter Interface is invalid.
C080_8100	0000_0000	Error at call of the instruction: The hardware ID at the parameter Interface does not address a PROFINET interface.
C080_8700	0000_0000	Error at call of the instruction: Incorrect length of the data block at the parameter CONF_DATA.

STATUS*	ERR_LOC*	Explanation
C080_8800	0001_0000	Error in the system data type IF_CONF_HEADER: The parameter FieldType has an invalid value. Use the value "0" for FieldType.
C080_8900	0001_0000	Error in the system data type IF_CONF_HEADER: The parameter FieldId has an invalid value or was used several times. Use the value "0" for FieldId.
C080_8A00	0001_0000	Error in the system data type IF_CONF_HEADER: Incorrect number at the parameter SubfieldCount. Enter the correct number of system data types IF_CONF_V4 and IF_CONF_NOS in use.
C080_8B00	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: The parameter Id has an invalid value. Use "30" and IF_CONF_NOS "40" for IF_CONF_V4.
C080_8C00	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: Incorrect data type system used, wrong order or multiple use of a system data type.
C080_8D00	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: The parameter Length has an incorrect or invalid value.
C080_8E00	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: The parameter Mode has an incorrect or invalid value. Only the values "1" (permanent) or "2" (temporary) are valid.
C080_9000	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: Configuration data cannot be applied. Possible cause: The setting "Set IP address on device" or "Set PROFINET device name on device" was not selected in the hardware configuration.
C080_9400	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: A parameter value is undefined or invalid.
C080_9500	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: The values of two parameters are inconsistent.
C080_C200	0000_0000	Error at call of the instruction: The configuration data can not be transferred. Possible cause: The PROFINET interface is not accessible.
C080_C300	0000_0000	Error at call of the instruction: Insufficient resources (e.g., multiple calling of "T_CONFIG" with different parameters).
C080_C400	0000_0000	Error at call of the instruction: Temporary communication error. Time indication for change to daylight saving time.
C080_D200	0000_0000	Error at call of the instruction: Call not possible. Instruction is not supported by the selected PROFINET interface.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Differences in Open User Communication libraries V3.1 and V4.0

New library version OUC V4.0

Introduction

CPU S7-1200 as of firmware version 4.1 supports new instruction versions for Open User Communication (OUC). The new instructions are included in the library with version 4.0.

If you want to use this library, changes must be made to the application program, since some of the new instructions for Open User Communication behave differently.

This section describes the differences in detail, especially in the call behavior of the instructions.

Note

Description only relevant when upgrading from the CPU S7-1200 ≤ V4.0 to S7-1200 ≥ V4.1

If you are using an S7-1500 CPU, changes between the library versions are not relevant. The same applies if you are using an S7-1200 ≥ Version 4.1 and do not convert the library for the Open User Communication to Version 4.0.

Differences between version 3.1 and 4.0 of the Open User Communication library

The following table shows the instructions in the Open User Communication library that differ between version 3.1 and 4.0. Click on the name of the instruction for detailed information.

Instruction	Version in library V3.1 (CPU FW ≤ V4.0)	Version in library V4.0 (CPU FW ≥ V4.1)
TSEND_C (Page 3709)	V2.1	V3.0
TRCV_C (Page 3711)	V2.1	V3.0
TMAIL_C *	V2.1	V3.0
TCON (Page 3713)	V3.0	V4.0
TDISCON	V2.1	V2.1 (identical to library V3.1)
TSEND (Page 3715)	V3.0	V4.0
TRCV (Page 3717)	V3.0	V4.0
TUSEND (Page 3715)	V3.0	V4.0
TURCV (Page 3717)	V3.0	V4.0
T_RESET *	V1.1	V1.2
T_DIAG *	V1.1	V1.2
T_CONFIG	V1.0	V1.0 (identical to library V3.1)
MB_CLIENT (Page 3720)	V3.1	V4.0
MB_SERVER (Page 3720)	V3.1	V4.0
* There are no differences in the versions that affect the user program.		

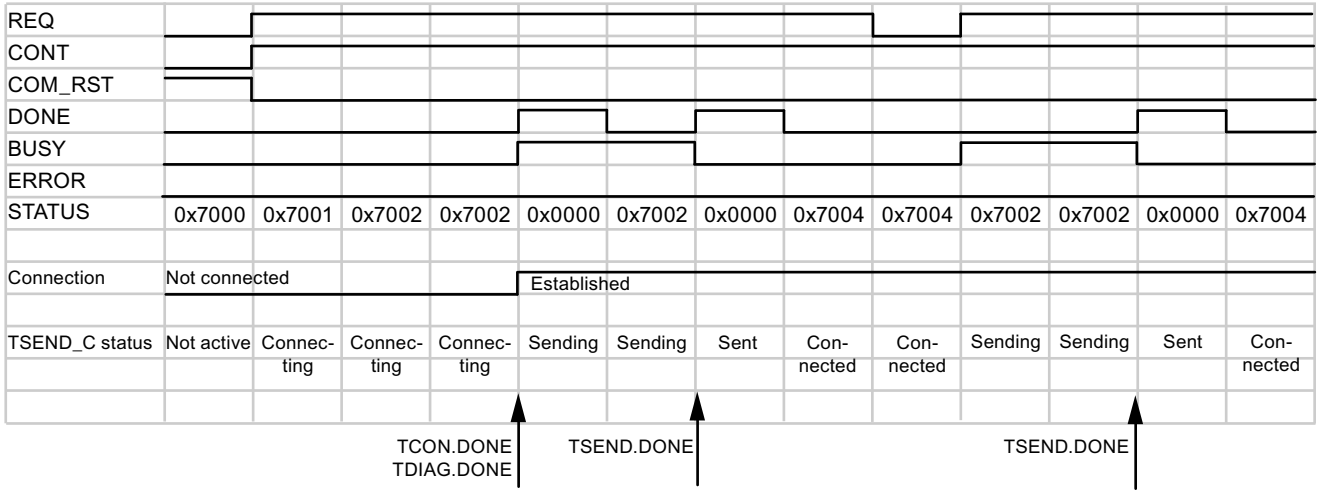
Changed behavior of instructions

Changes with the TSEND_C instruction

Call behavior of TSEND_C (V<3.0)

Up to version 2.1 of the TSEND_C instruction, the DONE output parameter is set twice: Once when a connection is established by the internally used TCON instruction and then after a send operation by the internally used TSEND instruction.

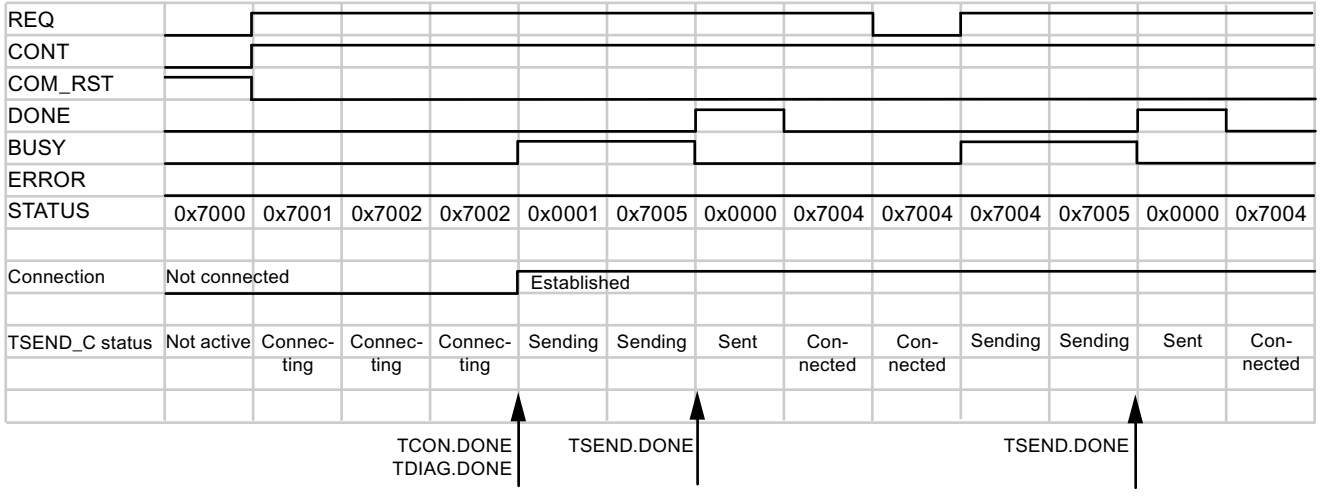
The following diagram shows the connection establishment and the sending of data with TSEND_C V2.1:



When setting the COM_RST parameter to "1", the current connection establishment or a current data transmission can be reset at any time. This terminates the existing communication connection and a new connection is established.

Call behavior of TSEND_C (V≥3.0)

As of the TSEND_C version, the DONE parameter is only set when data transfer has been completed by the internally used TSEND instruction (STATUS = 0000).



The existing connection is temporarily interrupted and reset when the COM_RST parameter is set to "1". However, unlike TSEND_C V2.1, the connection endpoint is retained.

Note

Additional protocols with TSEND_C as of version 3.0

Version 3.0 of the TSEND_C instruction also supports UDP and UDP Broadcast via the interface of the CPU and via CM/CP.

See also

TSEND_C: Send data via Ethernet (Page 3625)

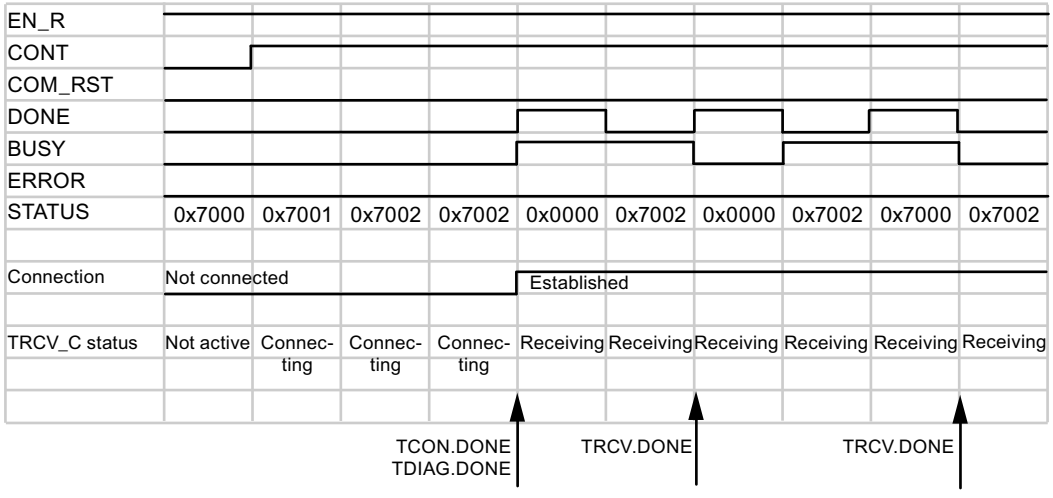
TSEND_C: Send data via Ethernet (Page 3629)

Changes with the TRCV_C instruction

Call behavior of TRCV_C (V<3.0)

Up to version 2.1 of the TRCV_C instruction, the DONE output parameter is set after the connection is established. The STATUS output parameter does not distinguish whether the connection or the data transfer has been completed.

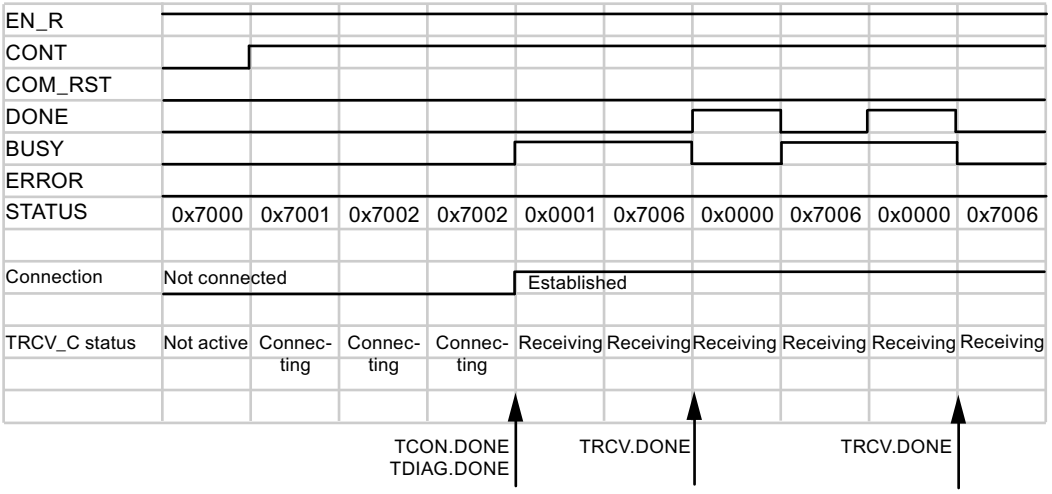
The following diagram shows the connection establishment and the sending of data with TRCV_C V2.1:



The current connection establishment or a current data transmission can be reset at any time by setting the COM_RST parameter to "1". This terminates the existing communication connection and a new connection is established.

Call behavior of TRCV_C (V≥3.0)

As of the TRCV_C version, the DONE parameter is only set when data transfer has been completed by the internally used TRCV instruction (STATUS = 0000). The completion of the connection establishment by the internally used TCON instruction is displayed by the output value 0x0001 at the STATUS parameter.



The existing connection is temporarily interrupted and reset when the COM_RST parameter is set to "1". However, unlike TSEND_C V2.1, the connection endpoint is retained.

Note

Additional protocols with TRCV_C as of version 3.0

Version 3.0 of the TRCV_C instruction also supports UDP and UDP Broadcast via the interface of the CPU and via CM/CP.

Note

ADHOC mode for the TCP protocol variant

In accordance with the TRCV instruction, with TRCV_C up to version 2.1, ADHOC mode is activated by assigning the value "0" to the LEN parameter. As of version 3.0 of the instruction, you use the ADHOC parameter for this. You can find detailed information in the descriptions of the instructions.

See also

Changes with the TRCV/TURCV instructions (Page 3717)

TRCV_C: Receive data via Ethernet (Page 3636)

TRCV_C: Receive data via Ethernet (Page 3641)

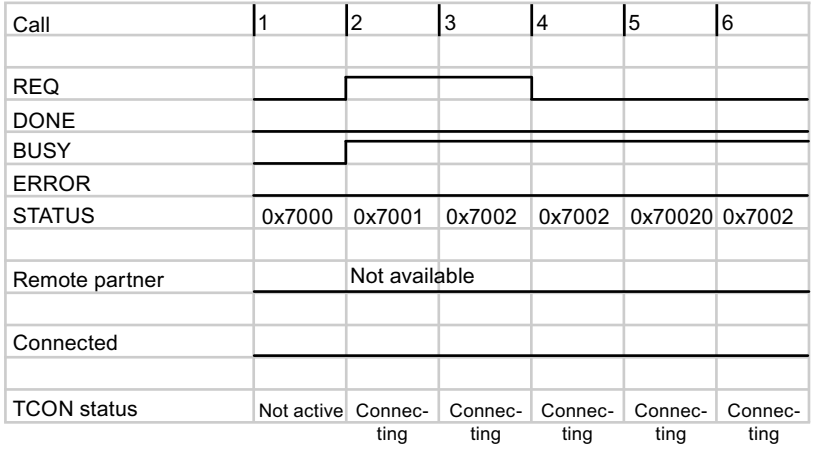
Changes with the TCON instruction

Changed call behavior with connection errors

Previous call behavior of TCON (V<4.0)

- The previous TCON instruction (V<4.0) is activated with a rising edge at the REQ input parameter.
- If the remote communication partner cannot be reached, the instruction sets the BUSY output parameter.
- An error message is not output.

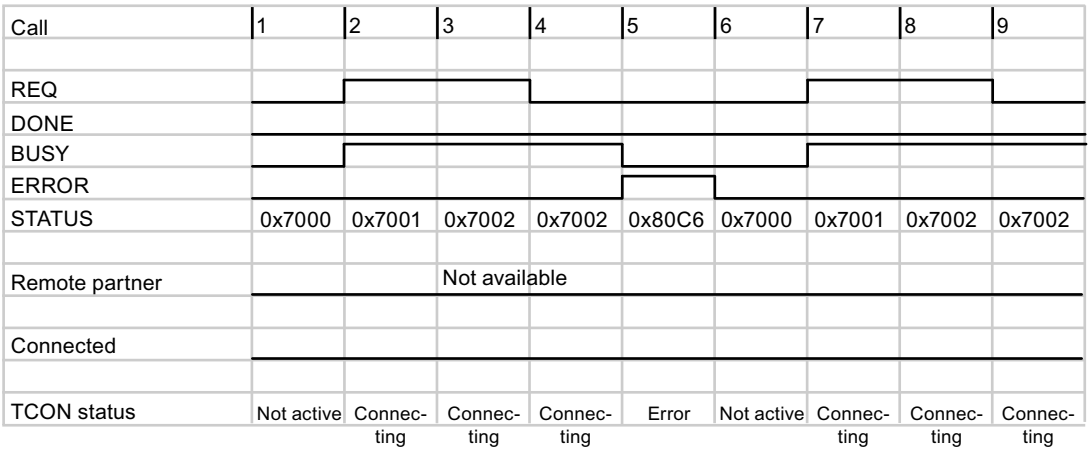
The following diagram shows the behavior of TCON on the part of the active connection partner. If no connection is made, the instruction is not called again.



New call behavior of TCON (V≥4.0)

- The new TCON instruction is also activated with a rising edge at the REQ input parameter.
- Unlike before, an error message is output when the remote communication partner cannot be reached. The error can be queried and the call started again by a new edge at the REQ parameter.
- The following changes result for the user program:
  - Additional error messages of TCON that can be evaluated (see description of TCON (Page 3664)).
  - The connection establishment can be re-initialized via a new rising edge. To do this, query the DONE and ERROR parameters to ensure that an error is present.

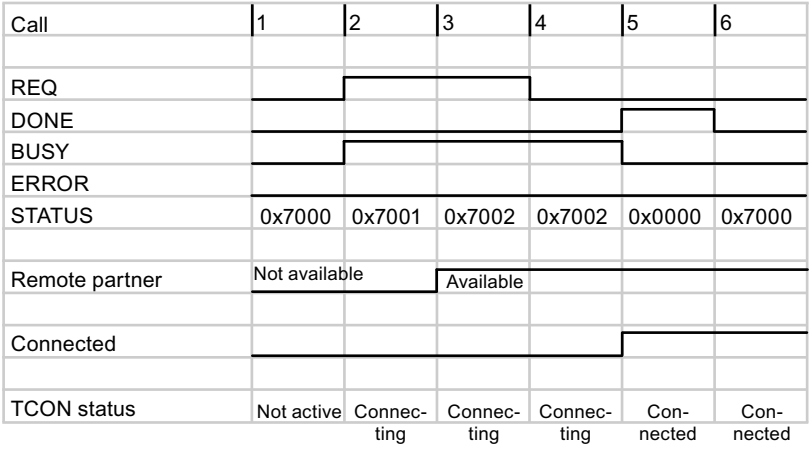
The following diagram shows the behavior of TCON on the part of the active connection partner. The instruction is called again after a network error (error code 80C6).



Call behavior without communication errors

If there is no connection error, the TCON versions have the same behavior.

The following diagram shows the behavior of TCON on the part of the active connection partner.



See also

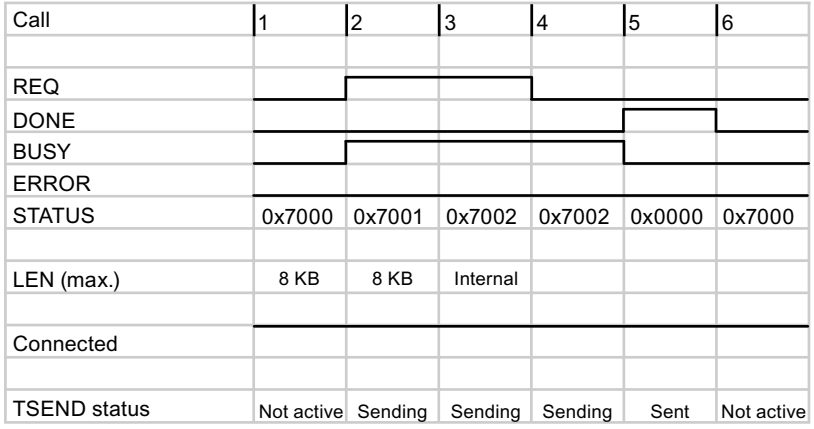
TCON: Establish communication connection (Page 3662)

Changes with the TSEND/TUSEND instructions

Send operation with an established connection

If no delay or interruption occurs during sending, versions 3.0 and 4.0 of the TSEND/TUSEND instructions behave the same way:

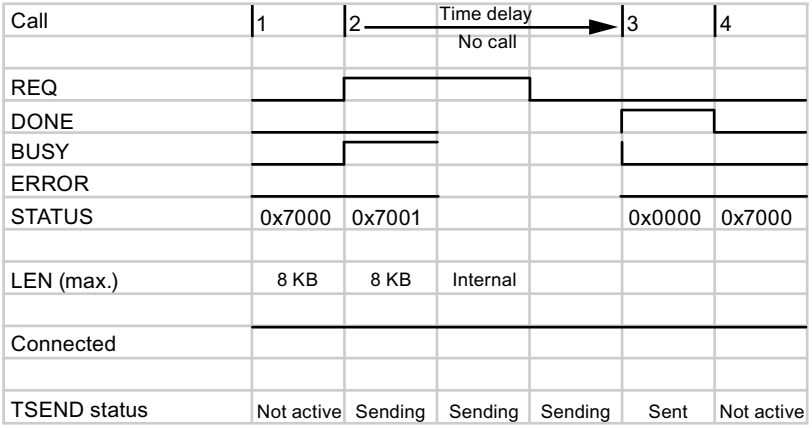
- The instruction is started with a positive edge at the "REQ" parameter. The instruction is executed asynchronously, i.e. it must be called until the full execution is indicated by the DONE parameter.
- A maximum of 8 KB of data can be sent with TSEND. A maximum of 1472 bytes can be sent with TUSEND. The amounts are based on the full execution of the instruction, i.e. all required calls until the DONE parameter is set.
- Data transmission takes place in three steps:
  - The data is written from the operand area to an internal buffer (indicated by STATUS=7001).
  - The transmission to the remote communication partner follows.
  - If the transmission is successful, DONE is set to "1" and STATUS is reset to "0" (see call 5 in the diagram below).



Send operation with time-delayed call

If the send operation is not performed via a communication module (CM) or a communication processor (CP), versions 3.0 and 4.0 of the TSEND/TUSEND instructions behave the same way:

- In the following example, TSEND or TUSEND are only called once with a rising edge at the REQ parameter (see call 2 in the diagram below).
- If the time delay is sufficiently long and the data could be transferred, the DONE parameter is set directly to "1" in the next call (call 3 here).

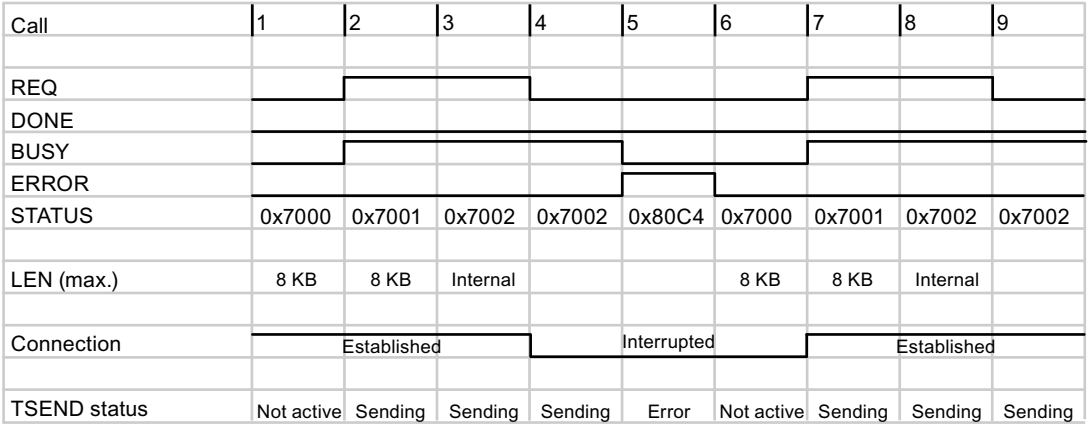


If the send operation is performed via a CM/CP, version 4.0 of the TSEND/TUSEND instructions behaves differently. In this case, the instruction must be called multiple times until the receipt of data is confirmed by the NDR parameter of the TRCV/TURCV instructions.

Send operation with interrupted connection

If the connection is interrupted during the send operation, the ERROR and STATUS parameters indicate the error and its cause. In the following example, the connection was interrupted during call number 5 and this error is indicated accordingly with the STATUS parameter.

As of version 4.0 of the TSEND/TUSEND instructions, more STATUS alarms are available that can be correspondingly evaluated (see description of TSEND (Page 3674)/TUSEND (Page 3687)) .



See also

TSEND: Send data via communication connection (Page 3671)

Changes with the TRCV/TURCV instructions

Receive operation with an established connection

If no delay or interruption occurs during sending, version 3.0 and 4.0 of the TRCV/TURCV instructions behave the same way:

- A maximum of 8 KB of data can be sent with TRCV. A maximum of 1472 bytes can be received with TURCV. The amounts are based on the full execution of the instruction, i.e. all required calls until the DONE parameter is set.
- The instruction receives data if the EN_R parameter is set to "1".
- The data reception is not complete until the length of data specified at the the LEN parameter has been completely received. The length of the received data is output at the RCVD_LEN output parameter. Only then does the data in the area defined at the DATA parameter become available.

Call	1	2	3	4	5	6
EN_R						
LEN (max.)	8 KB	8 KB	Internal			
NDR						
BUSY						
ERROR						
STATUS	0x7000	0x7001	0x7002	0x7002	0x0000	0x7000
RCVD_LEN	0	0	0	0	max. 8 KB	0
Connection				Established		
TRCV status	Not active	Receiving	Receiving	Receiving	Receiving	Not active

Receive operation with time-delayed call

If the send operation is not performed via a communication module (CM) or a communication processor (CP), versions 3.0 and 4.0 of the TRCV/TURCV instructions behave the same way:

- In the following example, TRCV or TURCV are only called once with a rising edge at the EN_R parameter (see call 2 in the diagram below).
- If the time delay is sufficiently long and the data could be transferred, the NDR parameter is set directly to "1" in the next call (call 3 here).

Call	1	2	Delay / No call		3	4
EN_R						
LEN (max.)	8 KB	8 KB	Internal			
NDR						
BUSY						
ERROR						
STATUS	0x7000	0x7001			0x0000	0x7000
RCVD_LEN	0	0	0	0	max. 8 kB	0
Connection	Established					
TRCV status	Not active	Receiving	Receiving	Receiving	Receiving	Not active

If the send operation is performed via a CM/CP, version 4.0 of the TSEND/TUSEND instructions behaves differently. In this case, the instruction must be called multiple times until the receipt of data is confirmed by the NDR parameter.

Receive operation with interrupted connection

If the connection is interrupted during receiving, the ERROR and STATUS parameters indicate the error and its cause. In the following example, call number 5 resulted in a temporary communication error, which is accordingly indicated by the STATUS parameter.

Call	1	2	3	4	5	6	7	8	9
EN_R									
LEN									
NDR									
BUSY									
ERROR	0x7000	0x7001	0x7002	0x7002	0x80C4	0x7000	0x7001	0x7002	0x7002
STATUS	8 KB	8 KB	Internal			8 KB	8 KB	Internal	
RCVD_LEN								Established	
Connection	Established								
TRCV state	Not active	Sending	Sending	Sending	Error	Not active	Sending	Sending	Sending

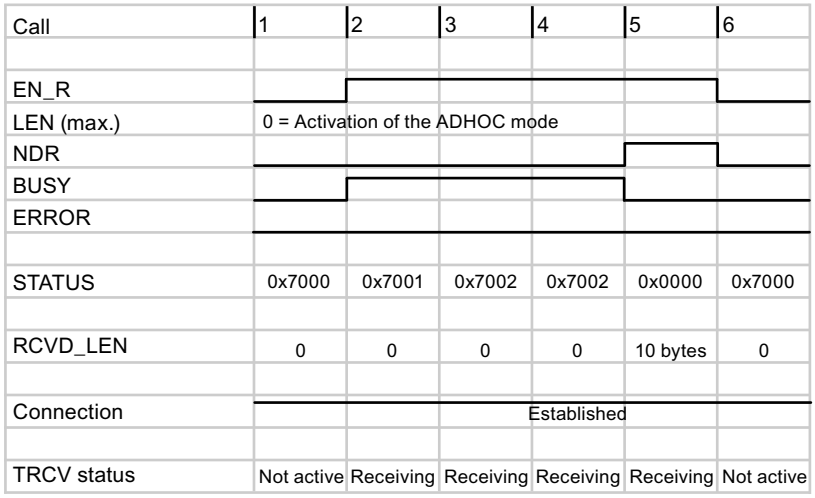
As of version 4.0 of the TRCV/TURCV instructions, more STATUS alarms are available that can be correspondingly evaluated (see description of TRCV (Page 3681)/TURCV (Page 3690)).

Receive operation using ADHOC mode

ADHOC mode is only available with the TCP protocol variant. You can use ADHOC mode to receive data of variable length with the TRCV/TURCV instruction. If ADHOC mode is active, the receipt of data is confirmed at the NDR parameter when at least one byte has been transferred.

Data reception in ADHOC mode with TRCV < 3.0 (S7-1200 < V4.0)

In the older version of TRCV (Page 3677), ADHOC mode was activated by setting the LEN parameter to "0". In the following example, 10 bytes of data are transferred with call 5.



Data receipt in ADHOC mode with TRCV ≥ 3.0 (S7-1200 ≥ V4.0 or S7-1500)

As of version 3.0 of the TRCV instruction, ADHOC mode is activated with its own parameter (ADHOC).

Call	1	2	3	4	5	6
EN_R						
LEN	Max. 8 KB	Max. 8 KB	Internal			
ADHOC						
NDR						
BUSY						
ERROR						
STATUS	0x7000	0x7001	0x7002	0x7002	0x0000	0x7000
RCVD_LEN	0	0	0	0	10 bytes	0
Connection			Established			
TRCV status	Not active	Receiving	Receiving	Receiving	Receiving	Not active

If an error occurs during execution, as of version 4.0 of the TRCV instructions, there are more STATUS alarms available which can be evaluated accordingly.

If the send operation is performed via a CM/CP, version 4.0 of the TRCV/TURCV instructions behaves differently. In this case, the instruction must be called multiple times until the receipt of data is confirmed by the NDR parameter.

Changes with the MB_SERVER/MB_CLIENT instructions

Differences between versions 3.1 and 4.0 of the Modbus instructions

The following differences between versions exist with the MB_SERVER/MB_CLIENT MODBUS instructions:

- Address parameter
  - In version 3.1, the address data for the Modbus TCP server is specified via the input parameters "IP_x".
  - Version 4.0 uses TCON_IP_V4 and TCON_Configured system data types at the CONNECT input parameter for this purpose.
- If an error occurs during execution, as of version 4.0 of the Modbus instructions, there are more STATUS alarms available which can be evaluated accordingly.

You can find detailed information in the instruction descriptions for MB_CLIENT (V3.1) (Page 3770) and MB_CLIENT (V4.0) (Page 3788) and in the descriptions for MB_SERVER (V3.1) (Page 3778) and MB_SERVER (V4.0) (Page 3800).

11.6.5.3 Web server

WWW: Synchronizing user-defined web pages

Description

The instruction WWW initializes the Web server of the CPU or synchronizes user-defined web pages with the user program in the CPU.

User-defined web pages together with the Web server make it possible for the CPU to access freely designed web pages of the CPU with a web browser.

Use script instructions (such as Javascript) and HTML code in user-defined web pages to transfer data via a web browser for further processing to the CPU and to display data from the operand area of the CPU in the web browser. Call the WWW instruction in the user program for synchronization of the user program and the Web server as well as initialization.

Initialization

User-defined web pages are "packaged" in data blocks for processing by the CPU. You will have to generate appropriate data blocks from the source files (HTML files, screens, Javascript files, ...) during configuration. The Web Control DB takes on a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. The data blocks with coded web pages are called fragment DBs.

When the data block is downloaded into the CPU, the CPU does not "know" that user-defined web pages are coded inside it. The instruction "WWW" in the startup OB, for example, will inform the CPU which DB is the Web Control DB. The user-defined web pages can be accessed via a web browser after this initialization.

Synchronization

If you want the user program to interact with the user-defined web pages, then the instruction WWW must be used in the cyclical program part.

Examples of interaction between user program and web page:

- Check received data
- Assemble and send back data to the web browser making the request

In this case it must be possible to evaluate the current status information and the Web server must receive control information, such as release of a web page requested by a web browser.

Parameter

The following table shows the parameters of the instruction "WWW":

Parameter	Declaration	Data type	Memory area	Description
CTRL_DB	Input	DB_WWW	I, Q, M, D, L or constant	Data block that describes the user-defined web pages (Web Control DB)
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to Overview of the valid data types (Page 1908).

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred. There are no web page requests that have to be released by the user program.
00xy	x: indicates whether an error has occurred during initialization of the Web Control DB (CTRL_DB): x=0: No errors occurred. x=1: Error occurred. The error is coded in the byte "CTRL_DB.last_error" of the Web Control DB, see description of Web Control DB. y: Number of the pending request. Several requests are possible (e. g. requests "0" and "1" are pending: y="3". y="1": Request "0" y="2": Request "1" y="4": Request "2" y="8": Request "3"
803A	The specified Web Control DB does not exist on the CPU.
8081	Incorrect version or incorrect format of the Web Control DB.
80C1	There are no resources to initialize the web application, for example, because only two or four web applications may be running.

See also

Overview of the valid data types (Page 1908)

11.6.5.4 Communications processor

Point-to-point

PORT_CFG: Configure communication parameters dynamically

Description

The instruction "PORT_CFG" allows dynamic configuration of communications parameters for a point-to-point communications port.

You set up the original static configuration of the port in the hardware configuration. You can change this configuration by executing the "PORT_CFG" instruction. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it.

With "PORT_CFG" you can influence the following communications parameter settings:

- Parity
- Baud rate
- Number of bits per character
- Number of stop bits
- Type and properties of flow control

The changes made by the "PORT_CFG" instruction are not stored permanently on the target system.

You can transfer serial data via the electrical connections RS-232 (half and full duplex) and RS-485 (half duplex).

Parameters

The following table shows the parameters of the "PORT_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the configuration change on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
PROTOCOL	Input	UINT	I, Q, M, D, L or constant	Transmission protocol: • 0: Point-to-point communication protocol • 1..n: Future definition for specific transmission protocols
BAUD	Input	UINT	I, Q, M, D, L or constant	Baud rate of the port: • 1: 300 baud • 2: 600 baud • 3: 1200 baud • 4: 2400 baud • 5: 4800 baud • 6: 9600 baud (default) • 7: 19200 baud • 8: 38400 baud • 9: 57600 baud • 10: 76800 baud • 11: 115200 baud
PARITY	Input	UINT	I, Q, M, D, L or constant	Parity of the port: • 1: No parity (default) • 2: Even parity • 3: Odd parity • 4: Mark parity • 5: Space parity

Parameter	Declaration	Data type	Memory area	Description
DATABITS	Input	UINT	I, Q, M, D, L or constant	Bits per character: • 1: 8 bits per character (default) • 2: 7 bits per character
STOPBITS	Input	UINT	I, Q, M, D, L or constant	Number of stop bits: • 1: 1 stop bit (default) • 2: 2 stop bits
FLOWCTRL	Input	UINT	I, Q, M, D, L or constant	Data flow control: • 1: None (default) • 2: XON/XOFF • 3: Hardware flow control (RTS always activated) • 4: Hardware flow control (RTS can be deactivated during transmission)
XONCHAR	Input	CHAR	I, Q, M, D, L or constant	Indicates the character used as XON character. The character DC1 (11H) is set as default.
XOFFCHAR	Input	CHAR	I, Q, M, D, L or constant	Indicates the character used as XOFF character. The character DC3 (13H) is set as default.
WAITTIME	Input	UINT	I, Q, M, D, L or constant	Specifies the wait time for XON or CTS after the start of the transmission. The specified value must be greater than 0. 2000 milliseconds are set as default.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

STATUS parameter

Error code* (W#16#...)	Description
80A0	The specified protocol is invalid.
80A1	The specified baud rate is invalid.
80A2	The specified parity rate is invalid.
80A3	The specified number of bits per character is invalid.
80A4	The specified number of stop bits is invalid.
80A5	The specified type of flow control is invalid.
80A6	Incorrect value at the WAITTIME parameter When the data flow control is enabled, the value at the WAITTIME parameter must be greater than zero.

Error code* (W#16#...)	Description
80A7	Invalid values at XONCHAR and XOFFCHAR parameters.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 3741)".

SEND_CFG: Configure serial transmission parameters dynamically

Description

The instruction "SEND_CFG" allows dynamic configuration of serial transmission parameters for a point-to-point communications port. All the messages waiting for transfer are discarded after execution of SEND_CFG.

You set up the original static configuration of the port in the hardware configuration. You can change this configuration by executing the "SEND_CFG" instruction. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it. With "SEND_CFG" you can influence the following transmission parameter settings:

- Time between the activation of RTS (Request to Send) and the start of the transmission
- Time between the end of transmission and the deactivation of RTS
- Define bit times for breaks

The changes made by the "SEND_CFG" instruction are not stored permanently on the target system.

You can transfer serial data via the electrical connections RS-232 (half and full duplex) and RS-485 (half duplex).

Parameters

The following table shows the parameters of the "SEND_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the configuration change on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
RTSONDLY	Input	UINT	I, Q, M, D, L or constant	The time that should elapse after activating RTS until the start of transmission. Valid values for this parameter are as follows: • 0 (default) • 0 to 65535 ms in steps of 1 ms This parameter does not apply to RS-485 modules.

Parameter	Declaration	Data type	Memory area	Description
RTSOFFDLY	Input	UINT	I, Q, M, D, L or constant	Time that should elapse after the end of transmission until deactivation of RTS. Valid values for this parameter are as follows: • 0 (default) • 0 to 65535 ms in steps of 1 ms This parameter does not apply to RS-485 modules.
BREAK	Input	UINT	I, Q, M, D, L or constant	Specifies the bit times for a break, which are sent at the start of the message. 12 bit times are set as default. A maximum of 25000 bit times can be specified.
IDLELINE	Input	UINT	I, Q, M, D, L or constant	Specifies the bit times for idle line after the break at the start of the message. 12 bit times are set as default. A maximum of 25000 bit times can be specified.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

STATUS parameter

Error code* (W#16#...)	Description
80B0	The configuration of a transmission interruption is not permitted.
80B1	The specified break time exceeds the permitted maximum of 25000 bit times.
80B2	The specified time for idle line exceeds the permitted maximum of 25000 bit times.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 3741)".

RCV_CFG: Configure serial receive parameters dynamically

Description

The instruction "RCV_CFG" allows dynamic configuration of serial receive parameters for a point-to-point communications port. You can use this instruction to configure the conditions that specify the start and end of the message to be transmitted. The receipt of messages that correspond to these conditions can be enabled by the "RCV_PTP (Page 3737)" instruction.

You set up the original static configuration of the port in the properties of the hardware configuration. Execute the "RCV_CFG" instruction in your program to change the configuration. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it. The changes made by the "RCV_CFG" instruction are not stored permanently on the target system.

All the messages waiting for transfer are discarded after execution of the "RCV_CFG" instruction.

Parameters

The following table shows the parameters of the "RCV_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the configuration change on a rising edge.
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
CONDITIONS	Input	CONDITIONS	D, L	Data structure defining the conditions for start and end of data transmission.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Data type CONDITIONS

You can use the CONDITIONS structure to define the start and end conditions for the message transmission. The structure CONDITIONS is included in the instance DB of the "RCV_CFG" instruction. Use the structure CONDITIONS to define the start and end conditions when the transmission of a message is complete and when the next message transfer is to start:

- You define the start conditions for the data transfer in the START structure.
- You define the end conditions for the data transfer in the END structure.

You can define one or more start and end conditions for this. If you specify multiple start or end conditions these are linked by an OR logic instruction.

The following table shows the "CONDITIONS" structure:

Parameters		Data type	Description
START		STRUCT	Start conditions
	STARTCOND	UINT	Specifies the start condition (details, see below). The start condition can be specified as a 16-bit hexadecimal value. Possible values for the start condition are: • 1: Start character • 2: Any character (default) • 4: Line break • 8: Idle line • 16: Character string 1 • 32: Character string 2 • 64: Character string 3 • 128: Character string 4 Multiple start conditions can also be defined at the STARTCOND parameter. The total from the values of the individual conditions is specified for this. If, for example, you want to define "Idle line" OR "Character string 1" OR "Character string 4" as start condition, the value "152" must be specified.
	IDLETIME	UINT	Specifies the maximum idle time of the line before receipt is started. Valid values for this parameter are as follows: • 40 bit times (default) • 0 to 2500 bit times
	STARTCHAR	BYTE	Specifies the start character. This setting is only enabled when the configured start condition is "Start character". Valid values for this parameter are as follows: • 02 (STX): Default setting • B#16#00 to B#16#FF
	SEQ[1].CTL	BYTE	Character string 1: Control sequence for each character You can use the bit position of the character to define which characters of the character string will be considered or ignored. To evaluate the characters, the corresponding bits have to be set. • Bit 0: 1 character • Bit 1: 2 characters • Bit 2: 3 characters • Bit 3: 4 characters • Bit 4: 5 characters A character is ignored when the corresponding bit is reset.
	SEQ[1].STR	CHAR[5]	Character string 1: Start character (5 characters)
	SEQ[2].CTL	BYTE	Character string 2: Ignore/compare control sequence for each character
	SEQ[2].STR	CHAR[5]	Character string 2: Start character (5 characters)
	SEQ[3].CTL	BYTE	Character string 3: Ignore/compare control sequence for each character
	SEQ[3].STR	CHAR[5]	Character string 3: Start character (5 characters)
	SEQ[4].CTL	BYTE	Character string 4: Ignore/compare control sequence for each character
	SEQ[4].STR	CHAR[5]	Character string 4: Start character (5 characters)

Parameters	Data type	Description
END	STRUCT	End conditions

Parameters		Data type	Description
	ENDCOND	UINT	Specifies the end condition (details, see below). The end condition can be specified as a 16-bit hexadecimal value. Possible values for the end condition are: • 1: Reply timeout • 2: Message timeout • 4: Timeout within the character string • 8: Maximum length • 16: N+LEN+M; the information on the message length in integrated in the message and will be evaluated. • 32: Character string 1 Multiple end conditions can also be defined at the ENDCOND parameter. The total from the values of the individual end conditions is specified for this. If, for example, you want to define the end condition "Maximum length" OR "Sequence 1", the value "40" must be specified.
	MAXLEN	UINT	Specifies the maximum number of characters in a message. Valid values* for this parameter are as follows: • 1 character (default) • 0 to 1024 characters This setting is only enabled if the "Maximum length" end condition is set at the ENDCOND parameter.
	N	UINT	Offset of the length field in the message Valid values for this parameter are as follows: • 0 characters (default) • 0 to 1024 characters This setting is only enabled if the "N+LEN+M" end condition is set at the END-COND parameter.
	LENGTHSIZE	UINT	Size of the length field in bytes Valid values* for this parameter are as follows: • 0 bytes (default) • 1 byte • 2 bytes • 4 bytes This setting is only enabled if the "N+LEN+M" end condition is set at the END-COND parameter.
	LENGTHM	UINT	Specifies the number of end characters that follow the length field but are not contained in the length of the message. Valid values for this parameter are as follows: • 0 characters (default) • 0 to 255 characters This setting is only enabled if the "N+LEN+M" end condition is set at the END-COND parameter.
	RCVTIME	UINT	Specifies the maximum duration for the receipt of the first character of a message. Valid values for this parameter are as follows: • 200 ms (default)

Parameters		Data type	Description
			• 0 to 65535 ms in steps of 1 ms This setting is only enabled if the "Reply timeout" end condition is set at the ENDCOND parameter.
	MSGTIME	UINT	Specifies the maximum duration of the receipt of a message. Valid values for this parameter are as follows: • 200 ms (default) • 0 to 65535 ms in steps of 1 ms This setting is only enabled if the "Message timeout" end condition is set at the ENDCOND parameter.
	CHARGAP	UINT	Specifies the time interval between received consecutive characters. Valid values for this parameter are as follows: • 12 bit times (default) • 0 to 2500 bit times This setting is only enabled if the "Timeout within the character string" end condition is set at the ENDCOND parameter.
	SEQ.CTL	BYTE	Character string: Control sequence for each character You can use the bit position of the character to define which characters of the character string will be considered or ignored. To evaluate the characters, the corresponding bits have to be set. • Bit 0: 1 character • Bit 1: 2 characters • Bit 2: 3 characters • Bit 3: 4 characters • Bit 4: 5 characters A character is ignored when the corresponding bit is reset.
	SEQ.STR	CHAR[5]	Character string: Start character (5 characters)
	* These value ranges also apply to the corresponding hardware settings for specifying the end of message.		

Start conditions for the message receipt (STARTCOND parameter)

The start of the message is recognized by the receiver if a configured start condition applies. The following conditions can be defined as start conditions for message receipt:

- Start character: The start of a message is recognized when a certain character occurs. This character is stored as first character of the message. All characters received before the start character are rejected.
- Any character: Any character can define the start of a message. This character is stored as first character of the message.
- Line break: The start of a message is recognized if the received data stream is interrupted for longer than one character length.

- Idle line: The start of a message is recognized when the send transmission line is in the idle state for a certain time (specified in bit times) followed by renewed transmission of characters.
- Character string (sequence): The start of a message is recognized when a specified character sequence occurs in the data stream. You can specify up to four character sequences with up to five characters each.
  Example: A received hexadecimal message includes the following characters: "68 10 aa 68 bb 10 aa 16". The configured start character sequences are listed in the following table. Start character sequences will be evaluated once the first character 68H has been received successfully. After the fourth character has been received successfully (the second 68H), the start condition "1" has been met. Once the start conditions have been met, evaluation of the end conditions will start.
  Processing of the start character sequence can end due to different errors in parity, framing or time intervals between characters. These errors will prevent reception of the message, because the start condition has not been met.

Start condition	First character	First character +1	First character +2	First character +3	First character +4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the message receipt (ENDCOND parameter)

The start of a message is recognized by the receiver if a configured end condition applies. The following conditions can be defined as end conditions for message receipt:

- Reply timeout: The receipt of messages will end when the specified maximum duration for the receipt of a character is exceeded. The maximum duration is defined at the RCVTIME parameter. The defined time starts to run down as soon at the last transmission is completed and the RCV_PTP instruction enables the receipt of the message. If no character was received within the defined time (RCVTIME), the RCV_PTP instruction reports an error.
- Message timeout: The receipt of messages will end when the specified maximum duration for the receipt of a message is exceeded. The maximum duration is defined at the MSGTIME parameter. The defined time starts to run down as soon as the first character of the message is received.
- Timeout within the character string: The receipt of messages will end when the time interval between the receipt of two consecutive characters is longer than the value at the CHARGAP parameter.
- Maximum length: The receipt of messages will end when the length of the message defined at the MAXLEN parameter is exceeded.

- Reading message length (N+LEN+M): The receipt of messages will end when a certain message length is reached. This length is calculated by the values of the following parameters:
  - N: Position of the character in the message from which the length field begins.
  - LENGTHSIZE: Size of the length field in bytes
  - LENGTHM: Number of end characters that follow the length field. These characters are not taken into account in the evaluation of the message length.
- Character string: The receipt of messages will end when a defined character sequence is received. The character string can contain a maximum of five characters. For each character of the character string, you can use the bit position to define if this will be considered or ignored in the evaluation.

STATUS parameter

Error code* (W#16#...)	Description
80C0	Error in start condition
80C1	• Error in end condition • No end condition defined
80C2	Receive interrupt enabled
80C3	A value that is equal to 0 or greater than 4132 was entered at the MAXLEN parameter while the "Maximum length" end condition was set.
80C4	A value that is greater than 4131 was entered at the N parameter while the "N+LEN+M" end condition was set.
80C5	A value that is equal to 0 or invalid was entered at the LENGTHSIZE parameter while the "N+LEN+M" end condition was set.
80C6	A value that is greater than 255 was entered at the LENGTHM parameter while the "N+LEN+M" end condition was set.
80C7	A message length greater than 4132 was calculated while the "N+LEN+M" end condition was set.
80C8	A value that is equal to 0 was entered at the RCVTIME parameter while the "Reply timeout" end condition was set.
80C9	A value that is equal to 0 or greater than 2500 was entered at the CHARGAP parameter while the "Timeout within a character string" end condition was set.
80CA	A value that is equal to 0 or greater than 2500 was entered at the IDLETIME parameter while the "Idle line" start condition was set.
80CB	All characters of the character string are marked as "Don't care" even though "Character string" is set as the end condition.
80CC	All characters of the character string are marked as "Don't care" even though "Character string" is set as the start condition.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 3741)".

SEND_PTP: Transmit send buffer data

Description

You use the "SEND_PTP" instruction to start the transmission of data. The "SEND_PTP" instruction does not execute the actual transmission of the data. The data of the send buffer is transmitted to the relevant point-to-point communication module (CM). The CM executes the actual transmission.

Parameters

The following table shows the parameters of the "SEND_PTP" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Enables the requested transmission on a rising edge of this enable input. The content of buffer is transmitted to the point-to-point communication module (CM).
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
BUFFER	Input	VARIANT	I, Q, M, D, L or constant	Pointer to the start address of the send buffer. Boolean values or Array of BOOL are not supported.
LENGTH	Input	UINT	I, Q, M, D, L or constant	Length of the send buffer
PTRCL	Input	BOOL	I, Q, M, D, L or constant	This parameter selects the buffer for normal point-to-point communication or for specific Siemens protocols implemented in the connected CM. FALSE = point-to-point operations controlled by the user program (only valid option)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

STATUS parameter

Error code* (W#16#...)	Description
7000	The send operation is not active.
7001	The send operation is processing the first call.
7002	The send operation is processing subsequent calls (queries following the first call).
8080	The identifier entered for the communications port number is invalid.

Error code* (W#16#...)	Description
8088	The length of the LENGHT parameter does not correspond to the length of data to be sent. See also: Parameters LENGHT and BUFFER.
80D0	A new send request was received while a transmission was taking place.
80D1	The transmission was interrupted because the CTS signal was not confirmed within the specified wait time.
80D2	The send request was interrupted because the communications partner (DCE) signaled that it was not willing to receive (DSR).
80D3	The send request was interrupted because the maximum size of the waiting loop was exceeded (more than 1024 Byte).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 3741)".

Parameters LENGTH and BUFFER

The minimum data size that can be sent by the "PTP_SEND" instruction is one byte. The parameter BUFFER defines the size of the data to be sent. You can use neither the BOOL nor the Array of BOOL data type for the BUFFER parameter.

LENGTH parameter	BUFFER parameter	Description
LENGTH = 0	Not used	The complete data is sent as defined by the BUFFER parameter. If LENGTH = 0, you do not need to specify the number of bytes transferred.
LENGTH > 0	Elementary data type	The LENGTH value must contain the byte count of this data type. Otherwise, data is not transferred and error 8088 is output.
	STRUCT	The LENGTH value can contain a byte count that is smaller than the complete byte length of the structure. In this case, only the first LENGTH bytes are transferred.
	ARRAY	The LENGTH value can contain a byte count that is smaller than the complete byte length of the field. In this case, only the field elements that fit completely in the LENGTH bytes are transferred. The LENGTH value must be a multiple of the byte count of the data elements. Otherwise, STATUS = 8088, ERROR = 1, and no data is transferred.
	STRING	The complete memory arrangement of the character sequence format will be transmitted as well as the information about maximum length of the character string and the actual length of the character string. The LENGTH value must contain bytes for maximum length, actual length, and the characters of the character string. With the data type STRING, all lengths and characters have the size of one byte. If a character string is used for the BUFFER parameter, the LENGTH value must also contain two bytes for the two length fields.

RCV_PTP: Enable receive messages

Description

With the RCV_PTP instruction you enable receipt of a sent message. Each message must be enabled individually. The sent data is only available in the receive area when the message has been acknowledged by the relevant communications partner.

Parameters

The following table shows the parameters of the "RCV_PTP" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Enables receipt on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
BUFFER	Input	VARIANT	I, Q, M, D, L or constant	Points to the start address of the receive buffer. Do not use a tag of the type STRING in the receive buffer.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
LENGTH	Output	UINT	I, Q, M, D, L	Length of the message in the receive buffer

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

STATUS parameter

Error code* (W#16#....)	Description
80E0	Receipt of messages was terminated because the receive buffer is full.
80E1	Receipt of messages was terminated as a result of a parity error.
80E2	Receipt of messages was terminated as a result of a framing error.
80E3	Receipt of messages was terminated as a result of an overflow error.
80E4	Receipt of messages was terminated because the calculated message length (N+LEN+M) exceeds the size of the receive buffer.
8080	The identifier entered for the communications port number is invalid.
8088	A data type STRING is referenced via the BUFFER parameter.
0094	Receipt of messages was terminated because the maximum character length was received.
0095	Receipt of messages was terminated as a result of a timeout.
0096	Receipt of messages was terminated because of a timeout within the character string.

Error code* (W#16#....)	Description
0097	Receipt of messages was terminated as a result of a reply timeout.
0098	Receipt of messages was terminated because the "N+LEN+M" length condition has been satisfied.
0099	Receipt of messages was terminated because the character string defined as the end condition was received.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 3741)".

RCV_RST: Delete receive buffer

Description

With the "RCV_RST" instruction, you delete the receive buffer of a communications partner.

Parameters

The following table shows the parameters of the "RCV_RST" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Enables deleting of the receive buffer on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 3741)".

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

SGN_GET: Query RS-232 signals

Description

With the "SGN_GET" instruction, you query the current state of several signals of an RS-232 communications module.

Parameters

The following table shows the parameters of the "SGN_GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Enables the query on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
NDR	Output	BOOL	I, Q, M, D, L	Is set for one cycle if new data are ready for sending and the instruction was executed error-free.
DTR	Output	BOOL	I, Q, M, D, L	Data terminal ready, module ready
DSR	Output	BOOL	I, Q, M, D, L	Data set ready, communications partner ready
RTS	Output	BOOL	I, Q, M, D, L	Send request, module ready to send
CTS	Output	BOOL	I, Q, M, D, L	Clear to send, communications partner can receive data (reaction to RTS = ON of the module).
DCD	Output	BOOL	I, Q, M, D, L	Data carrier detect, received signal level
RING	Output	BOOL	I, Q, M, D, L	Ring display, display of an incoming call
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

STATUS parameter

Error code* (W#16#....)	Description
80F0	The communication module is an RS-485 module and no signals are available.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 3741)".

SGN_SET: Set RS-232 signals

Description

With the "SGN_SET" instruction, you set the status of the output signals of an RS-232 communications module.

Parameters

The following table shows the parameters of the "SGN_SET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the action on a rising edge Initial value: FALSE
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID) Initial value: FALSE
SIGNAL	Input	BYTE	I, Q, M, D, L or constant	Specifies the signals to be set: • Set 01H = RTS • Set 02H = DTR • Set 04H = DSR Initial value: FALSE
RTS	Input	BOOL	I, Q, M, D, L or constant	Send request, module ready to send Initial value: FALSE
DTR	Input	BOOL	I, Q, M, D, L or constant	Data terminal ready, module ready Initial value: FALSE
DSR	Input	BOOL	I, Q, M, D, L or constant	Data set ready (applies only to interfaces of the DCE type) Initial value: FALSE
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors Initial value: FALSE
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred Initial value: FALSE
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction Initial value: 0

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

STATUS parameter

Error code* (W#16#...)	Description
80F0	The communication module is an RS-485 module and no signals are available.
80F1	No signals are settable because H/W flow control is enabled.
80F2	The DSR signal cannot be set because the module is a DTE device.
80F3	The DTR signal cannot be set because the module is a DCE device.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 3741)".

General status information of the communications blocks

General information on execution status of the communications blocks

The following table shows which general information can be output at the STATUS parameter of the communications blocks:

Error code* (W#16#....)	Description
0000	No error
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0.
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8x3A	Invalid pointer at parameter x.
8070	All internal instance memories are in use.
8080	The identifier entered for the communications port is invalid
8081	Timeout, module error, internal error
8082	Parameter assignment failed because parameter assignment is currently being performed in the background.
8083	Buffer overflow: The CM or CB has returned a receipt message with a length that is greater than permitted by the length parameter.
8085	Error specifying the length at the LENGHT parameter. The specified length is "0" or greater than the maximum permitted value.
8090	Message length invalid, module invalid, message invalid
8091	Incorrect version in parameterization message
8092	Invalid record length in parameterization message
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

USS

Overview of USS instructions

Introduction

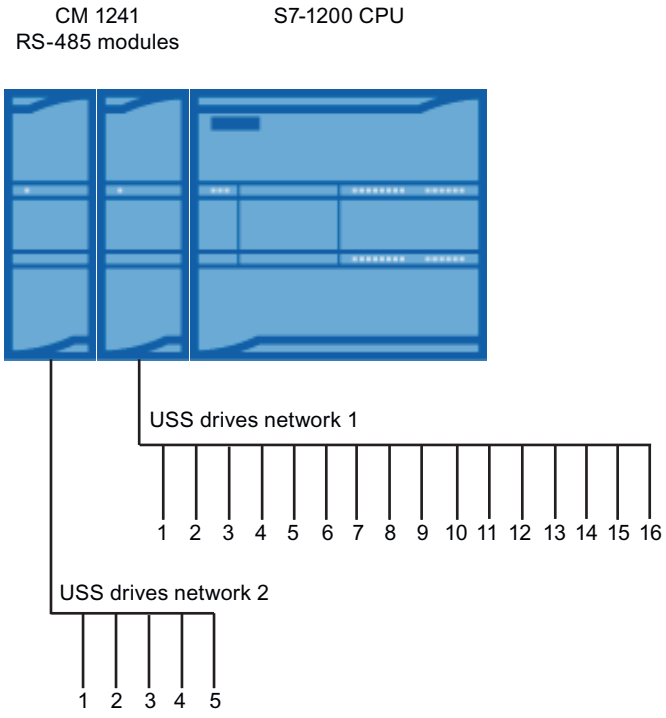
The USS instructions control the operation of drives that support the universal serial interface (USS). With the USS instructions, you can communicate with more than one drive via an RS-485 connection.

To do this, you require a CM 1241 RS-485 communications module or a CB 1241 RS-485 communications board. Up to three CM 1241 RS-485 modules and one CB 1241 RS-485 board can be installed in an S7-1200 CPU.

Each RS-485 port can operate up to sixteen drives.

The USS protocol uses a master/slave network for communication via a serial bus. The master uses an address parameter to send a message to a selected slave. A slave itself can never send without previously receiving a request. Direct exchange of messages between slaves is not possible. USS communication works in half duplex mode.

The following figure shows an example of a USS network diagram:



Requirements for using the USS protocol

General requirements for setting up a drive

- The use of 4 PKW words must be set up for the drives.
- The drives can be configured for 2, 4, 6 or 8 PZD words.
- The number of PZD words in the drive must correspond to the PZD_LEN input of the "USS_DRIVE (Page 3747)" instruction of the drive.
- The baud rate of all drives must match the baud rate of the BAUD input parameter of the "USS_PORT (Page 3746)" instruction.
- The drive must be set up for remote control.
- USS must be specified for the desired frequency value on the COM connection of the drive.
- 1 to 16 must be set as the drive address. This address must correspond to the address of the DRIVE input parameter of the "USS_DRIVE (Page 3747)" instruction.
- To control the direction of the drive, the polarity of the drive desired value must be set up.
- The RS-485 network must be connected correctly.

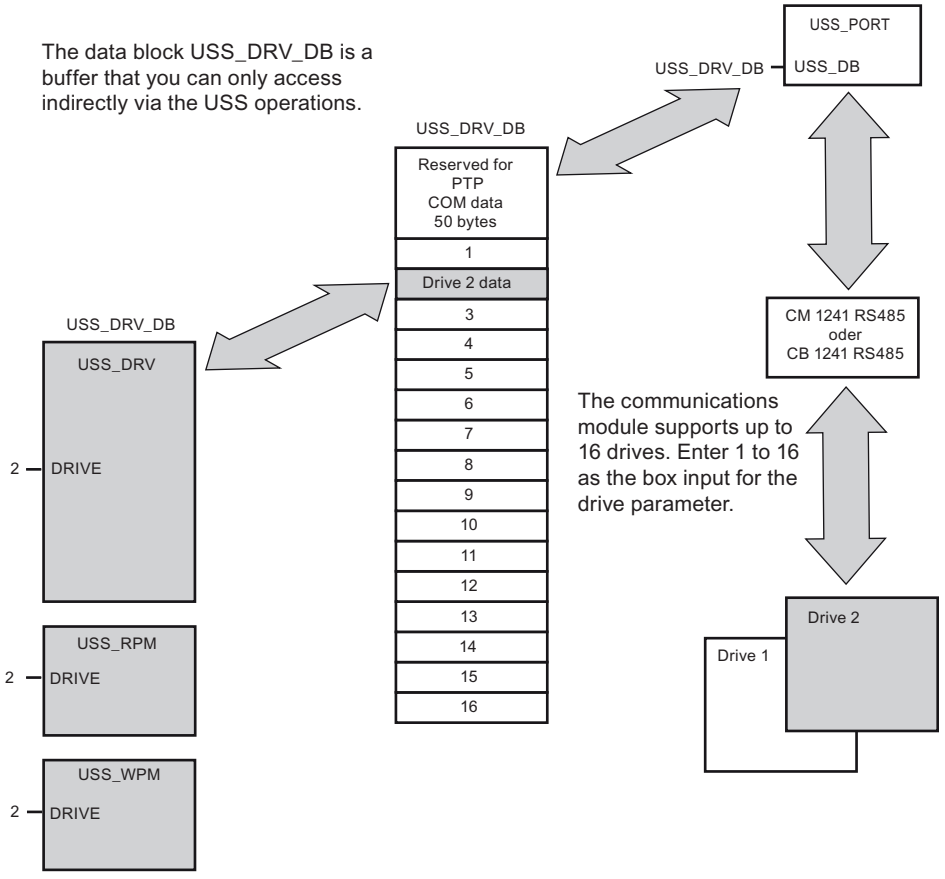
Definition: PKW / PZD area

- The PKW area relates to the handling of the parameter-identifier-value interface.
  The PKW interface is not a physical interface but describes a mechanism that controls the exchange of parameters between two communications partners, in other words, reading and writing parameter values, parameter descriptions and corresponding texts and the handling of parameter changes due to spontaneous messages. All the tasks handled via the PKW interface are essentially tasks for operator control and monitoring, service and diagnostics.
- The PZD area includes the signals required for automation:
  - Control word(s), and setpoint(s) from the master to the slave
  - Status word(s), and actual value(s) from the slave to the master.

Both areas together result in the user data field. This is transferred by the master to the slave as a job frame or by the slave to the master as a reply frame.

Description

Each communication module CM 1241 RS485 supports a maximum of 16 drives. A single instance data block contains temporary memory and buffer functions for all drives in the USS network that are connected to a PtP communications module you installed. The USS instructions for these drives have shared access to the information in this data block.



- All drives (max. 16) that are connected to an RS485 port are part of the same USS network. All drives that are connected to a different RS485 port are part of a different USS network. As the S7-1200 supports up to three CM 1241 RS485 modules, you can set up up to three USS networks, each having a maximum of 16 drives, thus a total of 48 USS drives are supported.
- Each USS network is managed via a unique data block (three data blocks are required for three USS networks with three CM 1241 RS485 modules). All instructions that belong to a USS network must share this data block. This includes all "USS_DRIVE (Page 3747)", "USS_PORT (Page 3746)", "USS_RPM (Page 3749)", and "USS_WPM (Page 3751)" instructions for controlling all drives in a USS network.

- The instruction "USS_DRIVE (Page 3747)" is a function block (FB). When you insert the instruction "USS_DRIVE" in the editor, the "Call options" dialog will ask you to assign a DB to the instruction.
  - If this is the first "USS_DRIVE" instruction in this program for this USS network, you can accept the default DB assignment (or, if necessary, change the name) and the new DB will be created.
  - If, however, this is not the first "USS_DRIVE" instruction for this network, you must select the DB that was previously assigned to this USS network in the drop-down list in the "Call options" dialog.
- All "USS_PORT (Page 3746), USS_RPM (Page 3749)", and "USS_WPM (Page 3751)" instructions are functions (FCs). When you insert these FCs in the editor, no DB is assigned. Instead, you have to assign the DB in question to the USS_DB input of these instructions (double-click on the parameter field and then on the icon to display the available DBs).
- The instruction "USS_PORT (Page 3746)" controls communication between the CPU and the drives via the PtP communications module. Whenever this instruction is called, communication with a drive is processed. Your program must call this function quickly enough so that the drives do not report a timeout. The instruction can be called from the main program or any interrupt OB.
- The function block "USS_DRIVE (Page 3747)" gives your program access to a specified drive in the USS network. Its inputs and outputs correspond to the states and operating functions of the drive. If there are 16 drives in the network, "USS_DRIVE" must be called at least 16 times in your program, i.e., once for each drive. How quickly these blocks are called depends on the required speed for controlling drive functions.
  You can only call the instruction "USS_DRIVE" from the main program OB.

Caution

Call "USS_DRIVE", "USS_RPM", "USS_WPM" only from the main program OB. The instruction "USS_PORT" can be called from any OB, it is usually called from a time-delay interrupt OB. If the instruction "USS_PORT" is interrupted during execution, this may result in unexpected errors.

The "USS_RPM" and "USS_WPM" instructions are used to read and write the operating parameters of the drive. These parameters control the internal mode of operation of the drive. A definition of these parameters can be found in the drive manual.

Your program may also contain any number of these instructions; however only one read or write request can be active for a drive. You may only call the instructions "USS_RPM" and "USS_WPM" from the main program OB.

Calculating the time for communication with the drive

Communication with the drive is runs asynchronous to the cycle of the S7-1200. The S7-1200 runs through several cycles before communication with a drive is completed.

The interval of "USS_PORT" is the time required for a drive transaction. The following table shows the minimum intervals for "USS_PORT" for each baud rate. Calling the "USS_PORT" more frequently than the "USS_PORT" interval will not increase the number of transactions. The timeout interval of the drive is the period of time available to a transaction if 3 attempts

are required to complete the transaction due to communications errors. By default, up to 2 further attempts are made for each transaction with the USS protocol.

Baud rate	Calculated minimum interval for calling USS_PORT (ms)	Drive message interval timeout per drive (ms)
1200	790	2370
2400	405	1215
4800	212.5	638
9600	116.3	349
19200	68.2	205
38400	44.1	133
57600	36.1	109
115200	28.1	85

USS_PORT: Edit communication via USS network

Description

The "USS_PORT" instruction handles communication over the USS network. In the program, use one "USS_PORT" instruction per PtP communications port to control the transmission to or from one drive.

All USS instructions that are assigned to one USS network and one PtP communications port must use the same instance data block.

Call

Your program must execute the "USS_PORT" instruction often enough to prevent timeouts in the drive. You should therefore call the "USS_PORT" instruction from a cyclic interrupt OB to prevent drive timeouts and keep the most recent USS data updates available for "USS_DRIVE (Page 3747)" calls.

Parameters

The following table shows the parameters of the "USS_PORT" instruction:

Parameter	Declaration	Data type	Memory area	Description
PORT	Input	PORT	D, L or constant	PtP communications port identifier Constant that can be referenced within the "Constants" tab of the default tag table.
BAUD	Input	DINT	I, Q, M, D, L or constant	Baud rate for USS communication.
USS_DB	InOut	USS_BASE	D	Reference to the instance DB of the "USS_DRIVE (Page 3747)" instruction.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	ERROR is set to TRUE if an error occurs. A corresponding error code will be output at the STATUS output.
STATUS (Page 3752)	Output	WORD	I, Q, M, D, L	Status value of the request. It indicates the result of the cycle or initialization. Additional information is available in the "USS_Extended_Error (Page 3752)" tag for some status codes.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

USS_DRIVE: Swap data with drive

Description

The "USS_DRIVE" instruction exchanges data with the drive by creating request messages and interpreting the drive response messages. A separate instruction must be used for each drive, but all USS instructions assigned to one USS network and one PtP communications module must use the same instance data block. You must create the DB name when you place the first "USS_DRIVE" instruction. Then reuse this DB that was created when the initial instruction was inserted.

When the "USS_DRIVE" instruction is executed the first time, the drive indicated by the USS address (parameter DRIVE) is initialized in the instance DB. After this initialization, subsequent "USS_PORT (Page 3746)" instructions can start communication with the drive at this drive number.

Changing the drive number requires a PLC STOP to RUN mode transition that initializes the instance DB. Input parameters are configured in the USS send buffer, and outputs are read from a "previous" valid response buffer if any exists. There is no data transmission during execution of the "USS_DRIVE" instruction. Communication with the drives takes place when "USS_PORT (Page 3746)" is executed. "USS_DRIVE" only configures the messages to be sent and interprets data received in a previous request.

You can control the drive direction of rotation using either the DIR (BOOL) input or using the sign (positive or negative) at the SPEED_SP (REAL) input. The following table explains how these inputs work together to determine the drive direction, assuming the motor is wired for forward rotation.

SPEED_SP	DIR	Direction of rotation of drive
Value > 0	0	Reverse
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Reverse

Parameters

Expand the box to display all the parameters by clicking the bottom of the box. The parameter connections that are grayed are optional and do not need to be assigned.

The following table shows the parameters of the "USS_DRIVE" instruction:

Parameter	Declaration	Data type	Memory area	Description
RUN	Input	BOOL	I, Q, M, D, L or constant	Drive start bit: If this parameter has the value TRUE, this input enables the drive to run at the preset speed.
OFF2	Input	BOOL	I, Q, M, D, L or constant	"Electrical stop" bit: If this parameter has the value FALSE, this bit cause the drive to coast to a stop without braking.
OFF3	Input	BOOL	I, Q, M, D, L or constant	Fast stop bit - If this parameter has the value FALSE, this bit causes a fast stop by braking the drive.
F_ACK	Input	BOOL	I, Q, M, D, L or constant	Fault acknowledge bit - This bit resets the fault bit on a drive. This bit is set after the fault is cleared to indicate to the drive that it no longer needs to indicate the previous fault.
DIR	Input	BOOL	I, Q, M, D, L or constant	Drive direction control - This bit is set to indicate that the direction is forward (when SPEED_SP is positive).
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PZD_LEN	Input	USINT	I, Q, M, D, L or constant	Word length - This is the number of words of PZD data. Valid values are 2, 4, 6, or 8 words. The default is 2.
SPEED_SP	Input	REAL	I, Q, M, D, L or constant	Speed setpoint - This is the speed of the drive as a percentage of configured frequency. A positive value specifies forward direction (when DIR has the value TRUE).
CTRL3	Input	WORD	I, Q, M, D, L or constant	Control word 3 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL4	Input	WORD	I, Q, M, D, L or constant	Control word 4 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL5	Input	WORD	I, Q, M, D, L or constant	Control word 5 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL6	Input	WORD	I, Q, M, D, L or constant	Control word 6 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive.
CTRL7	Input	WORD	I, Q, M, D, L or constant	Control word 7 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL8	Input	WORD	I, Q, M, D, L or constant	Control word 8 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
NDR	Output	BOOL	I, Q, M, D, L	New data ready - If this parameter has the value TRUE, the bit indicates that the output contains data from a new communication request.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred - If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT" instruction.

Parameter	Declaration	Data type	Memory area	Description
STATUS (Page 3752)	Output	WORD	I, Q, M, D, L	Status value of the request. It indicates the result of the cycle. This is not a status word returned from the drive.
RUN_EN	Output	BOOL	I, Q, M, D, L	Run enabled - This bit indicates whether the drive is running.
D_DIR	Output	BOOL	I, Q, M, D, L	Drive direction - This bit indicates whether the drive is running forward.
INHIBIT	Output	BOOL	I, Q, M, D, L	Drive inhibited - This bit indicates the state of the inhibit bit on the drive.
FAULT	Output	BOOL	I, Q, M, D, L	Drive fault - This bit indicates that the drive has registered a fault. The user must eliminate the fault and then set the F_ACK bit to clear this bit.
SPEED	Output	REAL	I, Q, M, D, L	Drive current speed (scaled value of drive status word 2) - The value of the speed of the drive as a percentage of configured speed.
STATUS1	Output	WORD	I, Q, M, D, L	Drive status word 1 - This value contains fixed status bits of a drive.
STATUS3	Output	WORD	I, Q, M, D, L	Drive status word 3 - This value contains a user-configurable status word on the drive.
STATUS4	Output	WORD	I, Q, M, D, L	Drive status word 4 - This value contains a user-configurable status word on the drive.
STATUS5	Output	WORD	I, Q, M, D, L	Drive status word 5 - This value contains a user-configurable status word on the drive.
STATUS6	Output	WORD	I, Q, M, D, L	Drive status word 6 - This value contains a user-configurable status word on the drive.
STATUS7	Output	WORD	I, Q, M, D, L	Drive status word 7 - This value contains a user-configurable status word on the drive.
STATUS8	Output	WORD	I, Q, M, D, L	Drive status word 8 - This value contains a user-configurable status word on the drive.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

USS_RPM: Readout parameters from the drive

Description

The "USS_RPM" instruction reads a parameter from the drive. All USS functions that are assigned to one USS network and one PtP communications module must use the same instance data block. "USS_RPM" must be called from the main program OB.

Parameter

The following table shows the parameters of the "USS_RPM" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Send request: If this parameter has the value TRUE, it indicates that a new read request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	Input	UINT	I, Q, M, D, L or constant	Parameter number: This input specifies which drive parameter is written. The range of this parameter is 0 to 2047. See your drive manual for details on how to access any parameters above this range.
INDEX	Input	UINT	I, Q, M, D, L or constant	Parameter index: This input specifies which drive parameter index is to be written. This is a 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for additional information.
USS_DB	InOut	USS_BASE	D	Reference to the instance DB that is created and initialized when a "USS_DRIVE" instruction is inserted in your program.
DONE	Output	BOOL	I, Q, M, D, L	If this parameter has the value TRUE, it indicates that the VALUE output holds the previously requested read parameter value. This bit is set when the "USS_DRIVE" instruction recognizes the read response from the drive. This bit is reset when either: • you request the response data via another "USS_RPM" poll or • the second of the next two calls of "USS_DRIVE (Page 3747)" is executed
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred - If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT (Page 3746)" instruction.
STATUS (Page 3752)	Output	WORD	I, Q, M, D, L	This is the status value of the request. It indicates the result of the read request. Additional information is available in the "USS_Extended_Error (Page 3752)" tag for some status codes.
VALUE	Output	VARIANT	I, Q, M, D, L	This is the value of the parameter that was read and is valid only when the DONE bit has the value TRUE.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

USS_WPM: Change parameters in the drive

Description

The "USS_WPM" instruction modifies a parameter in the drive. All USS functions that are assigned to one USS network and one PtP communications module must use the same instance data block. "USS_WPM" must be called from the main program OB.

Note

EEPROM write operations

Beware of overusing the EEPROM write operation. Minimize the number of EEPROM write operations to extend the EEPROM life.

Parameter

The following table shows the parameters of the "USS_WPM" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Send request: If this parameter has the value TRUE, it indicates that a new write request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	Input	UINT	I, Q, M, D, L or constant	Parameter number: This input specifies which drive parameter is written. The range of this parameter is 0 to 2047. See your drive manual for details on how to access any parameters above this range.
INDEX	Input	UINT	I, Q, M, D, L or constant	Parameter index: This input specifies which drive parameter index is to be written. This is a 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for additional information.
EEPROM	Input	BOOL	I, Q, M, D, L or constant	Store to drive EEPROM: If this parameter has the value TRUE, values written to the drive parameter will be stored in the drive EEPROM. If this parameter has the value FALSE, the value written is only temporarily saved and will be lost the next time the drive is switched on.
VALUE	Input	VARIANT	I, Q, M, D, L or constant	The value of the parameter that is to be written. It must be valid on the transition of REQ.
USS_DB	InOut	USS_BASE	D	This is a reference to the instance DB that is created and initialized when a "USS_DRIVE (Page 3747)" instruction is inserted in your program.

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	If this parameter has the value TRUE, the VALUE input was written to the drive. This bit is set when the "USS_DRIVE (Page 3747)" instruction recognizes the write response from the drive. This bit is reset when either: You request the drive's confirmation that the write operation has been completed via another "USS_WPM" query or when the second of the next two calls of "USS_DRIVE (Page 3747)" is executed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred: If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT (Page 3746)" instruction.
STATUS (Page 3752)	Output	WORD	I, Q, M, D, L	This is the status value of the request. It indicates the result of the write request. Additional information is available in the "USS_Extended_Error (Page 3752)" tag for some status codes.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Parameter STATUS of USS instructions

STATUS parameter

The following table contains the status codes of the USS operation that are output at the STATUS output of the USS instructions:

STATUS* (W#16#....)	Description
0000	No error
8180	The length of the drive response did not match the characters received from the drive. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8181	The parameter VALUE is not of the data type WORD, REAL, or DWORD
8182	User supplied a parameter value of the type word and received a DWORD or REAL from the drive in the response
8183	User supplied a parameter value of the type DWORD or REAL and received a word from the drive in the response
8184	Response telegram from drive had a bad checksum. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8185	Illegal drive address (valid drive address range: 1-16)
8186	Speed set point out of valid range (valid speed SP range: -200% to 200%)
8187	Wrong drive number responded to the request sent. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8188	Illegal PZD word length specified (valid range = 2, 4, 6 or 8 words)

STATUS* (W#16#....)	Description
8189	Illegal baud rate was specified
818A	Parameter request channel is in use by another request for this drive
818B	Drive has not responded to requests and retries. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
818C	Drive returned an extended error on a parameter request operation. See the extended error description below this table.
818D	Drive returned an invalid access error on a parameter request operation. See your drive manual for information of why parameter access may be limited
818E	Drive has not been initialized: This error code is output at "USS_RPM (Page 3749)" or "USS_WPM (Page 3751)" if the "USS_DRIVE (Page 3747)" instruction has not been called at least once for this drive. This prevents the initialization of the first cycle of "USS_DRIVE (Page 3747)" from overwriting a pending parameter read or write request since it initializes the drive as a new entry. To eliminate this error, call the "USS_DRIVE (Page 3747)" instruction for this drive.
80Ax-80Fx	Specific errors returned from PtP (Point-to-Point) communication instructions called by the USS library: These error code values are not modified by the USS library and are defined in the PtP instruction descriptions.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

USS_Extended_Error - USS drive extended error codes

USS drives support read and write access to a drive's internal parameters. This feature allows distributed control and configuration of the drive. Drive parameter access operations can fail due to errors such as values out of range or invalid requests in a drive's current mode. The drive generates an error code that is output in the "USS_Extended_Error" variable in the instance DB of the "USS_DRIVE (Page 3747)" instruction. This error code value is only valid for the last execution of the "USS_RPM (Page 3749)" or "USS_WPM (Page 3751)" instruction. The drive error code is put into the "USS_Extended_Error" tag when the value of STATUS is hexadecimal 818C. The error code of "USS_Extended_Error" depends on the drive variant. See the drive's manual for a description of the extended error codes for read and write parameter operations.

MODBUS (RTU)

MB_COMM_LOAD: Configure port on the PtP module for Modbus RTU

Description

The "MB_COMM_LOAD" instruction configures a port for communication using the Modbus RTU protocol. The following hardware can be used for this:

- Up to three point-to-point modules (PtP) CM 1241 RS485 or CM 1241 RS232
- A communications board CB 1241 RS485 in addition to this

After configuration of the port, you communicate over Modbus by executing the "MB_SLAVE" or "MB_MASTER" instruction.

Call

"MB_COMM_LOAD" must be called once to configure the port for the Modbus RTU protocol. On completion of the configuration, the port can be used by the "MB_MASTER (Page 3757)" and "MB_SLAVE (Page 3765)" instructions.

"MB_COMM_LOAD" only needs to be called again if one of the communication parameters has to be modified. Each "MB_COMM_LOAD" call deletes the communications buffer. To avoid data loss during communication, you should not call the instruction unnecessarily.

One instance of "MB_COMM_LOAD" must be used to configure the port of each communication module that is used for Modbus communication. You assign a unique "MB_COMM_LOAD" instance data block for each port that you use. The S7-1200 CPU is limited to three communication modules.

An instance data block is assigned when you insert the "MB_MASTER (Page 3757)" or "MB_SLAVE (Page 3765)" instruction. This instance data block is referenced when you specify the MB_DB parameter on the "MB_COMM_LOAD" instruction.

Parameters

The following table shows the parameters of the instruction "MB_COMM_LOAD":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Execution of the instruction upon a rising edge.
PORT	Input	PORT	I, Q, M, D, L or constant	ID of the communications port: After you have inserted the communications module in the device configuration, the port ID appears in the drop-down list at the PORT box connection. This constant can also be referenced within the "Constants" tab of the tag table.
BAUD	Input	UDINT	I, Q, M, D, L or constant	Baud rate selection: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 All other values are invalid.
PARITY	Input	UINT	I, Q, M, D, L or constant	Parity selection: • 0 – None • 1 – Odd • 2 – Even
FLOW_CTRL	Input	UINT	I, Q, M, D, L or constant	Flow control selection: • 0 – (default) No flow control • 1 – Hardware flow control with RTS always ON (does not apply to RS485 ports) • 2 - Hardware flow control with RTS switched

Parameter	Declaration	Data type	Memory area	Description
RTS_ON_DLY	Input	UINT	I, Q, M, D, L or constant	RTS on-delay selection: • 0 – (default) No delay of RTS active until the first character of the message is transmitted. • 1 to 65535 – Delay in milliseconds of "RTS active" until the first character of the message is transmitted (does not apply to RS-485 ports). RTS delays must be applied independent of the FLOW_CTRL selection.
RTS_OFF_DLY	Input	UINT	I, Q, M, D, L or constant	RTS off-delay selection: • 0 – (default) No delay after the last character transmitted until "RTS inactive" • 1 to 65535 – Delay in milliseconds after the last character transmitted until "RTS inactive" (does not apply to RS-485 ports). RTS delays must be applied independent of the FLOW_CTRL selection.
RESP_TO	Input	UINT	I, Q, M, D, L or constant	Response timeout: Time in milliseconds allowed by "MB_MASTER (Page 3757)" for the slave to respond. If the slave does not respond in this time, "MB_MASTER (Page 3757)" repeats the request or terminates the request with an error if the specified number of retries has been sent. 5 ms to 65535 ms (default = 1000 ms).
MB_DB	Input	MB_BASE	D	A reference to the instance data block of the "MB_MASTER (Page 3757)" or "MB_SLAVE (Page 3765)" instructions. After you insert "MB_SLAVE (Page 3765)" or "MB_MASTER (Page 3757)" in your program, the DB identifier appears in the drop-down list at the MB_DB box connection.
DONE	Output	BOOL	I, Q, M, D, L	Execution of instruction completed without error.
ERROR	Output	BOOL	I, Q, M, D, L	Error: • 0 – No error detected • 1 – Indicates that an error was detected. An error code is output in the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Port configuration error code

For additional information on valid data types, refer to "Overview of the valid data types (Page 1908)".

Parameter STATUS

Error code* (W#16#...)	Description
0000	No error
8180	Invalid value for the port ID (wrong address for the communications module).

Error code* (W#16#...)	Description
8181	Invalid baud rate value.
8182	Invalid parity value.
8183	Invalid flow control value.
8184	Invalid value for the timeout of the response (the time before which a timeout is reported must be at least 25 ms).
8185	Incorrect pointer in the MB_DB parameter to the instance DB of the "MB_MASTER (Page 3757)" or "MB_SLAVE (Page 3765)" instruction.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

MB_COMM_LOAD data block tags

The table below shows the public static tags in the instance DB of MB_COMM_LOAD that you can use in your program.

Table 11-97 Static tags in the instance DB

Tag	Data type	Default	Description
ICHAR_GAP	WORD	0	Delay for the character spacing between characters. This parameter is specified in milliseconds and is used to increase the anticipated period between the received characters. The corresponding number of bit times for this parameter is added to the Modbus standard value of 35 bit times (3.5 character times).
RETRIES	WORD	2	The number of repeated attempts by the master before the error code 0x80C8 is returned for "No response".
MODE	USINT	0	Mode Permitted modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS422) four-wire mode (multipoint master) • 3 = Full duplex (RS422) four-wire mode (multipoint slave) • 4 = Half duplex (RS485) two-wire mode

Tag	Data type	Default	Description
LINE_PRE	USINT	0	Receive line initial state Permitted defaults are: • 0 = "No" default • 1 = Signal R(A)=5V, signal R(B)=0 V (break detection): No break detection is possible with this default. Can only be selected with: "Full duplex (RS422) four-wire mode (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = Signal R(A)=0 V, signal R(B)=5 V: This default corresponds to the at-rest state (no send process is active). No break detection is possible with this default.
CABLE-BREAK	USINT	0	Activate cable break detection: • 0 - not activated • 1 - activated

MB_MASTER: Communicate via the PtP port as Modbus master

Description of MB_MASTER

Description

The "MB_MASTER" instruction allows your program to communicate as a Modbus master using a port on a point-to-point module (CM) or a communications board (CB). You can access data in one or more Modbus slave devices.

Before the "MB_MASTER" instruction can communicate with a port, "MB_COMM_LOAD (Page 3753)" must first execute.

An instance DB is created when you insert the "MB_MASTER" instruction in your program. You specify this instance DB in the MB_DB input parameter of the "MB_COMM_LOAD (Page 3753)" instruction.

Rules for Modbus master communication

- A port used for Modbus master requests cannot be used for "MB_SLAVE".
- A port can be used for one or more "MB_MASTER" calls if the same instance DB is used.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must poll the "MB_MASTER" instruction for completed send and receive operations.

- Calling the instruction:
  - Call the "MB_MASTER" instruction if possible in a cyclic program OB. The instruction can only be called in a time delay or cyclic interrupt OB.
  - Do not call more than one "MB_MASTER" instruction in organization blocks with different priority classes. If a "MB_MASTER" instruction executes "preemptively" from a higher priority class, the instruction may execute incorrectly.
  - Do not call the "MB_MASTER" instruction in a startup, diagnostics or time error OB.
- After a transfer has started, the EN parameter (LAD/FBD) must remain set to the value "1" until the DONE or ERROR output parameter is set to "1" by the instruction. A renewed call by the REQ parameter while the instruction is executing causes an error. After the instruction executes, the bit in the REQ parameter remains set for the time specified by the BLOCKED_PROC_TIMEOUT parameter in the instance DB.
- If "MB_MASTER" sends a request to a slave, make sure that "MB_MASTER" continues to execute until the response from the slave arrives.

Parameters

The following table shows the parameters of the "MB_MASTER" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ (Page 3760)	Input	BOOL	I, Q, M, D, L	Request input: • 0 – No request • 1 – Request to transmit data to Modbus slave(s)
MB_ADDR	Input	UINT	I, Q, M, D, L or constant	Modbus RTU station address: • Default address range: 0 to 247 • Extended address range: 0 to 65535 The value "0" is reserved for broadcasting a message to all Modbus slaves. Modbus function codes 05, 06, 15, and 16 are the only function codes supported for broadcast.
MODE (Page 3760)	Input	USINT	I, Q, M, D, L or constant	Mode selection: Specifies the type of request: Read, write, or diagnostics: Refer to the Modbus functions table for details.
DATA_ADDR (Page 3760)	Input	UDINT	I, Q, M, D, L or constant	Starting address in the slave: Specifies the starting address of the data to be accessed in the Modbus slave. You will find the valid addresses in the Modbus functions table.
DATA_LEN	Input	UINT	I, Q, M, D, L or constant	Data length: Specifies the number of bits or words to be accessed in this request. You will find the valid lengths in the Modbus functions table.
DATA_PTR (Page 3762)	Input	VARIANT	M, D	Points to the DB or bit memory address of the CPU for the data to be written or read. For a DB, this must be created with the "Standard - compatible with S7-300/400" access type.
DONE	Output	BOOL	I, Q, M, D, L	• 0: Transaction not completed • 1: Transaction completed without error

Parameters	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	• 0: No "MB_MASTER" transaction in progress • 1: "MB_MASTER" transaction in progress
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error • 1: Error, the error code is indicated by the STATUS parameter
STATUS	Output	WORD	I, Q, M, D, L	Execution condition code

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

STATUS parameter

Table 11-98 Communications and configuration error messages of the instruction

Error code* (W#16#....)	Description
0000	No error
80C8	Slave timeout. Check the baud rate, parity and the connectors on the slave.
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission within the wait time. This error is also generated during hardware flow control if the recipient does not detect CTS within the wait time.
80D2	The send request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full.
80E1	The message was terminated as a result of a parity error.
80E2	The message was terminated as a result of a framing error.
80E3	The message was terminated as a result of an overrun error.
80E4	The message was terminated as a result of the specified length exceeding the total buffer size.
8180	Invalid value for the port ID.
8186	Invalid Modbus station address
8188	The MODE parameter has an invalid value for a broadcast call.
8189	Invalid data address value.
818A	Invalid data length value.
818B	Invalid pointer to the local data source/destination: Size not correct
818C	The DATA_PTR parameter has an invalid pointer. Use a pointer to a bit memory area or a DB with the "Standard - compatible with S7-300/400" access type.
8200	Port is busy processing a send request
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Table 11-99 Error messages of the Modbus protocol

Error code* (W#16#....)	Response code of slave	Description
8380	-	CRC error
8381	01	Function code not supported
8382	03	Data length error
8383	02	Error in the data address or address outside the valid range of DATA_PTR
8384	> 03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)
8386	-	Function code of the response does not match the function code of the query.
8387	-	Response from wrong slave
8388	-	The response of the slave to a write call is not correct. The data sent by the slave does not match the query from the master.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Parameter REQ

Description

- REQ = FALSE: No request
- REQ = TRUE: Request to transmit data to Modbus slave(s)

You can control this input by means of a level-controlled or edge-controlled contact.

A state machine is started every time this input is activated to ensure that another instruction "MB_MASTER" that uses the same instance DB can only issue a request when the current request has been processed. All other input states are recorded and saved internally for the current request until the reply has been received or an error has been detected.

If the same instance of "MB_MASTER" is executed again with REQ input = 1 before the current request is completely processed, no subsequent transfers are made. However, if the request has been processed, a new request is issued at the time when "MB_MASTER" is executed again with REQ input = 1.

DATA_ADDR and MODE parameters

Description

You specify the start address for data access to the Modbus slave using the DATA_ADDR parameter.

With the MODE parameter and the Modbus address, you specify the function code to be transferred to the Modbus slave. The following table shows the relationship between the MODE parameter, the function code and Modbus address range.

MODE	Modbus function	Data length	Operation and data	Modbus address
0	01	1 to 2000 1 to 1992 (1)	Read output bits: 1 to (1992 or 2000) bits per query	1 to 9999
0	02	1 to 2000 1 to 1992 (1)	Read input bits: 1 to (1992 or 2000) bits per query	10001 to 19999
0	03	1 to 125 1 to 124 (1)	Read holding register: 1 to (124 or 125) WORD per query	40001 to 49999 or 400001 to 465535
0	04	1 to 125 1 to 124 (1)	Read input WORD: 1 to (124 or 125) WORD per query	30001 to 39999
1	05	1	Writing an output bit: One bit per query	1 to 9999
1	06	1	Writing a holding register: 1 WORD per query	40001 to 49999 or 400001 to 465535
1	15	2 to 1968 2 to 1960 (1)	Writing multiple output bits: 2 to (1960 or 1968) bits per query	1 to 9999
1	16	2 to 123 2 to 122 (1)	Writing multiple holding registers: 2 to (122 or 123) WORD per query	40001 to 49999 or 400001 to 465535
2	15	1 to 1968 2 to 1960 (1)	Writing one or more output bits: 1 to (1960 or 1968) bits per query	1 to 9999
2	16	1 to 123 2 to 122 (1)	Writing one or more holding registers: 1 to (122 or 123) WORD per query	40001 to 49999 or 400001 to 465535
11	11	0	Reading out the communications status word of the slaves and the event counter: The status word indicates execution of the instruction (0: is not executing; 0xFFFF: is executing). The event counter is incremented each time a message is transferred successfully. The DATA_ADDR and DATA_LEN parameters of the "MB_MASTER" instruction are ignored with this function.	-
80	08	1	Checking the slave status by reading the error code (0x0000): 1 WORD per query	-
81	08	1	Resetting the event counter of the slave with the diagnostics code 0x000A: 1 WORD per query	-
3 to 10, 12 to 79, 82 to 2555			Reserved	-
(1) For the "Extended address range", the maximum data length is reduced by one byte or one WORD depending on which data type is used for the function.				

Parameter DATA_PTR

Description

The DATA_PTR parameter is a pointer to a data block or bit memory from which the data should be written or read. If you use a data block, create a global data block with the "Standard - compatible with S7-300/400" access type.

Data block structures for the DATA_PTR parameter

- These data types are valid for **reading of words** of Modbus addresses 30001 to 39999, 40001 to 49999, and 400001 to 465536 and also for **writing of words** to Modbus addresses 40001 to 49999 and 400001 to 465536.
  - Standard array of WORD, UINT, or INT data types (see below).
  - Named WORD, UINT, or INT structure where each element has a unique name and 16 bit data type.
  - Named complex structure where each element has a unique name and 16 bit or 32 bit data type.
- For reading and writing of bits of Modbus addresses 00001 to 09999 and 10001 to 19999.
  - Standard array of Boolean data types.
  - Named Boolean structure of uniquely named Boolean variables.
- Although not required, it is recommended that each "MB_MASTER" instruction has its own separate memory area in a global data block. The reason for this recommendation is that there is a greater possibility of data corruption if multiple "MB_MASTER" instructions are reading and writing the same area of a global data block.
- The memory areas for DATA_PTR do not need to be in the same global data block. You can create one data block with multiple areas for Modbus read operations, one data block for Modbus write operations, or one data block for each slave station.

Instance DB of the "MB_MASTER" instruction

Static variables of the instance DB

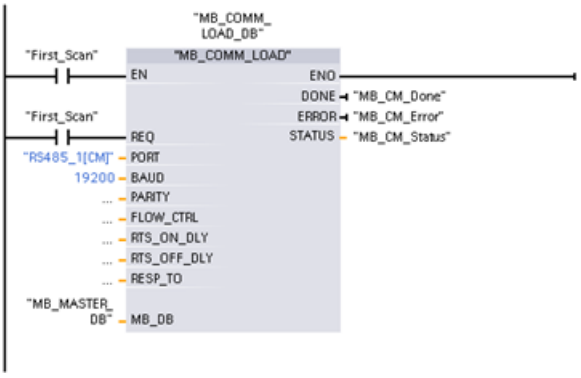
The following table describes the static variables of the instance DB of the instruction that you can use in the user program.

Variable	Data type	Description
MB_STATE	UINT	Internal status of the Modbus instruction
BLOCKED_ PROC_TIMEOUT	REAL	Time between completion of the instruction call and resetting the ACTIVE bit in the instance DB. The time buffer is used to avoid execution of the instruction being terminated before a job has been sent completely. The default time is 500 ms.
EXTENDED_ AD-DRESSING	BOOL	Configuring addressing: • 0: Default address area (1 byte) • 1: Extended address area (2 bytes) For additional information, refer to the section EXTENDED_AD-DRESSING: Instance DB of the "MB_SLAVE" instruction (Page 3768)

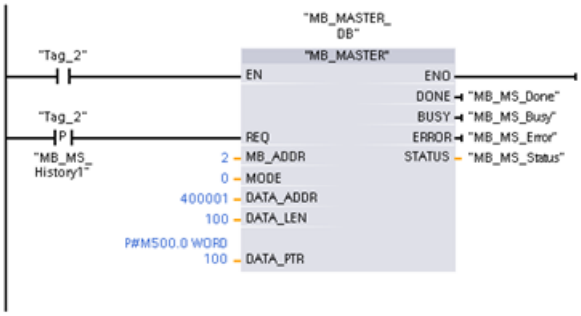
Sample program for a Modbus master

Networks (LAD)

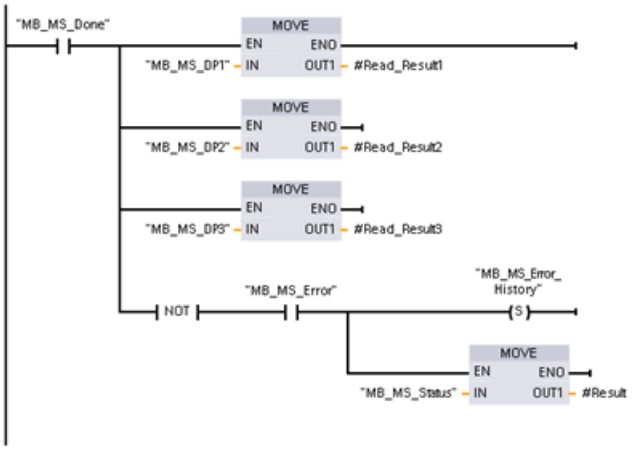
Network 1: Initialize parameters of the RS-485 module only once during the first cycle.



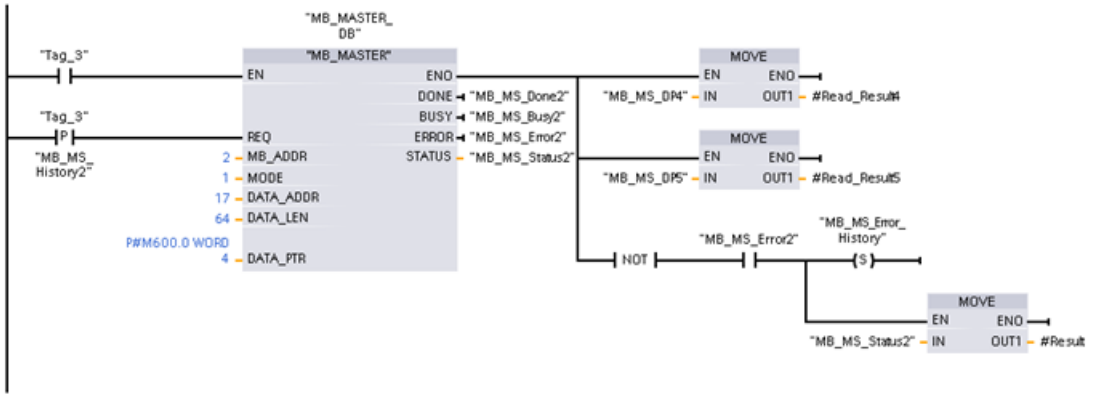
Network 2: Read 100 words from the holding register of the slave.



Network 3: This is an optional network that displays the values of the first 3 words as soon as the read operation has executed.



Network 4: Write 64 bits to the process image output, starting at slave address Q2.0.



MB_SLAVE: Communicate via the PtP port as Modbus slave

Description of MB_SLAVE

Description

The "MB_SLAVE" instruction allows your program to communicate as a Modbus slave using a port on a point-to-point module (PtP) or a communications board (CB). A Modbus RTU master can issue a request and then your program responds via "MB_SLAVE" execution.

You must assign a unique instance data block when you insert the "MB_SLAVE" instruction in your program. This instance data block is used when you specify it at the MB_DB parameter of the "MB_COMM_LOAD (Page 3753)" instruction.

Modbus communication function codes (1, 2, 4, 5, and 15) can read and write bits and words directly in the process image input and process image output in the target system. The following table shows the mapping of Modbus addresses to the process image in the CPU.

Modbus functions of "MB_SLAVE"						S7-1200	
Codes	Function	Data area	Address range			Data area	CPU address
01	Read bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
02	Read bits	Input	10001	to	18192	Process image input	I0.0 to I1023.7
04	Read words	Input	30001	to	30512	Process image input	IW0 to IW1022
05	Write bit	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
15	Write bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7

Modbus communication function codes (function codes 3, 6, 16) use a separate holding register. To do this, you can use bit memory or a data block with the "Standard - compatible with S7-300/400" access type.

You specify the type of the holding register using the MB_HOLD_REG parameter of the MB_SLAVE instruction. The following table shows the mapping of the Modbus holding register to the DB address of MB_HOLD_REG in the target system.

Modbus functions of "MB_SLAVE"				S7-1200	
Codes	Function	Data area	Address range (WORD number)	Address in the DB (BYTE number)	Bit memory address (BYTE number)
03	Read words	Holding register	40001 to 49999 or	DW0 to DW19998 or	MW0 to CPU limit
			400001 to 465535	DW0 to DW131068	
06	Write word	Holding register	40001 to 49999 or	DW0 to DW19998 or	
			400001 to 465535	DW0 to DW131068	
16	Write words	Holding register	40001 to 49999 or	DW0 to DW19998 or	
			400001 to 465535	DW0 to DW131068	

The table below shows the supported Modbus diagnostic functions.

S7-1200 "MB_SLAVE" Modbus diagnostic functions		
Codes	**Subfunction**	**Description**
08	0000H	Return query data echo test: The "MB_SLAVE" instruction returns the echo of a received data word to a Modbus master.
08	000AH	Clear communication event counter: The "MB_SLAVE" instruction clears the communication event counter that is used for Modbus function 11.
11	-	Get communication event counter: The "MB_SLAVE" instruction uses an internal communication event counter for recording the number of successful Modbus read and write requests that are sent to the Modbus slave. The counter is not incremented on any Function 8, Function 11, or broadcast requests. It is also not incremented on any requests that result in a communication error (for example, parity or CRC errors).

The "MB_SLAVE" instruction supports broadcast write requests from Modbus masters as long as the requests include access to valid addresses.

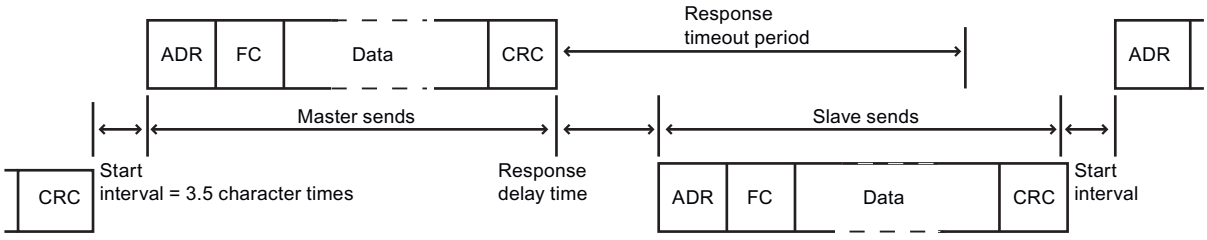
Regardless of the validity of a request, "MB_SLAVE" gives no response to a Modbus master as the result of a broadcast request.

Rules of Modbus slave communication

- "MB_COMM_LOAD" must be executed to configure a port, before the "MB_SLAVE" instruction can communicate with this port.
- If a port is to respond as a slave to a Modbus master, then that port cannot be used by "MB_MASTER (Page 3757)". Only one instance of "MB_SLAVE" can be used with a given port.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must control the communication process by polling the "MB_SLAVE" instruction for completed send and receive operations.
- The "MB_SLAVE" instruction must be executed periodically at a rate that allows it to make a timely response to incoming requests from a Modbus master. It is therefore advisable to call the instruction in a cyclic program OB. Calling the "MB_SLAVE" instruction in an interrupt OB is possible but not advisable since it can lead to long delays in execution.

Frequency of execution of "MB_SLAVE"

The "MB_SLAVE" instruction must be executed periodically to receive each request from the Modbus master and to respond as required. The frequency of execution of "MB_SLAVE" is dependent upon the specified response timeout period of the Modbus master. This is illustrated in the following diagram.



The response timeout period is the amount of time a Modbus master waits for the start of a response from a Modbus slave. This time period is not defined by the Modbus protocol, but rather by a parameter of each Modbus master. The frequency of execution (time between one execution and the next execution) of "MB_SLAVE" must be based on the particular parameters of your Modbus master. As a minimum, you should execute "MB_SLAVE" twice within the response timeout period of the Modbus master.

Parameters

The following table shows the parameters of the "MB_SLAVE" instruction:

Parameter	Declaration	Data type	Memory area	Description
MB_ADDR	Input	USINT	I, Q, M, D, L or constant	Station address of the Modbus slave (address range: 0 to 255)
MB_HOLD_REG	Input	VARIANT	D	Pointer to the Modbus holding register DB. The DB must be created with the "Standard - compatible with S7-300/400" access type.
NDR	Output	BOOL	I, Q, M, D, L	New data ready: • 0: No new data • 1: Indicates that new data has been written by the Modbus master
DR	Output	BOOL	I, Q, M, D, L	Read data: • 0: No data read • 1: Indicates that data has been read by the Modbus master
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error detected • 1: Error, a corresponding error code is output in the STATUS
STATUS	Output	WORD	I, Q, M, D, L	Error code

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

STATUS parameter

STATUS* (W#16#....)	Description
80C8	The specified response timeout (refer to RCVTIME or MSGTIME) is "0".
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission within the wait time. This error is also generated during hardware flow control if the recipient does not detect CTS within the wait time.
80D2	The send request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full
80E1	The message was terminated as a result of a parity error
80E2	The message was terminated as a result of a message frame error

STATUS* (W#16#....)	Description	
80E3	The message was terminated as a result of a overrun error	
80E4	The message was terminated as a result of the specified length exceeding the total buffer size	
8180	Invalid value for the port ID.	
8186	Invalid Modbus station address	
8187	Invalid pointer to MB_HOLD_REG-DB	
818C	Pointer to a type safe DB type MB_HOLD_REG (must be a Classic DB type)	
Response code sent to Modbus master (B#16#...)		
8380	No response	CRC error
8381	01	Function code not supported or not supported in a broadcast
8382	03	Data length error
8383	02	Error in the data address or address outside the valid range of MB_HOLD_REG
8384	03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Instance DB of the "MB_SLAVE" instruction

Static variables of the instance DB

The following table describes the static variables of the instance DB of the instruction that you can use in the user program. Your program can write values to the HR_Start_Offset and Extended_Addressing variables and control the Modbus slave operations.

The other variables can be read to monitor the Modbus status.

Variable	Data type	Description
HR_Start_Offset	WORD	Start address of the Modbus holding register (default = " 0")
Extended_ Addressing	BOOL	Configuring addressing: • 0: Default address area (1 byte) • 1: Extended address area (2 bytes)
Request_Count	WORD	Total number of queries received by the slave
Slave_Message_Count	WORD	Number of queries sent to this specific slave
Bad_CRC_Count	WORD	Number of received queries with CRC errors
Broadcast_Count	WORD	Number of received broadcast queries
Exception_Count	WORD	Number of Modbus-specific errors that require the return of an exception
Success_Count	WORD	The number of requests received for this specific slave without protocol errors

HR_Start_Offset

The addresses of the Modbus holding register start at 40001 or 400001. These addresses correspond to the start address of the holding register in the target system memory. Using the HR_Start_Offset variable, you can set the offset to a different start address.

Example: A holding register starts at MW 100 and has a length of 100 WORD. With an offset of 20 in the HR_Start_Offset parameter, the holding register begins at address 40021 instead of 40001. Each address below 40021 and above 400119 causes an addressing error.

	HR_Start_Offset = 0			**HR_Start_Offset = 20**	
	Modbus word address	S7-1200 byte address		Modbus word address	S7-1200 byte address
Minimum	40001	MW100		40021	MW100
Maximum	40099	MW198		40119	MW198

Extended_Addressing

To address the Modbus slave, a single byte (default address range) or a double byte (extended address range) can be configured. Extended addressing is used to address more than 247 devices in a single network. If you decide on extended addressing, you can address a maximum of 64,000 addresses. Below, you will see a frame of Modbus function 1 as an example.

Table 11-100 Slave address with one byte (byte 0)

Function 1	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	
Request	Slave address	F code	Start address		Length of the coils		
Valid response	Slave address	F code	Length	Coil data			
Error response	Slave address	0x81	E code				

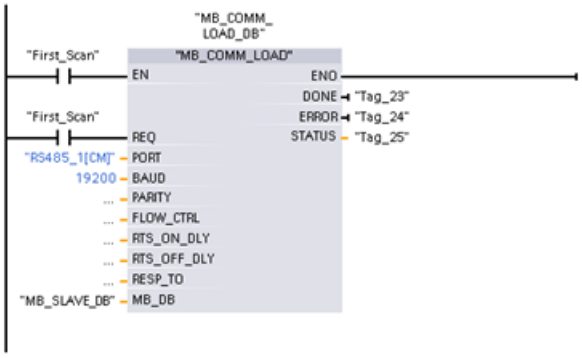
Table 11-101 Slave address with two bytes (byte 0 and byte 1)

Function 1	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**
Request	Slave address		F code	Start address		Length of the coils	
Valid response	Slave address		F code	Length	Coil data		
Error response	Slave address		0x81	F code			

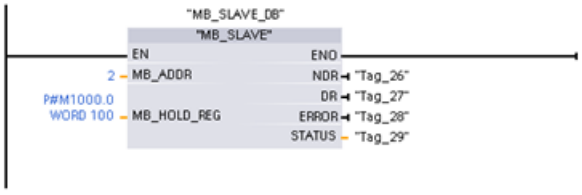
Sample program for a Modbus slave

Networks (LAD)

Network 1: Initialize parameters of the RS-485 module only once during the first cycle.



Network 2: Check the requests of the Modbus master in every cycle. 100 words starting at MW1000 are configured for the Modbus holding register.



MODBUS (TCP)

MODBUS (TCP)

MB_CLIENT: Communicating via PROFINET as a Modbus TCP client

Description of MB_CLIENT

Description

The "MB_CLIENT" instruction communicates as a Modbus TCP client via the PROFINET connection of the S7-1200 CPU. To use the instruction, you do not require any additional hardware module. With the "MB_CLIENT" instruction, you establish a connection between the client and the server, send requests and receive responses and control connection termination of the Modbus TCP server.

Parameters

The following table shows the parameters of the "MB_CLIENT" instruction:

Parameter	Declaration	Data type	Description
REQ (Page 3773)	Input	BOOL	Communication request with the Modbus TCP server The REQ parameter is level-controlled. This means that as long as the input is set (REQ=true), the instruction sends communication requests. • The instance DB for other clients is blocked with the communications request. • Changes to the input parameters will not become effective until the server has responded or an error message has been output. • If the parameter REQ is set again during an ongoing Modbus request, no additional transmission takes place afterwards.
DISCONNECT (Page 3773)	Input	BOOL	With this parameter, you control the establishment and termination of the connection to the Modbus server: • 0: Establish a communications connection to the specified IP address and port number. • 1: Disconnect the communication connection. No other function is executed during connection termination. The value 7003 is output at the STATUS parameter after successful connection termination. The request is sent immediately if the REQ parameter is set during connection establishment.
CONNECT_ID	Input	UINT	Unique ID to identify the connection. Each instance of the instructions "MB_CLIENT" and "MB_SERVER (Page 3778)" must be assigned a unique connection ID.
IP_OCTET_1	Input	USINT	1. Octet of the IP address* of the Modbus TCP server.
IP_OCTET_2	Input	USINT	2. Octet of the IP address* of the Modbus TCP server.
IP_OCTET_3	Input	USINT	3. Octet of the IP address* of the Modbus TCP server.
IP_OCTET_4	Input	USINT	4. Octet of the IP address* of the Modbus TCP server.
IP_PORT	Input	UINT	IP port number of the server to which the client establishes the connection and communicates using the TCP/IP protocol (default value: 502).
MB_MODE (Page 3774)	Input	USINT	Selects the mode of the request (read, write or diagnostics).
MB_DATA_ADDR (Page 3774)	Input	UDINT	Start address of the data accessed by the "MB_CLIENT" instruction.
DATA_LEN	Input	UINT	Data length: Number of bits or words for the data access (see "MB_MODE and MB_DATA_ADDR parameters" - data length).
MB_DATA_PTR (Page 3775)	InOut	VARIANT	Pointer to the Modbus data register: The register is a buffer for the data received from the Modbus server or to be sent to the Modbus server. The pointer must reference a global data block with standard access. The number of bits addressed must be divisible by 8.
DONE	Out	BOOL	The bit at output parameter DONE is set to "1" as soon as the last job is completed without errors.

Parameter	Declaration	Data type	Description
BUSY	Out	BOOL	• 0: No "MB_CLIENT " job in progress • 1: "MB_ CLIENT " job in progress
ERROR	Out	BOOL	• 0: No error • 1: Error occurred. The cause of error is indicated by the STATUS parameter.
STATUS (Page 3776)	Out	WORD	Error code of the instruction.
* 8-bit long component of the 32-bit IPv4 IP address of the Modbus TCP server.			

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Note

Consistent input data during an "MB_CLIENT" call

When a Modbus client calls a Modbus instruction, the status of the input parameters is stored internally and then compared at the next call. The comparison is used to determine whether this particular call initialized the current request. Several "MB_CLIENT" calls can be executed if you use a common instance DB. The values of the input parameters must not be changed while an "MB_CLIENT" instance is being executed. If the input parameters are changed during execution, "MB_CLIENT" cannot be used to check whether or not the instance is currently being executed.

Multiple client connections

A Modbus TCP client can support several TCP connections and the maximum number of connections depends on the CPU being used. The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections. Modbus TCP connections can also be shared by client and/or server connections.

With individual client connections, remember the following rules:

- Each "MB_CLIENT" connection must use a unique instance DB.
- For each "MB_CLIENT" connection, a unique server IP address must be specified.
- Each "MB_CLIENT" connection requires a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- Unique numbers for the IP port may or may not be required depending on the server configuration.

Static tags of the instruction

The following table describes the editable static tags of the instance data block of the "MB_CLIENT" instruction.

Tag	Data type	Start value	Description
Blocked_Proc_Time-out	REAL	3.0	Wait time in seconds before the static tag ACTIVE is reset if there is a blocked Modbus instance. This can, for example, occur if a client request is output and the execution of the client function aborts before the request was fully executed. The maximum wait time is 55 seconds.
MB_Transaction_ID	WORD	1	Transaction ID of the Modbus TCP protocol. The start value of "1" should only be changed if the Modbus TCP server requires a different value.
MB_Unit_ID	BYTE	255	Modbus device detection: A Modbus TCP server is addressed using its IP address. For this reason, the MB_UNIT_ID parameter is not used in the case of Modbus TCP addressing. The MB_UNIT_ID parameter corresponds to the field of the slave address in the Modbus RTU protocol. If a Modbus TCP server is used as a gateway for a Modbus RTU protocol, the slave device in the serial network can be identified using MB_UNIT_ID. The MB_UNIT_ID parameter would in this case forward the request to the correct Modbus RTU slave address. Please note that some Modbus TCP devices may require the MB_UNIT_ID parameter for the initialization within a limited value range.
RCV_TIMEOUT	REAL	2.0	Time interval in seconds in which the "MB_CLIENT" instruction waits for a response from the server.
Connected	BOOL	0	Indicates whether the connection to the assigned client has been established or not: 1 = connected, 0 = not connected.

See also

MB_CLIENT example 1: Send several requests via a TCP connection (Page 3784)

MB_CLIENT example 2: Send multiple requests via several TCP connections (Page 3785)

MB_CLIENT example 3: Coordinate several requests (Page 3786)

REQ and DISCONNECT parameters

Description

If no instance of the "MB_CLIENT" instruction is executing and if the value of the DISCONNECT parameter is "0", a new job executes if REQ=1. If there is not yet a connection, this is established during execution.

If the same instance of the "MB_CLIENT" instruction executes again (DISCONNECT = 0 and REQ = 1), before the active job was executed, this is not executed on completion of the active job. A new job can only be started on completion of the active job (REQ = 1).

You can monitor the status of execution with the DONE parameter. You can use this to monitor the status of execution if the "MB_CLIENT" instruction is executed sequentially.

See also

Description of MB_CLIENT (Page 3770)

MB_MODE and MB_DATA_ADDR parameters

Description

Instead of a function code, the "MB_CLIENT" instruction uses the MB_MODE parameter. The MB_DATA_ADDR parameter is used to specify the Modbus start address of the data you want to access. The combination of the parameters MB_MODE and MB_DATA_ADDRdefines the function code used in the current Modbus message.

The following table shows the relationship between the MB_MODE parameter, the Modbus function and the address space.

MB_MODE	Modbus function	Data length	Function and data type	MB_DATA_ADDR
0	01	1 to 2000	Read output bits: 1 to 2000 bits per call	1 to 9999
0	02	1 to 2000	Read input bits: 1 to 2000 bits per call	10001 to 19999
0	03	1 to 125	Read holding register: 1 to 125 WORD per call	40001 to 49999
0	04	1 to 125	Read input words: 1 to 125 WORD per call	30001 to 39999
1	05	1	Write an output bit: One bit per call	1 to 9999
1	06	1	Write a holding register: 1 WORD per call	40001 to 49999
1	15	2 to 1968	Write multiple output bits: 2 to 1968 bits per call	1 to 9999
1	16	2 to 123	Write several holding registers: 2 to 123 WORD per call	40001 to 49999
2	15	1 to 1968	Write one or more output bits: 1 to 1968 bits per call	1 to 9999
2	16	1 to 123	Write one or more holding registers: 1 to 123 WORD per call	40001 to 49999

MB_MODE	Modbus function	Data length	Function and data type	MB_DATA_ADDR
11	11	0	Read status word and event counter of server communication: • The status word reflects the the processing status (0 - not processing, 0xFFFF - processing). • The event counter is incremented each time a message is sent successfully. The MB_DATA_ADDR and MB_DATA_LEN parameters of the "MB_CLIENT" instruction are not evaluated when this function executes.	-
80	08	1	Check the server status with the error code 0x0000 (return loop test - the server sends the request back): 1 WORD per call	-
81	08	1	Reset the event counter of the server with the error code 0x000A: 1 WORD per call	
3 to 10, 12 to 79, 82 to 255			Reserved	

See also

Description of MB_CLIENT (Page 3770)

MB_DATA_PTR parameter

Description

The MB_DATA_PTR parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

For a buffer in the memory area (M), use a pointer in the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

The MB_DATA_PTR parameter uses a communications buffer:

- For the communication functions of the "MB_CLIENT" instruction:
  - Reading and writing of 1 bit of data of the Modbus server addresses 00001 to 09999 and 10001 to 19999.
  - Reading of 16-bit WORD data of the Modbus server addresses 30001 to 39999 and 40001 to 49999.
  - Writing 16-bit WORD data of the Modbus server addresses 40001 to 49999.
- During data transmission (length: bit or WORD) from or to the global DB or the memory area (M) that you assigned with the MB_DATA_PTR parameter.

If you use a data block for the buffer in the MB_DATA_PTR parameter, you will need to assign data types to the DB elements.

- Use the 1-bit data type BOOL for a Modbus bit address
- Use a 16-bit data type such as WORD, UINT, INT or REAL for a Modbus WORD address.
- Use a 32-bit data type (double word) such as DWORD, DINT or REAL for two Modbus WORD addresses.
- With MB_DATA_PTR, you can also access complex DB elements such as:
  - Standard arrays
  - Structures with unique element names
  - Complex structures with unique naming of the elements and data type lengths of 16 or 32 bits.
- The data areas for the MB_DATA_PTR parameter can also be in different global data blocks (or in different memory areas). You can, for example, use a data block for the the read jobs and another one for the write jobs or a separate data block for each "MB_CLIENT" station.

See also

Description of MB_CLIENT (Page 3770)

Parameter STATUS

Parameter STATUS (general status information)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active (REQ=0).
7001	First call with REQ=1: Processing started; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). Processing already active; BUSY has the value 1.
7003	Connection is being terminated.
7004	Connection established and monitored. No job processing active.
7005	Data was sent.
7006	Data was received.
80BB	Invalid value at ACTIVE_EST parameter (identifier for the type of connection establishment, see T_CON_PARAM): • Only passive connection establishment permitted for server (ACTIVE_EST = FALSE). • Only active connection establishment permitted for client (ACTIVE_EST = TRUE).
8380	Received Modbus frame has incorrect format or too few bytes were received.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS (protocol error)

STATUS* (W#16#)	Code of the response to the Modbus client (B#16#)	Description
8381	01	Function code is not supported.
8382	03	Error in data length.
8383	02	Error in the data address or access outside the address area of MB_DATA_PTR (Page 3775).
8384	03	Error in data value.
8385	03	Error codes of diagnostics not supported (function code 08).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Parameter STATUS (parameter error)

In addition to the errors listed in the following table, errors are also possible with the "MB_CLIENT" instruction caused by the communications instructions ("TCON", "TDISCON", "TSEND" and "TRCV").

STATUS* (W#16#)	Description
80C8	No response of the server in the defined period. Check the connection to the Modbus server. This error is only reported on completion of the configured repeated attempts. If the "MB_CLIENT" instruction does not receive an answer with the originally transferred transaction ID (MB_TRANSACTION_ID tag) within the defined period, this error code is output.
8188	The MB_MODE parameter has an invalid value.
8189	Invalid addressing of data at the MB_DATA_ADDR parameter.
818A	Invalid data length at the MB_DATA_LEN parameter.
818B	The MB_DATA_PTR parameter has an invalid pointer. You should also check the values of the MB_DATA_ADDR (Page 3774) and MB_DATA_LEN parameters.
818C	• The MB_DATA_PTR (Page 3775) pointer references an an optimized data block. Either use a data block with standard access or a memory area • Timeout at parameter BLOCKED_PROC_TIMEOUT (see static tags of instruction). The limit of 55 seconds has been exceeded.
818D	The transaction ID (MB_TRANSACTION_ID tag) does not correspond to the one sent originally (see static tags of instruction).
8200	• A further Modbus request is currently being processed via the port. • Another instance of MB_CLIENT with the same connection parameters is processing an existing Modbus request.
8380	Received block of transferred Modbus data is not well-formed or too few bytes were received.
8386	Received function code does not match the one sent originally.
8387	• The assigned connection ID is different from that used for previous requests. Only one connection ID can be used for each instance DB of the "MB_CLIENT" instruction. • The error code is also output when the ID of the Modbus TCP protocol received by the server is not "0".

STATUS* (W#16#)	Description
8388	The Modbus server sent a different data length than was requested. This error occurs only when using the Modbus functions 15 or 16.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

Error codes of internally used communications instructions.

With the "MB_CLIENT" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND" and "TRCV") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_CLIENT" instruction. The error codes are displayed for the respective instruction under STATUS in the Static section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

See also

Description of MB_CLIENT (Page 3770)

MB_HOLD_REG parameter (Page 3782)

MB_SERVER: Communicating via PROFINET as a Modbus TCP server

Description of MB_SERVER

Description

The "MB_SERVER" instruction communicates as a Modbus TCP server via the PROFINET connection of the S7-1200 CPU. To use the instruction, you do not require any additional hardware module. The "MB_SERVER" instruction processes connection requests of a Modbus TCP client, receives requests from Modbus functions and sends responses.

Notice

Security information

Note that each client of the network is given read and write access to the process image inputs and outputs and to the data block or bit memory area defined by the Modbus holding register.

The option is available to restrict access to an IP address to prevent unauthorized read and write operations. Note, however, that the shared address can also be used for unauthorized access.

Parameters

The following table shows the parameters of the "MB_SERVER" instruction:

Parameter	Declaration	Data type	Description
DISCONNECT	Input	BOOL	The instruction "MB_SERVER" enters into a passive connection with a partner module, which means the server responds to a TCP connection request from each requesting IP address. You can use this parameter to control when a connection request is accepted: • 0: A passive connection is established when there is no communications connection. • 1: Initialization of the connection termination. If the input is set, no other operations are executed. The value 7003 is output at the STATUS parameter after successful connection termination.
CONNECT_ID	Input	UINT	The parameter uniquely identifies a connection within the CPU. Each individual instance of the instructions "MB_CLIENT (Page 3770)" and "MB_SERVER" must have a unique CONNECT_ID parameter.
IP_PORT	Input	UINT	Start value = 502. The number of IP ports defines which IP port is monitored for connection requests of the Modbus client. The following TCP port numbers must not be used for the passive connection of the "MB_SERVER" instruction: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.
MB_HOLD_REG (Page 3782)	InOut	VARIANT	Pointer to the Modbus holding register of the "MB_SERVER" instruction: Use a global data block with standard access as holding register. The holding register contains the values that may be accessed by a Modbus client using the Modbus functions 3 (read), 6 (write) and 16 (read).
NDR	Output	BOOL	"New Data Ready": • 0: No new data • 1: New data written by the Modbus client written
DR	Output	BOOL	"Data Read": • 0: No data read • 1: Data read by the Modbus client
ERROR	Output	BOOL	If an error occurs during the call of the "MB_SERVER" instruction, the output of the ERROR parameter is set to TRUE. Detailed information about the cause of the problem is indicated by the STATUS parameter.
STATUS (Page 3783)	Output	WORD	Error code of the instruction.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Mapping of Modbus addresses to the process image

The "MB_SERVER" instruction allows incoming Modbus functions (1, 2, 4, 5 and 15) direct read and write access to the process image input and output of the S7-1200 CPU (use of the data types BOOL and WORD).

For the data transfer of the function codes 3, 6 and 16, the holding register (MB_HOLD_REG parameter) must be defined longer than one byte. The following table shows the mapping of the Modbus addresses to the process image of the CPU.

Modbus function						S7-1200	
Code	**Function**	**Data area**	**Address space**			**Data area**	**CPU address**
01	Read: Bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
02	Read: Bits	Input	10001	to	18192	Process image input	I0.0 to I1023.7
04	Read: WORD	Input	30001	to	30512	Process image input	IW0 to IW1022
05	Write: Bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
15	Write: Bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7

Incoming Modbus alarms with the function codes 3, 6 and 16 write to or read from the Modbus holding registers (you specify the holding registers with the MB_HOLD_REG parameter).

Multiple server connections

You can create multiple Server connections. This allows a single CPU to establish connections to more than one Modbus TCP client at the same time.

A Modbus TCP server can support several TCP connections and the maximum number of connections depends on the CPU being used.

The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections.

Modbus TCP connections can also be shared by client and/or server connections.

In the case of Server connections, remember the following rules:

- Each "MB_SERVER" connection must use a unique instance DB.
- Each "MB_SERVER" connection must be created with a unique IP port number. Only one connection is supported for each port.
- Each "MB_SERVER" connection must use a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- For each connection, the "MB_SERVER" instruction must be called individually.

Modbus diagnostics functions

The table below contains a description of the Modbus diagnostics functions.

Code	Subfunction	Description
08	0x0000	Echo test: The "MB_SERVER" instruction receives a data word and returns this unchanged to the Modbus master.
08	0x000A	Reset event counter: The "MB_SERVER" instruction resets the event counter for communication that is used for Modbus function 11.
11	-	Fetch event counter of the communication: The "MB_SERVER" instruction uses an internal event counter for communication to record the number of successfully executed read and write requests sent to the Modbus server. The event counter is not incremented by the functions 8, 11 or broadcast requests. The same applies to requests that result in a communications error (for example parity errors or CRC errors). The broadcast function is not available for Modbus TCP because only one client/server connection can exist at any one time.

Static tags of the instruction

The following table describes the static tags of the instance data block of the "MB_SERVER" instruction used in the program. You can write the HR_Start_Offset tag. You can read the other tags to monitor the Modbus status.

Tag	Data type	Start value	Description
HR_Start_Offset	WORD	0	Assign the start address of the Modbus holding register.
Request_Count	WORD	0	Total number of requests received by the server.
Server_Message_Count	WORD	0	Total number of received alarms for the relevant server.
Xmt_Rcv_Count	WORD	0	Counter for detecting the number of transfers during which an error occurred. The counter is also incremented if an invalid Modbus alarm is received.
Exception_Count	WORD	0	Counter for detecting the number of errors specifically for Modbus cause an exception.
Success_Count	WORD	0	Counter for detecting the number of requests that contain no error in the transferred protocol.
Connected	BOOL	0	Indicates whether the connection to the assigned client has been established or not: 1 = connected, 0 = not connected.

Example: Addressing via static tag HR_Start_Offset

The addresses of the Modbus holding register start at 40001. These addresses correspond to the address space of the CPU memory area for the holding register. You can also define the HR_Start_Offset tag so that the Modbus holding register has a start address other than 40001.

Example: The holding register begins at MW100, and has a length of 100 WORD. An offset value in the HR_Start_Offset parameter means that the start address of the holding register is

moved from 40001 to 40021. Whenever the holding register is addressed below the address 40021 and above the address 40119, this causes an error.

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (WORD)	40001	40099
	S7-1200 address	MW100	MW298
20	Modbus address (WORD)	40021	40119
	S7-1200 address	MW100	MW298

See also

MB_SERVER example: Multiple TCP connections (Page 3787)

MB_HOLD_REG parameter

Description

The MB_HOLD_REG parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

As pointer to a buffer in the memory area (M), use the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

The following table shows examples of mapping Modbus addresses to the holding register for the Modbus functions 3 (read WORD), 6 (write WORD) and 16 (write several WORD). The upper limit for the number of addresses in the data block is decided by the maximum work memory of the CPU. If you use a memory area, the upper limit for the addresses is decided by the size of the memory area of the CPU.

Modbus addresses	MB_HOLD_REG parameter - examples		
P#M100.0 WORD 5	P#DB10.DBx0.0 WORD 5	"Recipe".ingredient	
40001	MW100	DB10.DBW0	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	"Recipe".ingredient[5]

See also

Description of MB_SERVER (Page 3778)

Parameter STATUS

Parameter STATUS (general status information)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active (REQ=0).
7001	First call with REQ=1: Processing started; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). Processing already active; BUSY has the value 1.
7003	Connection is being terminated.
7004	Connection established and monitored. No job processing active.
7005	Data was sent.
7006	Data was received.
80BB	Invalid value at ACTIVE_EST parameter (identifier for the type of connection establishment, see T_CON_PARAM): • Only passive connection establishment permitted for server (ACTIVE_EST = FALSE). • Only active connection establishment permitted for client (ACTIVE_EST = TRUE).
8380	Received Modbus frame has incorrect format or too few bytes were received.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS (parameter error)

STATUS* (W#16#)	Code of the response to the Modbus server (B#16#)	Description
8187	No response	The MB_HOLD_REG parameter has an invalid pointer. Data area is too small.
818C	No response	• The MB_HOLD_REG parameter references an an optimized data block. Either use a data block with standard access or a memory area • Error due to timeout of execution (more than 55 seconds).
8381	01	Function code is not supported.
8382	03	Error in data length
8383	02	Error in data address or access outside the address area of the holding register (MB_HOLD_REG (Page 3782) parameter).
8384	03	Error in data value
8385	03	Value of the diagnostic code is not supported (only with function code 08).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Note

Error codes of internally used communications instructions.

With the "MB_SERVER" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND" and "TRCV") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_SERVER" instruction. The error codes are displayed for the respective instruction under STATUS in the Static section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

See also

Description of MB_SERVER (Page 3778)

Examples

MB_CLIENT example 1: Send several requests via a TCP connection

Description

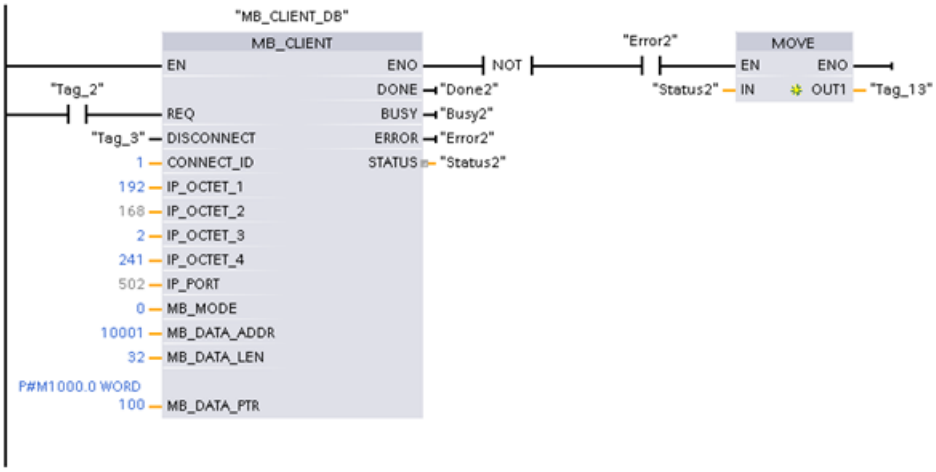
Several requests of the Modbus Client can be sent via a TCP connection. To do this, use the same instance DB, the same connection ID and the same port number.

Only one client can be active at any one time. After processing of a client has been completed, the next client is processed. The order of execution must be defined in the program.

In the following sample program, the value of the STATUS output parameter is also copied.

Network 1: Modbus function 1 - 16 read output bits

Network 2: Modbus function 2 - 32 read input bits



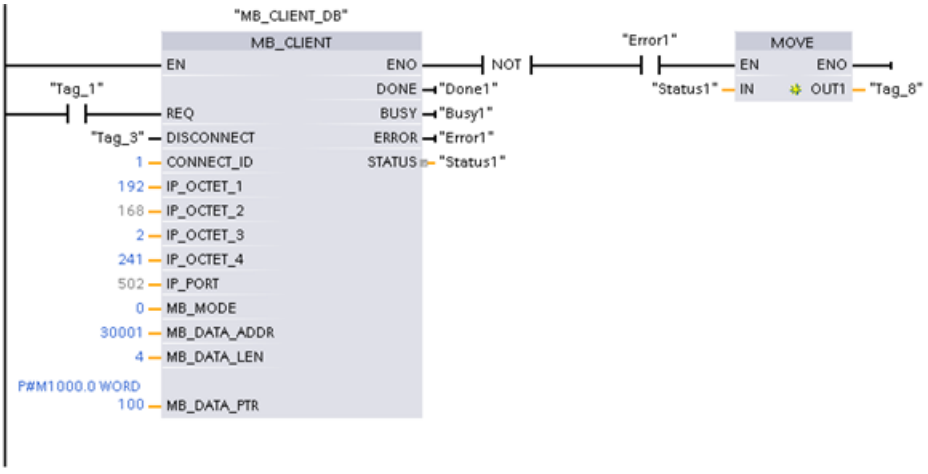
MB_CLIENT example 2: Send multiple requests via several TCP connections

Description

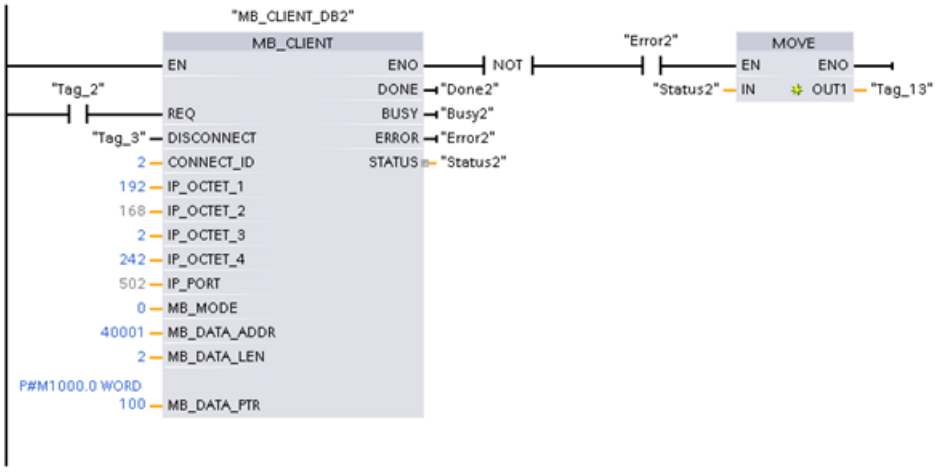
Requests from the Modbus client can be sent via different TCP connections. If you require this, use a different instance DB and a different connection ID.

Use a different port number, if the connections are to the same Modbus server. If the connections are to different Modbus servers, you can freely assign the port number.

Network 1: Modbus function 4 - read input (WORD)



Network 2: Modbus function 3 - read holding register (WORD)

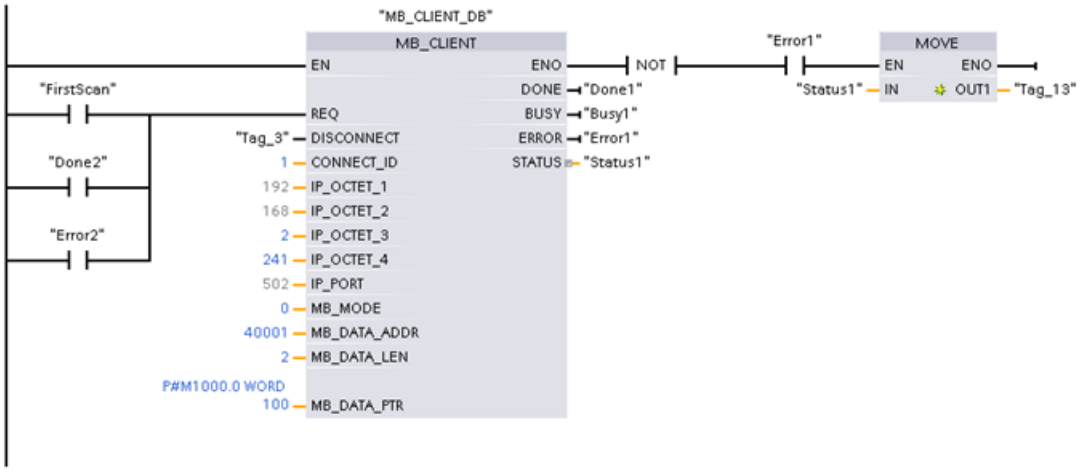


MB_CLIENT example 3: Coordinate several requests

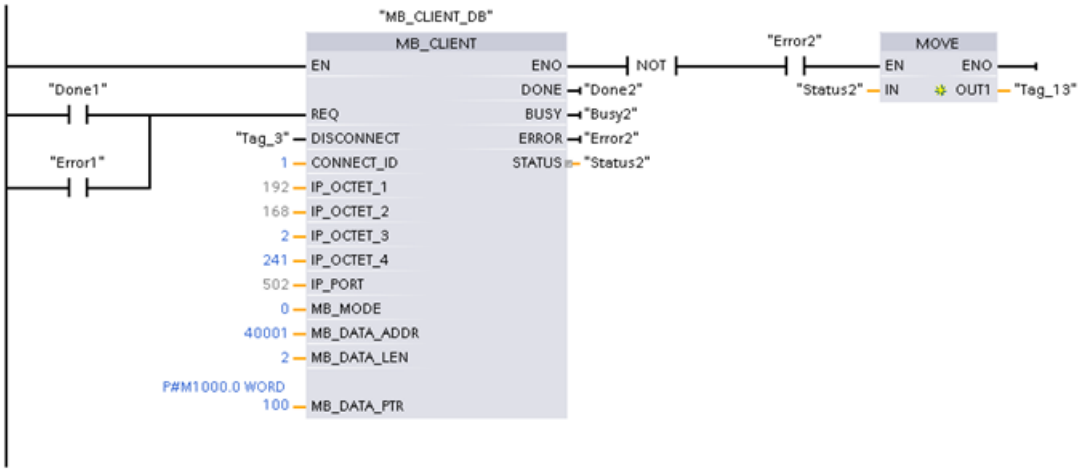
Description

Make sure that the individual Modbus requests are executed. You control coordination of requests with the program. The following example demonstrates how the output parameters of the first and second client request can be used to coordinate execution of the instructions.

Network 1: Modbus function 3 - read holding register (WORD)



Network 2: Modbus function 3 - read holding register (WORD)



MB_SERVER example: Multiple TCP connections

Description

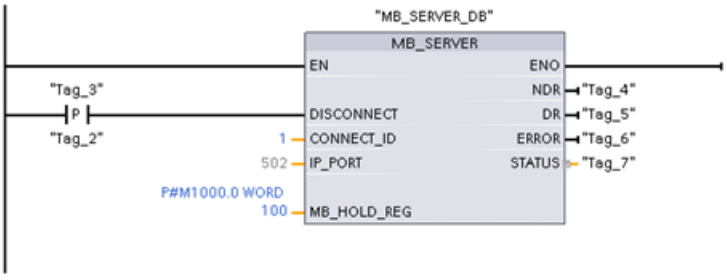
You can use several Modbus TCP server connections. To do this, the "MB_SERVER" instruction must be called independently for each connection.

Every connection requires the following:

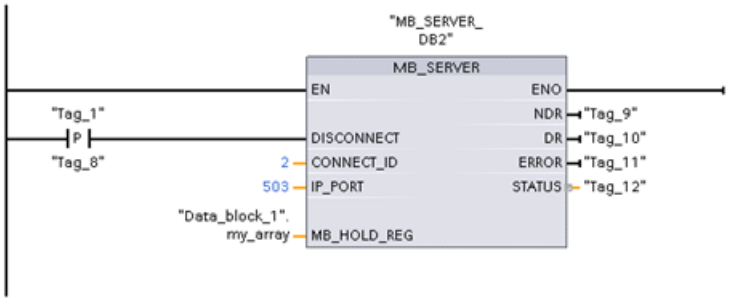
- An independent instance data block of the instruction
- A unique connection ID
- A separate IP port (on the S7-1200, only one connection is permitted per IP port)

To optimize the performance "MB_SERVER" should be executed once per program cycle for each connection.

Network 1: Connection #1 with associated IP port connection ID and instance DB



Network 2: Connection #1 with associated IP port connection ID and instance DB



MODBUS (TCP)

MB_CLIENT: Communicating via PROFINET as a Modbus TCP client

Description of MB_CLIENT

Description

The "MB_CLIENT" instruction communicates as a Modbus TCP client via the PROFINET connection. With the "MB_CLIENT" instruction, you establish a connection between the client and the server, send Modbus requests and receive responses and control connection termination of the Modbus TCP client.

In V3.0 and higher, the "MB_CLIENT" instruction can be used for the S7-1500 as well as for the S7-1200 version 4.0 and higher. The connection can take place via the local interface of the CPU or CM/CP.

To use the instruction, you do not require an additional hardware module.

Multiple client connections

A Modbus TCP client can support several TCP connections and the maximum number of connections depends on the CPU being used. The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections. Modbus TCP connections can also be shared by "MB_CLIENT" and/or "MB_SERVER" instances.

With individual client connections, remember the following rules:

- Each "MB_CLIENT" connection must use a unique instance DB.
- For each "MB_CLIENT" connection, a unique server IP address must be specified.

- Each "MB_CLIENT" connection requires a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- Unique numbers for the IP port may or may not be required depending on the server configuration.

Parameters

The following table shows the parameters of the "MB_CLIENT" instruction:

Parameter	Declaration	Data type	Description
REQ (Page 3792)	Input	BOOL	Communication request with the Modbus TCP server The REQ parameter is level-controlled. This means that as long as the input is set (REQ=true), the instruction sends communication requests. • The instance DB for other clients is blocked with the communications request. • Changes to the input parameters will not become effective until the server has responded or an error message has been output. • If the parameter REQ is set again during an ongoing Modbus request, no additional transmission takes place afterwards.
DISCONNECT (Page 3792)	Input	BOOL	With this parameter, you control the establishment and termination of the connection to the Modbus server: • 0: Establish communications connection to the connection partner configured at the CONNECT parameter (see CONNECT parameter). • 1: Disconnect the communication connection. No other function is executed during connection termination. The value 0003 is output at the STATUS parameter after successful connection termination. The Modbus request is sent immediately if the REQ parameter is set during connection establishment.
MB_MODE (Page 3792)	Input	USINT	Selects the mode of the Modbus request (read, write or diagnostics).
MB_DATA_ADDR (Page 3792)	Input	UDINT	Start address of the data accessed by the "MB_CLIENT" instruction.
MB_DATA_LEN	Input	UINT	Data length: Number of bits or words for the data access (see "MB_MODE and MB_DATA_ADDR parameters" - data length).
MB_DATA_PTR (Page 3794)	InOut	VARIANT	Pointer to the Modbus data register: The register is a buffer for the data received from the Modbus server or to be sent to the Modbus server. The pointer must reference a global data block with optimized access. The number of bits addressed must be divisible by 8.

Parameter	Declaration	Data type	Description
CONNECT (Page 3795)	InOut	VARIANT	Pointer to the structure of the connection description The following structures (system data types) can be used: • TCON_IP_v4: Includes all address parameters required for establishing a programmed connection. When using TCON_IP_v4, the connection is established when calling the instruction "MB_SERVER". • TCON_Configured: Includes the address parameters of a configured connection. When using TCON_Configured, an existing connection is used that was created by the CPU after download of the hardware configuration.
DONE	Out	BOOL	The bit at output parameter DONE is set to "1" as soon as the last job is completed without errors.
BUSY	Out	BOOL	• 0: No Modbus request in progress • 1: Modbus request being processed The output parameter BUSY is not set during connection establishment and termination.
ERROR	Out	BOOL	• 0: No error • 1: Error occurred. The cause of error is indicated by the STATUS parameter.
STATUS (Page 3797)	Out	WORD	Detailed status information of the instruction.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Note

Consistent input data during an "MB_CLIENT" call

When a Modbus client instruction is called, the status of the input parameters is stored internally and then compared at the next call. The comparison is used to determine whether this particular call initialized the current request. Several "MB_CLIENT" calls can be executed if you use a common instance DB. The values of the input parameters must not be changed while an "MB_CLIENT" instance is being executed. If the input parameters are changed during execution, "MB_CLIENT" cannot be used to check whether or not the instance is currently being executed.

Static tags of the instruction

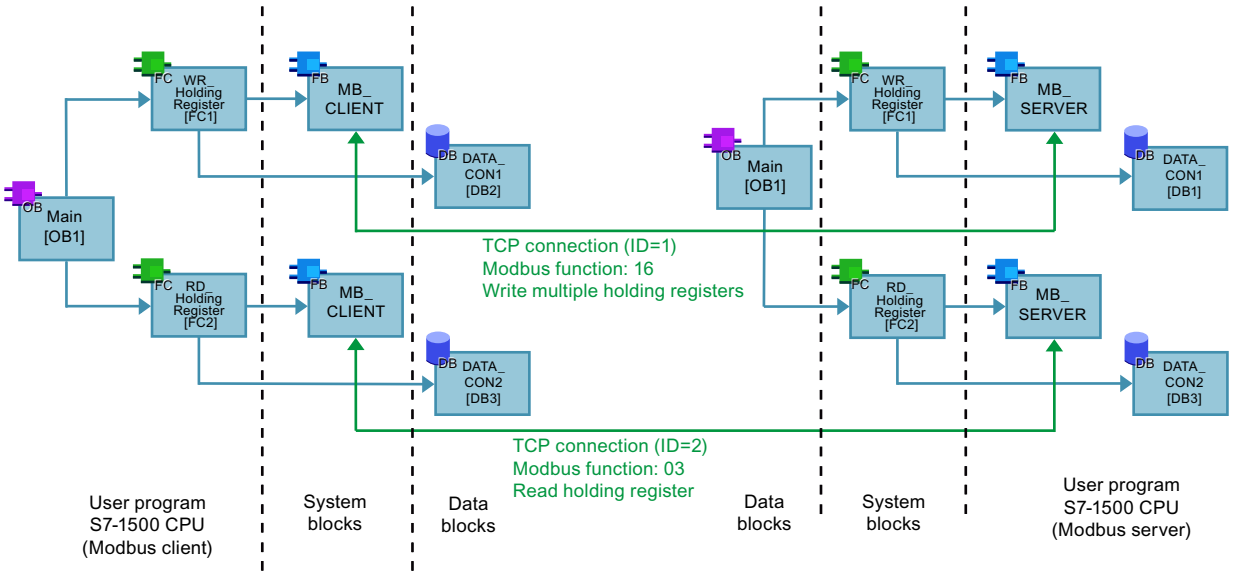
The following table describes the editable static tags of the instance data block of the "MB_CLIENT" instruction.

Tag	Data type	Start value	Description
Blocked_Proc_Time-out	REAL	3.0	Wait time in seconds before the static tag ACTIVE is reset if there is a blocked Modbus instance. This can, for example, occur if a client request is output and the execution of the client function aborts before the request was fully executed. The maximum wait time is 55 seconds.
MB_Transaction_ID	WORD	1	Transaction ID of the Modbus TCP protocol. The start value of "1" should only be changed if the Modbus TCP server requires a different value.
MB_Unit_ID	BYTE	255	Modbus device detection: A Modbus TCP server is addressed using its IP address. For this reason, the MB_UNIT_ID parameter is not used in the case of Modbus TCP addressing. The MB_UNIT_ID parameter corresponds to the field of the slave address in the Modbus RTU protocol. If a Modbus TCP server is used as a gateway for a Modbus RTU protocol, the slave device in the serial network can be identified using MB_UNIT_ID. The MB_UNIT_ID parameter would in this case forward the request to the correct Modbus RTU slave address. Please note that some Modbus TCP devices may require the MB_UNIT_ID parameter for the initialization within a limited value range.
RCV_TIMEOUT	REAL	2.0	Time interval in seconds in which the "MB_CLIENT" instruction waits for a response from the server.
Connected	BOOL	0	Indicates whether the connection to the assigned server has been established or not: 1 = connected, 0 = not connected.

Example

An example project for the Modbus TCP communication between two S7-1500 CPUs can be found in the Service and Support Portal under Entry ID 94766380 (http://support.automation.siemens.com/WW/view/en/94766380).

Two Modbus functions are used in this example. For each Modbus function, a Modbus TCP connection is established using a Modbus block pair (MB_CLIENT and MB_SERVER).



REQ and DISCONNECT parameters

Description

If no instance of the "MB_CLIENT" instruction is executing and if the value of the DISCONNECT parameter is "0", a new job executes if REQ=1. If there is not yet a connection, this is established during execution.

If the same instance of the "MB_CLIENT" instruction executes again (DISCONNECT = 0 and REQ = 1), before the active job was executed, this is not executed on completion of the active job. A new job can only be started on completion of the active job (REQ = 1).

The status of the execution is output by the output parameters. You can use it to monitor the execution status when the "MB_CLIENT" instruction is executed sequentially.

MB_MODE and MB_DATA_ADDR parameters

Description

Instead of a function code, the "MB_CLIENT" instruction uses the MB_MODE parameter. The MB_DATA_ADDR parameter is used to specify the Modbus start address of the data you want to access.

The combination of the parameters MB_MODE, MB_DATA_ADDR and MB_DATA_LEN defines the function code used in the current Modbus message. For example:

- Function code 5
  - MB_MODE=1
  - MB_DATA_ADDR=1
  - MB_DATA_LEN=1
- Function code 15
  - MB_MODE=1
  - MB_DATA_ADDR=1
  - MB_DATA_LEN=2

The following table shows the relationship between the input parameters of the "MB_CLIENT" instruction and the Modbus function.

MB_MODE	MB_DATA_ADDR	MB_DATA_LEN	Modbus function	Function and data type
0	Start address: • 1 to 9999	Data length (bits) per call: • 1 to 2000	01	Read output bits: • 1 to 2000
0	Start address: • 10001 to 19999	Data length (bits) per call: • 1 to 2000	02	Read input bits: • 1 to 2000
0	Start address: • 40001 to 49999 • 400001 to 465535	Data length (WORD) per call: • 1 to 125 • 1 to 125	03	Read holding register: • 0 to 9998 • 0 to 65534
0	Start address: • 30001 to 39999	Data length (WORD) per call: • 1 to 125	04	Read input words: • 0 to 9998
1	Start address: • 1 to 9999	Data length (bits) per call: • 1	05	Writing an output bit: • 0 to 9998
1	Start address: • 40001 to 49999 • 400001 to 465535	Data length (WORD) per call: • 1 • 1	06	Write a holding register: • 0 to 9998 • 0 to 65534
1	Start address: • 1 to 9999	Data length (bits) per call: • 2 to 1968	15	Write multiple output bits: • 0 to 9998

MB_MODE	MB_DATA_ADDR	MB_DATA_LEN	Modbus function	Function and data type
1	Start address: • 40001 to 49999 • 400001 to 465535	Data length (WORD) per call: • 2 to 123 • 2 to 123	16	Write multiple holding registers: • 0 to 9998 • 0 to 65534
2	Start address: • 1 to 9999	Data length (bits) per call: • 1 to 1968	15	Write one or more output bits: • 0 to 9998
2	Start address: • 40001 to 49999 • 400001 to 465535	Data length (WORD) per call: • 1 to 123 • 1 to 123	16	Write one or more holding registers: • 0 to 9998 • 0 to 65534
11	The MB_DATA_ADDR and MB_DATA_LEN parameters are not evaluated when this function is executed.		11	Read status word and event counter of the server: • The status word reflects the the processing status (0 - not processing, 0xFFFF - processing). • The event counter is incremented when the Modbus request was executed successfully. If an error occurred during execution of a Modbus function, a message is sent by the server but the event counter is not incremented.
80	-	Data length (WORD) per call: • 1	08	Check the server status with the diagnostic code 0x0000 (return loop test - the server sends the request back): • 1 WORD per call
81	-	Data length (WORD) per call: • 1	08	Reset the event counter of the server with the diagnostic code 0x000A: • 1 WORD per call
3 to 10, 12 to 79, 82 to 255				Reserved

MB_DATA_PTR parameter

Description

The MB_DATA_PTR parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

For a buffer in the memory area (M), use a pointer in the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

The MB_DATA_PTR parameter uses a communications buffer:

The communications buffer is used:

- For the communication functions of the "MB_CLIENT" instruction:
  - Reading and writing of 1 bit of data of the Modbus server addresses 00001 to 09999 and 10001 to 19999.
  - Reading of 16-bit WORD data of the Modbus server addresses 30001 to 39999 and 40001 to 49999.
  - Writing of 16-bit WORD data of the Modbus server addresses 40001 to 49999.
- During data transmission (length: bit or WORD) from or to the global DB or the memory area (M) that you assigned with the MB_DATA_PTR parameter.

If you use a data block for the buffer in the MB_DATA_PTR parameter, you will need to assign data types to the DB elements.

- Use the 1-bit data type BOOL for a Modbus bit address
- Use a 16-bit data type such as WORD, UINT, INT or REAL for a Modbus WORD address.
- Use a 32-bit data type (double word) such as DWORD, DINT or REAL for two Modbus WORD addresses.
- With MB_DATA_PTR, you can also access complex DB elements such as:
  - Standard arrays
  - Structures with unique element names
  - Complex structures with unique naming of the elements and data type lengths of 16 or 32 bits.
- The data areas for the MB_DATA_PTR parameter can also be in different global data blocks (or in different memory areas). You can, for example, use a data block for the the read jobs and another one for the write jobs or a separate data block for each "MB_CLIENT" station.

CONNECT parameter

Connection descriptions at CONNECT parameter

Two different connection descriptions can be used for the "MB_CLIENT" instruction:

- Programmed connections with the structure TCON_IP_v4
  The connection parameters are stored in the TCON_IP_v4 structure and the connection is set up with the call of the instruction "MB_CLIENT".
- Configured connections with the structure TCON_Configured
  The configured connection has already been established by the CPU. Use the structure TCON_Configured to specify which existing connection is to be used for the instruction.

Each instance of the instruction "MB_CLIENT" requires a unique connection. Create a separate structure TCON_IP_v4 or TCON_Configured for each instance of the instruction to describe the connection.

Connection description for programmed connections

Use the following structure for connection description to TCON_IP_v4 for programmed connections at the CONNECT parameter:

- Make sure that only connections of the type TCP are specified in the TCON_IP_v4 structure.
- The connection may not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). The parameter uniquely identifies a connection within the CPU. Each individual instance of the instruction "MB_CLIENT" must use a unique ID.
4	ConnectionType	BYTE	11	Connection type Select 11 (decimal) for TCP. Other connection types are not permitted. If another connection type (e.g. UDP) is used, a corresponding error message is output at the STATUS parameter of the instruction.
5	ActiveEstablished	BOOL	TRUE	ID for the manner in which the connection is established Select TRUE for active connection establishment.
6 … 9	RemoteAddress	ARRAY [1..4] of BYTE	-	IP address of the connection partner (Modbus server), for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1
10 … 11	RemotePort	UINT	502	Port number of the remote connection partner (value range: 1 to 49151). Use the IP port number of the server to which the client establishes the connection and communicates using the TCP/IP protocol (default value: 502).
12 … 13	LocalPort	UINT	0	Port number of the local connection partner: • Port numbers: 1 to 49151 • Any port: "0"

Note

Migration of the instruction "MB_CLIENT" version 2.1

The parameters CONNECT_ID, IP_PORT and IP_OCTET_x are mapped in version 3.0 of the "MB_CLIENT" instruction in the TCON_IP_v4 structure:

- The parameter CONNECT_ID of the "MB_CLIENT" V2.1 instruction corresponds to the ID parameter of TCON_IP_v4.
- The parameter IP_PORT of the "MB_CLIENT" V2.1 instruction corresponds to the RemotePort parameter of TCON_IP_v4.
- The four parameters IP_OCTET_x of the "MB_CLIENT" V2.1 instruction correspond to the array of the RemoteAddress parameter of TCON_IP_v4.

Connection description for configured connections

For programmed connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured.

- Make sure that only connections of the type TCP are specified in the TCON_Configured structure.
- The connection may not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceId	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	0	Connection type Select 254 (decimal) for a configured connection.

Parameter STATUS

Parameter STATUS (general status information)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active and no connection established (REQ=0).
7001	Connection establishment triggered.
7002	Intermediate call. Connection is being established.
7003	Connection is being terminated.
7004	Connection established and monitored. No job processing active.

STATUS* (W#16#)	Description
7005	Data is being sent.
7006	Data is being received.
80BB	Invalid value at ActiveEstablished parameter (identifier for the type of connection establishment, see CONNECT parameter (Page 3795)): • Only passive connection establishment permitted for server (ActiveEstablished = FALSE). • Only active connection establishment permitted for client (ActiveEstablished = TRUE).
8380	Received Modbus frame has incorrect format or too few bytes were received.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS (protocol error)

STATUS* (W#16#)	Error code in the error message from MB_SERVER (B#16#)	Description
8381	01	Function code is not supported.
8382	03	Error in data length: • Invalid length specification in received Modbus frame. • The length of the Modbus frame in the frame header does not match the number of received bytes. • The number of bytes does not match the number of actually transmitted bytes (only functions 1-4). • The received start address does not match the one sent originally (functions 5, 6, 15, 16). • The number of words does not match the number of actually transmitted words (only functions 15 and 16).
8383	02	Error reading or writing data or access outside the address area of MB_DATA_PTR (Page 3794). The error can occur locally as well as with the instruction "MB_SERVER".
8384	03	Error in data value: • Error in data value for function 5 (error on server). • A different data value was received than was originally sent by the client (functions 5 and 6) (local error). • Invalid exception code received.
8385	03	Diagnostic code is not supported (function code 08). The error can occur locally as well as on the server.
8386	-	Received function code does not match the one sent originally.
8387	-	• The assigned connection ID is different from that used for previous requests. Only one connection ID can be used for each instance DB of the "MB_CLIENT" instruction. • The error code is also output when the protocol ID of the Modbus TCP frame received by the server is not "0".

STATUS* (W#16#)	Error code in the error message from MB_SERVER (B#16#)	Description
8388	15 or 16	The Modbus server sent a different data length than was requested. This error occurs only when using the Modbus functions 15 or 16.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Parameter STATUS (parameter error)

STATUS* (W#16#)	Description
80B6	Invalid connection type, only TCP connections are supported.
80C8	No response of the server in the defined period. Check the connection to the Modbus server. This error is only reported on completion of the configured repeated attempts. If the "MB_CLIENT" instruction does not receive an answer with the originally transferred transaction ID (see static MB_TRANSACTION_ID tag) within the defined period, this error code is output.
8188	The MB_MODE parameter has an invalid value.
8189	Invalid addressing of data at the MB_DATA_ADDR parameter.
818A	Invalid data length at the MB_DATA_LEN parameter.
818B	The MB_DATA_PTR parameter has an invalid pointer. You should also check the values of the MB_DATA_ADDR (Page 3794) and MB_DATA_LEN parameters.
818C	Timeout at parameter BLOCKED_PROC_TIMEOUT or RCV_TIMEOUT (see static tags of instruction). The limit of 55 seconds has been exceeded.
818D	The transaction ID (MB_TRANSACTION_ID tag) does not correspond to the one sent originally (see static tags of instruction).
8200	• A different Modbus request is currently being processed via the port. • Another instance of MB_CLIENT with the same connection parameters is processing an existing Modbus request.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

Error codes of internally used communications instructions

With the "MB_CLIENT" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND", "TRCV", "T_DIAG" and "TRESET") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_CLIENT" instruction. The error codes are displayed for the respective instruction under STATUS in the "Static" section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

See also

GET_ERR_ID: Get error ID locally (Page 2417)

MB_SERVER: Communicating via PROFINET as a Modbus TCP server

Description of MB_SERVER

Description

The "MB_SERVER" instruction communicates as Modbus TCP server via a PROFINET connection. The "MB_SERVER" instruction processes connection requests of a Modbus TCP client, receives and processes Modbus requests and sends responses.

In V3.0 and higher, the "MB_SERVER" instruction can be used for the S7-1500 as well as for the S7-1200 version 4.0 and higher. The connection can take place via the local interface of the CPU or CM/CP.

To use the instruction, you do not require an additional hardware module.

Notice

Security information

Note that each client of the network is given read and write access to the process image inputs and outputs and to the data block or bit memory area defined by the Modbus holding register.

The option is available to restrict access to an IP address to prevent unauthorized read and write operations. Note, however, that the shared address can also be used for unauthorized access.

Multiple server connections

You can create multiple Server connections. This allows a single CPU to accept connections from multiple Modbus TCP clients at the same time.

A Modbus TCP server can support several TCP connections and the maximum number of connections depends on the CPU being used.

The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections.

Modbus TCP connections can also be shared by "MB_CLIENT" and/or "MB_SERVER" instances.

In the case of Server connections, remember the following rules:

- Each "MB_SERVER" connection must use a unique instance DB.
- Each "MB_SERVER" connection must use a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- For each connection, the "MB_SERVER" instruction must be called individually.

Parameters

The following table shows the parameters of the "MB_SERVER" instruction:

Parameter	Declaration	Data type	Description
DISCONNECT	Input	BOOL	The "MB_SERVER" instruction is used to enter into a passive connection with a partner module. The server responds to a connection request from the IP address which is entered in the SDT "TCON_IP_v4" in the CONNECT parameter. You can use this parameter to control when a connection request is accepted: • 0: A passive connection is established when there is no communications connection. • 1: Initialization of the connection termination. If the input is set, no other operations are executed. The value 0003 is output at the STATUS parameter after successful connection termination.
MB_HOLD_REG (Page 3804)	InOut	VARIANT	Pointer to the Modbus holding register of the "MB_SERVER" instruction The holding register contains the values that may be accessed by a Modbus client using the Modbus functions 3 (read), 6 (write) and 16 (multiple write). As the holding register, use either a global data block with optimized access or the memory area of the bit memories.
CONNECT (Page 3805)	InOut	VARIANT	Pointer to the structure of the connection description The following structures (SDTs) can be used: • TCON_IP_v4: Includes all address parameters required for establishing a programmed connection. The default address is 0.0.0.0 (any IP address), but you can enter a specific IP address so that the server only responds to requests from this address. When using TCON_IP_v4, the connection is established when calling the instruction "MB_SERVER". • TCON_Configured: Includes the address parameters of a configured connection. When using TCON_Configured, the connection is established by the CPU after download of the hardware configuration.
NDR	Output	BOOL	"New Data Ready": • 0: No new data • 1: New data written by the Modbus client written
DR	Output	BOOL	"Data Read": • 0: No data read • 1: Data read by the Modbus client
ERROR	Output	BOOL	If an error occurs during the call of the "MB_SERVER" instruction, the output of the ERROR parameter is set to "1". Detailed information about the cause of the problem is indicated by the STATUS parameter.
STATUS (Page 3807)	Output	WORD	Detailed status information of the instruction.

You can find additional information on valid data types under "Overview of the valid data types (Page 1908)".

Static tags of the instruction

The following table describes the static tags of the instance data block of the "MB_SERVER" instruction used in the program. You can write the HR_Start_Offset tag. You can read the other tags to monitor the Modbus status.

Tag	Data type	Start value	Description
HR_Start_Offset	WORD	0	Assign the start address of the Modbus holding register.
Request_Count	WORD	0	Total number of requests received by the server.
Server_Message_Count	WORD	0	Total number of received alarms for the relevant server.
Xmt_Rcv_Count	WORD	0	Counter for detecting the number of transfers during which an error occurred. The counter is only incremented when an invalid Modbus request is received.
Exception_Count	WORD	0	Counters for detecting the number of errors specifically for Modbus which cause an error message to "MB_CLIENT".
Success_Count	WORD	0	Event counter for detecting the number of requests that were successfully executed by the server.
Connected	BOOL	0	Indicates whether the connection to the assigned client has been established or not: 1 = connected, 0 = not connected.

Mapping of Modbus addresses to the process image

The "MB_SERVER" instruction allows incoming Modbus functions (1, 2, 4, 5 and 15) direct read and write access to the process image inputs and outputs of the CPU (use of the data types BOOL and WORD).

For the data transfer of the function codes 3, 6 and 16, the holding register (MB_HOLD_REG parameter) must be defined longer than one byte. The following table shows the mapping of the Modbus addresses to the process image of the CPU.

Modbus function						S7-1500, S7-1200 V4.0	
Function code	Function	Data area	Address space			Data area	CPU address
01	Read: Bits	Output	1	to	9999	Process image output	Q0.0 to Q1249.6
02	Read: Bits	Input	1	to	9999	Process image input	I0.0 to I1249.6
04	Read: WORD	Input	1	to	9999	Process image input	IW0 to IW19996
05	Write: Bits	Output	1	to	9999	Process image output	Q0.0 to Q1249.6
15	Write: Bits	Output	1	to	9999	Process image output	Q0.0 to Q1249.6

Incoming Modbus requests with the function codes 3, 6, 16 and 23 write to or read from the Modbus holding register (you specify the holding register with the MB_HOLD_REG parameter).

Example: Addressing via static tag HR_Start_Offset

The addresses of the Modbus holding register start at 40001. These addresses correspond to the address space of the CPU memory area for the holding register. You can also define the HR_Start_Offset tag so that the Modbus holding register has a start address other than 40001.

Example: The holding register begins at MW100, and has a length of 100 WORD. An offset value in the HR_Start_Offset parameter means that the start address of the holding register is

moved from 40001 to 40021. Whenever the holding register is addressed below the address 40021 and above the address 40120, this causes an error.

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (WORD)	40001	40100
	CPU address	MW100	MW298
20	Modbus address (WORD)	40021	40120
	CPU address	MW100	MW298

Modbus diagnostics functions

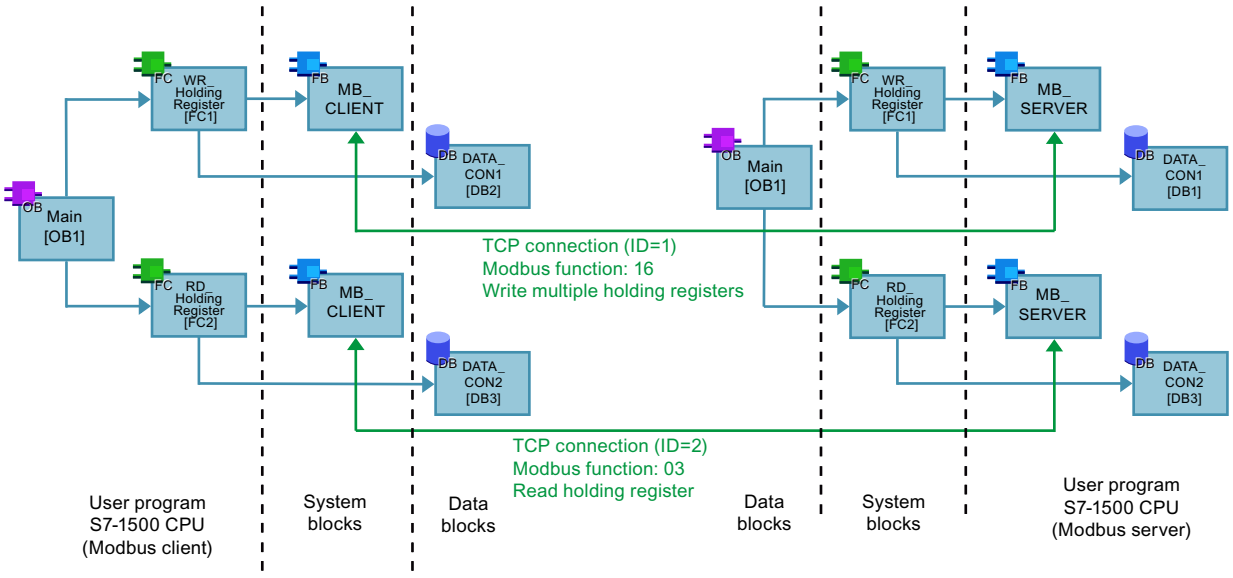
The table below contains a description of the Modbus diagnostics functions.

Function code	Diagnostic code	Description
08	0x0000	Echo test: The "MB_SERVER" instruction receives a data word and returns this unchanged to the Modbus client.
08	0x000A	Reset event counter: The "MB_SERVER" instruction resets the following event counters: "Success_Count", "Xmt_Rcv_Count", "Exception_Count", "Server_Message_Co" and "Request_Count".
11	-	Fetch event counter of the communication: The "MB_SERVER" instruction uses an internal event counter for communication to record the number of successfully executed read and write requests sent to the Modbus server. The event counter is not incremented with the functions 8 or 11. The same holds true for requests that cause a communications error, for example, if a protocol error has occurred (e.g., the function code in the received Modbus request is not supported).

Example

An example project for the Modbus TCP communication between two S7-1500 CPUs can be found in the Service and Support Portal under Entry ID 94766380 (http://support.automation.siemens.com/WW/view/en/94766380).

Two Modbus functions are used in this example. For each Modbus function, a Modbus TCP connection is established using a Modbus block pair (MB_CLIENT and MB_SERVER).



MB_HOLD_REG parameter

Description

The MB_HOLD_REG parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. You can use a global data block or a bit memory (M) as memory area.

- The high limit for the number of addresses in the data block (D) is determined by the maximum work memory of the CPU.
- The high limit for the number of bit memories (M) is determined by the size of the CPU memory area.

The following table shows examples of mapping Modbus addresses to the holding register for the Modbus functions 3 (read WORD), 6 (write WORD), 16 (write several WORD) and 23 (write and read several words).

Modbus addresses	MB_HOLD_REG parameter - examples		
40001	MW100	DB10.DBW0	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	"Recipe".ingredient[5]

CONNECT parameter

Connection descriptions at CONNECT parameter

Two different connection descriptions can be used for the "MB_SERVER" instruction:

- Programmed connections with the structure TCON_IP_v4
  The connection parameters are stored in the TCON_IP_v4 structure and the connection is set up with the call of the instruction "MB_SERVER".
- Configured connections with the structure TCON_Configured
  The configured connection has already been established by the CPU. Use the structure TCON_Configured to specify which existing connection is to be used for the instruction.

Each instance of the instruction "MB_SERVER" requires a unique connection. Create a separate structure TCON_IP_v4 or TCON_Configured for each instance of the instruction to describe the connection.

Connection description for programmed connections

For programmed connections at the CONNECT parameter, use the following structure for connection description to TCON_IP_v4.

- Make sure that only connections of the type TCP are specified in the TCON_IP_v4 structure.
- The connection may not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). The parameter uniquely identifies a connection within the CPU. Each individual instance of the instruction "MB_SERVER" must use a unique ID.
4	ConnectionType	BYTE	11	Connection type Select 11 (decimal) for TCP. Other connection types are not permitted. If another connection type (e.g. UDP) is used, a corresponding error message is output at the STATUS parameter of the instruction.
5	ActiveEstablished	BOOL	FALSE	ID for the manner in which the connection is established Select FALSE for passive connection establishment.
6 … 9	RemoteAddress	ARRAY [1..4] of BYTE	0.0.0.0	IP address of the connection partner, for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1 If the instruction "MB_SERVER" is to accept connection requests from any connection partner, use "0.0.0.0" as IP address.

Byte	Parameter	Data type	Start value	Description
10 … 11	RemotePort	UINT	0	Port number of the remote connection partner (value range: 1 to 49151). If the instruction "MB_SERVER" is to accept connection requests from any remote partner, use "0" as port number.
12 … 13	LocalPort	UINT	502	Port number of the local connection partner (value range: 1 to 49151). The number of the IP port defines which IP port is monitored for connection requests of the Modbus client. The following TCP port numbers must not be used for the passive connection of the "MB_SERVER" instruction: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Note

Migration of the instruction "MB_SERVER" version 2.1

The parameters CONNECT_ID and IP_PORT are mapped in version 3.0 of the "MB_SERVER" instruction in the TCON_IP_v4 structure:

- The parameter CONNECT_ID of the "MB_SERVER" V2.1 instruction corresponds to the ID parameter of TCON_IP_v4.
- The parameter IP_PORT of the "MB_SERVER" V2.1 instruction corresponds to the LocalPort parameter of TCON_IP_v4.

Connection description for configured connections

For configured connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured.

- Make sure that only connections of the type TCP are specified in the TCON_Configured structure.
- The connection may not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	-	Connection type Select 254 (decimal) for a configured connection.

Parameter STATUS

Parameter STATUS (general status information)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active (REQ=0).
7001	First call. Connection establishment triggered.
7002	Intermediate call. Connection is being established.
7003	Connection is being terminated.
7005	Data is being sent.
7006	Data is being received.
80BB	Invalid value at ActiveEstablished parameter (identifier for the type of connection establishment, see CONNECT parameter (Page 3805)): • Only passive connection establishment permitted for server (active_established = FALSE). • Only active connection establishment permitted for client (active_established = TRUE).
8380	Received Modbus frame has incorrect format or too few bytes were received.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS (parameter error)

STATUS* (W#16#)	Error code in the error message from "MB_SERVER" (B#16#)	Description
8187	No response	The MB_HOLD_REG parameter has an invalid pointer. Data area is too small.
8381	01	Function code is not supported.
8382	03	Error in data length: • Invalid length specification in received Modbus frame • The frame length entered in the header of the Modbus frame does not match the number of actually received bytes. • The number of bytes entered in the header of the Modbus frame does not match the number of actually received bytes (functions 15 and 16).
8383	02	Error in data address or access outside the address area of the holding register (MB_HOLD_REG (Page 3804) parameter).
8384	03	Invalid data value (function 5).
8385	03	Diagnostic code is not supported (only with function code 08).
* The status codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Note

Error codes of internally used communications instructions

With the "MB_SERVER" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND", "TRCV", "T_DIAG" and "T_RESET") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_SERVER" instruction. The error codes are displayed for the respective instruction under STATUS in the "Static" section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

See also

GET_ERR_ID: Get error ID locally (Page 2417)

SIMATIC NET CM/CP

S7-1500 CM/CP

Industrial Ethernet

Instructions for FTP services

FTP_CMD for FTP services

Overview of FTP_CMD

Meaning

Using the FTP_CMD instruction, you can establish FTP connections and transfer files from and to an FTP server.

Data transfer is possible using FTP or FTPS (secure SSL connections).

Note

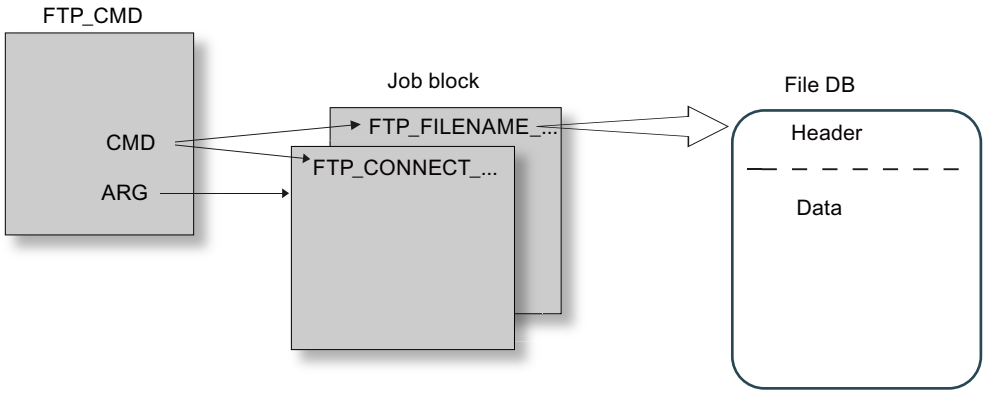
FTPS: Comparing certificates

FTPS requires a comparison of the certificates between FTP server and FTP client. If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server. Import the certificate of the FTP server as a trusted certificate in the certificate manager.

How it works

The FTP_CMD instruction references a job block (ARG) in which the FTP command is specified. Depending on the type of FTP command (CMD), this job block uses different data structures for parameter assignment. Suitable data types (UDTs) are available for these various structures.

The following diagram shows the call structure:



Job blocks

The following data structures are used for the job blocks:

- Connection establishment
  Various data structures are available for the connection establishment using the following types of access:
  - FTP_CONNECT_IPV4: Connection establishment with IP addresses according to IPv4
  - FTP_CONNECT_IPV6: Connection establishment with IP addresses according to IPv6
  - FTP_CONNECT_NAME; Connection establishment with server name (DNS)
- Data transfer
  For the data transfer, two different data structures are available:
  - FTP_FILENAME: Data structure for access to a complete file
  - FTP_FILENAME_PART: Data structure for read access to a data area

Data transfer in the File_DB

The data transfer is achieved using data blocks containing a header for job data and the area for the user data. The data block is specified in the job buffer.

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.
- For all data blocks being used as file DBs, disable the "Optimized block access" attribute

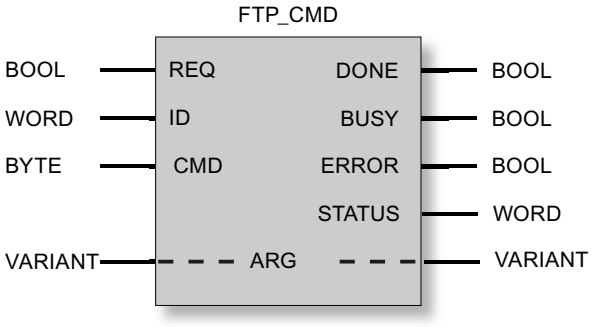
Validity

The FTP_CMD instruction can be used with the following module types:

- CP 1543-1

Call interface

Call interface in FBD representation



See also

Output parameters and status information FTP_CMD (Page 3817)

Input parameter - FTP_CMD (Page 3810)

Structure of the data blocks (file DBs) for FTP services - FTP client mode (Page 3821)

Input parameter - FTP_CMD

Explanation of the input parameters

You supply the FTP_CMD instruction with the following input parameters:

Table 11-102 Formal parameters of the FTP_CMD instruction - input parameters

Parameter	Declaration	Data type	Memory area	Meaning / remarks
REQ	Input	BOOL	I, Q, M, D, L	Starts the send job on a rising edge.
ID *	INPUT	INT	1, 2 ... 64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.

Parameter	Declaration	Data type	Memory area	Meaning / remarks
CMD *	INPUT	BYTE	See following table "Commands".	FTP command to be executed when the instruction is called. You will find value ranges for the FTP command types after the table. Note: The FTP command specified here must be specified identically in the job block (ARG parameter). If a command is not supported by the CP firmware, an error message with STATUS = $8F6B_H$ is output.
ARG *	INPUT	VARIANT	See following table "Commands".	Job block References the data area with the execution parameters suitable for the FTP command. Specific data types (UDT) are used depending on the FTP command. The UDTs are shown below. The ANY data type is not permitted for the pointer to be specified here!

* The values of the input parameters "ID" and "CMD" overwrite the value of the input parameter "ARG".

FTP commands in the "CMD" parameter

The following table shows you the significance of the commands of the "CMD" parameter and which UDTs you use to supply the job blocks.

Table 11-103 Command types

CMD (command type)	Relevant job blocks / UDT	Meaning / handling
0 (NOOP)	*	The called FC does not execute any action. The status codes are set as follows when these parameters are supplied: DONE=1; ERROR=0; STATUS=0
1 (CONNECT)	FTP_CONNECT_IPV4 FTP_CONNECT_IPV6 FTP_CONNECT_NAME	FTP connection establishment With this command, the FTP client establishes an FTP connection to an FTP server (port 21). The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user.
2 (STORE)	FTP_FILENAME	This function call transfers a data block (file DB) from the FTP client (S7-CPU) to the FTP server. Caution: If the file (file DB) already exists on the FTP server, it will be overwritten.
3 (RETRIEVE)	FTP_FILENAME	This function call transfers a file from the FTP server to the FTP client (S7-CPU). Caution: If the data block (file DB) on the FTP client already contains a file, it will be overwritten.
4 (DELETE)	FTP_FILENAME	With this function call, you delete a file on the FTP server.
5 (QUIT)	*	With this function call, you establish the FTP connections selected with the ID.

CMD (command type)	Relevant job blocks / UDT	Meaning / handling
6 (APPEND)	FTP_FILENAME	Similar to "STORE", the "APPEND" command saves a file on the FTP server. With "APPEND", the file on the FTP server is, however, not overwritten. The new content is appended to the existing file. If the file (file DB) does not exist on the FTP server, it will be created.
7 (RETR_PART)	FTP_FILENAME_PART	Using the "RETR_PART" command (retrieve part) , you can request a section of a file from the FTP server. If very large files are involved, this allows you to restrict the read to the part you currently require. To do this, you need to know the structure of the file. Enter the required part of the file using the two parameters "OFFSET" and "LEN" in FB40.

***** With the command types 0 (NOOP) and 5 (QUIT) a freely selectable job block (UDT) must be specified. This is not evaluated.

See also

Overview of FTP_CMD (Page 3808)

Job blocks for FTP_CMD

Meaning

You supply the FTP_CMD instruction with a job block using the ARG parameter. The structure depends on the FTP command type. By using the default data types (UDT), the instruction recognizes the type of the job block. Below, you will find the relevant data types (UDTs) for the following job blocks:

- FTP connection establishment with IP address according to IPv4
- FTP connection establishment with IP address according to IPv6
- FTP connection establishment with server name
- Write and read access and other FTP commands
- FTP command RETR_PART

Job block for FTP connection establishment with IP address according to IPv4

For FTP connection establishment with IP address according to IPv4, the following data structure is used.

Table 11-104 FTP_CONNECT_IPV4

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPCmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You will find the ranges of values in Table 11-103 Command types (Page 3811) Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	'benutzer'	User name for the login on the FTP server
Password	STRING[32]	'passwort'	Password for the login on the FTP server
FTPserverIPaddr	IP_V4	ADDR(1) ... ADDR(4)	IP address of the FTP server as Array[1..4] of Byte, where 1 byte specifies one block of the address. Example: ADDR(1) specifies the first address block (the first byte of the address).

Job block for FTP connection establishment with IP address according to IPv6

For FTP connection establishment with IP address according to IPv6, the following data structure is used.

Table 11-105 FTP_CONNECT_IPV6

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPCmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You will find the ranges of values in Table 11-103 Command types (Page 3811) Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	'user'	User name for the login on the FTP server
Password	STRING[32]	'password'	Password for the login on the FTP server
FTPserverIPaddr	IP_V6	ADDR(1) ... ADDR(16)	IP address of the FTP server as Array[1..16] of Byte, where 2 bytes specify one block of the address. Example: ADDR(1) + ADDR(2) specify the first address block.

Job block for FTP connection establishment with server name

For FTP connection establishment specifying the server name, the following data structure is used. The server name is assigned to an IP address using DNS.

Table 11-106 FTP_CONNECT_NAME

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPcmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You will find the ranges of values in Table 11-103 Command types (Page 3811) Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	'benutzer'	User name for the login on the FTP server
Password	STRING[32]	'passwort'	Password for the login on the FTP server
FTPserverName	STRING[254]		IP address of the FTP server

Job block for write and read access and other FTP commands

The following data structure is used for the FTP commands store, retrieve, delete and append.

Table 11-107 FTP_FILENAME

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	

Parameter	Type	Range of values	Meaning / remarks
FTPcmd	BYTE	2, 3, 4, 6	FTP command "STORE / RETRIEVE / DELETE / APPEND" FTP command that executes when the instruction is called. You will find the ranges of values in Table 11-103 Command types (Page 3811) Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
DataBlockNumber	UINT		The data block specified here contains the file DB to be read / written.
LenFilename	UINT	0...1000	The "LenFilename" parameter for specifying the total length of the file name is not evaluated. Instead the length information in the string of the "Filename" parameter is evaluated.
Filename	ARRAY[0..3] OF STRING[220]		File name of the destination or source file.

Job block for the RETR_PART FTP command

The following data structure is used for the RETR_PART FTP command.

Table 11-108 FTP_FILENAME_PART

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPcmd	BYTE	7	FTP command "RETR_PART" FTP command that is executed when the instruction is called. You will find value ranges in Table 11-103 Command types (Page 3811) Note: The FTP command specified here must be specified identically in the CMD input parameter.

Parameter	Type	Range of values	Meaning / remarks
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
Offset	DWORD		Offset in bytes starting at which the file will be read.
Length	DWORD		Sublength in bytes that is read starting at the value specified in "OFFSET". Special features: • If "DW#16#FFFFFFFF" is specified, the available rest of the file will be read. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET > length of the original file: Length of the destination file (ACT_LENGTH in file DB): 0 bytes on the CPU. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET + LEN > length of the original file (and LEN ≠ 0xFFFFFFFF): Length of the destination file (ACT_LENGTH in file DB): Available bytes starting at "OFFSET". Result OK (DONE = 1, STATUS = 0) if no other error occurred.
DataBlockNumber	UINT		The data block specified here contains the file DB to be read / written.
LenFilename	UINT	0...1000	The "LenFilename" parameter for specifying the total length of the file name is not evaluated. Instead the length information in the string of the "Filename" parameter is evaluated.
Filename	ARRAY[0..3] OF STRING[220]		File name of the destination or source file.

See also

Overview of FTP_CMD (Page 3808)

Input parameter - FTP_CMD (Page 3810)

Output parameters and status information FTP_CMD

Parameters BUSY, DONE and ERROR

You control the execution status with the parameters BUSY, DONE, ERROR and STATUS. The BUSY parameter indicates the processing status. With the DONE parameter, you check whether or not a job was correctly executed. The ERROR parameter is set if errors occur during execution of "FTP_CMD". Error information is output in the STATUS parameter.

The following table shows the relationship between the parameters BUSY, DONE and ERROR:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was terminated with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Evaluating status codes

Note

For entries coded with 8FxxH in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 11-109 FTP_CMD: Meaning of the STATUS parameter in conjunction with DONE and ERROR

DONE	ERROR	STATUS	Meaning
0	0	0000_H	No job is being executed.
1	0	0000_H	the job was completed without error.
0	0	7001_H	The job was triggered the first time (BUSY=1)
0	0	7002_H	The job is still being executed (BUSY=1)
0	0	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).
0	0	8183_H	The configuration does not match the job parameters.
0	1	8401_H	Unknown error Possible causes: • A timeout was detected on the connection. • The FTP server has aborted the connection. Remedy: • Send the QUIT and CONNECT commands again to re-establish the connection.
0	1	8402_H	The connection has an error status. The timeout of the connection may have been exceeded or the FTP server may have terminated the connection. Remedy: Send the QUIT and CONNECT commands to re-establish the connection.
0	1	8403_H	Login has failed.
0	1	8404_H	FTP server is not obtainable.
0	1	8405_H	Transfer has failed.

DONE	ERROR	STATUS	Meaning
0	1	8406_H	Timeout for current action
0	1	8407_H	File was not found on the FTP server.
0	1	8408_H	Transfer not possible.
0	1	8409_H	File could not be fetched.
0	1	8410_H	Setting the TCP port for the data connection has failed.
0	1	8411_H	Offset information does not match.
0	1	8412_H	Error changing the specified directory
0	1	8413_H	Error receiving data
0	1	8414_H	Error sending data
0	1	8415_H	Specified CMD (command type) was rejected by the client.
0	1	8416_H	Connection was closed by the FTP server.
0	1	8418_H	Error in the user data. Possible causes: • File name is empty. • Data length is "0" • etc.
0	1	8419_H	There is no socket resource for opening a data connection.
0	1	8420_H	There is no socket resource for opening a control connection.
0	1	8421_H	Error opening the file DB to be read.
0	1	8422_H	Error opening the file DB to be written.
0	1	8423_H	Connection establishment to the FTP server has failed.
0	1	8424_H	Internal error
0	1	8425_H	Format error in the domain name
0	1	8426_H	There are too many DNS queries pending.
0	1	8427_H	The specified DNS server could not assign the specified domain name.
0	1	8428_H	There is no connection resource available.
0	1	8429_H	Unknown channel ID
0	1	8430_H	The file DB is too short.
0	1	8431_H	Error when writing to the file DB.
0	1	8432_H	Error when reading from the file DB.
0	1	8433_H	Error when accessing the file DB.
0	1	8434_H	Action was aborted.
0	1	8435_H	Channel will be reset.
0	1	8436_H	Unexpected server reply
0	1	8437_H	Certificate could not be verified.
0	1	8438_H	Unknown error occurred
0	1	8439_H	The FTP command causes an error. The cause must be looked for on the FTP server (REST command).
0	1	8440_H	The FTP server does not support the requested SSL protocol.
0	1	8446_H	After the FTP password was sent to the FTP server, an unexpected code was returned by the FTP server.
0	1	8451_H	An error was signaled when attempting to change the transmission mode from binary to ASCII.

DONE	ERROR	STATUS	Meaning
0	1	8455_H	A memory request has failed on the CM/CP.
0	1	8460_H	A problem has occurred handling SSL/TLS.
0	1	8469_H	Interface error The specified output interface could not be used. Remedy: Set the interface to be used for outgoing connections.
0	1	8475_H	The SSL certificate or the SSH md5 fingerprint was not considered trusted.
0	1	8476_H	Nothing was received from the FTP server. In the current status, an incorrect response must be assumed.
0	1	8477_H	The specified "Crypto engine" (cryptographic module) was not found.
0	1	8478_H	The attempt to set the selected SSL "Crypto engine" as the default failed.
0	1	8480_H	A problem has occurred with the certificate of the FTP client.
0	1	8481_H	The specified number could not be used.
0	1	8482_H	The FTP server uses a coding that is not supported.
0	1	8484_H	The maximum file size was exceeded.
0	1	8485_H	The file DB was modified while being processed to be sent or the file DB is incorrectly structured.
0	1	8489_H	Data could not be sent. There is not enough memory available for the action on the FTP server.
0	1	8492_H	The file already exists. The file will not be overwritten.
0	1	8496_H	A problem occurred reading the SSL CA certificate.
0	1	8497_H	An unexpected error occurred in the SSH session.
0	1	8498_H	It was not possible to terminate the SSL connection.
0	1	8499_H	The socket is not ready for sending/receiving. Wait until it is ready and try again.
0	1	8501_H	The SSL certificate check by the FTP server has failed.
0	1	8507_H	A timeout has occurred establishing the connection during the active FTP session while waiting for the FTP server.
0	1	$8F55_H$	Header status bit: Locked
0	1	$8F56_H$	The NEW bit in the file DB header was not reset
0	1	$8F6B_H$	Possible causes: • Bad value for the CMD parameter Values from 0 to 15 are permitted. • An FB40 command is not supported. Possible cause: Wrong firmware on the CP Remedy: Firmware update (with older CPs, use the functions FC 40...FC 44 instead of FB 40.)
0	1	$8F7F_H$	Internal error, for example illegal ANY reference

See also

Overview of FTP_CMD (Page 3808)

Structure and use of the file DB - FTP client mode

Structure of the data blocks (file DBs) for FTP services - FTP client mode

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.
- For all data blocks being used as file DBs, disable the "Optimized block access" attribute.

File DB header for FTP client mode

Note: The file DB header described here is largely identical to the file DB header for server mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data (file does not exist). • 1: The file DB contains valid user data (file exists).	The DELETE FTP command sets EXIST=0. The STORE FTP command sets EXIST=1.
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The "STORE" and "RETRIEVE" FTP commands set LOCKED=1 when they are executed. The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program: 1. Check LOCKED bit (if = 0) 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit (if = 0) 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program in the S7-CPU must set NEW=0 to allow a new "RETRIEVE" command.
WRITE_ACCESS	BOOL	• 0: The user program (FTP client blocks) has write access rights for the file DBs on the S7 CPU. • 1: The user program (FTP client blocks) has no write access rights for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.

Parameter	Type	Value / meaning	Supply
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	Unsigned integer (16-bit) containing the last reply code from FTP as a binary value. The content of this field is only valid when EXIST = 1.	This is updated by the FTP client when the FTP command is executed.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. The reference is the startup time of the (initialization value: 01.01.1994 00:00h).

See also

FILE_DB_HEADER data block as template - FTP client mode (Page 3823)

FILE_DB_HEADER data block as template - FTP client mode

Meaning

The data type FILE_DB_HEADER is predefined for creating a file DB header.

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)

Follow the steps outlined below:

1. In the CPU on which you create the user program with the FTP instructions, create a data block of the type "Global DB".
2. Select the line you want to use as the start line for the file DB.

3. In the "Data type" column, select a structure element of the type "FILE_DB_HEADER" (in the drop-down list at the very bottom).
   Result: A data structure with the header structure required for the file DB is created.
4. Select the properties of the newly created data block (shortcut menu) and disable the "Optimized block access" attribute.

Note

"Add new block" function - type selection

When you create new data blocks, the "FILE_DB_HEADER" block type is also available under the "Type" entry in the drop-down list. Do not use this option! The data block created in this way only contains the header structure and cannot be expanded by the area required for storing user data.

FILE_DB_HEADER data block - example and template for the file DB header

In the declaration view, you will see the following structure:

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	bit08	BOOL	FALSE	reserved
+0.1	bit09	BOOL	FALSE	reserved
+0.2	bit10	BOOL	FALSE	reserved
+0.3	bit11	BOOL	FALSE	reserved
+0.4	bit12	BOOL	FALSE	reserved
+0.5	bit13	BOOL	FALSE	reserved
+0.6	bit14	BOOL	FALSE	reserved
+0.7	bit15	BOOL	FALSE	reserved
+1.0	EXIST	BOOL	FALSE	if TRUE: File DB does not contain valid data
+1.1	LOCKED	BOOL	FALSE	if TRUE: The file DB is blocked due to a change to the content.
+1.2	NEW	BOOL	FALSE	if TRUE: The content of the File DB is new and must not be overwritten.
+1.3	WRITE_ACCESS	BOOL	FALSE	if TRUE: The FTP server has write access.
+1.4	bit04	BOOL	FALSE	reserved
+1.5	bit05	BOOL	FALSE	reserved
+1.6	bit06	BOOL	FALSE	reserved
+1.7	bit07	BOOL	FALSE	reserved

Address	Name	Type	Initial value	Comment
+2.0	ACT_LENGTH	DINT	L#0	Current length of the content in bytes (not including 20 bytes for the header)
+6.0	MAX_LENGTH	DINT	L#0	Maximum length of the content in bytes (not including 20 bytes for the header)
+10.0	FTP_REPLY_CODE	INT	0	Last replay information from the remote FTP server.
+12.0	DATE_TIME	DATE_AND_TIME	DT#00-1-1-0:0:0.000	Date and time of the last change to the content of the file DB.
=20.0		END_STRUCT		

Differences in the modes

File DB header for FTP client mode

The file DB header described here is largely identical for the FTP client mode and FTP server mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

S7-1200 CM/CP

Telecontrol

Telecontrol instructions

TC_CON: Establish connection via the GSM network

Meaning

The TC_CON instruction allows an S7-1200 with a CP 1242-7 to establish a connection of the following types:

- ISO-ON-TCP
  Connection partner is a CP 1242-7.
  ISO-ON-TCP connections are used only in "GPRS direct" mode.
- UDP
  Any connection partner is possible.

- SMS
  The connection partner is an SMS client.
- Telecontrol connection
  The connection partner is either a telecontrol server or another station that can be reached via the telecontrol server.

A TC_CON establishes exactly one connection. Depending on the mode of the CP 1242-7 and the protocol you are using, a maximum of 3 to 5 simultaneous connections with unique IDs (see below) are supported per CP. You will find the maximum number of simultaneous connections in the performance data of the CP.

The CONNECT parameter uses a data block (DB) with the structure of a system data type (SDT) for the connection description.

The required connection type is defined using a connection-specific SDT "TCON_..." (see below). For each of the connection types listed above, one of the following SDTs must be assigned:

- TCON_IP_RFC for ISO-ON-TCP connections
- TCON_IP_V4 for UDP connections
- TCON_PHONE for SMS connections
- TCON_WDC for telecontrol connections

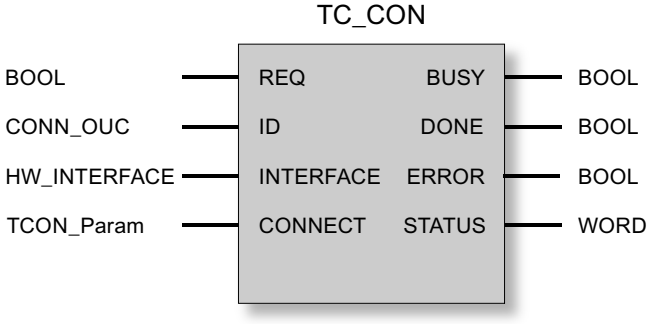
The "ActiveEstablished" parameter of these SDTs also specifies whether or not connection establishment is active or passive.

For parameter settings for these SDTs, see TCON_...: SDTs for the telecontrol connection establishment (Page 3841).

The ID parameter references the GPRS connection. The ID is assigned and must be unique within the CPU.

The INTERFACE parameter references the GPRS interface of the required local CP. This must be taken from STEP 7.

Call interface in FBD representation



Explanation of the formal parameters

The following table explains all the formal parameters for the TC_CON instruction.

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection. The ID is assigned. The value of ID is also required by the system data type (SDT) of the CONNECT parameter.
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
CONNECT	INOUT	TCON_Param	See also "TCON_...: SDTs for telecontrol connection establishment"	Reference to a data block for connection establishment. The SDTs of the type TCON_IP_RFC, TCON_IP_V4, TCON_PHONE or TCON_WDC specify the structure of the data block suitable for the relevant connection. In the SDTs, note the parameter "ActiveEstablished" (active / passive connection establishment).
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

When called, the instruction remains in the BUSY = 1 state for several seconds. In the following situations, the BUSY state = 1 can last for a longer time:

- On active ISO-on-TCP connections if the partner cannot be reached.
- On passive connections when no frame is received.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	8087_H	Maximum number of connections reached, no further connection possible
0	1	$80E3_H$	The ID is already being used by another connection. This means for TC_SEND, TC_RECV or TC_DISCON, BUSY is currently TRUE. The status code is output when EN_R of TC_RECV is permanently TRUE. This mostly results in TC_RECV being called. Remedy for this situation: Turn EN_R off before TC_CON or TC_DISCON is called. EN_R may only be turned on again if TC_CON executed free of errors.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80E8_H$	Remote partner cannot be reached. Check the connection parameters. In the "GPRS direct" mode, the message is output if the partner can be reached but is not accepting a connection request.
0	1	$80EB_H$	Request temporarily denied (TC_CON has already been called with the same destination address.)
0	1	$80EC_H$	Opening the Listener Port failed: Check the connection parameters.
0	1	$80F2_H$	The CP is in the wrong mode: • Telecontrol connections are permitted only in "Telecontrol" mode. • ISO-ON-TCP connections are permitted only in "GPRS direct" mode.
0	1	$80F3_H$	No free connection endpoint for sending data: • Use less connections or • Use less passive connections or • Turn off NTP. Remember the maximum number of simultaneous connections of the CP 1242-7.

DONE	ERROR	STATUS	Meaning
0	1	$80F4_H$	Connection endpoint cannot be generated: Repeat the call. If necessary, check the connection parameters.
0	1	$80F5_H$	Invalid connection endpoint: Connection establishment by TC_CON failed. Repeat the block call.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format, invalid value or phone number in TCON_Phone longer than 20 characters) Check the configuration of the "TC_CON..." SDT.

TC_DISCON: Terminate connection via the GSM network

Meaning

The TC_DISCON instruction on an S7-1200 with CP 1242-7 terminates an ISO-ON-TCP, UDP, SMS or telecontrol connection that was established with the TC_CON instruction.

You will find detailed information on the connection types in the description of the TC_CON instruction.

TC_DISCON terminates the connection to the telecontrol server only logically. At the TCP/IP level, the connection is retained.

If you want the connection to the telecontrol server to be terminated physically, configure the connection as a "Temporary connection" in the "telecontrol server" parameter group in STEP 7. Temporary stations terminate the connection automatically after sending the data.

Note

Stopping the execution of further program blocks by TC_DISCON

Calling up TC_DISCON ends the execution of TC_CON, TC_SEND, and TC_RECV blocks, that were called up with the same connection ID (parameter "ID") and interface (parameter "INTERFACE"). These blocks then signal an ERROR.

Do not call up TC_DISCON if TC_CON indicates "Error = 1".

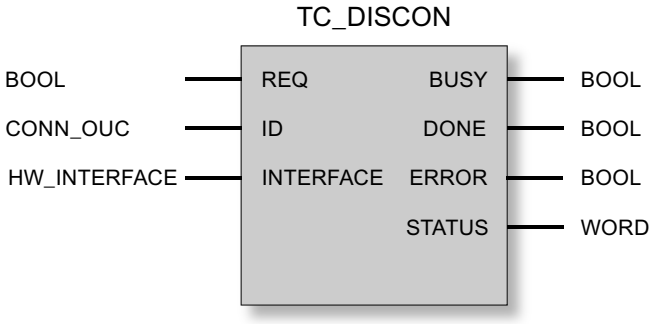
If TC_CON indicates "ERROR", then the connection is not established. In this case TC_DISCON must not be called.

If TC_DISCON is called in this case, the connection ID ("ID") is briefly reserved, and a TC_CON called up immediately afterwards would indicate ERROR and STATUS 80E3.

The ID parameter references the GPRS connection. The ID must be unique within the CPU and the same as the ID used with TC_CON.

The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

Call interface in FBD representation



Explanation of the formal parameters

The following table explains all the formal parameters for the TC_DISCON instruction

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	The instruction has not yet been called.

You will find all other code combinations of DONE and ERROR in the following table.

Note

When called, the instruction remains in the BUSY = 1 state for several seconds.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	$80E4_H$	Unknown ID: No connection with this ID has been established by TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed or • Connection terminated by remote partner.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the configuration of the "TC_CON..." SDT.

TC_SEND: Send data via the GSM network

Meaning

The TC_SEND instruction allows the sending of data via programmed connections of the following types:

- ISO-ON-TCP connections
- UDP connections

- SMS connections
  The sending of SMS messages is supported only if this was set up in the STEP 7 configuration of the CP.
- Telecontrol connections

Note

Sending SMS messages to multiple recipients

If you want to send an identical SMS message to several recipients, you need to establish a connection to each recipient.

You will find more detailed information on the connection types in the description of the TC_CON instruction.

The ID parameter references the GPRS connection. The value of ID must correspond to the value used for ID by TC_CON.

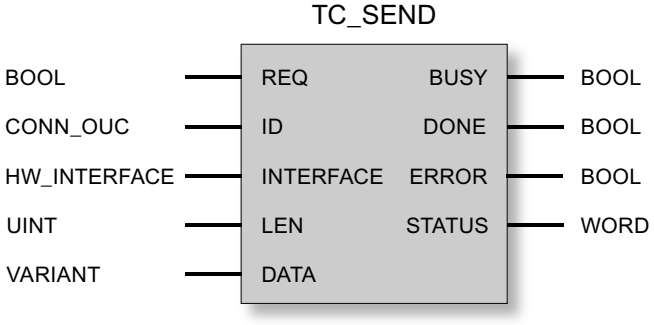
The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

The amount of data to be sent is specified with the LEN parameter.

The size of the data area specified in DATA must be at least as large as the number of bytes configured for LEN. Permitted data types in the data area specified in DATA are all except BOOL and ARRAY of BOOL.

The destination address (connection partner) for the data to be sent is configured in the TC_CON instruction.

Call interface in FBD representation



Explanation of the formal parameters

The following table explains all the formal parameters for the TC_SEND instruction.

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
LEN	INPUT	UINT	1...2048	Number of bytes of data to be sent. The value must be ≥ 1 and ≤ 2048. The value should match the size of the range of DATA.
DATA	INOUT	VARIANT		Address reference to the send data area of the CPU *
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. ** For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

* For special features of the DATA parameter for SMS texts, refer to the next section.

** After sending a frame, TC_SEND sets DONE = 1. Note the following response:
The loss of an ISO-on-TCP connection is only recognized by the sender after 1 to 2 minutes. The transferred data may be lost although TC_SEND has set DONE = 1 at the sender.
If an ISO-on-TCP connection is aborted after receiving a frame before TC_RECV was started, the transferred data may be lost even if TC_SEND sets DONE = 1 at the sender.

Configuring SMS texts with the DATA parameter

The instruction sends the data referenced by the pointer of the type VARIANT of the DATA parameter as an SMS text.

If an operand of the data type STRING is referenced by DATA for SMS texts, the first two bytes are transferred with length information of the string.

One option for the correct text representation of SMS messages to be sent is to convert the text string into an Array of BYTE or Array of CHAR using the conversion function Strg_TO_Chars. Strg_TO_Chars at the EN parameter is linked to the output parameter ENO by TC_SEND.

For SMS texts, the CP does not support all special characters, for example umlauts (ü, ä etc.). The specification GSM 03.38 applies. There may be additional restrictions imposed by the GSM network provider.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS *	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	$80E0_H$	Internal error If you send the frames directly to the telecontrol server (mode "Telecontrol"), make sure that the send cycle time is ≥ 1 second.
0	1	$80E1_H$	Timeout: • Increase the value of the "Connection monitoring time" in the configuration of the CP 1242-7 or • Check the connection partner.
0	1	$80E4_H$	Unknown ID: First call TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80E7_H$	Data to be sent not completely transferred: Repeat the job.

DONE	ERROR	STATUS *	Meaning
0	1	$80E8_H$	Remote partner cannot be reached. Check the connection parameters. In the "GPRS direct" mode, the message is output if the partner can be reached but is not accepting a connection request.
0	1	$80E9_H$	Connection establishment by remote partner: Check the connection partner. If necessary, terminate the connection with TC_DISCON and establish it again with TC_CON.
0	1	$80EA_H$	Error message from remote partner: • Check the connection partner. Enable the "TC_RECV" instruction on the communications partner. • If necessary, terminate the connection with TC_DISCON and establish it again with TC_CON.
0	1	$80EF_H$	SMS could not be sent: • Check whether the destination address (telephone number of the destination subscriber) exists. • Check whether the inserted SIM card allows sending of SMS. • Check the length of the SMS text that was sent. SMS texts > 160 characters will not be sent. • Make sure that when the data block TCON_PHONE was created, the "Standard" option was selected for block access.
0	1	$80F1_H$	Sending of SMS messages is not enabled in the STEP 7 configuration of the CP: Enalble the "Allow SMS" option in the configuration of the CP.
0	1	$80F4_H$	Connection endpoint cannot be generated: Check the connection partner.
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed. or • Connection terminated by remote partner: First call TC_DISCON.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value): Check the configuration of the "TC_CON..." SDT.

* Further statuses that are not listed here can be found in the status display is of the "RDREC" or "WRREC" instructions in the two middle status bytes (STATUS[2], STATUS[3]).

TC_RECV: Receive data via the GSM network

Meaning

The TC_RECV instruction allows the reception of data via programmed connections of the following types:

- ISO-ON-TCP connections
- SMS connections
  To receive SMS messages, the phone number of the sender must be configured in the STEP 7 configuration of the receiving CP (authorized phone numbers). The sender must support the CLIP function.
  The phone number of the connection partner must be entered in the "TCON_PHONE" SDT.
  Wake-up SMS messages are filtered out.
- Telecontrol connections

Note

Receiving SMS messages from different senders

If you want to receive SMS messages from different senders, you have two alternatives:

- You configure several connections (TC_CON, TC_RECV, TC_DISCON).
  or
- You may only enter no telephone number for only one configured connection in the required data block "TCON_PHONE" in the "PhoneNumber" parameter. When receiving messages, this is then interpreted as a placeholder for all authorized connection partners.

You will find more detailed information on the connection types in the description of the TC_CON instruction.

The ID parameter references the GPRS connection. The value of ID must correspond to the value used for ID by TC_CON.

The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

The maximum amount of data to be received is specified with the LEN parameter.

The size of the data area specified in DATA must be at least as large as the number of bytes configured for LEN. Permitted data types in the data area specified in DATA are all except BOOL and ARRAY of BOOL. The received data is interpreted as if the sending partner had used the same data types.

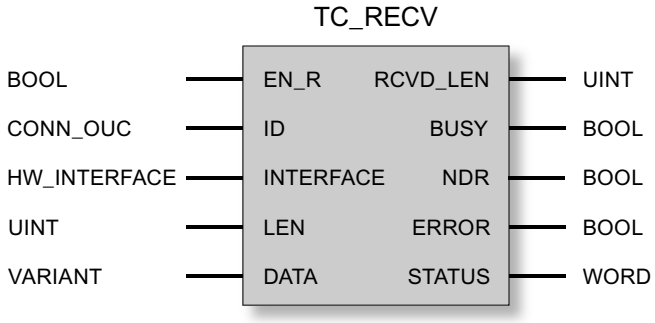
The DB (system data type) used for the connection description of TC_RECV must differ from a DB used for TC_SEND.

Storing SMS messages

Received SMS messages are stored retentively in the CP 1242-7 (25 storage spaces) and on the SIM card (varying number of storage spaces).

- After an SMS message has been read by TC_RECV the SMS message will be deleted from its storage space.
- If all of the storage spaces have been allocated and a new SMS message is received, the oldest SMS message is deleted.

Call interface in FBD representation



Explanation of the formal parameters

The following table explains all the formal parameters for the TC_RECV instruction

Parameter	Declaration	Data type	Range of values	Description
EN_R	INPUT	BOOL	0: Data reception locked 1: Data reception enabled	Enables / locks the reception of data. • Block version 1.1: After setting 1 to 0, the block is inactive. • Block version 1.0: After setting 1 to 0, the program block receives data again (until DONE = 0 and ERROR = 0). Note the information on the status code 80E3 of TC_CON.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTER-FACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
LEN	INPUT	UINT	1...2048	(minimum) number of bytes of data to be received, maximum 2048
DATA	INOUT	VARIANT		Address reference to the receive data area of the CPU *
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
RCVD_LEN	OUTPUT	UINT		Number of bytes of received data

Parameter	Declaration	Data type	Range of values	Description
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

* For special features of the DATA parameter for SMS texts, refer to the next section.

Configuring SMS texts with the DATA parameter

The instruction references the received SMS text with the pointer of the type VARIANT of the DATA parameter to the data area of the CPU.

If DATA references an operand of the data type STRING for the SMS text, the first two bytes of the SMS text will be interpreted as length information of the data type STRING and not as SMS text.

One option for the correct text representation of SMS messages to be received is to convert an Array of BYTE or Array of CHAR to a text string using the conversion function Chars_TO_Strg. Chars_TO_Strg at the EN parameter is linked to the output parameter ENO of TC_RECV.

For SMS texts, the CP does not support all special characters, for example umlauts (ü, ä etc.). The specification GSM 03.38 applies. There may be additional restrictions imposed by the GSM network provider.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS *	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	$80A3_H$	• An attempt is made to re-establish an existing connection. • An attempt is made to terminate a non-existent connection.
0	1	$80E0_H$	Internal error
0	1	8086_H	Illegal value for ID
0	1	$80E4_H$	Unknown ID: First call TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed. or • Connection terminated by remote partner: First call TC_DISCON.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the configuration of the "TC_CON..." SDT.

* Further statuses that are not listed here can be found in the status display is of the "RDREC" or "WRREC" instructions in the two middle status bytes (STATUS[2], STATUS[3]).

TC_CONFIG: Transferring configuration data to a CP

Meaning

With the TC_CONFIG instruction, parameters of a the CP 1242-7 configured in STEP 7 can be modified. The configured values are not overwritten retentively. The overwritten values remain valid until TC_CONFIG is called again or until the station starts up again (cold restart after cycling power).

If the STEP 7 configuration data of the CP needs to be changed permanently, the instruction needs to be called again each time the station restarts (cold restart) or a modified project must be downloaded to the station.

The CONFIG parameter points to the memory area with the configuration data. The configuration data is stored in a data block (DB). The structure of the DB is specified by the system data type (SDT) IF_CONF.

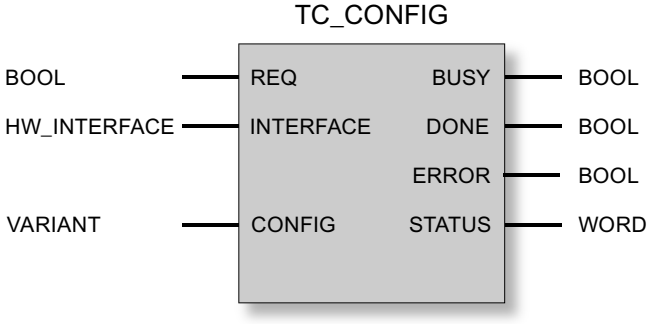
The configuration data to be modified on the CP is put together as necessary in blocks in IF_CONF "IF_CONF_..." for the individual parameters.

Parameters that are not intended to change as a result of the instruction are not entered in IF_CONF. They retain the value configured in STEP 7.

For detail information on assigning value to IF_CONF, refer to the section IF_CONF: SDT for telecontrol configuration data (Page 3846).

The INTERFACE parameter references the GPRS interface of the required local CP.

Call interface in FBD representation



Explanation of the formal parameters

The following table explains all the formal parameters for the TC_CONFIG instruction

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
INTERFACE	INPUT	HW_INTERFACE (WORD)		Reference to the interface of the local CP 1242-7
CONFIG	INOUT	VARIANT	See also "IF_CONF: SDT for telecontrol configuration data	Reference to the memory area with the collected configuration data to be modified
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.

Parameter	Declaration	Data type	Range of values	Description
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80EB_H$	Query temporarily rejected (the CP is currently being configured by STEP 7).
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the "IF_CONF" SDT.
0	1	$80F7_H$	Wrong ID in the parameter fields of the configuration data: Check the "IF_CONF" SDT.

TCON_...: SDTs for the telecontrol connection establishment

System data types TCON_... for the TC_CON instruction

To configure a telecontrol connection using the TC_CON instruction, the CONNECT parameter of the instruction is used for the connection description.

The connection description is specified by the structure of the system data type (SDT). The structure of the relevant SDT contains the parameters necessary to establish the connection with the remote communications partner.

For different connection types that depend on the remote communications partner, the following SDTs are used:

- TCON_IP_RFC for ISO-on-TCP connections to IPv4 stations with CP 1242-7
- TCON_IP_V4 for UDP connections to IPv4 stations (sending only)
- TCON_PHONE for connections to SMS clients
- TCON_WDC for connections to telecontrol servers or stations that can be reached via the telecontrol server.

The parameter assignment of the connection description is made in a data block of the same type as the SDT.

Creating a DB of the type TCON_...

You will need to type in the data types of the relevant DBs with the keyboard. They are not displayed in the selection list. The data types are not case-sensitive.

To create a TCON_... DB, follow the steps outlined below:

1. Create a data block of the type "global DB" with block access "Standard".
2. Create an SDT in the table of the parameter configuration of the DB by assigning the name and typing in the required type (for example "TCON_IP_RFC") in the cell of the data type. The SDT and its parameters are created (see below).
3. Configure the parameters that are described below for each SDT type.

Reserved bits are not displayed.

System data type TCON_IP_RFC for connections to IPv4 stations

This connection type is supported only on ISO-on-TCP connections to communications partners with a fixed IP address. The CP must be configured for the "GPRS direct" mode.

Table 11-110 Parameters of TCON_IP_RFC

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0C	Protocol variant 12 (C_h): ISO-on-TCP connection
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment • 1: Active connection establishment
6 ... 7	-	-	-	- reserved -

Byte	Parameter	Data type	Initial value	Description
8 ... 11	RemoteAddress	IP_V4		IP address of the connection partner
	ADDR	Array [1...4] of Byte		IP address of the relevant connection partner
12 ... 13	RemoteTSelector	TSelector		Remote T selector
	TSelLen	UINT		Length of the remote T selector "RemoteTSelector"
14 ... 45	TSel	Array [1...32] of Byte	any	Remote transport selector of the connection • When "ActiveEstablished" = 1: With active connection establishment, the T selector of the local partner must be the same as the T selector of the connection partner (passive connection establishment on the remote partner). • When "ActiveEstablished" = 0 correspondingly (passive connection establishment local, active connection establishment remote)
46 ... 47	LocalTSelector	TSelector		Local T selector
	TSelLen	UINT		Length of the local T selector "LOCAL_TSel"
48 ... 79	TSel	Array [1...32] of Byte	any	Local transport selector of the connection • When "ActiveEstablished" = 1: With active connection establishment, the T selector of the local partner must be the same as the T selector of the connection partner (passive connection establishment on the remote partner). • When "ActiveEstablished" = 0 correspondingly (passive connection establishment local, active connection establishment remote)

System data type TCON_IP_V4 for connections to IPv4 stations

This connection type is supported only for sending on UDP connections to communications partners with a fixed IP address.

To receive, ActiveEstablished = 0 must be set.

Table 11-111 Parameters of TCON_IP_V4

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0B	Protocol variant 11 (B_h): UDP connection

Byte	Parameter	Data type	Initial value	Description
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment Setting for sending and receiving data. • 1: Active connection establishment Setting for sending data only.
6 ... 9	RemoteAddress	IP_V4		IP address of the connection partner
	ADDR	Array [1...4] of Byte		The four bytes (ADDR[1] ... ADDR[4]) specify the four blocks of the IP address.
10 ... 11	RemotePort	UINT	1...65535	IP port of the connection partner Not relevant if ActiveEstablished = 0.
12 ... 13	LocalPort	UINT	1...65535	Local IP port ("0" is not permitted) Not relevant if ActiveEstablished = 1.

System data type TCON_PHONE for SMS connections

Note

Authorized phone numbers

The CP only accepts an SMS if the sending communication partner is authorized based on its phone number. These numbers are in configured for the CP in STEP 7 in the "authorized phone numbers" list.

SMS text

- Programmed SMS texts for SMS messages to be sent are accessed using the DATA parameter of the TC_SEND instruction.
- The text of a received SMS message is assigned to the address area of the CPU by the DATA parameter of the TC_RECV instruction.

Table 11-112 Parameters of TCON_PHONE

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0E	Protocol variant 14 (E_h): SMS connection
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment (not relevant for the CP 1242-7): • 0: Passive connection establishment (not relevant here) • 1: Active connection establishment

Byte	Parameter	Data type	Initial value	Description
6...7	-	-	-	- reserviert -
8 ... 31	PhoneNumber	STRING[22]		Call number of the connection partner Permitted values: Plus character (+) and numbers Note the exact notation of the international dialing code of the relevant phone number assigned by the network provider ("+" character or zeros). Without an entry for the PhoneNumber parameter, no connection partner is specified and SMS messages can be received reception from all authorized connection partners. Note the following during startup: Without an entry, TC_RECV first delivers the oldest received SMS message.

System data type TCON_WDC for connections to telecontrol servers or remote stations

You can configure the connection to the telecontrol server assigned to the S7-1200 or to a remote station that can be reached via the telecontrol server with TCON_WDC. The address data of the telecontrol server assigned to the CP can be found in STEP 7 in the "Telecontrol interface > Mode" tab of the CP. The telecontrol server or a remote station is addressed using the host name or the IP address.

The "RemoteWdcAddress" parameter of TCON_WDC specifies the Access ID of the connection partner.

Table 11-113 Parameters of TCON_WDC

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0F	Protocol variant 15 (F$_h$): Telecontrol connection using an IP address
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment • 1: Active connection establishment

Byte	Parameter	Data type	Initial value	Description
6 ... 7	-	-	-	- reserved -
8 ... 11	RemoteWdcAddress	DWORD		Specifies the Access ID (hex). The access ID depends on the connection partner. • Connection to a remote CP: The access ID is made up of the following: – STEP 7 project number – Station number – Slot If the remote station has more than one GPRS-CP and you do not want to specify the path, the last byte for the slot must be set to 0. You will find the access ID in the STEP 7 project in the "CP authentication of the CP" parameter group. • Connection to the telecontrol server: Access ID = 0 • To only write to the process image of the CP: Access ID = DW#16#FEEDDADA

IF_CONF: SDT for telecontrol configuration data

Structure of the system data type IF_CONF for the TC_CONFIG instruction

The CONFIG parameter of the TC_CONFIG instruction references the memory area with the configuration data of the CP 1242-7 to be modified. The configuration data stored in a data block is described as a structure of the system data type (SDT) IF_CONF.

IF_CONF is made up of a header followed by fields that correspond to the parameters or parameter areas of the CP in the device properties of the STEP 7 project.

The CP configuration data to be modified is collected together as IF_CONF fields. Parameters that will not be modified are ignored in the IF_CONF structure and remain as they were configured in the STEP 7 project.

Creating the DB and the IF_CONF structures

You can create the parameters of the CP within the IF_CONF DB in one or more structures each with one or more fields.

You will need to type in the data types of the fields using the keyboard. They are not displayed in the selection list. The data types are not case-sensitive.

Follow the steps below to create IF_CONF:

1. Create a data block of the type "global DB" with block access "Standard".
2. Create a structure (data type "Struct") in the table of the parameter configuration of the DB. You can specify any name.

3. Under this structure add a header by assigning the name of the header and typing it in the cell of the data type "IF_CONF_Header".
   The header of the structure and its three parameters (see below) is created.
4. Create a field for the first parameter to be changed by typing in the required data type (for example "IF_CONF_APN") in the cell of the data type.
5. Repeat the last step for all parameters you want to change on the CP using the TC_CONFIG instruction.
6. Finally, update the number of fields in the header in the "subfieldCnt" parameter.

Header of IF_CONF

Table 11-114 IF_CONF_Header

Byte	Parameter	Data type	Initial value	Description
0 ... 1	fieldType	UINT		Field type: Must always be 0.
2 ... 3	fieldId	UINT		Field ID: Must always be 0.
4 ... 5	subfieldCnt	UINT		Total number of fields contained in the structure

General parameters of the parameter fields

Each field has the following general parameters:

- Id
  This parameter identifies the field and must not be modified.
- Length
  This parameter indicates the length of the field. The value serves as information.
  Fields with strings and / or arrays have a variable length. Due to hidden bytes, the actual length of fields can be greater than the sum of the displayed parameters.
- Mode
  The following values are permitted to these parameters:

Table 11-115 Values of "Mode"

Value	Meaning
1	Permanent validity of the configuration data Not relevant for the CP 1242-7
2	Temporary validity of the configuration data, including deleting of existing permanent configuration data The permanent configuration data is replaced by the parameter fields of IF_CONF.

Field for the parameter area "GPRS access"

Table 11-116 IF_CONF_APN

Parameter	Data type	Initial value	Description
Id	UINT	4	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 174
Mode	UINT		Validity (1: permanent, 2: temporary)
AccesspointGPRS	STRING [98]		APN: Name of the access point of the GSM network provider to the Internet
AccesspointUser	STRING [42]		APN user name
AccesspointPassword	STRING [22]		APN password

Field for the parameter area "CP identification"

Table 11-117 IF_CONF_Login

Parameter	Data type	Initial value	Description
Id	UINT	5	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 54
Mode	UINT		Validity (1: permanent, 2: temporary)
ModemName	STRING [22]		Access ID The value cannot be set.
ModemPassword	STRING [22]		Telecontrol password (max. 20 characters)

Field for the parameter area "Telecontrol server access"

This field is only used when the telecontrol server is addressed with a name that can be resolved by DNS. If the telecontrol server is addressed with its IP address, the "IF_CONF_TCS_IP_V4" field is used.

In STEP 7, the corresponding data is located in the "Mode" parameter area.

If there is more than one telecontrol server, use the field once per server.

Table 11-118 IF_CONF_TCS_Name

Parameter	Data type	Initial value	Description
Id	UINT	6	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 266
Mode	UINT		Validity (1: permanent, 2: temporary)
TcsName	-	-	- reserved -
	STRING [254]		Name of the telecontrol server that can be resolved by DNS
RemotePort	UINT		Port of the telecontrol server
Rank	UINT		Priority of the server [1, 2] 1 = first telecontrol server, 2 = second telecontrol server

Field for the parameter area "Telecontrol server access"

This field is only used when the telecontrol server is addressed by its IP address. If the telecontrol server is addressed by its DNS name, the "IF_CONF_TCS_Name" field is used.

In STEP 7, the corresponding data is located in the "Mode" parameter area.

If there is more than one telecontrol server, use the field once per server.

Table 11-119 IF_CONF_TCS_IP_v4

Parameter	Data type	Initial value	Description
Id	UINT	7	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
RemoteAddress	IP_V4		IP address of the telecontrol server
RemotePort	UINT		Port of the telecontrol server
Rank	UINT		Priority of the server [1, 2] 1 = first telecontrol server, 2 = second telecontrol server

Field for the "Mode" parameter area

In STEP 7, the corresponding data is located in the parameter areas "Mode" and Modem settings".

Table 11-120 IF_CONF_GPRS_Mode

Parameter	Data type	Initial value	Description
Id	UINT	8	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 10
Mode	UINT		Validity (1: permanent, 2: temporary)
GPRSmode	UINT		Mode of the CP: • 0 = Telecontrol • 1 = GPRS direct
TemporaryStation	BOOL		Bit 0: Temporary connection If this option is selected, the CP only establishes a temporary connection to send data. Once the frames have been transferred, the CP terminates the connection again. • 1: activated (temporary connection) • 0: deactivated (permanent connection)
SMS_Enabled	BOOL		Bit 1: Allow SMS Selecting the option allows the S7 station to send SMS messages. • 1: activated (SMS allowed) • 0: deactivated (no SMS)

Field for the "SMSC" parameter

In STEP 7, the corresponding data is located in the parameter area "Modem settings".

Table 11-121 IF_CONF_SMS_Provider

Parameter	Data type	Initial value	Description
Id	UINT	10	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 28
Mode	UINT		Validity (1: permanent, 2: temporary)
SMSProvider	STRING [20]		Node number of the SMS center (SMSC) of the GSM network provider with which the contract was signed for this station.

Field for the "PIN" parameter

In STEP 7, the corresponding data is located in the parameter area "Modem settings".

Table 11-122 IF_CONF_PIN

Parameter	Data type	Initial value	Description
Id	UINT	11	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 16
Mode	UINT		Validity (1: permanent, 2: temporary)
Pin	STRING [8]		PIN of the SIM card inserted in the SIM card The parameter is not relevant if the PIN was correctly configured. If the PIN was incorrectly configured, the correct PIN can be entered.

Field for monitoring times

In STEP 7, the corresponding data is located in the parameter areas "Keepalive timeout" and Operating mode".

Table 11-123 IF_CONF_TC_Timeouts

Parameter	Data type	Initial value	Description
Id	UINT	12	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 12
Mode	UINT		Validity (1: permanent, 2: temporary)
KeepAliveTimeout	-	-	- Reserved - (cannot be set)

Parameter	Data type	Initial value	Description
SendTimeout	UINT		Connection monitoring time: Monitoring time of the connection to the communications partner (seconds) Relevant in the modes "Telecontrol" and "GPRS direct"
RedialTimeout	UINT		Reconnection delay: Basic value for the wait time until the next attempt to establish a connection following an unsuccessful connection establishment. After every 3 attempts, the basic value is doubled up to a maximum of 900 s. Range of values: 10 to 600 s. Example: Basic value 20 results in the following dialing intervals: three times 20 s, three times 40 s, three times 80 s etc. up to a maximum of 900 s. If a second telecontrol server / router is configured, the CP attempts to connect to the second partner at the 4th attempt. If the second partner is also unreachable, the 7th time the CP attempts to connect to the first partner again. Not relevant for SMS connections

Field for the "Wake up right" parameter area

Table 11-124 IF_CONF_WakeupList

Parameter	Data type	Initial value	Description
Id	UINT	13	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 246
Mode	UINT		Validity (1: permanent, 2: temporary)
WakeupPhone [1...10]	ARRAY [1...10] of STRING [22]		Phone number subscriber authorized to wake up The asterisk (*) at the end of a call number is used a placeholder for direct dialing numbers.

Field for the "Preferred GSM networks" parameter area

Table 11-125 IF_CONF_PrefProvider

Parameter	Data type	Initial value	Description
Id	UINT	14	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 46
Mode	UINT		Validity (1: permanent, 2: temporary)
Provider [1...5]	ARRAY [1...5] of STRING [6]		Alternative GSM networks with priority 1 to 5 into which the CP dials. Up to 5 networks can be configured. No. 1 with highest priority, no. 5 with lowest priority. Entry of the Public Land Mobile Network (PLMN) of the network provider consisting of Mobile Country Code (MCC) and Mobile Network Code (MNC). Example (test network of Siemens AG): 26276

Field for the "DNS configuration" parameter area

Table 11-126 IF_CONF_DNS

Parameter	Data type	Initial value	Description
Id	UINT	16	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
DNS_IP [1]	IP_V4		IP address of the 1st domain name system server
DNS_IP [2]	IP_V4		IP address of the 2nd domain name system server

Field for the "Time-of-day synchronization" parameter area

Table 11-127 IF_CONF_NTP

Parameter	Data type	Initial value	Description
Id	UINT	17	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 24
Mode	UINT		Validity (1: permanent, 2: temporary)
NTP_IP [1]	ARRAY [1...4] of IP_V4		IP address of NTP server 1
...	...		(IP address of NTP server 2...3)
NTP_IP [4]	ARRAY [1...4] of IP_V4		IP address of NTP server 4

Block for activating / deactivating TeleService users

SDT for activating or deactivating TeleService users already configured in the STEP 7 project of the CP. In STEP 7, the corresponding data can be found in the parameter area "TeleService settings" > "TeleService user management".

Table 11-128 IF_CONF_GPRS_UserList

Parameter	Data type	Initial value	Description
Id	UINT	19	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 506
Mode	UINT		Validity (1: permanent, 2: temporary)
GPRS_User [1...10]	ARRAY [1...10] of GPRS_User		TeleService user no. 1 to max. no. 10

The array is formed from the parameter records for the TeleService users ("GPRS_User" [1...n]).

Table 11-129 GPRS_User [n] (parameter for TeleService user)

Parameter	Data type	Initial value	Description
UserName [n]	STRING [22]		TeleService user name
Password [n]	STRING [22]		- The string must be empty! -
Diag_Allowed [n]	BOOL		- Reserved - (cannot be set)
Teleserv_Allowed [n]	BOOL		Activation of the TeleService user • 0 = user is deactivated • 1 = user is activated
FW_Load_Allowed [n]	BOOL		- Reserved - (cannot be set)

Field for setting the parameters for TeleService access (DNS name of the server)

Access data of the TeleService server (switching station).

In STEP 7, the corresponding data is located in the parameter area "TeleService settings".

If there is more than one TeleService server, use the field once per server.

Table 11-130 IF_CONF_TS_Name

Parameter	Data type	Initial value	Description
Id	UINT	20	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 266
Mode	UINT		Validity (1: permanent, 2: temporary)
ts_name	String [254]		Name of the TeleService server that can be resolved by DNS
RemotePort	UINT		Port of the engineering station
Rank	UINT		Priority of the server [1] or [2] 1 = server 1, 2 = server 2

Field for setting the parameters for TeleService access (IP address of the server)

Access data of the TeleService server (switching station).

In STEP 7, the corresponding data is located in the parameter area "TeleService settings".

If there is more than one TeleService server, use the field once per server.

Table 11-131 IF_CONF_TS_IF_V4

Parameter	Data type	Initial value	Description
Id	UINT	21	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
RemoteAddress	IP_V4		IP address of the TeleService server

Parameter	Data type	Initial value	Description
RemotePort	UINT		Port of the TeleService server
Rank	UINT		Priority of the server [1] or [2] 1 = server 1, 2 = server 2

11.6.5.5 TeleService

TM_MAIL: Transfer email

Description of TM_MAIL

Description

The "TM_MAIL" instruction works asynchronously, in other words, its execution extends over multiple calls. You must specify an instance when you call the instruction "TM_MAIL". The attribute "retentive" may not be set in the instance. This attribute ensures that the instance is initialized when the CPU switches from STOP to RUN and that a new e-mail send job can be triggered afterwards.

You start the sending of an e-mail with an edge change from "0" to "1" for the REQ parameter. The job status is indicated by the output parameters BUSY , "DONE", "ERROR", "STATUS" and "SFC_STATUS". "SFC_STATUS" corresponds in this case to the "STATUS" output parameter of the called communication blocks.

The output parameters DONE, ERROR, STATUS, and SFC_STATUS are each displayed for only one cycle if the status of the BUSY output parameter changes from "1" to "0". The following table shows the relationship between BUSY, DONE, and ERROR. Using this table, you can determine the current status of the instruction "TM_MAILand when the sending of the e-mail is complete.

DONE	BUSY	ERROR	Description
0	1	0	The job is being processed.
1	0	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS and SFC_STATUS parameters.
0	0	0	The "TM_MAIL" instruction was not assigned a (new) job.

If the CPU changes to STOP mode while "TM_MAIL" is active, the communication connection to the mail server aborts. The communication connection to the mail server will also be lost if

communication problems occur on the Industrial Ethernet bus. In this case, the transfer of the e-mail will be interrupted and it will not reach its recipient.

Notice

Changing user programs

You can change the parts of your user program that directly affect calls of "TM_MAIL" only:

- when the CPU is in "STOP" mode or
- when no mail is being sent (REQ = 0 and BUSY = 0).

This relates, in particular, to deleting and replacing program blocks that contain "TM_MAIL" calls or calls for the instance of "TM_MAIL".

Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status for the TPC/IP communication functions via Industrial Ethernet.

A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The ADDR_MAIL_SERVER input parameter of the instruction is taken from the "TM_MAIL" instruction again each time the sending of an e-mail is triggered. If a change is made during operation, the "new" value only becomes effective once an e-mail is triggered again.

In contrast, the WATCH_DOG_TIME, TO_S, CC, FROM, SUBJECT, TEXT, ATTACHMENT, and, if applicable, USERNAME and PASSWORD parameters are taken from the "TM_MAIL" instruction while it is running, which means that they may only be changed after the job is complete (BUSY = 0)

Setting the parameters of the TS Adapter IE

On the TS Adapter IE, you need to specify parameters for outgoing calls in such a way as to enable the TS Adapter IE to establish a connection to the dial-up server of your service provider.

If you set "When required" for connection establishment, the connection will only be established when an e-mail needs to be sent.

Connection establishment can take longer (approx. one minute) for an analog modem connection. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection.

Parameters

The following table shows the parameters of the "TM_MAIL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter REQUEST: Activates the sending of an e-mail on a rising edge.
ID	Input	CONN_OUC (Word)	D, L or constant	Reference to the connection to be established. See ID parameter of the TCON (Page 3662), TDISCON (Page 3669), TSEND (Page 3674), and TRCV (Page 3677) instructions. A number that is not used for any additional instances of this instruction in the user program must be entered here.
TO_S (Page 3858)	Input	STRING	D	Input parameter for receiver addresses: STRING with a maximum length of 240 characters (see example call).
CC (Page 3858)	Input	STRING	D	Input parameter for CC recipient addresses (optional): STRING with a maximum length of 240 characters (see example call). If an empty string is assigned here, the e-mail is not sent to a CC recipient.
SUBJECT	Input	STRING	D	Input parameter for subject of the e-mail: STRING with a maximum length of 240 characters.
TEXT	Input	STRING	D	Input parameter for text of the e-mail (optional): Reference to a data string with a maximum length of 240 characters. If an empty string is assigned at this parameter, the e-mail is sent without text.
ATTACHMENT	Input	VARIANT	I, Q, M, D, L	Input parameter for e-mail attachment (optional): Reference to a byte/word/double word field with a maximum length of 65534 bytes. If no value is assigned, the e-mail is sent without an attachment.
DONE	Output	BOOL	I, Q, M, D, L	• DONE = 0: Job not yet started or still executing. • DONE = 1: Job was executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: E-mail transmission is not yet complete. • BUSY=0: The processing of "TM_MAIL" was stopped.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR=1: An error occurred during processing. STATUS and SFC_STATUS supply detailed information about the type of error.
STATUS (Page 3859)	Output	WORD	I, Q, M, D, L	Output/status parameter STATUS: Return value or error information of the "TM_MAIL" instruction.

Parameter	Declaration	Data type	Memory area	Description
ADDR_MAIL_SERVER	Static*	DWORD	I, Q, M, D, L	IP address input parameter of the mail server: Specify as a data word in HEX format, for example: IP address = 192.168.0.200. ADDR_MAIL_SERVER = DW#16#C0A800C8, where: • 192 = 16#C0, • 168 =16#A8 • 0 = 16#00 and • 200 = 16#C8.
WATCH_DOG_TIME	Static*	TIME	I, Q, M, D, L	Input parameter for max. period of time: The "TM_MAIL" instruction should establish a connection within the period specified by WATCH_DOG_TIME. The block is exited with an error if this period is exceeded. The time before the block is exited and the error is output can exceed the WATCH_DOG_TIME because connection termination also takes a certain amount of time. To begin with, you should set a period of 2 minutes. If you have an ISDN telephone connection, a significantly shorter period can be selected.
USERNAME	Static*	STRING	D	Input parameter for user name: STRING with a maximum length of 180 characters. A user name is an absolute requirement for authentication.
PASSWORD	Static*	STRING	D	Input parameter for password: STRING with a maximum length of 180 characters. A password is an absolute requirement for authentication.
FROM (Page 3858)	Static*	STRING	D	Input parameter for sender address: STRING with a maximum length of 240 characters (see example call).
SFC_STATUS (Page 3859)	Static*	WORD	I, Q, M, D, L	Output/status parameter "SFC_STATUS": Error information of the called communication blocks.
*The values of the parameter are not modified at every call of the instruction "TM_MAIL". The values are in the static parameters of the instance and are only written once at the first call of the instruction.				

You will find more detailed information on valid data types in "Overview of the valid data types (Page 1908)".

Note

Optional parameters

The optional parameters CC, TEXT, and ATTACHMENT are only sent with the e-mail if the corresponding parameters contain a string of length > 0.

SMTP authentication

Authentication refers here to a procedure for ensuring an identity, for example, by a password query.

The "TM_MAIL" instruction supports the SMTP authentication procedure AUTH-LOGIN that is requested by most mail servers. For information about the authentication procedure of your mail server, please refer to your mail server manual or the website of your Internet service provider.

To use the AUTH-LOGIN authentication procedure, the "TM_MAIL" instruction requires the user name with which it can log onto the mail server. This user name corresponds to the user name with which you set up a mail account on your mail server. It is made available to the "TM_MAIL" instruction in the USERNAME parameter.

To log on, the "TM_MAIL" instruction also requires the associated password. This password corresponds to the password you specified when you set up your mail account. It is made available to the "TM_MAIL" instruction in the PASSWORD parameter.

The user name and the password are each transferred to the mail server unencrypted (BASE64-coded).

If no user name is specified in the DB, the AUTH-LOGIN authentication procedure is not used. The e-mail is then sent without authentication.

Parameters TO_S, CC, and FROM

Description

The TO_S, CC, and FROM parameters are strings with, for example, the following content:

- TO: <wenna@mydomain.com>, <ruby@mydomain.com>,
- CC: <admin@mydomain.com>, <judy@mydomain.com>,
- FROM: <admin@mydomain.com>

Note the following rules when entering the parameters:

- The "TO:", "CC:", and "FROM:" characters must be entered.
- A space and an opening pointed bracket "<" must be entered before each address.
- A closing pointed bracket ">" must be entered after each address.
- A comma must be entered after each address in TO and CC.
- Only one e-mail may be entered in FROM, and there must not be a comma at the end of this address

Because of runtime and memory space, the "TM_MAIL" instruction does not perform any syntax check of the parameters TO_S, CC and FROM.

STATUS and SFB_STATUS parameters

Description

The return values of the "TM_MAIL" instruction can be classified as follows:

- W#16#0000: "TM_MAIL" was completed successfully
- W#16#7xxx: Status of "TM_MAIL"
- W#16#8xxx: An error was reported during the internal call of a communication block or from the mail server.

The following table shows the return values of "TM_MAIL" except for error codes of the internally called communication blocks.

Return value STATUS* (W#16#...):	Return value SFB_STATUS (W#16#...):	Explanation	Notes
0000	-	The processing of "TM_MAIL" was completed without error.	A without error completion of "TM_MAIL" does not mean that the sent e-mail will necessarily arrive (see below - note on point 1)
7001		"TM_MAIL" is active (BUSY = 1).	Initial call; job has started
7002	7002	"TM_MAIL" is active (BUSY = 1).	Intermediate call; job already active
8xxx	xxxx	The processing of "TM_MAIL" was completed with an error code of the internally called communication instructions.	For detailed information on the evaluation of the SFB_STATUS parameter, refer to the descriptions of the STATUS parameter of the communication instructions.
8010	xxxx	Error during connection establishment.	For further information on the evaluation of the SFB_STATUS parameter, refer to the descriptions of the STATUS parameter of the "TCON (Page 3662)" instruction.
8011	xxxx	Error sending the data.	For further information on the evaluation of SFB_STATUS, refer to the descriptions of the STATUS parameter of the "TSEND (Page 3674)" instruction.
8012	xxxx	Error receiving the data.	For further information on the evaluation of SFB_STATUS, refer to the descriptions of the STATUS parameter of the "TRCV (Page 3677)" instruction.
8013	xxxx	Error during connection establishment.	For further information on the evaluation of SFB_STATUS, refer to the descriptions of the STATUS parameter of the "TCON (Page 3662)" and "TDISCON (Page 3669)" instructions.

Return value STATUS* (W#16#...):	Return value SFB_STATUS (W#16#...):	Explanation	Notes
8014	-	Establishment of a connection is not possible.	You have possibly entered an incorrect mail server IP address (ADDR_MAIL_SERVER) or a time span that is too short (WATCH_DOG_TIME) to establish the connection. It is also possible that the CPU has no connection to the network or that the CPU configuration is incorrect.
8016	-	Error copying the attachment	-
82xx, 84xx, or 85xx	-	The error message originates from the mail server and corresponds, except for the "8", to the error number of the SMTP protocol. The following columns list several error codes that can occur:	See point 2 in the note.
8450	-	Action not executed: Mailbox not available/ not reachable.	Try again later.
8451	-	Action aborted: Local processing error	Try again later.
8500	-	Syntax error: Error not recognized. This also includes the error when a command string is too long. This can also be occur when the e-mail server does not support the LOGIN authentication procedure.	Check the parameters of "TM_MAIL". Try to send an e-mail without authentication. To do this, replace the USERNAME parameter with an empty string.
8501	-	Syntax error: Parameter or argument incorrect	You have possibly entered an incorrect address in TO_S or CC.
8502	-	Command unknown or not implemented.	Check your entries, in particular the FROM parameter. This is possibly incomplete and you have forgotten "@" or ".".
8535	-	SMTP authentication incomplete.	You have possibly entered an incorrect user name or incorrect password.
8550	-	Mail server cannot be reached, you have no access rights.	You have possibly entered an incorrect user name or password, or the mail server does not support your LOGIN. Another cause of error could be an incorrect entry of the domain name after the "@" in TO_S or CC.
8552	-	Action aborted: Assigned memory size has been exceeded	Try again later.
8554	-	Transmission failed.	Try again later.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".			

Note

Status error

1. An incorrect entry of the recipient addresses does not generate a status error of the "TM_MAIL" instruction. In this case, there is no guarantee that the e-mail will be sent to other recipients, even if these were entered correctly.
2. You can find more detailed information on the SMTP error code and other error codes in SMTP protocol on the Internet or in the error documentation of the mail server. You can also view the most recent message from the mail server in your instance DB in the BUFFER1 parameter. If you look under "Data", you will find the data most recently sent by the "TM_MAIL" instruction.

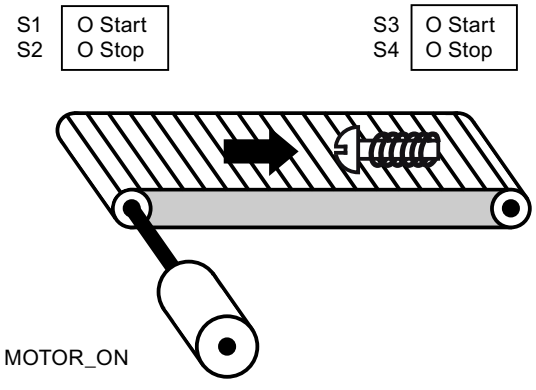
11.7 Programming examples

11.7.1 LAD programming examples

11.7.1.1 Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbuttons at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbuttons at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

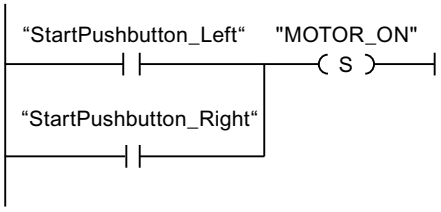
The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
StartPushbutton_Left (S1)	Input	BOOL	Start pushbutton on the left side of the conveyor belt
StopPushbutton_Left (S2)	Input	BOOL	Stop pushbutton on the left side of the conveyor belt
StartPushbutton_Right (S3)	Input	BOOL	Start pushbutton on the right side of the convey-or belt
StopPushbutton_Right (S4)	Input	BOOL	Stop pushbutton on the right side of the convey-or belt
MOTOR_ON	Output	BOOL	Turn on the conveyor belt motor

The following networks show the LAD programming for solving this task:

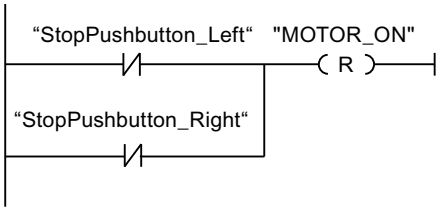
Network 1:

The conveyor belt motor is switched on when start pushbutton "S1" or "S3" is pressed.



Network 2:

The conveyor belt motor is switched off when stop pushbutton "S2" or "S4" is pressed.



See also

---(S)---: Set output (Page 2204)

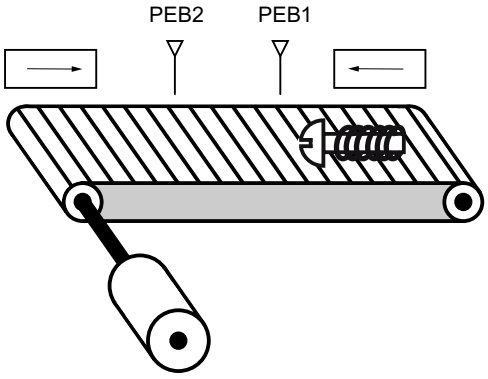
---| |---: Normally open contact (Page 2199)

---(R)---: Reset output (Page 2203)

11.7.1.2 Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The detected running direction of the belt is indicated by a RIGHT arrow or a LEFT arrow. If additional conveyed material is approaching PEB1 from the right or PEB2 from the left, the displayed arrow is initially switched off until, after both photoelectric barriers are passed, the running direction can be detected again and the corresponding arrow displayed. For the solution of the task, 2 edge memory bits are needed that detect the signal change from "0" to "1" at the two photoelectric barriers.



Implementation

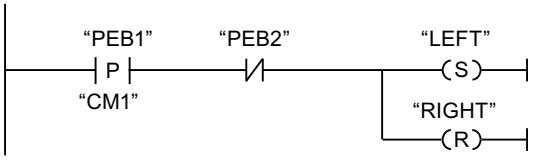
The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RIGHT	Output	BOOL	Display during movement to right
LEFT	Output	BOOL	Display during movement to left
CM1	Input	BOOL	Edge bit memory 1
CM2	Input	BOOL	Edge bit memory 2

The following networks show the LAD programming for solving this task:

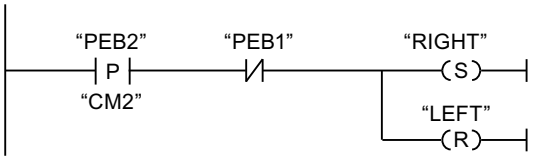
Network 1:

If the signal state changes from "0" to "1" (positive edge) at photoelectric barrier "PEB1" and, at the same time, the signal state at "PEB2" is "0", the object on the belt is moving to the left.



Network 2:

If the signal changes from "0" to "1" (positive edge) at photoelectric barrier "PEB2" and, at the same time, the signal state at "PEB1" is "0", the object on the belt is moving to the right.



See also

---(S)---: Set output (Page 2204)

---| / |---: Normally closed contact (Page 2200)

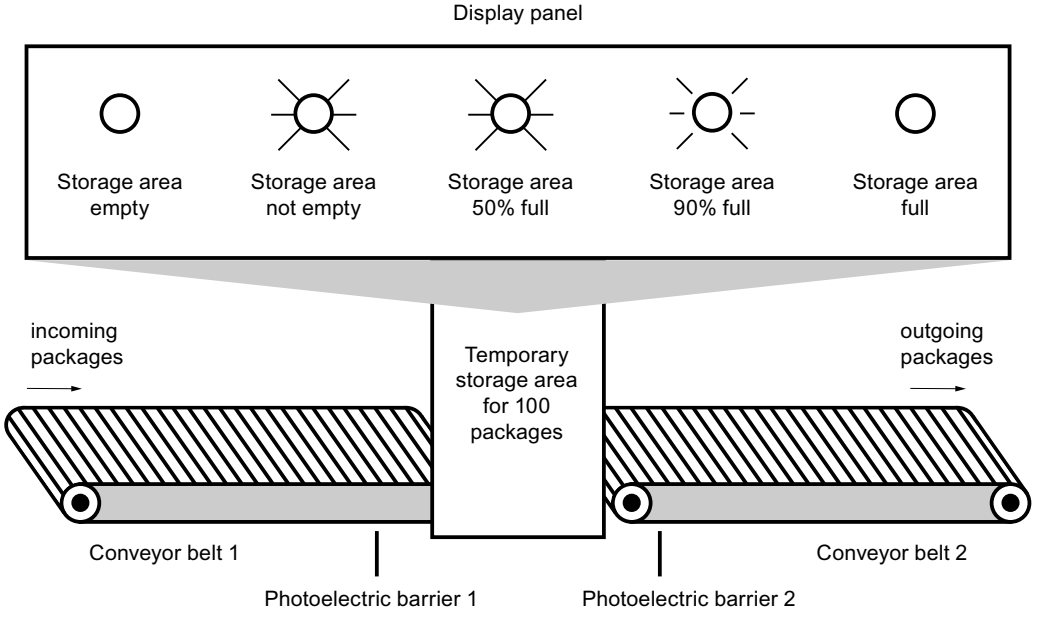
---(R)---: Reset output (Page 2203)

--|P|--: Scan operand for positive signal edge (Page 2209)

11.7.1.3 Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.



Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RESET	Input	BOOL	Reset counter
LOAD	Input	BOOL	Adjust the current counter value to the value of the PV parameter.
MAX STORAGE AREA FILL AMOUNT	Input	INT	Maximum possible number of packages in the storage area
PACKAGECOUNT	Output	INT	Number of packages in the storage area (current count value)
STOCK_PACKAGES	Output	BOOL	Is set when the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag.
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50 % full
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90 % full
STOR_FULL	Output	BOOL	Display lamp: Storage area full
VOLUME_50	Input	INT	Comparison value: 50 packages
VOLUME_90	Input	INT	Comparison value: 90 packages
VOLUME_100	Input	INT	Comparison value: 100 packages

The following networks show the LAD programming for activating the lamps:

Network 1:

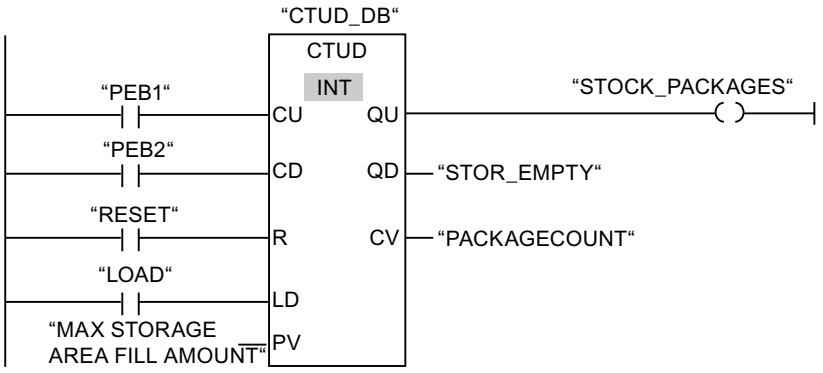
When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

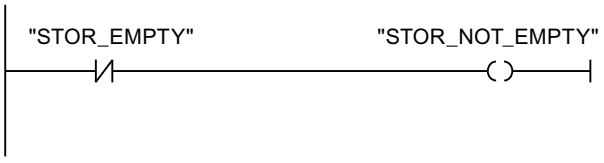
The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "MAX STORAGE AREA FILL AMOUNT" tag. As long as the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".



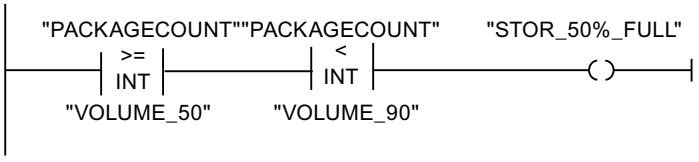
Network 2:

As long as there are packages in the storage area, the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.



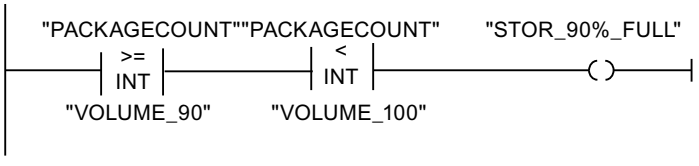
Network 3:

If the number of packages in the storage area is greater than or equal to 50, the "Storage area 50 % full" lamp switches on.

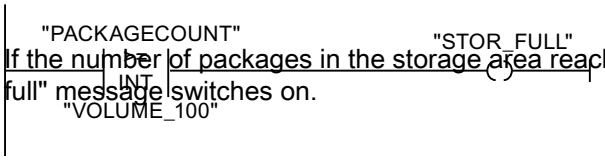


Network 4:

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.



Network 5:



If the number of packages in the storage area reaches 100, the lamp for the "Storage area full" message switches on.

See also

---( )---: Assignment (Page 2201)

CTUD: Count up and down (Page 2264)

CMP >=: Greater or equal (Page 2281)

CMP <: Less than (Page 2286)

---| |---: Normally open contact (Page 2199)

---| / |---: Normally closed contact (Page 2200)

11.7.1.4 Example of calculating an equation

Calculating a complex equation

Do you want to program a complex calculation consisting of multiple arithmetic operations or logic operations within one instruction box? The "CALCULATE" instruction is available to you for this purpose.

Depending on the data type you choose, different arithmetic operations are available to you which can be combined with one another.

The following programming example shows how you can enter and calculate a complex equation:

RESULT = ((5 + 10) x 4) / 6

Implementation

1. First, define the following tags in the block interface of your LAD block:

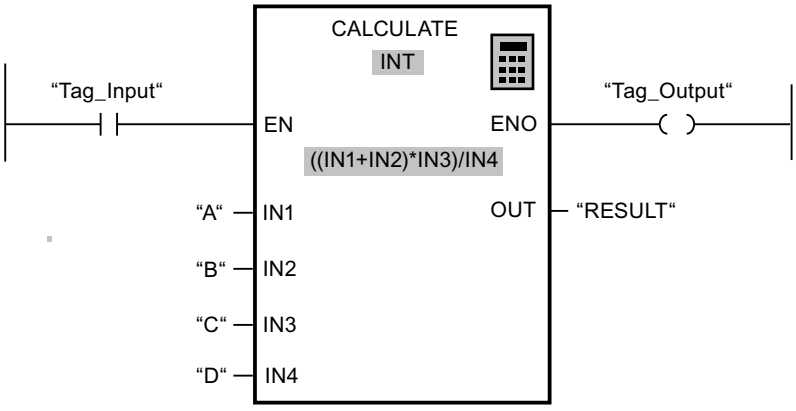
Name	Declaration	Data type	Comment	Values
A	Input	INT	First value for addition	5
B	Input	INT	Second value for addition	10
C	Input	INT	Multiplier	4
D	Input	INT	Divisor	6
RESULT	Output	INT	End result	10

2. You can select the data type INT for the instruction from the "<???>" drop-down list.
3. Interconnect the tags declared in the block interface with the inputs and outputs of the instruction box.

4. Click the "Calculator" icon on the top right corner of the instruction box to enter the equation to be calculated.
   The dialog "Edit "Calculate" instruction" opens.
5. Enter the following expression in the "OUT:=" field:
   ((IN1 + IN2) * IN4) / IN3
   The equation is shown in the instruction box.

Result

The following network shows the result in the LAD programming language:



If the "Tag_Input" has the signal state "1", the instruction is executed. The value of operand "A" is added to the value of operand "B". The subtotal is multiplied by "C" and then divided by the value of the "D" operand. The end result is saved in the "RESULT" operand.

See also

CALCULATE: Calculate (Page 2300)

11.7.1.5 Example of controlling room temperature

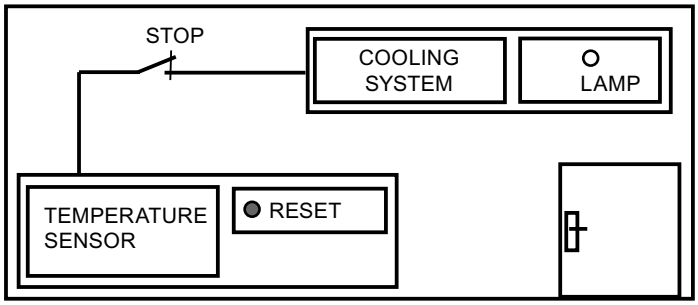
Controlling room temperature

In a cold room, the temperature must be maintained below zero degrees Celsius. A sensor is used to check for any temperature fluctuations. If the temperature rises above zero degrees Celsius, the cooling system switches on for a preset time. The "Cooling system on" lamp is lit during this time.

The cooling system and the lamp are turned off if one of the following conditions is met:

- The sensor reports a temperature fall below zero degrees Celsius.
- The preset cooling time has elapsed.
- The "STOP" pushbutton is pressed.

If the preset cooling time has expired and the temperature in the cold room is still too high, the cooling system can be restarted with the "RESET" pushbutton.



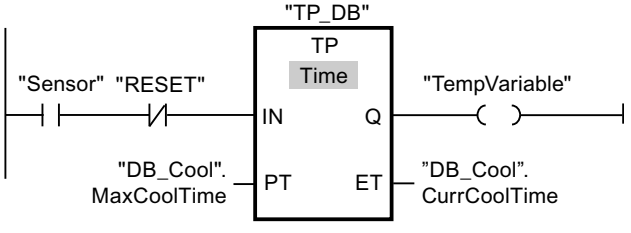
Implementation

The following table shows the definition of the tags used:

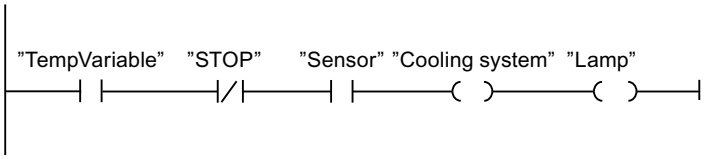
Name	Declaration	Data type	Comment
Sensor	Input	BOOL	Temperature sensor signal
RESET	Input	BOOL	Restart
STOP	Input	BOOL	The cooling system is switched off.
MaxCoolTime	-	TIME	Preset cooling time This tag is defined in the "DB_Cool" data block.
CurrCoolTime	-	TIME	Currently elapsed cooling time This tag is defined in the "DB_Cool" data block.
Cooling system	Output	BOOL	The cooling system is switched on.
Lamp	Output	BOOL	The lamp for the "Cooling system on" message is switched on.
TempVariable	Temp	BOOL	Temporary tag This tag stores the signal state of the IEC timer TP.

The following network shows the LAD programming for controlling room temperature:

Network 1:



Network 2:



When the temperature in the cold room rises above zero degrees Celsius, the signal state at the "Sensor" operand switches from "0" to "1" (positive signal edge). In the case of a positive signal edge at the IN input of the timer function, the preset cooling time is started and the "TempVariable" receives the signal state "1". The signal state "1" of the "TempVariable" causes the cooling system as well as the display lamp to be turned on in network 2. The "Sensor", "Cooling system" and "Lamp" outputs must be programmed in network 2, because program only one coil can be programmed at the Q output of the timer function.

If the temperature in the cold room falls below zero degrees Celsius, the signal state of the sensor switches back to "0". This switches the cooling system and lamp off.

If the sensor does not signal a temperature drop, the cooling system and lamp are switched off after the preset cooling time has elapsed, at the latest. In this case, the cooling process can be restarted with the "RESET" pushbutton. Pressing and releasing the pushbutton generates a new positive signal edge at the IN input that restarts the cooling system.

The cooling system and the display lamp can be turned off with the "STOP" pushbutton at any time.

See also

---( )---: Assignment (Page 2201)

TP: Generate pulse (Page 2218)

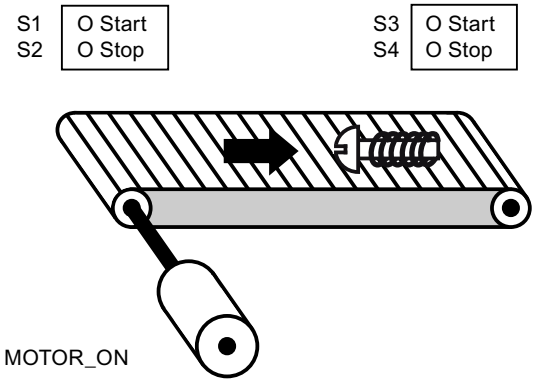
---| |---: Normally open contact (Page 2199)

11.7.2 FBD programming examples

11.7.2.1 Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbuttons at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbuttons at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

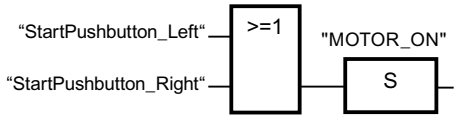
The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
StartPushbutton_Left (S1)	Input	BOOL	Start pushbutton on the left side of the conveyor belt
StopPushbutton_Left (S2)	Input	BOOL	Stop pushbutton on the left side of the conveyor belt
StartPushbutton_Right (S3)	Input	BOOL	Start pushbutton on the right side of the convey-or belt
StopPushbutton_Right (S4)	Input	BOOL	Stop pushbutton on the right side of the convey-or belt
MOTOR_ON	Output	BOOL	Turn on the conveyor belt motor

The following networks show the FBD programming for solving this task:

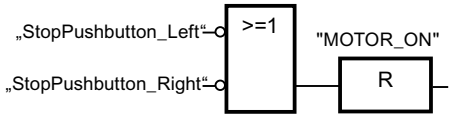
Network 1:

The conveyor belt motor is switched on when start pushbutton "S1" or "S3" is pressed.



Network 2:

The conveyor belt motor is switched off when stop pushbutton "S2" or "S4" is pressed.



See also

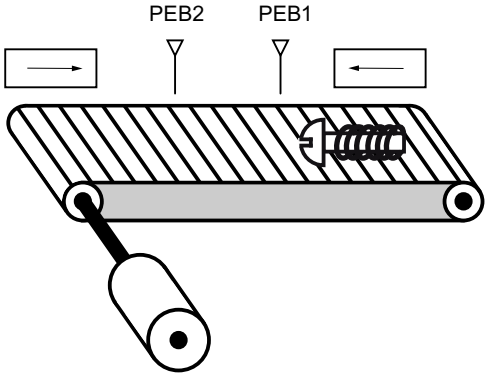
>=1: OR logic operation (Page 2473)

R: Reset output (Page 2479)

11.7.2.2 Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The detected running direction of the belt is indicated by a RIGHT arrow or a LEFT arrow. If additional conveyed material is approaching PEB1 from the right or PEB2 from the left, the displayed arrow is initially switched off until, after both photoelectric barriers are passed, the running direction can be detected again and the corresponding arrow displayed. For the solution of the task, 2 edge memory bits are needed that detect the signal change from "0" to "1" at the two photoelectric barriers.



Implementation

The following table shows the definition of the tags used:

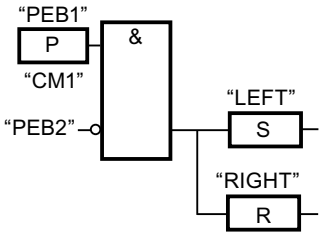
Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2

Name	Declaration	Data type	Description
RIGHT	Output	BOOL	Display during movement to right
LEFT	Output	BOOL	Display during movement to left
CM1	Input	BOOL	Edge bit memory 1
CM2	Input	BOOL	Edge bit memory 2

The following networks show the FBD programming for solving this task:

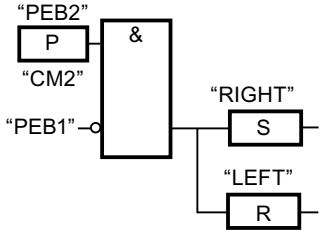
Network 1:

If the signal state changes from "0" to "1" (positive edge) at photoelectric barrier "PEB1" and, at the same time, the signal state at "PEB2" is "0", the object on the belt is moving to the left.



Network 2:

If the signal changes from "0" to "1" (positive edge) at photoelectric barrier "PEB2" and, at the same time, the signal state at "PEB1" is "0", the object on the belt is moving to the right.



See also

&: AND logic operation (Page 2471)

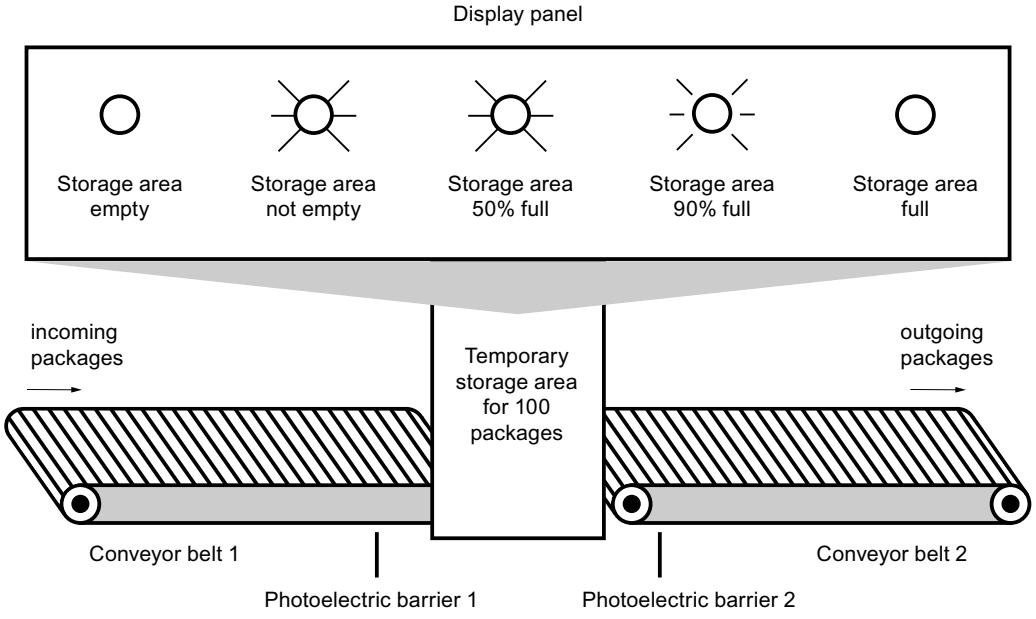
R: Reset output (Page 2479)

P: Scan operand for positive signal edge (Page 2485)

11.7.2.3 Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.



Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RESET	Input	BOOL	Reset counter
LOAD	Input	BOOL	Adjust the current counter value to the value of the PV parameter.
MAX STORAGE AREA FILL AMOUNT	Input	INT	Maximum possible number of packages in the storage area
PACKAGECOUNT	Output	INT	Number of packages in the storage area (current count value)

Name	Declaration	Data type	Description
STOCK_PACKAGES	Output	BOOL	Is set when the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag.
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50 % full
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90 % full
STOR_FULL	Output	BOOL	Display lamp: Storage area full
VOLUME_50	Input	INT	Comparison value: 50 packages
VOLUME_90	Input	INT	Comparison value: 90 packages
VOLUME_100	Input	INT	Comparison value: 100 packages

The following networks show the FBD programming for activating the lamps:

Network 1:

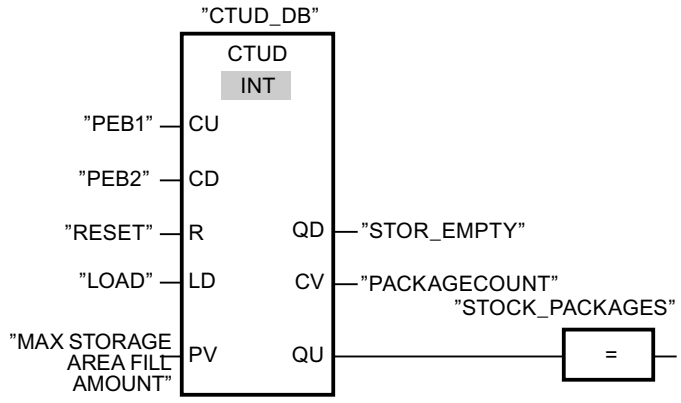
When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

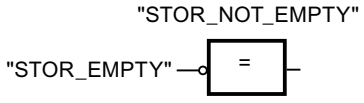
The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "MAX STORAGE AREA FILL AMOUNT" tag. As long as the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".



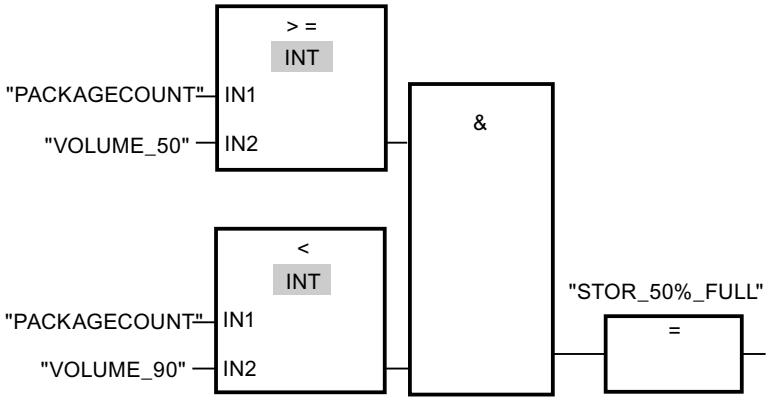
Network 2:

As long as there are packages in the storage area, the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.



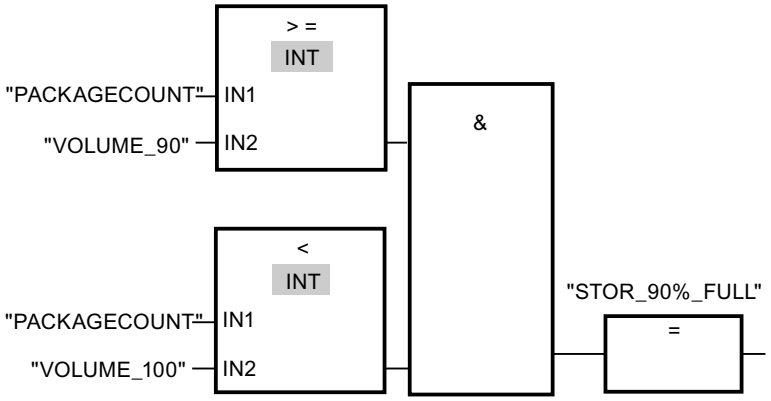
Network 3:

If the number of packages in the storage area is greater than or equal to 50, the "Storage area 50 % full" lamp switches on.



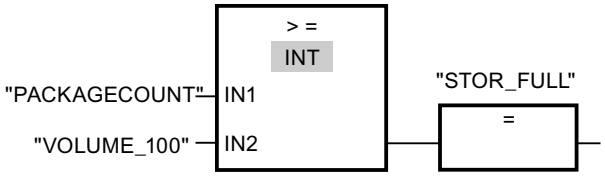
Network 4:

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.



Network 5:

If the number of packages in the storage area reaches 100, the lamp for the "Storage area full" message switches on.



See also

CTUD: Count up and down (Page 2264)

=: Assignment (Page 2477)

CTUD: Count up and down (Page 2541)

CMP >=: Greater or equal (Page 2558)

CMP <: Less than (Page 2562)

11.7.2.4 Example of calculating an equation

Calculating a complex equation

Do you want to program a complex calculation consisting of multiple arithmetic operations or logic operations within one instruction box? The "CALCULATE" instruction is available to you for this purpose.

Depending on the data type you choose, different arithmetic operations are available to you which can be combined with one another.

The following programming example shows how you can enter and calculate a complex equation:

RESULT = ((5 + 10) x 4) / 6

Implementation

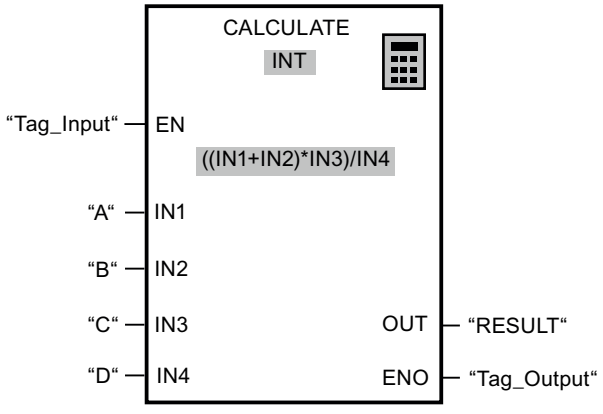
1. First, define the following tags in the block interface of your FBD block:

Name	Declaration	Data type	Comment	Values
A	Input	INT	First value for addition	5
B	Input	INT	Second value for addition	10
C	Input	INT	Multiplier	4
D	Input	INT	Divisor	6
RESULT	Output	INT	End result	10

2. You can select the data type INT for the instruction from the "<???>" drop-down list.
3. Interconnect the tags declared in the block interface with the inputs and outputs of the instruction box.
4. Click the "Calculator" icon on the top right corner of the instruction box to enter the equation to be calculated.
   The dialog "Edit "Calculate" instruction" opens.
5. Enter the following expression in the "OUT:=" field:
   ((IN1 + IN2) * IN4) / IN3
   The equation is shown in the instruction box.

Result

The following network shows the result in the FBD programming language:



If the "Tag_Input" has the signal state "1", the instruction is executed. The value of operand "A" is added to the value of operand "B". The subtotal is multiplied by "C" and then divided by the value of the "D" operand. The end result is saved in the "RESULT" operand.

See also

CALCULATE: Calculate (Page 2575)

11.7.2.5 Example of controlling room temperature

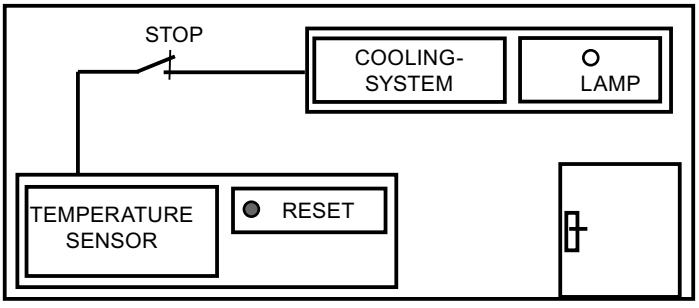
Controlling room temperature

In a cold room, the temperature must be maintained below zero degrees Celsius. A sensor is used to check for any temperature fluctuations. If the temperature rises above zero degrees Celsius, the cooling system switches on for a preset time. The "Cooling system on" lamp is lit during this time.

The cooling system and the lamp are turned off if one of the following conditions is met:

- The sensor reports a temperature fall below zero degrees Celsius.
- The preset cooling time has elapsed.
- The "Stop" pushbutton is pressed.

If the preset cooling time has expired and the temperature in the cold room is still too high, the cooling system can be restarted with the "RESET" pushbutton.



Implementation

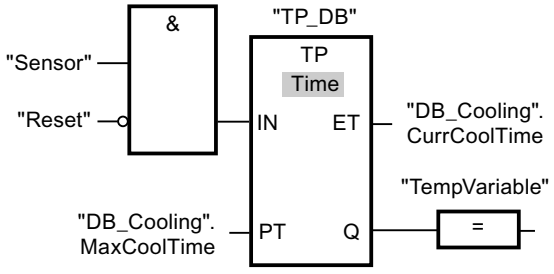
The following table shows the definition of the tags used:

Name	Declaration	Data type	Comment
Sensor	Input	BOOL	Temperature sensor signal
RESET	Input	BOOL	Restart
STOP	Input	BOOL	The cooling system is switched off.
MaxCoolTime	-	TIME	Preset cooling time This tag is defined in the "DB_Cool" data block.
CurrCoolTime	-	TIME	Currently elapsed cooling time This tag is defined in the "DB_Cool" data block.
Cooling system	Output	BOOL	The cooling system is switched on.

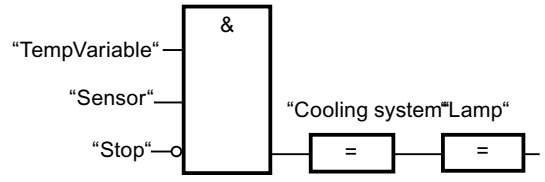
Name	Declaration	Data type	Comment
Lamp	Output	BOOL	The lamp for the "Cooling system on" message is switched on.
TempVariable	Temp	BOOL	Temporary tag This tag stores the signal state of the IEC timer TP.

The following network shows the FBD programming for controlling room temperature:

Network 1:



Network 2:



When the temperature in the cold room rises above zero degrees Celsius, the signal state at the "Sensor" operand switches from "0" to "1" (positive signal edge). In the case of a positive signal edge at the IN input of the timer function, the preset cooling time is started and the "TempVariable" receives the signal state "1". The signal state "1" of the "TempVariable" causes the cooling system as well as the display lamp to be turned on in network 2. The "Sensor", "Cooling system" and "Lamp" outputs must be programmed in network 2, because program only one coil can be programmed at the Q output of the timer function.

If the temperature in the cold room falls below zero degrees Celsius, the signal state of the sensor switches back to "0". This switches the cooling system and lamp off.

If the sensor does not signal a temperature drop, the cooling system and lamp are switched off after the preset cooling time has elapsed, at the latest. In this case, the cooling process can be restarted with the "RESET" pushbutton. Pressing and releasing the pushbutton generates a new positive signal edge at the IN input that restarts the cooling system.

The cooling system and the display lamp can be turned off with the "STOP" pushbutton at any time.

See also

TP: Generate pulse (Page 2494)

=: Assignment (Page 2477)

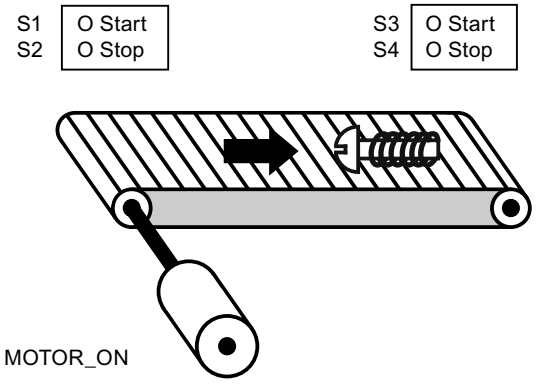
&: AND logic operation (Page 2471)

11.7.3 SCL programming examples

11.7.3.1 Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbuttons at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbuttons at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

The following table shows the definition of the tags used:

Operand	Declaration	Data type	Description
StartPushbutton_Left (S1)	Input	BOOL	Start pushbutton on the left side of the conveyor belt
StopPushbutton_Left (S2)	Input	BOOL	Stop pushbutton on the left side of the conveyor belt
StartPushbutton_Right (S3)	Input	BOOL	Start pushbutton on the right side of the convey-or belt
StopPushbutton_Right (S4)	Input	BOOL	Stop pushbutton on the right side of the convey-or belt

Operand	Declaration	Data type	Description
MOTOR_ON	Output	BOOL	Turn on the conveyor belt motor
MOTOR_OFF	Output	BOOL	Turn off the conveyor belt motor

The following SCL program shows how to implement this task:

SCL

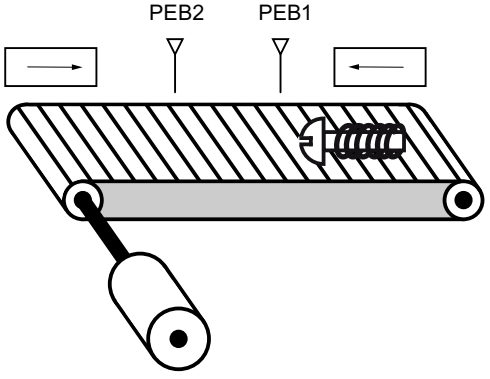
```
IF "StartPushbutton_Left_S1" OR "StartPushbutton_Right_S3" THEN
"MOTOR_ON" := 1;
"MOTOR_OFF" := 0;
END_IF;
IF "StopPushbutton_Left_S2" OR "StopPushbutton_Right_S4" THEN
"MOTOR_ON" := 0;
"MOTOR_OFF" := 1;
END_IF;
```

The conveyor belt motor is switched on when start pushbutton "StartPushbutton_Left_S1" or "StartPushbutton_Right_S3" is pressed. The conveyor belt motor is switched off when stop pushbutton "StopPushbutton_Left_S2" or "StopPushbutton_Right_S4" is pressed.

11.7.3.2 Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The detected running direction of the belt is indicated by a RIGHT arrow or a LEFT arrow. If additional conveyed material is approaching PEB1 from the right or PEB2 from the left, the displayed arrow is initially switched off until, after both photoelectric barriers are passed, the running direction can be detected again and the corresponding arrow displayed. For the solution of the task, 2 edge memory bits are needed that detect the signal change from "0" to "1" at the two photoelectric barriers.



Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
Photoelectric barrier PEB1	Input	BOOL	Photoelectric barrier 1
Photoelectric barrier PEB2	Input	BOOL	Photoelectric barrier 2
RIGHT	Output	BOOL	Display for movement to the right
LEFT	Output	BOOL	Display for movement to the left
Auxiliary flag PEB1	Input	BOOL	Edge bit memory 1
Auxiliary flag PEB2	Input	BOOL	Edge bit memory 2

The following SCL program shows how to implement this example:

SCL

```
// Program code for left running
IF "Photolelectric barrier PEB1" = 1 AND "Auxiliary flag PEB2" = 0 THEN
          "Auxiliary flag PEB1" := 1; // Set auxiliary flag for PEB1
          "LEFT" := 0; // Switch off arrow display left
          "RIGHT" := 0; // Switch off arrow display right
END_IF;

IF "Auxiliary flag PEB1" = 1 AND "Photoelectric barrier PEB2" = 1 THEN // The conveyor belt
is running to the left
          "LEFT" = 1;
          "RIGHT" := 0;
END_IF;

IF "LINKS" = 1 AND "Photoelectric barrier PEB2" = 0 THEN // Reset auxiliary flag for PEB1
          "Auxiliary flag PEB1" = 0
END_IF;
```

SCL

```
// Program code for right running
IF "Photolelectric barrier PEB2" = 1 AND "Auxiliary flag PEB1" = 0 THEN
          "Auxiliary flag PEB2" := 1; // Set auxiliary flag for PEB2
          "LEFT" := 0; // Switch off arrow display left
          "RIGHT" := 0; // Switch off arrow display right
END_IF;

IF "Auxiliary flag PEB2" = 1 AND "Photoelectric barrier PEB1" = 1 THEN // The conveyor belt
is running to the right
          "LEFT" = 0;
```

SCL

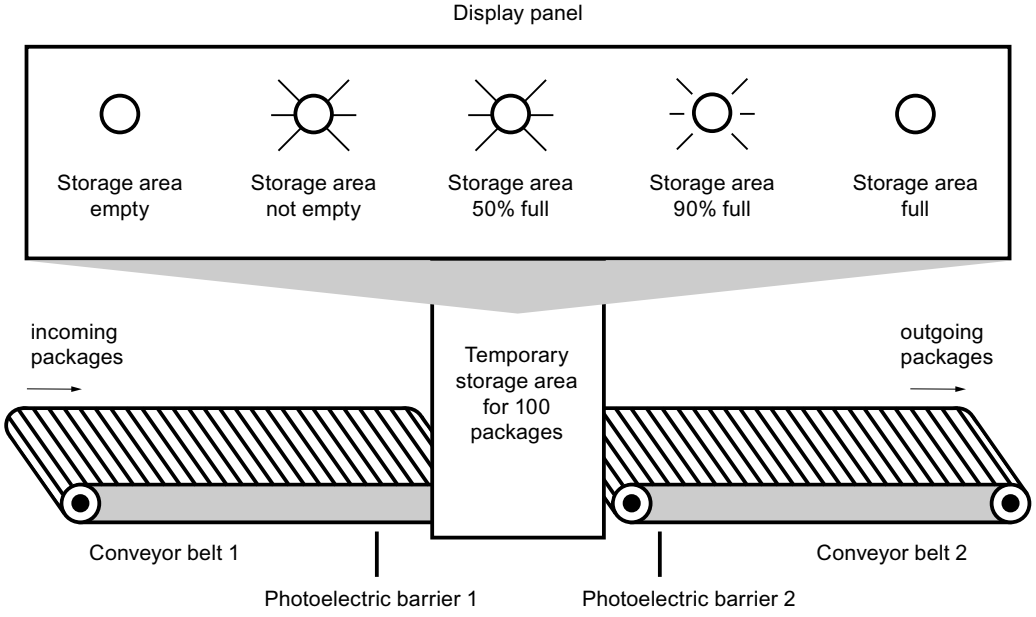
```
          "RIGHT" := 1;
END_IF;

IF "RIGHT" = 1 AND "Photoelectric barrier PEB1" = 0 THEN // Reset auxiliary flag for PEB2
          "Auxiliary flag PEB2" := 0;
END_IF;
```

If the photoelectric barrier "PEB1" has signal state "1" and the photoelectric barrier "PEB2" has signal state "0" at the same time, the object on the belt is moving to the left. If the photoelectric barrier "PEB2" has signal state "1" and the photoelectric barrier "PEB1" has signal state "0" at the same time, the object on the belt is moving to the right. The displays for a movement to the left or right will be turned off when the signal state at both photoelectric barriers is "0".

11.7.3.3 Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.



Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RESET	Input	BOOL	Reset counter
LOAD	Input	BOOL	Adjust the current counter value to the value of the PV parameter.
MAX STORAGE AREA FILL AMOUNT	Input	INT	Maximum possible number of packages in the storage area
PACKAGECOUNT	Output	INT	Number of packages in the storage area (current count value)
STOCK_PACKAGES	Output	BOOL	Is set when the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag.
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50 % full
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90 % full
STOR_FULL	Output	BOOL	Display lamp: Storage area full
VOLUME_50	Input	INT	Comparison value: 50 packages
VOLUME_90	Input	INT	Comparison value: 90 packages
VOLUME_100	Input	INT	Comparison value: 100 packages

The following SCL program shows how to implement this example:

When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "MAX STORAGE AREA FILL AMOUNT" tag. As long as the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".

SCL

```
"VOLUME_50" := 5; // Preassigning of the comparison value to 50 packages (for the test only
5 packages)
"VOLUME_90" := 9; // Preassigning of the comparison value to 90 packages (for the test only
9 packages)
"VOLUME_100" := 10; // Preassigning of the comparison value to 100 packages (for the test
only 10 packages)
"MAX STORAGE AREA FILL AMOUNT" := 10; // Preassigning of the maximum amount in storage area
to 100 packages (for the test only 10 packages)


"IEC_Counter_0_DB".CTUD(CU := "PEB1",
                        CD := "PEB2",
                        R := "RESET",
                        LD := "LOAD",
                        PV := "MAX STORAGE AREA FILL AMOUNT",
                        QU => "STOCK_PACKAGES",
                        QD => "STOR_EMPTY",
                        CV => "PACKAGECOUNT");
```

As long as the storage area contains packages, the "Storage area not empty" lamp is switched on.

SCL

```
"STOR_NOT_EMPTY" := NOT "STOR_EMPTY"
```

If the number of packages in the storage area is lower than 50%, the lamps for the alarms "Storage area 50% full", "Storage area 90% full" and "Storage area full" switch off.

SCL

```
IF "PACKAGECOUNT" < "VOLUME_50" THEN
"STOR_50%_FULL" := 0;
"STOR_90%_FULL" := 0;
"STOR_FULL" := 0;
END_IF;
```

If the number of packages in the storage area is greater than or equal to 50 %, the "Storage area 50 % full" lamp switches on.

```
IF "PACKAGECOUNT" >= "VOLUME_50" AND "PACKAGECOUNT <= "VOLUME_90" THEN
"STOR_50%_FULL" := 1;
"STOR_90%_FULL" := 0;
"STOR_FULL" := 0;
END_IF;
```

If the number of packages in the storage area is greater than or equal to 90 %, the "Storage area 90 % full" lamp switches on. The display lamp for 50 % full also remains on.

SCL

```
IF "PACKAGECOUNT" >= "VOLUME_90" AND "PACKAGECOUNT < "VOLUME_100" THEN
"STOR_50%_FULL" := 1;
"STOR_90%_FULL" := 1;
"STOR_FULL" := 0;
END_IF;
```

If the number of packages in the storage area reaches 100 %, the lamp for the "Storage area full" message switches on. The display lamps for 50 % and 90 % full also remain on.

SCL

```
IF "PACKAGECOUNT" >= "VOLUME_100" THEN
"STOR_50%_FULL" := 1;
"STOR_90%_FULL" := 1;
"STOR_FULL" := 1;
END_IF;
```

Visualize processes

12

12.1 Creating screens

12.1.1 Basics

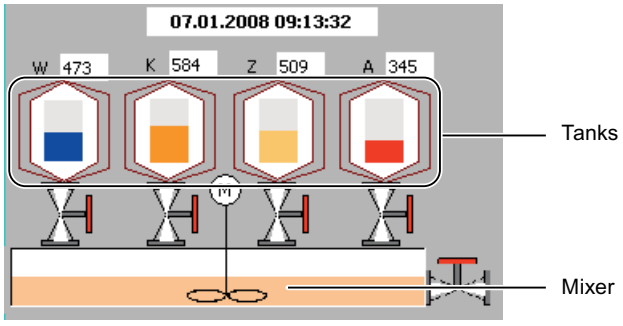
12.1.1.1 Screen basics

Introduction

In WinCC you create screens that an operator can use to control and monitor machines and plants. When you create your screens, the object templates included support you in visualizing processes, creating images of your plant, and defining process values.

Application example

The figure shows a screen that was created in WinCC. With the help of this screen, the operator can operate and monitor the mixing station of a fruit juice manufacturing system. Fruit juice base is supplied from various tanks to a mixing unit. The screen indicates the fill level of the tanks.



Screen design

Insert an object you need to represent a process into your screen. Configure the object to correspond to the requirements of your process.

A screen may consist of static and dynamic elements.

- Static elements such as text or graphic objects do not change their status in runtime. The tank labels (W, K, Z, A) shown in this example of a mixing unit are static elements.
- Dynamic elements change their status based on the process. Visual current process values as follows:
  - From the memory of the PLC
  - From the memory of the HMI device in the form of alphanumeric displays, trends and bars.

  Input fields on the HMI device are also considered dynamic objects. The fill level values of the tanks in our example of a mixing plant are dynamic objects.

Process values and operator inputs are exchanged between the controller and the HMI device via tags.

Screen properties

The screen layout is determined by the features of the HMI device you are configuring. It corresponds with the layout of the user interface of this device. If the set HMI device has function keys, the screen shows these function keys. Other properties such as the screen resolution, fonts and colors are also determined by the characteristics of the selected HMI.

Function keys

A function key is a key on the HMI device to which you can assign one or several functions in WinCC. The functions are triggered as soon as the operator presses the key on the HMI device.

A function key can be assigned global or local functions:

Global function keys always trigger the same action, regardless of the currently displayed screen.

Local function keys trigger different actions depending on the currently displayed screen on the HMI device. This assignment applies only to the screen in which you have defined the function key.

Navigation

You configure a screen navigation to enable the operator to call a screen on the HMI device in Runtime.

- The "Screen" editor is used to configure buttons for calling other screens.
- You use the "Global Screen" editor to configure globally assigned function keys.

See also

Steps (Page 3895)

Basics on working with templates (Page 3900)

Task cards (Page 3892)

Device-dependent functional scope of screens (Page 3891)
Overview of objects (Page 3935)
Basics on dynamizing screens (Page 4012)
Basics on working with layers (Page 4051)
Basics on libraries (Page 6771)
Basics on faceplates (Page 4057)
Bar (Page 4097)
Basics on screen navigation (Page 4194)
Basics on text lists (Page 3990)
Basic principles of graphics lists (Page 4001)

12.1.1.2 Device-dependent functional scope of screens

Introduction

The functions of an HMI device determine the display of the device in WinCC and the scope of functions of the editors.

The following screen properties are determined by the functions of the selected HMI device:

- Device layout
- Screen resolution
- Number of colors
- Fonts
- Objects available

Device layout

The device layout of a screen forms the image of the HMI device in your configuration. The device layout of the screen shows all the function keys available on the HMI device, for example.

Screen resolution

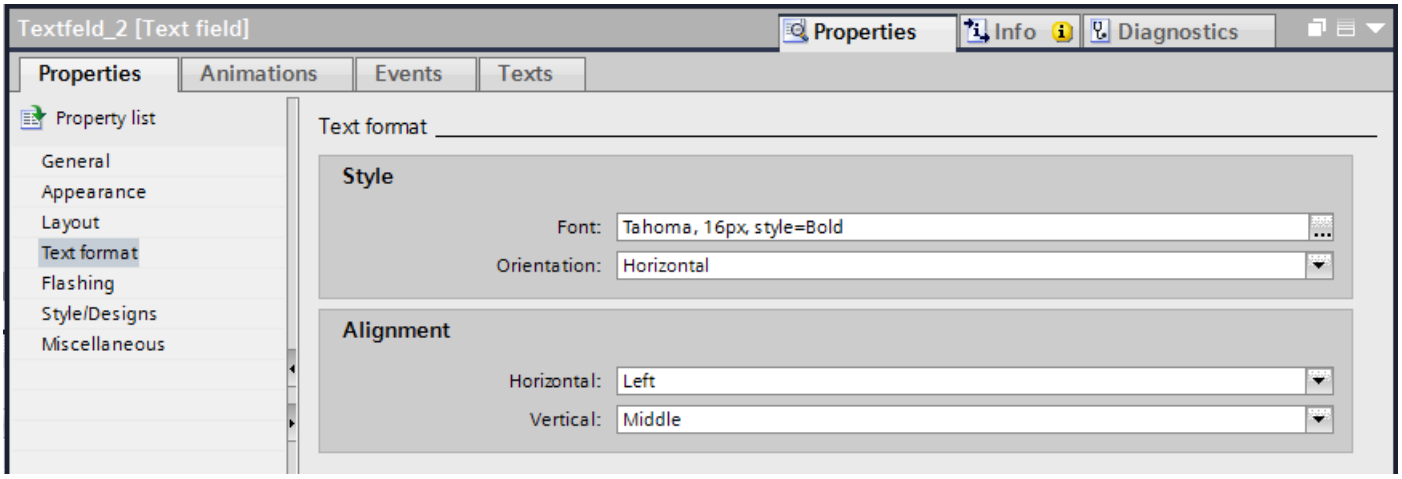
The screen resolution is determined by the different display dimensions of the various operator panels. You can only change the screen resolution if you configure the "WinCC Runtime Advanced" or "WinCC Runtime Professional" HMI device.

Number of colors

You can assign colors to the screen objects. The number of possible colors is determined by the color depth and specific colors supported on the selected HMI device.

Fonts

You can customize the appearance of the texts in all the screen objects that contain static or dynamic text. How to highlight individual texts in a screen. Select the font, font style and size, and set additional effects such as underscoring, for example.



The settings for the text markups such as font style and effects always refer to the entire text of a screen object. That is, you can display the complete title in bold format, but not its individual characters or words, for example.

Objects available

Some of the screen objects can not be configured globally for all HMI devices. These screen objects are not displayed in the "Tools" task card. For a KTP 1000 touch panel unit you can not configure a slider, for example.

See also

Screen basics (Page 3889)

12.1.1.3 Elements and basic settings

Task cards

Introduction

The following task cards are available in the "Screens" editor:

- Tools: Display and operating objects
- Animations: Templates for dynamic configuration

- Layout: Aid for customizing the display
- Libraries: Administration of the project library and of the global libraries

Note
WinCC Basic

The "Animations" task card is not available in WinCC Basic.

Tools

The "Tools" task card contains objects in different panes:

- Basic objects
- Elements
- Controls
- User controls (optional)
- Graphics

You paste objects from the palettes into your screens by drag&drop or a double click. The objects available for selection are determined by the features of the HMI device you are configuring. The following icons are used to change the display mode:

Icon	Meaning
	Displays the objects as a list.
	Displays the objects as a graphic.

Animations

The "Animations" task card contains the possible dynamizations of a screen object in the palettes. You paste the animations to a screen object by drag&drop or a double click from the "Movements", "Display" and "Tag Binding" palettes.

Layout

The "Layout" task card contains the following panes for displaying objects and elements:

- Layers: Serves to manage screen object layers The layers are displayed in a tree view and contain information about the active layer and the visibility of all layers.
- Grid: You specify whether you want to align the objects to a grid or to other objects and set the grid size for a grid.
- Objects out of range: Objects that lie outside the visible area are displayed with name, position and type.

Libraries

The "Libraries" task card show the following libraries in separate panes:

- Project library: The project library is stored together with the project.
- Global library: The global library is stored in a separate file in the specified path on your configuration PC.

See also

Moving the view (Page 3894)

Zooming the view (Page 3895)

Screen basics (Page 3889)

Moving the view

Introduction

To display only a section of the entire screen in the work area, use the icon of the "Screens" editor:

Requirement

- A screen is open.
- The view shows only a screen section.

Procedure

To move the view:

1. Click the icon at the bottom right corner of the work area and press the left mouse button. A miniature view of the full screen is shown. An orange frame shows the currently selected area.
2. Hold down the mouse button and drag the frame to the desired area.

Note

The screen is scrolled when you drag a screen object from the visible to a currently hidden section.

See also

Task cards (Page 3892)

Zooming the view

Introduction

To view a small screen section in closer detail, use the zoom tool to magnify the screen in the working area. The maximum zoom amounts to 800%.

You can zoom with the toolbar in the work area or with the "Layout > Zoom" task card.

There are different ways to increase the screen, for example, with the zoom factor or by adapting the work area to the height of the screen.

Requirement

The screen is opened.

Procedure

Proceed as follows to zoom a view with the selection frame:

1. Click the toolbar button.
2. Use the mouse to draw a selection frame in the screen.

After you have released the mouse button, the section enclosed by the selection frame is zoomed to fit the complete work area.

Alternatively, use the slider in the lower right-hand corner of the screen.

Result

The selected screen section is magnified.

See also

Task cards (Page 3892)

12.1.1.4 Working with screens

Steps

Steps

To create screens, you need to take the following initial steps:

- Plan the structure of the process visualization: Number of screens and their layout
  Example: Subprocesses are visualized in separate screens, and merged in a master screen.
- Define your screen navigation control strategies.

- Adapt the templates and the global screen.
  You define objects centrally and assign function keys for example.
- Create the screens. Use the following options of efficient screen creation:
  - Working with libraries
  - Working with layers
  - Working with faceplates

See also

Creating a new screen (Page 3896)

Managing screens (Page 3897)

Defining the start screen of the project (Page 3898)

Screen basics (Page 3889)

Creating a new screen

Introduction

Create screens to display processes in your system.

Requirement

- The project has been created.
- The Inspector window is open.

Procedure

1. Double-click "Screens > Add New Screen" in the project navigation.
   The screen is generated in the project and appears in your view. The screen properties are shown in the Inspector window.
2. Enter a meaningful name for the screen.
3. Configure the screen properties in the Inspector window:
   - Specify whether and on which template the screen is based.
   - Set the "Background Color" and the "Screen Number."
   - Specify a documenting text under "Tooltip".
   - Specify the layers to be displayed under "Layers" in the engineering system.
   - Select dynamic screen update under "Animations."
   - Select "Events" to define which functions you want to execute in Runtime when you call and exit the screen or at other events.

     Note

     Not all HMI devices support the "Visibility" animation.

Result

You created the screen in your project. You can now paste objects and control elements from the "Tools" task card and assign function keys in further work steps.

See also

Steps (Page 3895)

Managing screens

Introduction

You can move screens within a project to other groups, or copy, rename, and delete them.

Moving screens in a group

1. Select the "Screens" folder in the project tree.
2. Select the "Add group" command from the shortcut menu.
   A folder called "Group_x" is inserted.
3. Select the screen in the project tree.
4. Drag-and-drop the screen to the required group.
   The screen is moved into this group.

Copy screen

1. Select the screen in the project tree.
2. Select the "Copy" command in the shortcut menu to copy the screen to the clipboard.
3. In the project tree, select the screen insert position.
4. Select "Paste" from the shortcut menu to insert the screen.
   A copy of the screen is inserted. A consecutive number is appended to the name of the original in the copy.

Alternatively, press <Ctrl> while you drag the screen to the required position.

Note

If you copy a screen with interconnected template for several devices and projects, the template will also be copied. Any existing matching template is not used. This holds particularly true when you copy the screens with drag-and-drop.

Rename screen

1. Select the screen in the project tree.
2. Select "Rename" from the shortcut menu.
3. Type in a new name.
4. Press <Enter>.

As an option, use the <F2> function key to rename the screen.

Delete screen

1. Select the screen in the project tree.
2. Select "Delete" from the shortcut menu.
   The screen and all its objects are deleted from the current project.

See also

Steps (Page 3895)

Defining the start screen of the project

Introduction

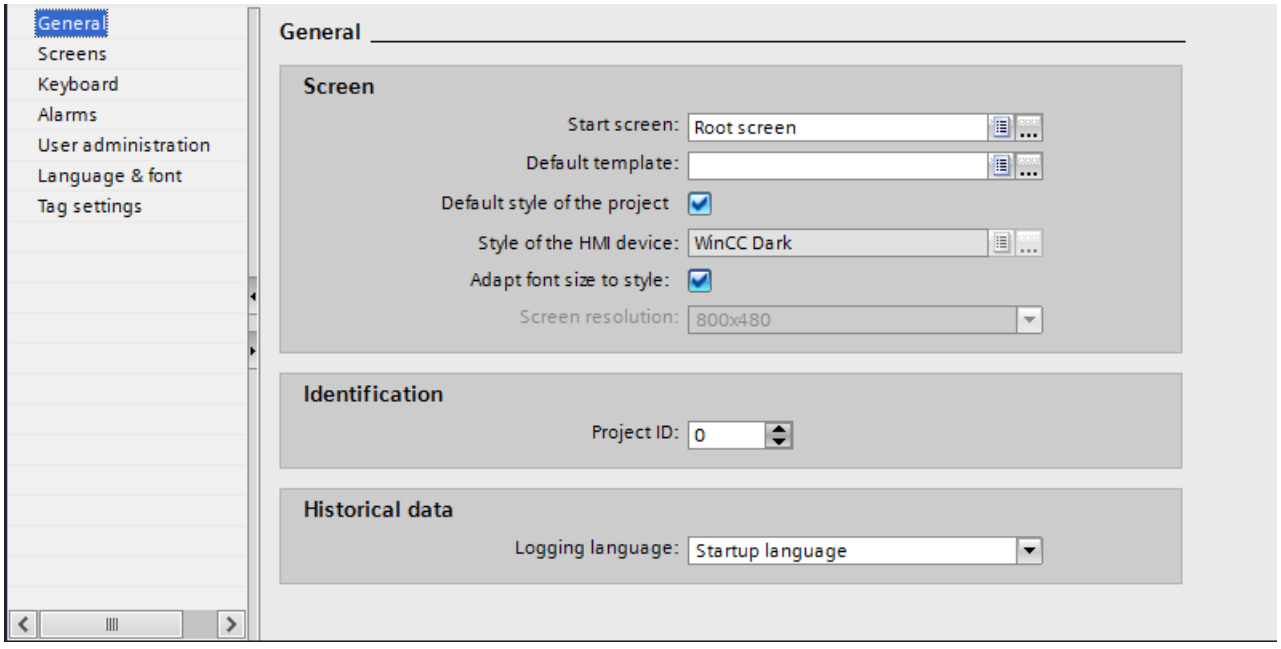
The start screen is the initial screen opened at the start of project in Runtime. You can define a different start screen for each one of the HMI devices. Beginning at this start screen, the operator calls the other screens.

Requirement

The project contains the screen you want to use as the start screen.

Procedure

1. Double-click "Runtime settings > General" in the project tree.



2. Select the desired screen as "Start screen".

Alternatively select a screen in the project tree and select "Use as start screen" in the shortcut menu.

Result

The start screen opens on the HMI at the start of Runtime.

See also

Steps (Page 3895)

Changing the screen resolution

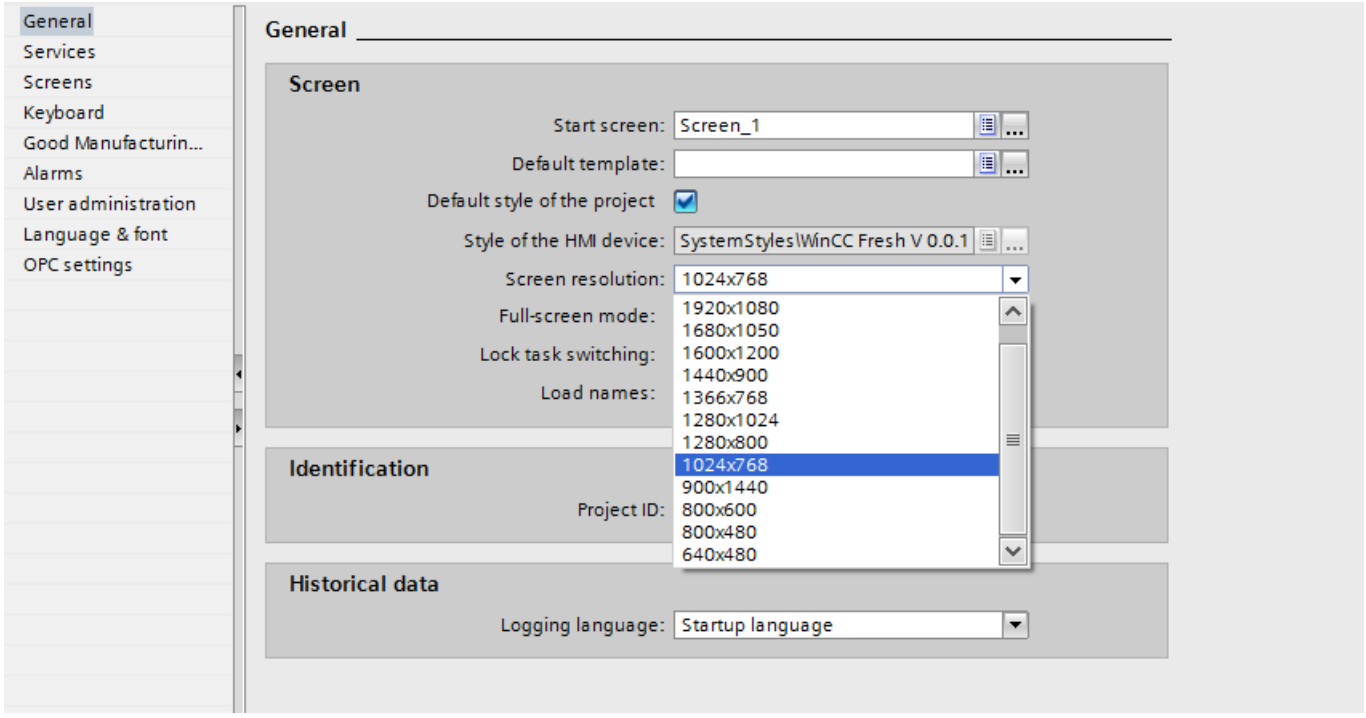
You change the screen resolution in the Runtime settings of the HMI device.

The screen resolution can be configured for the following HMI devices:

- RT Advanced

Configuring a fixed screen resolution

1. Open the Runtime settings of the HMI device.
2. Select the screen resolution under "General > Screen > Screen resolution".



Configuring a user-defined screen resolution

1. Open the Runtime settings of the HMI device.
2. Select the "User-defined" function under "General > Screen > Screen resolution".
   A dialog appears to input the width and height of the user-defined screen resolution.
3. Enter the width and height of the screen resolution.

12.1.1.5 Working with templates

Basics on working with templates

Introduction

Configure the objects in a template which are to be displayed in all screens based on this template.

Note the following rules:

- A screen must not be based on a template.
- A screen is only based on one template.
- You can create several templates for one device.
- A template cannot be based on another template.

Objects for a template

You determine functions and objects in the template which are to apply for all screens based on this template:

- Assignment of function keys: You also assign the function keys in the template for HMI devices with function keys. This assignment overwrites a possible global assignment.
- Permanent window: Some devices support a permanent window for all screens in the top area of the screen. In contrast to the template, the permanent window occupies an area of the screen for itself alone.
- Operator controls: You can paste all screen objects which you also use for a screen into a template.

Application examples

- You want to assign the ActivateScreen" function to a function key in the template. The operator uses this key to switch to another screen in runtime. This configuration applies to all screens that are based on this template.
- A graphic with your company logo can be added to the template. The logo appears on all screens that are based on this template.

Note

If an object from the template has the same position as an object in the screen, the template object is covered.

See also

Managing templates (Page 3903)

Creating a new template (Page 3904)

Global screen (Page 3902)

Screen basics (Page 3889)

Using a template in the screen (Page 3906)

Configuring a permanent window in the template (Page 3905)

Global screen

Introduction

You define global elements for all screens of an HMI device independently of the used template.

Function keys

For HMI devices with function keys you assign the function keys globally in the "Global Screen" editor. This global assignment applies for all screens of the HMI device.

Proceed as follows to assign function keys locally in screens or templates:

1. Click the function key in your screens or templates.
2. Deactivate "Properties > Properties > General > Use Global Assignment" in the inspection window.

Indicator and control objects for alarms

The "Alarm window" and "Alarm indicator" objects that are available as global objects are configured within the "Global screen" editor.

The "Alarm window" and "Alarm indicator" are always shown in the foreground.

For Comfort Panels you also have the possibility of configuring a "System diagnostic view" in the global screen.

Note

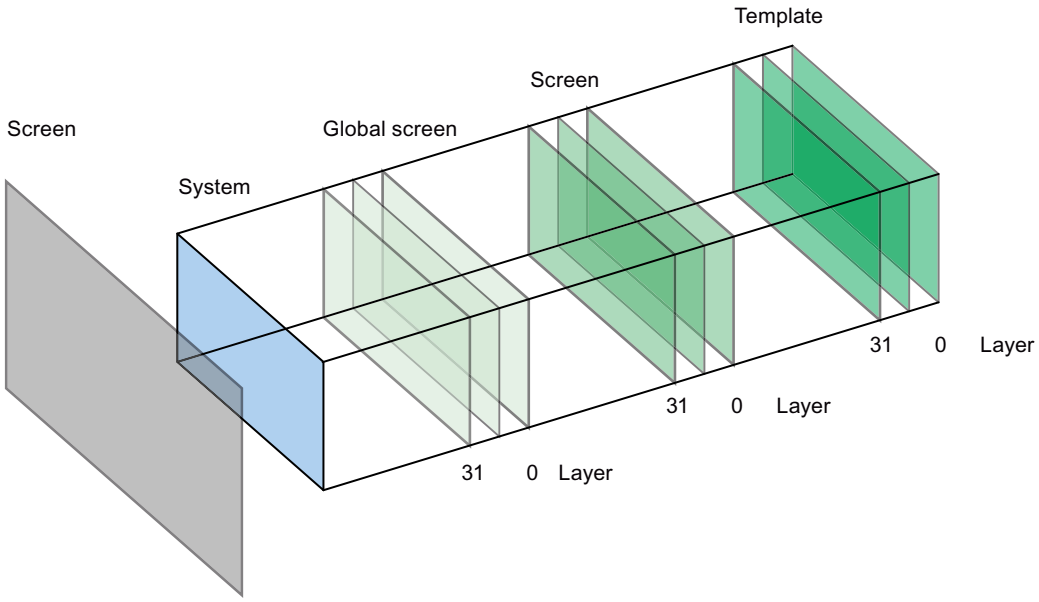
If you have configured a permanent window in a template, do not position the alarm window and alarm indicator in the area of the permanent window. Otherwise, the alarm window and the alarm indicator will not be displayed in Runtime.

The permanent window is not visible in the "Global screen" editor.

Order of configuration of screens

The following order applies for the configuration:

- The global screen comes before screens and templates
- Screens come before templates



The system layer is not configurable. This contains

- input dialog boxes
- alarms from the operating system
- the direct keys for touch panels

See also

Basics on working with templates (Page 3900)

Managing templates

Introduction

You can move, copy, rename, and delete templates within a project in the Project window.

Moving a template into a group

1. Select the templates in the project navigation "Screen management > Templates".
2. Select "Add group" in the shortcut menu.
   A folder called "Group_x" is inserted.

3. Select the template in the project navigation.
4. Drag-and-drop the template to the required group.
   The template is moved to this group.

Copying templates

1. Select the template in the project navigation.
2. Select "Copy" in the shortcut menu.
3. Select the position in the project navigation where you want to paste the template.
4. Select "Paste" from the shortcut menu to insert the template.
   A unique name is assigned automatically to the copy.

Alternatively, you can hold down the <Ctrl> key, and drag the template into position.

Deleting a template

1. In the project navigation, select the template to be deleted.
2. Select "Delete" in the shortcut menu.
   The template, and all its objects are deleted from the current project.

Assigning a template to a screen

1. In the project navigation, select the screen to which you want to assign the template.
2. In the Inspector window, select "Properties > Properties > General".
3. Select the desired template under "Template."
   The selected template and all the objects contained in it are assigned to the screen.

See also

Basics on working with templates (Page 3900)

Creating a new template

Introduction

In a template, you can centrally modify objects and function keys. Changes to an object or of a function key assignment in the template are applied to the object in all the screens which are based on this template.

Note

HMI device dependency

Function keys are not available on all HMI devices.

Requirement

- The project has been created.
- The Inspector window is open.

Procedure

1. Select "Screen management > Templates" in the project tree and then double-click "Add new template".
   The template is created in the project, and appears in your view.
   The properties of the template are displayed in the Inspector window.
2. Define the name of the template under "Properties > Properties > General" in the Inspector window.
3. Specify the layers in the engineering system that are displayed under "Properties >Properties >Layers" in the Inspector window.
4. Add the necessary objects from the "Tools" task card.
5. Configure the function keys.

Result

The template is created in your project.

See also

Basics on working with templates (Page 3900)

Configuring a permanent window in the template

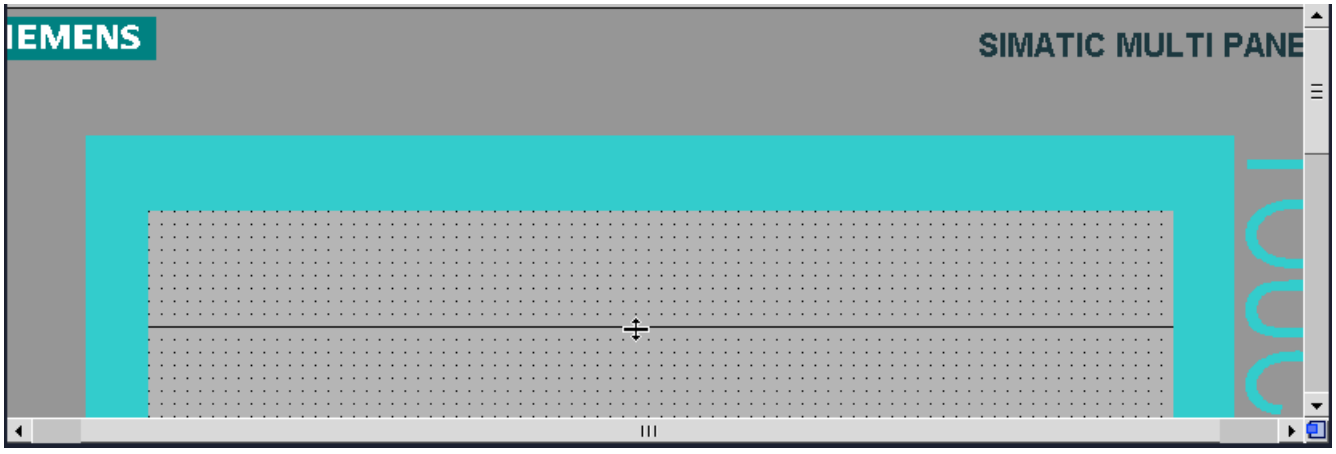
Introduction

In the permanent window you configure objects which are visible in all screens. In contrast to the template, the permanent window occupies an area of the screen for itself alone.

Configuring permanent windows

Proceed as follows to configure a permanent window:

1. Open a template for a HMI device with a permanent window.
2. Use the mouse (cursor shape:) to drag the top edge of the editable screen area down.



3. Configure the desired objects in the area of the permanent window.

The default height of the permanent window is "0". The maximum height is the maximum height of the screen minus 10 pixels.

Result

The content of the permanent window appears on every screen, and in the template. The area above this line is now used as a permanent window in all the screens of the HMI device. Already configured objects in the screens are moved down by the height of the permanent window.

See also

Basics on working with templates (Page 3900)

Using a template in the screen

Introduction

You use a template in the screen. All the objects configured in the template are also available in the screen.

Requirement

A template has been created.

A screen has been created.

Procedure

Proceed as follows to use a template in a screen:

1. Double click a screen in the project tree. The screen opens in the work area.
2. Open "Properties > Properties > General" in the inspector window.
3. Select a template that is to be applied to the screen under "Template".

Show template in screen

When you edit a screen, you can show an existing template in the screen.

Proceed as follows to show a template in the screen:

1. Activate "Extras > Settings > Visualization > Show template in screens" in the menu.

Result

The screen is based on the selected template. All objects which you have configured in the template are available in the screen. The template is displayed in the screen.

See also

Basics on working with templates (Page 3900)

12.1.1.6 Working with pop-up screens

Basics on working with pop-up screens

Introduction

You use the pop-up screen to configure a screen for additional content, for example, object settings. The pop-up screen is displayed on top of the current screen as soon as you call a system function.

Only one pop-up screen at a time is displayed in a screen.

Note

Device dependency of the "Pop-up screen" object

The "Pop-up screen" object is only available for KTP Mobile Panels, Comfort Panels and RT Advanced.

Application

You configure a button in the screen and assign it the function "ShowPopupScreen". When operating the button, a pop-up screen is opened or closed. In the pop-up screen, you make the settings.

Objects in the pop-up screen

You configure the following objects in the pop-up screen:

- Basic objects, such as lines and polygon-based objects
- Elements, such as buttons
- Controls, such as trend view or alarm view
- Faceplates

Note

You cannot configure alarm windows, system diagnostics windows and alarm indicators in the pop-up screen.

Softkeys and hotkeys are not supported in the "Pop-up screen" object.

Note

Objects in the pop-up screen as well as the pop-up screen itself do not support access with VB scripting.

Creating a new pop-up screen

Introduction

A new pop-up screen is created in Project tree > Screen management. The size of a pop-up screen is defined in the properties. The maximum size of a pop-up screen is limited by the screen size.

Requirements

- The project has been created.
- The Inspector window is open.

Creating a new pop-up screen

1. Double-click "Screen management > Pop-up screens > Add new pop-up screen" in the project tree.
   The pop-up screen is generated in the project and is displayed in the work area.
   The pop-up screen properties are displayed in the Inspector window.
2. Define the name of the pop-up screen under "Properties > Properties > General" in the Inspector window.
3. Specify the layers in the engineering system that are displayed under "Properties >Properties >Layers" in the Inspector window.

4. Define the height and width of the pop-up screen under "Properties > Properties > Layout" in the Inspector window.
5. Add the necessary objects from the "Tools" task card.
6. Configure the function keys.

Note

You cannot address the "Pop-up screen" object and the objects configured in it with VB scripts.

Note

Objects and controls can be dynamized in the pop-up screen. The pop-up screen itself does not support dynamization.

Result

You have created a new pop-up screen in which you have configured objects and functions.

Managing pop-up screens

Introduction

You can move, copy, rename, and delete pop-up screens within a project in the Project window.

Moving pop-up screens in a group

1. Select "Screen management > Pop-up screens" in the project tree.
2. Select "Add group" in the shortcut menu.
   A folder called "Group_x" is inserted.
3. Select the pop-up screen in the project tree.
4. Drag-and-drop the pop-up screen to the required group.
   The pop-up screen is moved into this group.

Copying pop-up screen

1. Select the pop-up screen in the project tree.
2. Select "Copy" from the shortcut menu.
3. In the project tree, select the pop-up screen insert position.
4. Select "Paste" in the shortcut menu.
   A unique name is assigned automatically to the copy.

Deleting pop-up screen

1. Select the pop-up screen you want to delete in the project tree.
2. Select "Delete" in the shortcut menu.
   The pop-up screen and all its objects are deleted from the current project.

Using the pop-up screen

Introduction

You configure the system function to open a pop-up screen to the event of a screen object. The linked system function is executed when the event occurs in runtime.

Requirements

- A pop-up screen has been configured.
- A screen is open.
- The Inspector window is open.

Procedure

1. Configure a button.
2. Click "Properties > Events" in the Inspector window.
3. Select an event, for example, "Click".
4. Click on "Add function" in the table.
5. Select the "ShowPopupScreen" system function.

Result

The pop-up screen opens when the operator clicks the button in runtime.

12.1.1.7 Working with slide-in screens

Basics on working with slide-in screens

Introduction

The "Slide-in screen" object provides quick navigation between the start screen and the slide-in screens containing additional configured contents stored outside the start screen. Navigation is supported by the multi-touch functions. The configurable handles appearing at the edges of the start screen provide quick access to the stored slide-in screens.

The size of the slide-in screens depends on the HMI device used.

Note

Device dependency of the "Slide-in screen" object

The "Slide-in screen" object is only available for KTP Mobile Panels, Comfort Panels and RT Advanced.

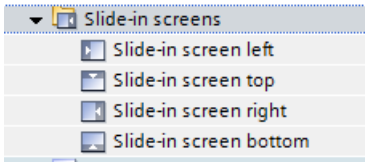
Application

The "Slide-in screen" is used to configure up to four slide-in screens for additional contents and operator controls, for example, a virtual keyboard or a system dialog.

By default, the following slide-in screens are set up for each device:

- "Slide-in screen top": appears at the top of the start screen in Runtime
- "Slide-in screen bottom": appears at the bottom of the start screen in Runtime
- "Slide-in screen left": appears at the left of the start screen in Runtime
- "Slide-in screen right": appears at the right of the start screen in Runtime

It is possible to configure objects in each of these slide-in screens.



Note

It is not possible to delete the default slide-in screens or to set up your own slide-in screens.

Objects in the slide-in screen

You configure the following objects in the slide-in screen:

- Basic objects, such as lines and polygon-based objects
- Elements, such as buttons
- Controls, such as trend view or alarm view
- Faceplates

Note

Objects in the slide-in screen as well as the slide-in screen itself do not support access with VB scripting.

Note

Release buttons and locked operator controls cannot be configured in the slide-in screen.

Softkeys and hotkeys are not supported in the slide-in screen.

See also

Configuring slide-in screens (Page 3912)

Configuring slide-in screens for devices without multi-touch function (Page 3914)

Using slide-in screens (Page 3915)

Configuring slide-in screens

Introduction

By default, four slide-in screens are preset for each device in the project tree under "Screen management > Slide-in screens". For each of these slide-in screens, you configure the size, the layout and the handle for operation in Runtime using the properties.

Requirement

- The project has been created.
- An HMI device has been created.
- The Inspector window is open.

Configuring properties for slide-in screens

1. Select the slide-in screen you wish to configure in the project tree under "Screen management > Slide-in screens.
   The slide-in screen properties are displayed in the Inspector window.

   Note

   The names of all slide-in screens are write-protected and cannot be changed.

2. Define the background color of the slide-in screen in the "Background color" selection list of the "Properties > Properties > General" Inspector window.
3. Define the grid color of the slide-in screen in the "Grid color" selection list.
4. Enable the "Activate" check box.

5. Specify the layers in the engineering system that are displayed under "Properties >Properties >Layers" in the Inspector window.
6. Define the height and width of the slide-in screen in "Properties > Properties > Layout" in the Inspector window:
   - Define the height only for "Slide-in screen top" and "Slide-in screen bottom".
   - Define the width only for "Slide-in screen left" and "Slide-in screen right".

Note

If you do not need all four slide-in screens, enable the slide-in screens relevant to you only.

If you do not need any slide-in screens, leave the "Activate" check box empty for all four slide-in screens.

Note

Slide-in screens cannot be deleted nor copied to a different HMI device.

However, it is possible to copy the objects configured in a slide-in screen to another slide-in screen.

Configuring handles for slide-in screens

1. Define the color of the first line of the handle in the "Lines" area of the "Color" selection list under "Properties > Properties > Handle" in the Inspector window.
2. Define the color of the second line of the handle in the "Alternative color" selection list.
3. Define the color for the operable area of the handle in the "Color" selection list under "Operable area".
4. Select one of the following options from the "Visibility" area:
   - "Hide handle automatically" (activated by default)
   - "Always display handle"
   - "Never display handle"

Note

When selecting the "Hide handle automatically" option, the handle only becomes visible once the user clicks the operable area.

The "Never display handle" option is selected when the slide-in screen is configured using the "ShowSlideInScreen" system function.

Note

You are notified by a warning if a handle which is configured as visible covers a configured object in a screen, such as a button, completely or partially.

Note

The following dimensions are defined for the handle:

- The width of a handle for the "Slide-in screen top" and "Slide-in screen bottom" objects is one-third of the screen length.
- The height of a handle for the "Slide-in screen left" and "Slide-in screen right" objects is one-third of the screen width.

Result

You have configured the properties of the slide-in screens and the handle for operation in Runtime.

See also

Basics on working with slide-in screens (Page 3910)

Configuring slide-in screens for devices without multi-touch function (Page 3914)

Configuring slide-in screens for devices without multi-touch function

Introduction

You use the "ShowSlideInScreen" system function to make the slide-in screens available on devices without multi-touch functions, for example Key Panels. You link this system function to buttons. Buttons are used to navigate between the start screen and the slide-in screens.

Requirements

- The project has been created.
- An HMI device has been created.
- A screen is open.
- The Inspector window is open.

Procedure

1. Drag the "Button" object to the screen from the toolbox.

   Note

   Navigation in Runtime is much easier if the buttons for operating the slide-in screens are created directly in the start screen.

2. Enter a suitable text of any length for the button.
3. In the Inspector window, select "Properties > Events > Click".
   The "Function list" dialog box is opened.

4. Select the "ShowSlideInScreen" system function from the "Keyboard operation for screen objects" group.
5. Assign the ID of the slide-in screen appearing in Runtime when the button is operated to the "ShowSlideInScreen" function:
   - "Top"
   - "Bottom"
   - "Left"
   - "Right"
6. Assign the appropriate mode to the "ShowSlideInScreen" function:
   - "Switching"
   - "Off"
   - "On"

Result

You have configured the slide-in screens for operation on devices without multi-touch function. When clicking on the respective button in the screen in Runtime, the slide-in screen behaves in accordance with the configuration.

See also

Basics on working with slide-in screens (Page 3910)

Configuring slide-in screens (Page 3912)

Using slide-in screens

Introduction

In slide-in screens, you configure additional contents and operator controls, for example, navigation buttons, an alarm view or a trend view.

Requirements

- The project has been created.
- An HMI device has been created.
- A slide-in screen is open.
- The Inspector window is open.

Procedure

1. Add the necessary objects from the "Tools" task card to the slide-in screen.
2. Configure the function keys.

Note

You cannot address the "Slide-in screen" object and the objects configured in it with VB scripts.

Note

Objects and controls can be dynamized in the slide-in screens. The slide-in screens as such do not support dynamization.

Result

You have configured objects and functions in the slide-in screen.

See also

Basics on working with slide-in screens (Page 3910)

12.1.1.8 Working with styles

Basics on working with styles

Introduction

The style editor is a global editor. You define a uniform appearance for display and operating elements in the style editor. In this way, you harmonize the display of objects in Runtime. The style editor offers predefined styles. Additional predefined styles are available for download at Siemens Industry Online Support (http://support.automation.siemens.com/WW/view/en/91174767).

You can choose one of the predefined styles or create your own style. You use the style editor independently of devices and projects.

Views in the style editor

Objects are grouped in the style editor according to their main visual features.

The following groups are available:

- Buttons, lines, and polygon-based objects
- Text-based objects
- Trend-based objects, such as trend view,
- graphic or value-based objects, such as value table.

Style items

You use a style item to define the appearance of a screen object within a style. You can design several objects of the same type differently in one style. You define, for example, different style items for simple buttons and navigation buttons.

You create and manage style items in the style editor.

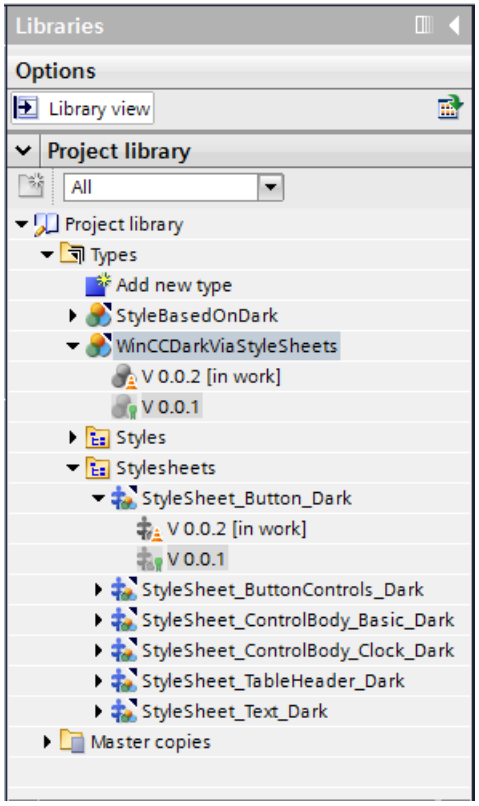
Style sheets

You use style sheets to configure the same properties for a group of objects. In a style sheet you can, for example, define the width and background color of table headers for all table-based objects. Style sheets are divided into predefined types depending on the included properties, for example, buttons, limits, table headers.

You use style sheets to define the design of shared object properties. Some special object properties are not included in style sheets. You use styles to design all objects of a project with visual consistency.

Managing styles and style sheets

You create and edit styles and style sheets in the project library. To get a better overview of types in expansive libraries, you save styles and style sheets in folders with meaningful names. You should also assign meaningful names to the styles and style sheets you create yourself.



Support of styles in faceplates

Styles and style items can be assigned to faceplate instances.

However, there is no direct connection between the faceplate types and styles, so selection and preview of a style item in the faceplate editor is not possible. To assign a style item to a faceplate, enter the name of the style item under "Style item appearance".

The configured style item is used in the instance of a faceplate, provided the name of the style item exists in the style used. Ensure that the name is written correctly.

See also

Displaying predefined styles (Page 3918)

Defining a style (Page 3919)

Managing styles (Page 3921)

Using style items (Page 3924)

Working with style sheets (Page 3926)

Displaying predefined styles

Introduction

In the "Style" editor, you see predefined styles and create your own styles for representing display and operating objects in Runtime.

Open "Styles" editor

1. Double-click "Common data" in the project navigation.
2. Click the "Styles" editor. The editor opens.

   Note
   Style editor in the project navigation

   The style editor is only visible in the project navigation if a device that supports styles, e.g., Comfort Panels, is created in the project.

3. The standard styles are displayed in the work area of the "Style" editor:

- WinCC Light
- WinCC Dark
- WinCC Fresh
- WinCC Wireframe.

Result

The settings for the groups and individual objects are displayed in the work area.

You will assign predefined styles or define your own styles in the next steps.

Predefined styles are write-protected and cannot be changed.

See also

Managing styles (Page 3921)

Defining a style (Page 3919)

Defining a style

Introduction

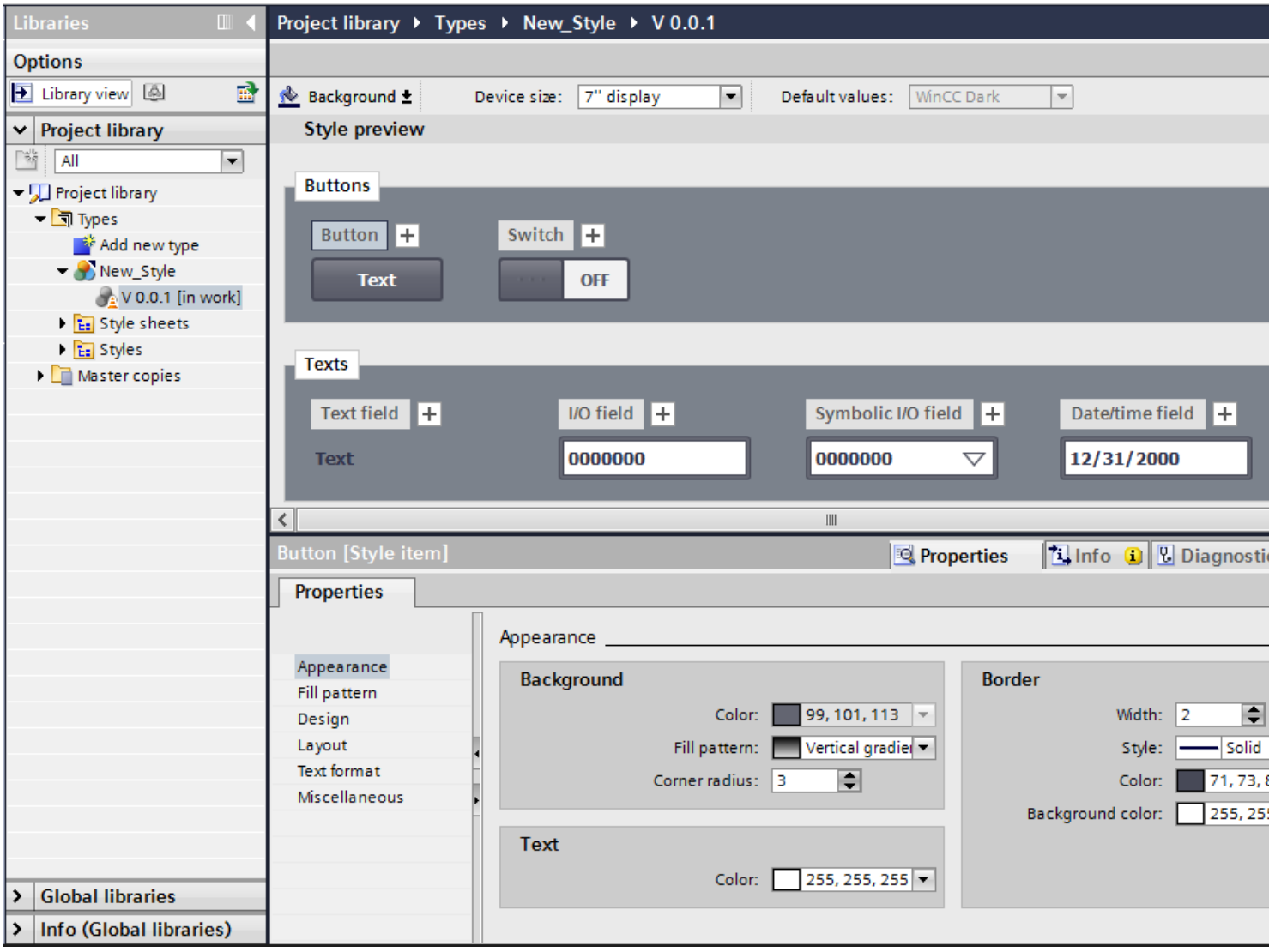
In the "Styles" editor, you see predefined styles. Predefined styles are write-protected and cannot be changed.

In addition, you have the option of defining your own styles and applying them to individual objects, groups, and even projects.

Defining a style

1. Open the "Common data" folder in the project navigation.
2. Open the "Style" editor.
3. Click "Add" in the "Styles" area.
   A new style is created.
   The library view opens. The version of the type is displayed in the "Types" folder. The version 0.0.1 has the status "in progress".

4. In the Inspector window, select the colors, borders and other settings for individual objects.



5. In order to activate the new style, select the current version "in progress" and select "Release version" in the shortcut menu.
   The new style appears in the "Styles" editor.

Result

You have added a new style which can be applied to objects or projects.

Defining a style based on a predefined style

1. Open the "Common data" folder in the project navigation.
2. Open the "Style" editor.
3. Select a predefined style in the work area.

4. Select the "Duplicate type" command in the shortcut menu.
   The "Duplicate type" dialog opens.
5. Overwrite the suggested name with a meaningful name.
6. In the "Comment" column, enter a description of the style, e.g. the intended use.
7. Click "OK" to confirm your input.
8. Select the created type in the project library.
9. Select "Edit new type" in the shortcut menu.
   The library view opens. The version of the type is displayed in the "Types" folder. The version 0.0.2 has the status "in progress".
10. Configure the type as required.
11. Select the "Release version" command from the shortcut menu of the edited version.

Result

They have configured a new style based on the predefined style of WinCC.

See also

Displaying predefined styles (Page 3918)

Managing styles

Deleting a style

You can delete styles that you have defined yourself.

1. Open the "Common data" folder in the project navigation.
2. Open the "Style" editor.
3. Select the style in the work area that you want to delete.

   Note

   You cannot delete the predefined styles of WinCC.

4. Select the "Delete" command in the shortcut menu.

The style is deleted.

Editing a style

You can edit styles that you have defined yourself.

1. Open the project library.
2. Select the released version of the style that you want to change.
3. Select "Edit type" in the shortcut menu.
   A new version "in progress" is created.

4. Make the required changes in the Inspector window.
5. Select the "Release version" command from the shortcut menu of the edited version to put the changes into effect.
   The changes made take effect.

Determining a standard style for a project

A project can be assigned a standard style to achieve a visual conformity of all display objects in the project.

1. Open the "Common data" folder in the project navigation.
2. Open the "Style" editor.
3. Activate a style in the work area.

The activated style is used as standard style in the project. When you add a device to the project, this device uses the standard style.

Changing the style for a specific device

Within a project, you change the style for a specific device.

1. Open the "Runtime settings" editor of the HMI device.
2. Select a device style under "General".

The selected style is applied to the device.

If the device style is not assigned, the device uses the standard style of the project.

See also

Displaying predefined styles (Page 3918)

Adjusting the font size to the device size

Introduction

You define the size of the device to whose objects you want to apply a style in the style editor. This means you specify device-dependent font sizes in a style. All font sizes are automatically adapted to the device size by selecting a specific reference size.

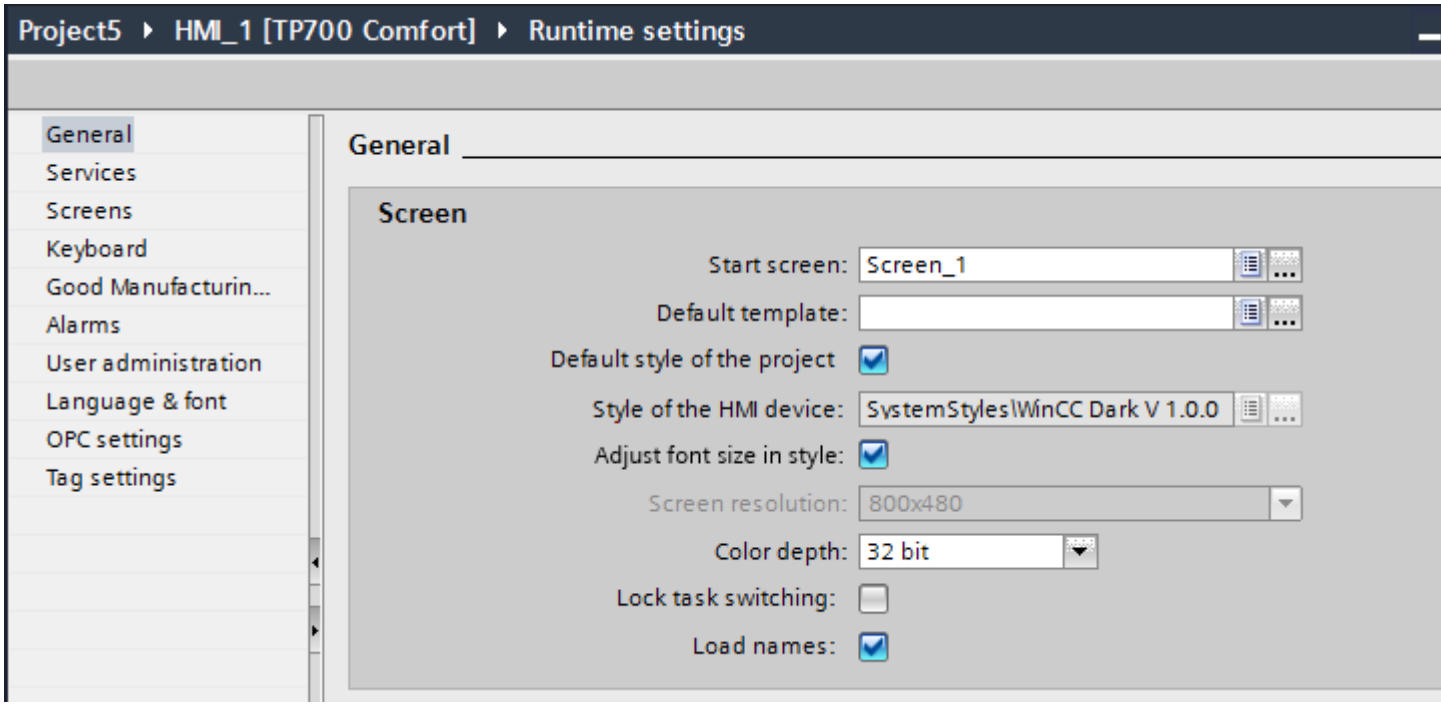
The table below shows the device sizes and the reference font sizes in pixels:

Device size	Font size in pixels
4" display	15 px
7" display	17 px
9" display	15 px
12" display	19 px
15" display	19 px

Device size	Font size in pixels
19" display	17 px
22" display	21 px
WinCC RT Advanced	17 px

Activating automatic adjustment of font sizes

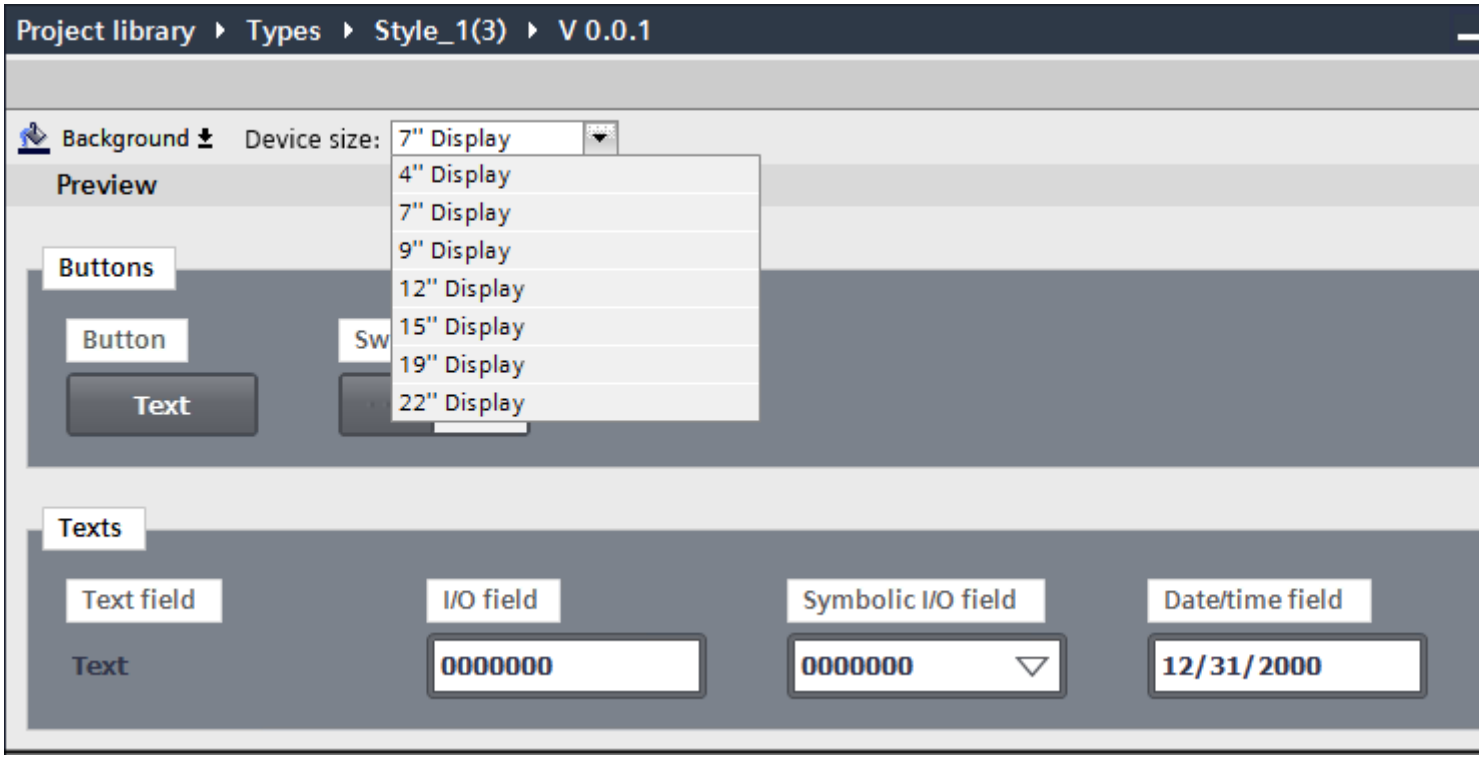
1. Double-click the "Runtime settings" editor in the project window.
2. Click "General".
3. Activate the option "Adjust font size in style" under "General > Screen".



Automatic adjustment of font sizes to the device size was activated in a style.

Adjusting font size in a device style

1. Open the corresponding style for editing in the project library.
2. Select the reference size of the HMI device in the work area.



All font sizes in the style are automatically adjusted to the selected device size.

Using style items

Introduction

You use style items to change the style of a screen object without changing the entire device style. You can, for example, define types of buttons with the help of different style items that are to receive the same appearance in one or several devices.

You define style items in the style editor.

Adding a new style item

1. Open the project library.
2. Select the released version of the style that you want to change.
3. Select "Edit type" in the shortcut menu.
   A new version "in progress" is created.
4. Select the object that you want to use in a different style.

5. Click the plus sign next to the object name.

   A new style item has been created.
   You can also select "Duplicate style item" in the shortcut menu of the object.

6. Make the required changes in the inspector window.
7. Assign a meaningful name to the new style item under "Properties > Miscellaneous > Object > Name".
8. Release the version of the style.

Deleting a style item

You can delete style items that you have defined yourself.

1. Open the project library.
2. Select the style that includes the style item.
3. Open the "in progress" version.
4. Select the style item that you want to delete in the work area.
5. Select the "Delete" command in the shortcut menu.

   Note

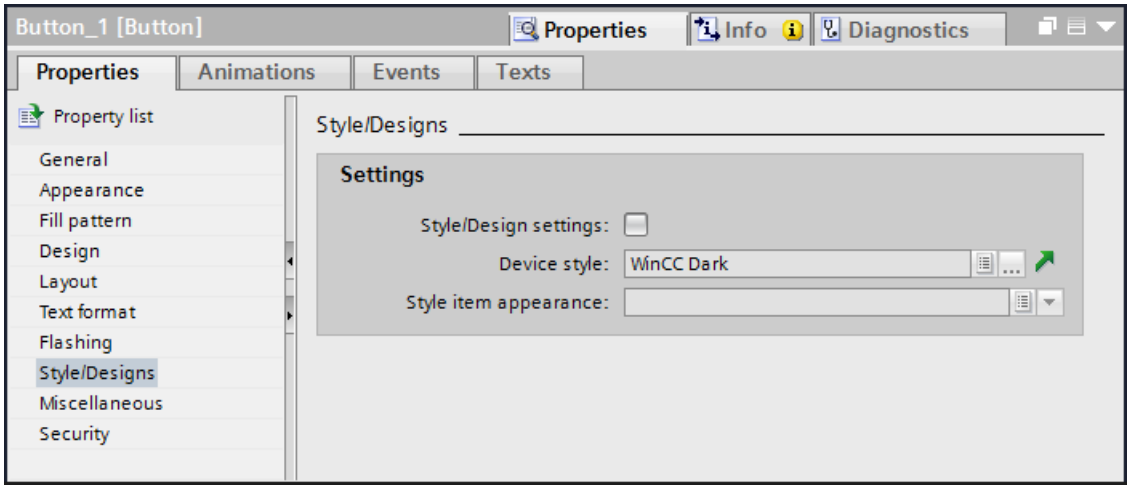
   You cannot rename or delete predefined WinCC style items.

The style item is deleted.

Specifying a style item for an object

If you have configured additional style items for an object in the device style, you can use the predefined style item for a screen object.

1. Open "Properties > Styles/Designs > Settings".
2. Enable the option "Use style/design".
3. Select the predefined style item under "Style item appearance".



The object is displayed as selected style item.

Working with style sheets

Basics on working with style sheets

Introduction

Some screen objects have the same properties which can be configured similarly. These general properties are grouped in style sheets. You can configure several properties for several objects simultaneously and consistently by using style sheets.

Types of style sheets

Style sheets are divided into several predefined categories. Each category contains a specific number of common properties shared by different objects. Select the required category when you create a style sheet. A style sheet can only be applied to objects for which the corresponding category is defined.

The table below shows the available categories and objects that can be configured with them.

Category/Objects	Polylines/ Lines	Text-based objects	Buttons	Value-based objects	Table-based objects	Diagrams
Button	-	-	X	-	X	X
Object body	X	-	-	X	X	X

Category/Objects	Polylines/ Lines	Text-based objects	Buttons	Value-based objects	Table-based objects	Diagrams
Diagram body	-	-	-	-	-	X
Focus and selection	-	X	X	-	X	X
Limits	-	X	-	X	-	-
Labels	-	-	-	X	-	-
Scale	-	-	-	X	-	-
Table body	-	-	-	-	X	X
Table header	-	-	-	-	X	X
Text field	-	-	-	-	-	-

Use of style sheets

You apply a style sheet to a style item when editing a style. The properties configured in the style sheet are transferred to the style item.

To apply a style sheet to a style item, drag the required style sheet from the task card to the style item in the work area using drag-and-drop.

Managing style sheets

Style sheets are library types. You manage style sheets in the project library. Style sheet categories are available in the project library in the "Style sheets > Categories" task card during style editing.

The style sheets that are available for a style sheet type are available in the "Style sheets > Visual attributes" task card. The task card includes predefined style sheets as well as style sheets you have created yourself and released.

Creating a new style sheet

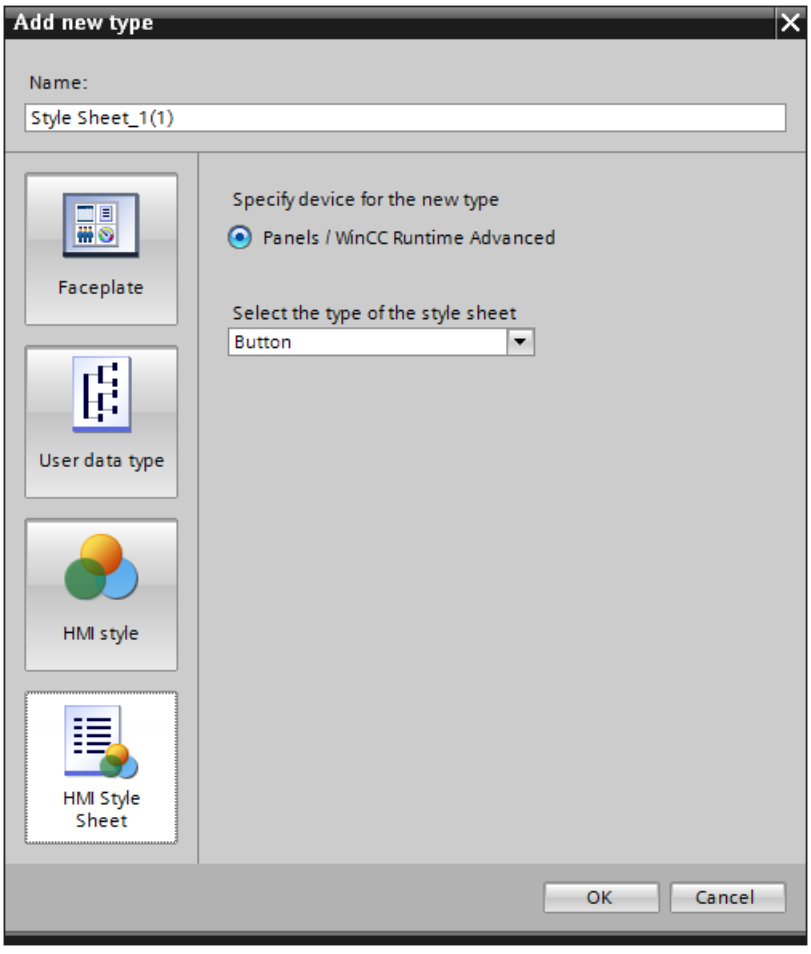
Introduction

You configure the properties for several objects of a similar type in a style sheet.

Create new style sheet

1. Open the project library.
2. Click the "Add new type" button.
   The "Add new type" dialog opens.
3. Select the option "HMI style sheet" in the dialog.

4. Select the type of the style sheet.



5. Assign a meaningful name to the style sheet.
6. Click "OK" to confirm your input.
   The new style sheet was created.

Create new style sheet based on a style

To automatically configure the properties for similar screen objects when editing a style, create a new style sheet based on properties of a style item that have already been configured.

1. Select the style item you want to use to create a style sheet.
2. Select "Create new style sheet" in the shortcut menu.
3. Select the category for the style sheet in the submenu.
   The "Add type" dialog opens.

4. Assign a meaningful name to the style sheet.
5. Click "OK" to confirm.
   The new style sheet based on a style was created and has been added to the "Style sheets > Visual attributes" task card.
   The properties in the style sheet were applied from the selected style item. The properties of the style sheet not included in the selected style have been assigned default values.

Managing style sheets

Introduction

You edit, copy and delete the style sheets in the project library.

You can also open, copy, create and delete style sheets with the commands in the shortcut menu of the "Style sheets > Categories" task card.

Edit style sheet

You can edit style sheets that you have created yourself.

1. Open the project library.
2. Select the released version of the style sheet that you want to change.
3. Select "Edit type" in the shortcut menu.
   A new version "in progress" is created.
4. Make the required changes in the Inspector window.
5. Select the "Release version" command from the shortcut menu of the edited version to put the changes into effect.
   The changes made take effect.

Delete style sheet

You can delete style sheets that you have defined yourself.

1. Open the project library.
2. Select the style sheet that you want to delete in the project library.

   Note

   You cannot delete the predefined style sheets of WinCC.

   Note

   You cannot delete the style sheet which includes an "in progress" type. To delete the style sheet, start by releasing all versions of the style sheet.

3. Select the "Delete" command in the shortcut menu.

The style sheet is deleted.

Copy style sheet

1. Open the project library.
2. Select the style sheet that you would like to copy.
3. Select "Duplicate type" in the shortcut menu.
4. Overwrite the suggested name with a meaningful name.
5. In the "Comment" column, enter a description of the style sheet, for example, the intended use.
6. Click "OK" to confirm your input.
   The required style sheet was copied and stored in the same folder.

Using style sheets

Introduction

You are using a style sheet while editing a style in the style editor. Style sheets are object-specific and are only applied to style items with which they are compatible.

Only a released version of a style sheet can be applied to a style item.

You have the following options to use a style sheet:

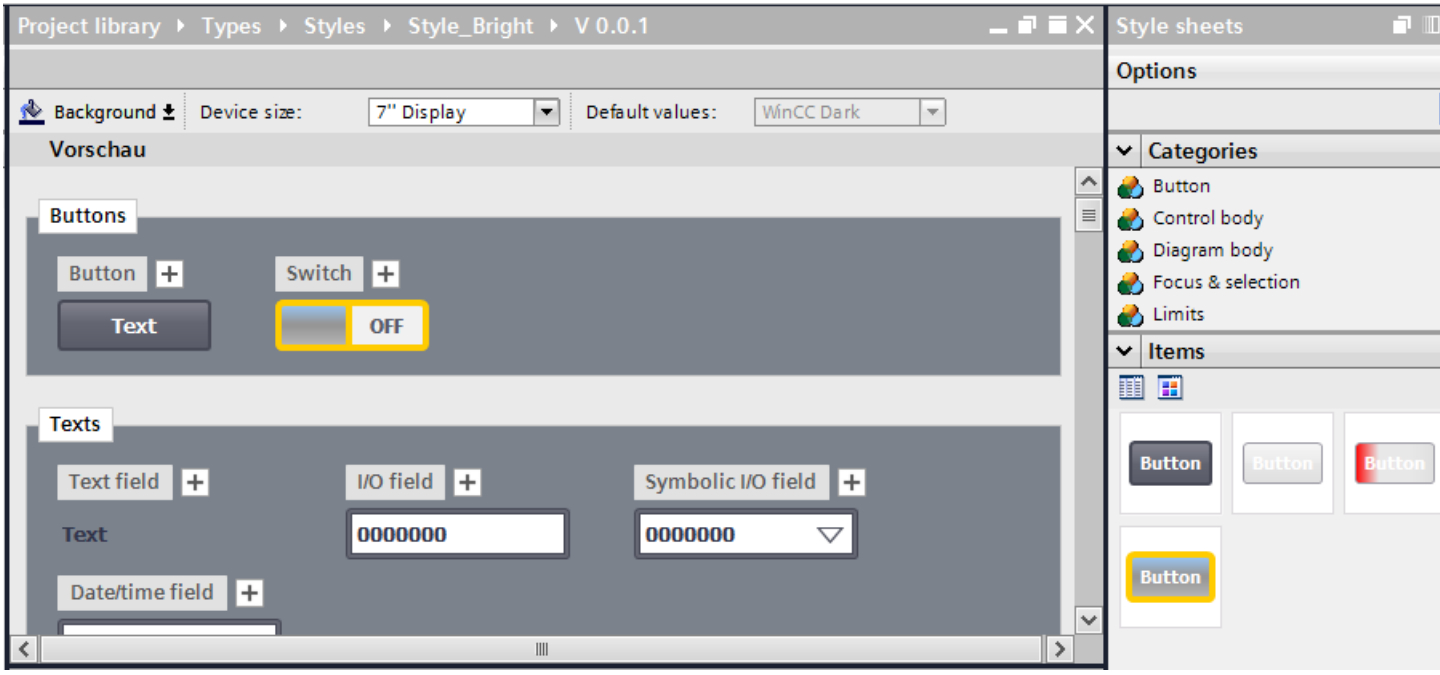
- To design the selected style item, drag the style sheet to a style item in the style editor.
- To design several style items of a group, drag the style sheet to a group of style items.
- To design all style items of the style which are compatible with the style sheet, drag the style sheet to an empty area in the style editor.

Requirement

- You have created a style.
- A version of the style is opened "in progress".

Apply style sheet to style item

1. Select the style sheet category in the "Style sheets > Categories" task card.
2. Select the required style sheet in the "Style sheets > Visual attributes" task card.
3. Drag the selected style sheet to a style item with drag&drop.



When you drop the style sheet on the style item, the style item applies the settings from the style sheet.

Note

There is no connection between the style item and the used style sheet. When you change the style sheet, the changes are not automatically applied in the style item. Apply the style sheet again to make the changes to the style sheet effective in the style item.

Note

Style sheets are not linked with screen objects. Unused or outdated style sheet versions are deleted by cleaning up or versioning the library.

Result

You have applied a style sheet to a style item in a style.

12.1.2 Working with objects

12.1.2.1 Overview of objects

Introduction

Objects are graphics elements which you use to design the screens of your project.

The "Tools" task card contains all objects that can be used for the HMI device. Display the Task Card with menu command "View" by activating the "Task Card" option.

The toolbox contains various palettes, depending on the currently active editor. If the "Screens" editor is open, the toolbox contains the following palettes:

- "Basic objects"
  The basic objects include basic graphic objects such as "Line", "Circle", "Text field" or "Graphic view".
- "Elements"
  The elements include basic operator controls, e.g. "I/O field", "Button" or "Gauge".
- "Controls"
  The controls provide advanced functions. They also represent process operations dynamically, for example Trend view and Recipe view.

- "Graphics"
  Graphics are broken down into subjects in the form of a directory tree structure. The various folders contain the following graphic illustrations:
  - Machine and plant areas
  - Measuring equipment
  - Operator controls
  - Flags
  - Buildings

  You can create links to your own graphic folders. The external graphics are located in these folders and subfolders. They are displayed in the toolbox and incorporated into the project using links.
- "Libraries" task card
  In addition to the display and operating elements, the library objects are available. They are located within the palettes of the "Libraries" task card. A library contains preconfigured objects such as graphics of pipes, pumps, or preconfigured buttons. You can also integrate multiple instances of library objects into your project without having to reconfigure them. The WinCC software package includes libraries, e.g. "HMI Buttons & Switches".
  You can also store customized objects, and faceplates in user libraries.
  Faceplates are objects that you create from existing screen objects, and for which you define the configurable properties.

Note

HMI device dependency

Some of the toolbox objects are either available with restricted functionality, or not at all. This depends on the HMI device you are configuring. Unavailable properties of an object are displayed as deactivated, and cannot be selected.

Basic objects

Symbol	Object	Instructions
	"Line"	-
	"Ellipse"	-
	"Circle"	-
	"Rectangle"	-
	"Text field"	One or more lines of text. The font and layout are adjustable.
	"Graphic view"	Displays graphics from external graphic programs, and inserts OLE objects. The following graphic formats can be used: "*.emf", "*.wmf", "*.dib", "*.bmp", "*.jpg", "*.jpeg", "*.gif" and "*.tif".

Elements

Symbol	Object	Instructions
	"I/O field"	Outputs the values of a tag, and/or writes values to a tag. You can define limits for the tag values shown in the I/O field. To hide the operator input in runtime, configure "Hidden input".
	"Button"	Executes a function list, or a script as configured.
	"Symbolic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A text from a text list is displayed in relation to the tag value.
	"Graphic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A graphic from a graphics list is displayed in relation to the tag value.
	"Date/time field"	Outputs the system date and time, or the time and date from a tag. This allows the operator to enter new values. The display format is adjustable.
	"Bar"	The bar represents a value from the PLC in the form of a scaled bar graph.
	"Switch"	Toggles between two defined states. You can label a switch with text, or a graphic.

Controls

Symbol	Object	Description
	"Alarm view"	Shows currently pending alarms or alarm events from the alarm buffer or alarm log.
	"Trend view"	Represents multiple curves with values from the PLC, or from a log.
	"User view"	Allows an administrator to administer users on the HMI device. It also allows an operator without administrator rights to change his password.
	"HTML Browser"[1)]	Displays HTML pages.
	"Recipe view"	Displays data records, and allows them to be edited.
	"System diagnostics view"	Provides an overview of all devices capable of diagnostics. Displays errors in the plant.

[1)]Available for Basic Panels 2nd Generation.

See also

Overview of objects (Page 3935)

Designing an object (Page 3952)

12.1.2.2 Overview of objects

Introduction

Objects are graphics elements which you use to design the screens of your project.

The "Tools" task card contains all objects that can be used for the HMI device. Display the Task Card with menu command "View" by activating the "Task Card" option.

The toolbox contains various palettes, depending on the currently active editor. If the "Screens" editor is open, the toolbox contains the following palettes:

- "Basic objects"
  The basic objects include basic graphic objects such as "Line", "Circle", "Text field" or "Graphic view".
- "Elements"
  The elements include basic operator controls, e.g. "I/O field", "Button" or "Gauge".
- "Controls"
  The controls provide advanced functions. They also represent process operations dynamically, for example Trend view and Recipe view.
- "Graphics"
  Graphics are broken down into subjects in the form of a directory tree structure. The various folders contain the following graphic illustrations:
  - Machine and plant areas
  - Measuring equipment
  - Operator controls
  - Flags
  - Buildings

  You can create links to your own graphic folders. The external graphics are located in these folders and subfolders. They are displayed in the toolbox and incorporated into the project using links.
- "Libraries" task card
  In addition to the display and operating elements, the library objects are available. They are located within the palettes of the "Libraries" task card. A library contains preconfigured objects such as graphics of pipes, pumps, or preconfigured buttons. You can also integrate multiple instances of library objects into your project without having to reconfigure them. The WinCC software package includes libraries, e.g. "HMI Buttons & Switches".
  You can also store customized objects, and faceplates in user libraries.
  Faceplates are objects that you create from existing screen objects, and for which you define the configurable properties.

Note

HMI device dependency

Some of the toolbox objects are either available with restricted functionality, or not at all. This depends on the HMI device you are configuring. Unavailable properties of an object are displayed as deactivated, and cannot be selected.

Basic objects

Symbol	Object	Instructions
	"Line"	-
	"Polyline"	The polyline is an open object. Even if the start and end points have the same coordinates, the area they enclose cannot be filled in. If you want to fill a polygon, select the "Polygon" object.
	"Polygon"	-
	"Ellipsis"	-
	"Circle"	-
	"Rectangle"	-
	"Text field"	One or more lines of text. The font and layout are adjustable.
	"Graphic view"	Displays graphics from external graphic programs, and inserts OLE objects. The following graphic formats can be used: "*.emf", "*.wmf", "*.dib", "*.bmp", "*.jpg", "*.jpeg", "*.gif" and "*.tif".

Elements

Symbol	Object	Instructions
	"I/O field"	Outputs the values of a tag, and/or writes values to a tag. You can define limits for the tag values shown in the I/O field. To hide the operator input in runtime, configure "Hidden input".
	"Button"	Executes a list of functions, or a script as configured.
	"Symbolic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A text from a text list is displayed in relation to the tag value.
	"Graphic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A graphic from a graphics list is displayed in relation to the tag value.
	"Date/time field"	Outputs the system date and time, or the time and date from a tag. This allows the operator to enter new values. The display format is adjustable.
	"Bar"	The bar represents a value from the PLC in the form of a scaled bar graph.
	"Charge condition"	Shows the charge condition of the battery for Mobile Wireless

Symbol	Object	Instructions
	"Switch"	Toggles between two defined states. You can label a switch with text, or a graphic.
	"Slider"	Displays a current value from the PLC, or sends a numeric value to the PLC.
	"Symbol library"	Used to add screen objects based on controls of the same name.
	"Effective range name"	Shows the name of the effective range.
	"Effective range name (RFID)"	Displays the name of the effective range (RFID).
	"Effective range signal"	Indicates how accurately the Mobile Panel Wireless is positioned within an effective range.
	"WLAN reception"	Indicates how good the WLAN radio connection is.
	"Gauge"	Displays numeric values. The appearance of the gauge is adjustable.
	"Zone name"	Displays the name of the zone.
	"Zone signal"	Indicates how accurately the Mobile Panel Wireless is positioned within a zone.
	"Clock"	Displays the system time in analog or digital format.

Controls

Symbol	Object	Description
	"Alarm view"	Shows currently pending alarms or alarm events from the alarm buffer or alarm log.
	"Trend view"	Represents multiple curves with values from the PLC, or from a log.
	"User view"	Allows an administrator to administer users on the HMI device. It also allows an operator without administrator rights to change his password.
	"HTML browser"	Displays HTML pages.
	"Camera view"[1)]	Displays pictures from a connected network camera.
	"PDF view"	Displays PDF documents.
	"Status/Force"	This function gives the operator direct read / write access from the HMI device to individual address areas in the connected SIMATIC S7.

Symbol	Object	Description
	"Sm@rtClient view"	Allows an operator to monitor and maintain a connected device remotely.
	"Recipe view"	Displays data records, and allows them to be edited.
	"f(x) trend view"	Represents the values of a tag as a function of another tag.
	"System diagnostics view"	Provides an overview of all diagnostics capable devices. Displays errors in the plant.
	"Media Player"	Media files are shown in the media player.

[1] Available for Comfort Panels and KTP Mobile Panels (as of device version V13).

See also

Basics on groups (Page 3977)

Overview of keyboard access (Page 3984)

Screen basics (Page 3889)

Example: Configuring a rectangle (Page 3987)

Storing an external image in the graphics library (Page 3974)

Managing external graphics (Page 3973)

External graphics (Page 3971)

Repositioning and resizing multiple objects (Page 3968)

Selecting multiple objects (Page 3966)

Inserting multiple objects of the same type (stamping) (Page 3964)

Flashing (Page 3952)

Flipping objects (Page 3951)

Rotating objects (Page 3949)

Showing objects outside the screen area (Page 3948)

Moving an object forwards or backwards (Page 3947)

Aligning objects (Page 3946)

Resizing objects (Page 3943)

Positioning an object (Page 3942)

Deleting an object (Page 3941)

Inserting an object (Page 3939)

Overview of objects (Page 3932)

12.1.2.3 Options for editing objects

Introduction

Objects are graphics elements which you use to design the screens of your project.

You have the following options for editing objects:

- Copying, pasting or deleting objects using the shortcut menu If you copy an object in a screen and the screen already includes an object of the same name, the name of the object is changed.
- Maintaining the standard size of the objects you are inserting or customizing their size on insertion.
- Moving an object in front of or behind other objects
- Rotating objects
- Mirroring objects
- Defining the tab sequence for objects
- Stamping: Inserting several objects of the same type
- Selecting several objects simultaneously
- Repositioning and resizing multiple objects
- You can assign external graphics to objects, e.g. in the Graphic View.
  You can view only the images you have previously stored in the graphic browser of your WinCC project.
  You can save graphics in the graphic browser:
  - Via drag & drop from the "Graphics" object group to the working area
  - As graphic files in the following formats: *.bmp, *.dib, *.ico, *.emf, *.wmf, *.gif, *.tif, *.jpeg or *.jpg
  - As an OLE object
    You either create a new OLE object or save an existing graphic file as an OLE object. To save an OLE object, an OLE-compatible graphics program must be installed on the configuration computer.

12.1.2.4 Inserting an object

Introduction

In the "Screens" or "Reports" editor, insert the objects to the "Toolbox" task card. Use the mouse to drag the objects into the work area. You either keep the objects in their original size, or scale them up or down when you paste them.

In addition, you can copy or move objects via the clipboard from one editor to another, for example to transfer a screen object to a report. Alternatively, you can also use the mouse instead of the clipboard for copying and moving:

- Copying: <Ctrl + Drag&Drop>
- Moving: Drag&drop

Note

Basic Panels

The "Reports" editor is not available for Basic Panels.

Requirement

The "Tools" task card is open.

Inserting objects in their standard size

1. In the "Toolbox" task card, select the desired graphic object or the desired graphic in the WinCC graphics folder.
   When you move the cursor across the work area, it turns into a crosshair with an appended object icon.
2. Click the location in the work area where you want to insert the object or graphic.
   The object is inserted with its standard size at the desired position in the work area.

Alternatively, double-click the object in the "Toolbox" task card.

Copying an object

1. Select the desired object.
2. Select "Copy" in the shortcut menu.
3. Click the desired location and select "Paste" in the shortcut menu.

WinCC inserts a copy of the object at the desired location. You can only change the properties that are appropriate for the relevant context.

Example: In the "Screens" editor, you can set for I/O fields and the mode for input and output. In the "Reports" editor, the mode is set to "Output".

Original and copy are not interconnected and are configured independently from one another.

Inserting lines

1. Select the desired graphic object in the "Tools" task card.
2. Click on a location in the work area. A line in the standard size is inserted.

Inserting a polygon or polyline

1. Select the desired object "Polyline" or "Polygon" in the "Tools" task card.
2. Click on a location in the work area. This fixes the starting point of the object.
3. Click another location in the work area. A corner point is set.
4. For every additional corner point, click on the corresponding location in the work area.
5. Double-click on a location in the work area. The last corner point is set.
   This fixes all the points of the polygon or polyline.

Note

Basic Panels

The "Polyline" and "Polygon" objects are not available for Basic Panels.

Note

If you want to insert several objects of the same type, use the "Stamp" function. This avoids having to reselect the object in the "Tools" task card every time before inserting it. To do so, select the icon in the toolbar of the "Tools" task card.

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.5 Deleting an object

Introduction

You can delete objects individually or with a multiple selection.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to delete.
   To delete multiple objects, keep the <Shift> key pressed and select the objects to be deleted one after the other. Alternatively, drag and maximize an area around the desired objects with the mouse.
2. Select "Delete" from the shortcut menu.

Result

The selected objects are deleted.

See also

Overview of objects (Page 3935)

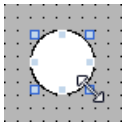
Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.6 Positioning an object

Introduction

When you select an object, it is enclosed by a rectangle with resizing handles. This rectangle is the rectangle which surrounds the object. The position of an object is defined by the coordinates of the top left corner of the rectangle surrounding the object.



Note

If the position is outside the work area the object is not displayed in Runtime.

Position and align

You have the possibility of having a grid displayed in the work area. You have three options for easier positioning of objects:

- "Snap to grid" When you resposition objects, they are automatically snapped and pasted to the grid. If you hold down the <Alt> key, the object is no longer snapped to the grid.
- "Snap to objects" When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.
- "None": You position the objects in any position.

You activate and deactivate the grid and options as follows:

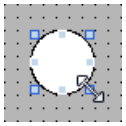
- In the "Options > Settings > Visualization > Screens" menu
- In the "Layout > Grid" task card

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object you want to move.
   The selected object is framed by a rectangle with resizing handles.



2. Left-click the object and keep the mouse button pressed.
3. Move the mouse pointer onto the new position.
   The contour of the object moves with the mouse and displays the new position for the object.

   The object initially remains at its original position.
4. Now release the mouse button.
   The object is moved into the position indicated by the contour of the selection rectangle.

Alternative procedure

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the X and Y values for the position under "Position & Size".

Result

The object appears at its new position.

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.7 Resizing objects

Introduction

When you select an object, it is enclosed by a rectangle with handles. You have the following options of resizing an object:

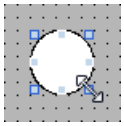
- Drag the handles using the mouse.
- Modify the "Size" property in the Inspector window.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object you want to resize.
   The selection rectangle appears. The following figure shows a selected object:



2. Drag a resizing contact of the rectangle to a new position.
   The size of the object changes.
   - The size of the object is aligned to the grid pattern, provided the "Snap to grid" function is set.
   - Press <ALT> to disable this function while you drag the object.
     In order to scale the object proportionally, keep the <Shift> key pressed while changing the size with the mouse.

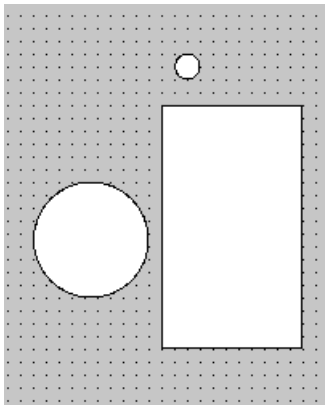
Alternative procedure

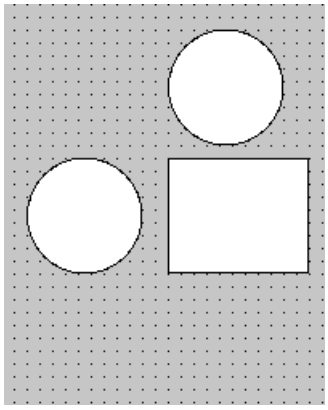
1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the size of the object under "Position & Size".

Harmonizing the object size

1. Select the objects.
2. Now, click one of the following buttons: or or
   The size of the selected objects is matched to each other.

The following screen shows how the selected objects are adapted to the height of the reference object:





Icon	Description
	Aligns the selected objects to the width of the reference object.
	Aligns the selected objects to the height of the reference object.
	Aligns the selected objects to the width and height of the reference object.

Result

The object now appears with its new size.

See also

Overview of objects (Page 3935)

Designing an object (Page 3952)

12.1.2.8 Aligning objects

Procedure

1. Select the objects via multiple selection.
2. Specify an object as the reference object.
3. Select the desired command in the toolbar or the shortcut menu - see table below.
   The selected objects will be aligned.

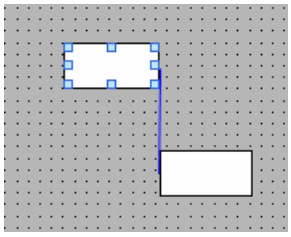
Aligning objects flush

The selected objects will be aligned flush to the reference object.

Icon	Description
	Aligns the selected objects to the left edge of the reference object.
	Aligns the selected objects to the vertical center axis of the reference object.
	Aligns the selected objects to the right edge of the reference object.
	Aligns the selected objects to the upper edge of the reference object.
	Aligns the selected objects to the horizontal center axis of the reference object.
	Aligns the selected objects to the lower edge of the reference object.
	Centers the selected objects to the center points of the reference object.
	Centers the selected objects vertically in the screen.

Snap to object

When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.



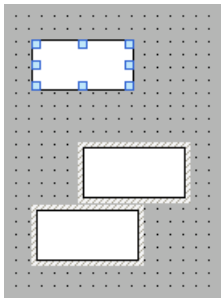
If you are working with the keyboard, press the Alt key. When you move the selected object with the arrow keys, the next anchor point is displayed.

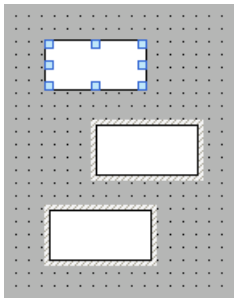
Distributing objects evenly

You need at least three selected objects. A reference object is not required.

1. Select the objects.
2. Click one of the buttons "Distribute horizontally equal" or "Distribute vertically equal".
   The selected objects are distributed at equal distances.

The following screen shows how you align the vertical spacing of the selected objects:





Icon	Description
	Aligns the horizontal distance between the objects. The position of the objects on the extreme left and right side remains unchanged. All other objects are distributed evenly between them.
	Aligns the vertical distance between the objects. The position of the objects at the extreme top and bottom (right and left) remains unchanged. All other objects are distributed evenly between them.

See also

Overview of objects (Page 3935)

Designing an object (Page 3952)

12.1.2.9 Moving an object forwards or backwards

Introduction

You can use the "Order" functions in the shortcut menu of a selected object or in the toolbar to move a selected object in front of or behind other objects within an object layer.

Note

ActiveX controls are always positioned in front of an object layer (.NET property).

Requirement

You have opened a screen which contains a layer with multiple objects.

Procedure

1. Select the object you want to move forward or backward.
2. Select the "Sort" command and one of the following commands from the shortcut menu:

Icon	Description
	Moves the selected object before all the other objects of the same layer
	Moves the selected object to the lowest position in the same layer
	Moves the selected object up by one position
	Moves the selected object down by one position

Alternative procedure

1. Open the "Layers" palette of the "Layout" task card.
2. Navigate to the required object.
3. Hold down the mouse button, and drag the object in the tree topology to the required position in the layer.
4. Now release the mouse button.

Result

The object is moved up or down.

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.10 Showing objects outside the screen area

Introduction

If you assign objects to positions that are outside the configurable area, these objects will be hidden. The functions of the "Objects outside the visible area" palette in the "Layout" task card are used to move these objects back into the screen.

Requirement

- You have opened a screen which contains objects that are outside the configurable area.
- The "Layout" task card is open.

Procedure

1. Open the "Layout > Objects outside the area" task card.
   This displays a list of objects that are outside the configurable area.
2. Select the the object which you want to move back into the screen from the list.
3. Select "Move to screen" in the object shortcut menu.

Alternatively open the "Layout > Layer" task card. Objects outside the area are indicated by the icon. If you click this icon, the object is moved back into the screen.

Result

The objects are moved to the configurable area.

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.11 Rotating objects

Introduction

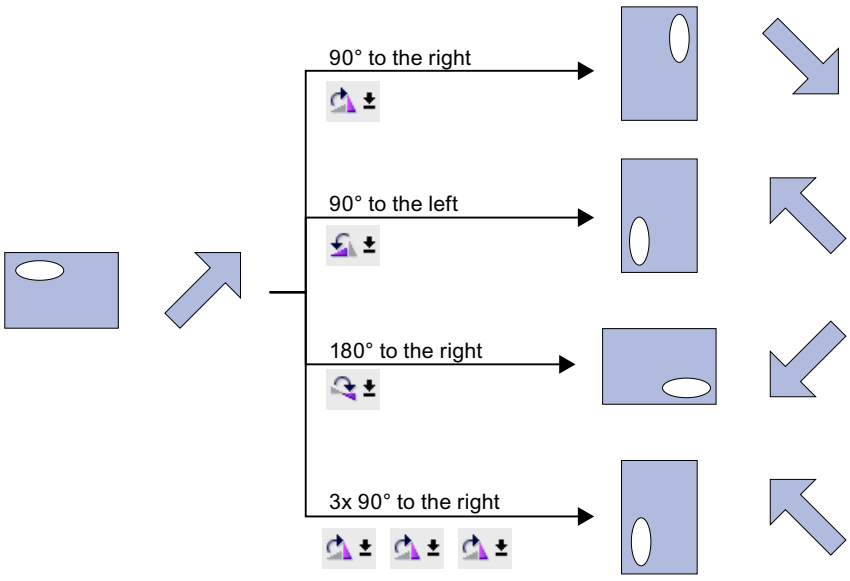
You can rotate a suitable object clockwise or counterclockwise around its center axis in steps of 90°.

Note

Not all the objects can be rotated. Some objects that can be rotated in screens cannot be rotated in reports.

You can also rotate multiple objects using the multiple selection function. Certain WinCC objects such as buttons cannot be rotated.

The alignment of elements in an object will change in a rotated object. The following figure shows how a rectangle and an ellipse behave under the different commands for rotating an object:



Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to rotate.
2. Click one of the following toolbar icons:
   , to rotate the object clockwise around its center point. The angle of rotation is 90°.
   , to rotate the object counterclockwise around its center point. The angle of rotation is 90°.
   , to rotate the object clockwise by 180°.

Result

The object is shown at its new angle.

See also

Overview of objects (Page 3935)

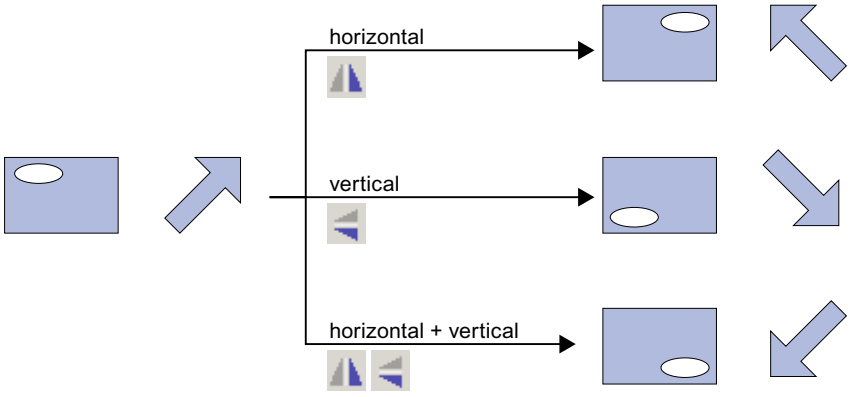
Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.12 Flipping objects

Introduction

You can flip an object along its vertical or horizontal center axis. The alignment of elements in an object will change when you flip an object. The following figure shows how a rectangle and an ellipse behave under the different commands for flipping an object.



Requirement

You have opened a screen which contains at least one object.

Procedure

1. Select the object that you want to flip.
2. Click the "Flip" command in the shortcut menu and select one of the options displayed:
   - , to flip the selected object along its vertical center axis.
   - , to flip the selected object along its horizontal center axis.

Result

The object is shown at its flipped position.

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.13 Flashing

Introduction

You want an object to flash in Runtime.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to flash.
2. In the Inspector window, select the type "Default" under ""Properties > Properties > Flashing".

Result

The object flashes in runtime.

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.14 Designing an object

Introduction

You design the border and background of an object.

Requirements

A line has been created in a screen.

Procedure

1. Select the line on your screen.
2. In the Inspector window, select "Properties > Properties > Appearance":
3. Select "Dash" as the style.
4. To display the dashed line in two colors, select the line width "1".
5. Select the setting "Arrow" in the "Line ends" area.

Result

The line is displayed in two colors as a dashed line. The end of the line is an arrow.

See also

Overview of objects (Page 3932)

Inserting an object (Page 3939)

Deleting an object (Page 3941)

Positioning an object (Page 3942)

Resizing objects (Page 3943)

Aligning objects (Page 3946)

Moving an object forwards or backwards (Page 3947)

Showing objects outside the screen area (Page 3948)

Rotating objects (Page 3949)

Flipping objects (Page 3951)

Flashing (Page 3952)

Inserting multiple objects of the same type (stamping) (Page 3964)

Selecting multiple objects (Page 3966)

Repositioning and resizing multiple objects (Page 3968)

Managing own controls (Page 3968)

External graphics (Page 3971)

Managing external graphics (Page 3973)

Storing an external image in the graphics library (Page 3974)

12.1.2.15 Designing the fill pattern

Introduction

WinCC lets you design the background color and the fill pattern of an object. The design options change in the Inspector window depending on object for which you are making the filling pattern.

For certain objects, you can not only define the color, but also a transparent background or a background with a color gradient.

Note

Device-specific nature of the properties

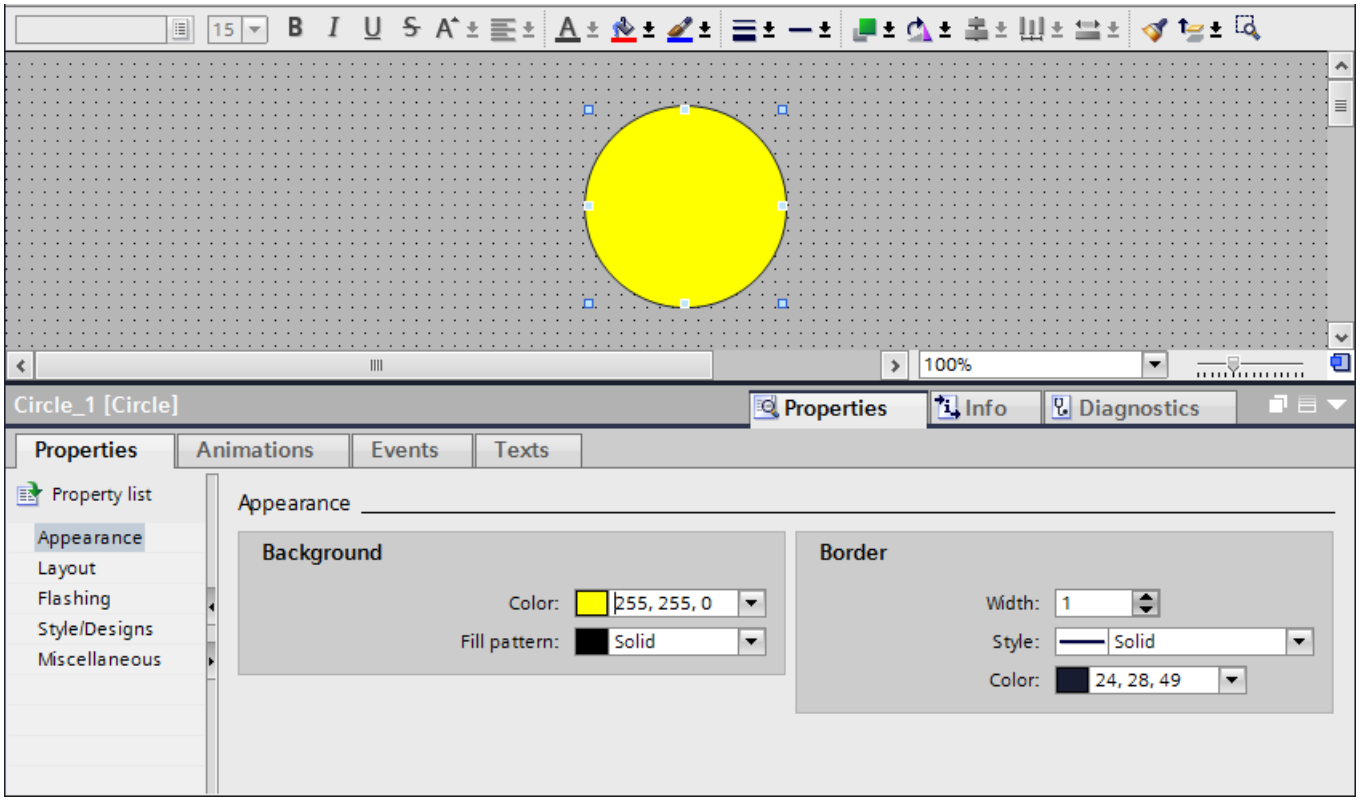
The fill style that is available depends on the object and the HMI device used.

Requirement

The object has been created and selected.

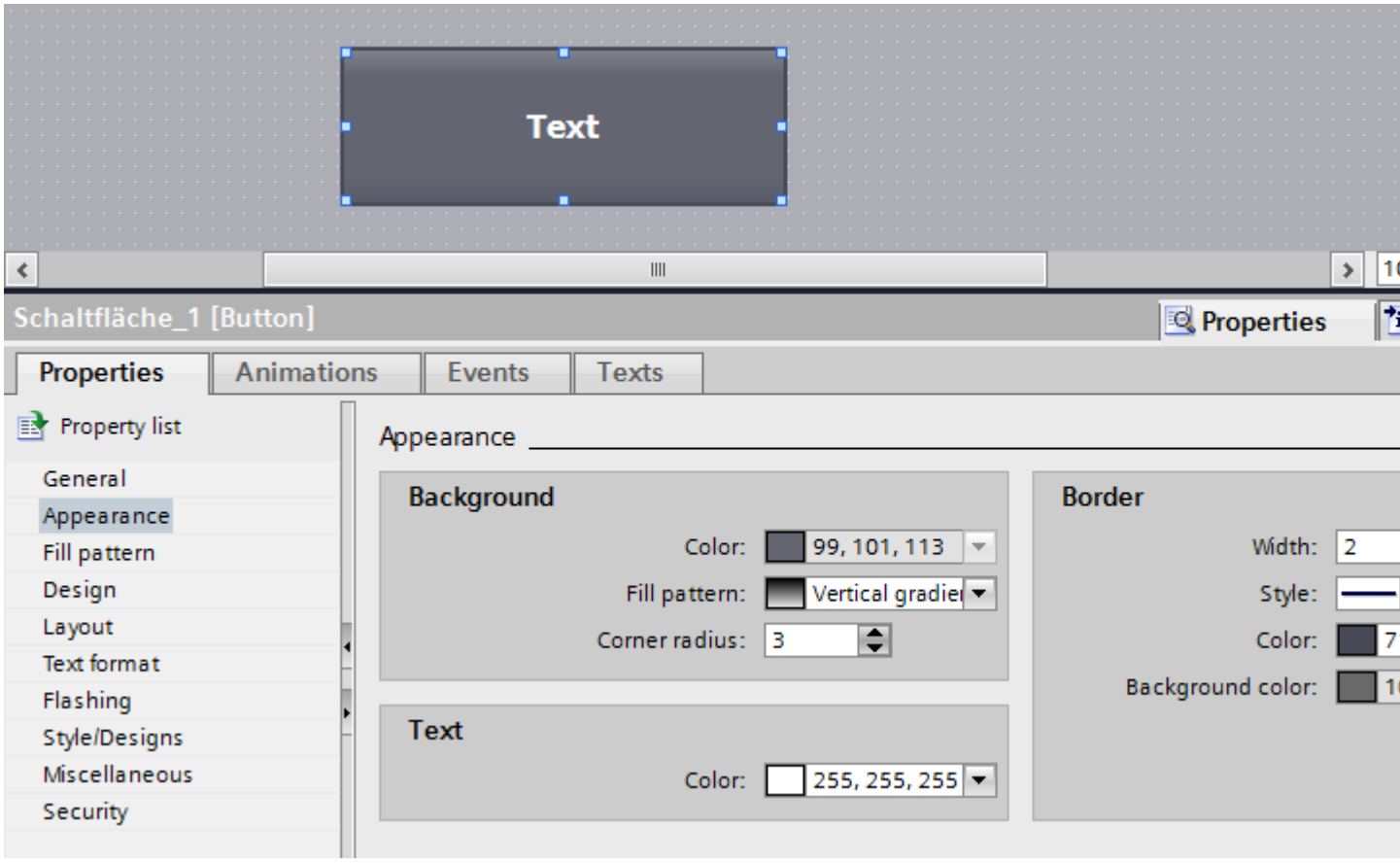
Designing the background color of an object

1. In the Inspector window, select "Properties > Properties > Appearance".
2. Select a color for the background of the object, for example, yellow.
   The object is filled with the selected color.



Designing the fill pattern of an object

1. In the Inspector window, select "Properties > Properties > Appearance".
2. To define a transparent background for the object, select transparent.
   You can find information about designing a fill pattern with a color gradient in the section Defining color gradients (Page 3962).
   The object is shown as transparent.



You can optionally set the fill pattern in the Inspector window under "Properties > Properties > Fill Pattern".

See also

Defining color gradients (Page 3962)

12.1.2.16 Formatting text in an object

Introduction

Some objects, for example, the I/O field, support text within the object.

You have several options to align text.

Configuring the distance from the object border

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the value "5" for the left border, for example.

The text is aligned five pixels from the left border of the object.

Aligning the text position

1. In the Inspector window, select "Properties > Properties > Text format".
2. Select the horizontal alignment, for example, centered.
3. Select the vertical alignment, for example, top.
   The text is shown at the top center in the object.

Defining the text orientation

1. In the Inspector window, select "Properties > Properties > Text format".
2. Select the orientation of the text, for example, vertical, right.
   The text flow is shown from bottom to top.

12.1.2.17 Formatting graphics in an object

Introduction

WinCC lets you insert graphics into some objects and format them. You can change the size, orientation and distances of a graphic to the object border. You change the properties of the graphic in the Inspector window of the relevant object.

Requirement

- The selected object is in graphic mode.
- The selected object contains at least one graphic.

Resizing a graphic

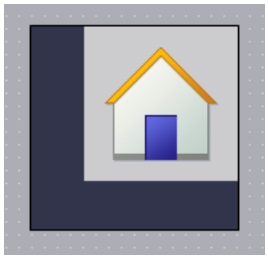
1. In the Inspector window, select "Properties > Properties > Layout".
2. Select "No stretching of picture" or "Stretch picture".
   The contained graphic is displayed in its original size or stretched to the size of the object.

Note

To adjust the object size to the contained graphic, select "Fit object to contents".

Aligning a graphic horizontally and vertically

1. In the Inspector window, select "Properties > Properties > Layout".
2. To determine the horizontal position of the graphic, select "Right", for example.
3. To determine the vertical position of the graphic, select "Top", for example.
   The graphic is then shown at the top right of the object.



Defining the distance from the object border

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the value for the distance to the object border, for example, "20" for the top border.
   The graphic is shown with a distance of 20 pixels to the top border of the object.

Icon	Description
	Sets the distance from the left border of the object.
	Sets the distance from the right border of the object.
	Sets the distance from the top border of the object.
	Sets the distance from the bottom border of the object.

12.1.2.18 Connecting tags and text lists in the text

Introduction

Comfort Panels and WinCC Runtime Advanced give you the option to add output fields for tags or entries to a text list for the following objects:

- Symbolic I/O field
- Text field

Note

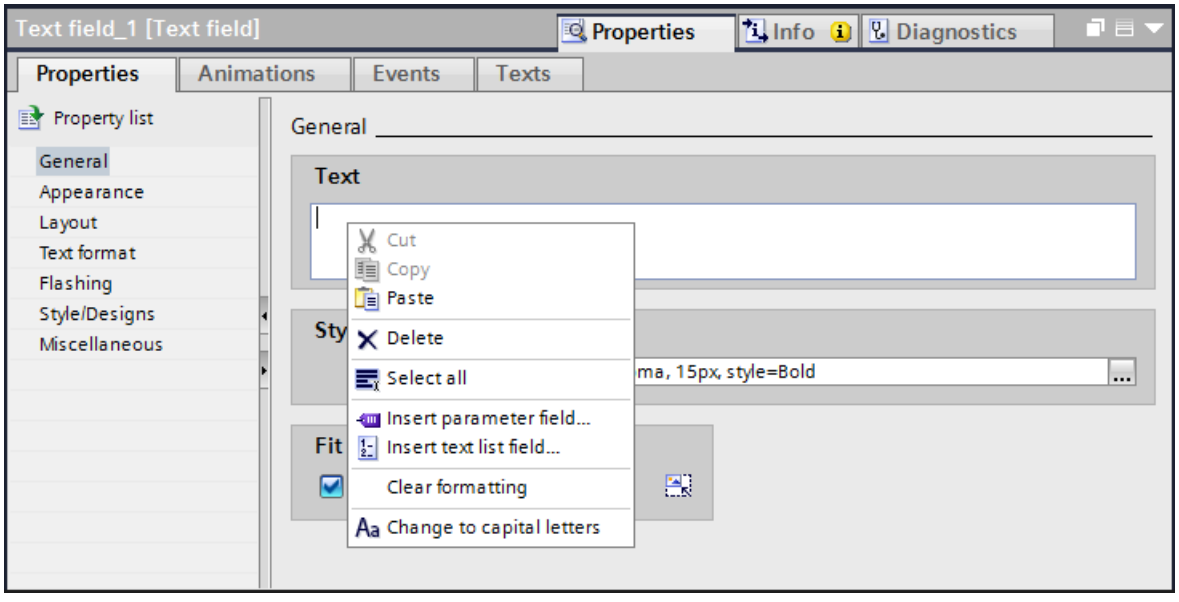
The symbolic I/O field supports the connection of tags and text list entries in the "Two states" mode.

Requirement

- The object has been created and selected

Connecting tags in a text field

1. Open "Properties > Properties > General" in the inspector window.
2. Open the shortcut menu of the input field.



3. Select the command "Insert parameter field ...".
   A dialog opens.
4. Select the tag and the format in which you want to display the tag.
5. Confirm your entries.

The name of the connected tag is displayed in the text field of the object.

Connecting the text list

1. Open "Properties > Properties > General" in the inspector window.
2. Open the shortcut menu of the input field.
3. Select the command "Insert text list field ...".
   A dialog opens.
4. Select the text list from which a text entry is displayed.
5. Confirm your entries.

The reference to the text list entry is displayed in the text field of the object.

Note

The number of references to text list entries that in turn include references to text list entries or tags is limited to three layers.

12.1.2.19 Designing a border

Introduction

WinCC offers various properties for designing display and operating objects.

The I/O field example below shows the possible settings for designing borders.

You change the appearance in the Inspector window of the relevant object.

Designing a border

1. Open "Properties > Properties > Appearance".
2. Select the width of the border, e.g. "5".
3. Select the style of the border, e.g. double line.
4. Select the foreground color.
5. Select the background color.

Figure 12-1 "Double line" border style

Border style

The figure below shows a border in "3D style".

The figure below shows a border in "Solid" style.

Note

Device-specific nature of the properties

The border style that is available depends on the object and the HMI device used.

12.1.2.20 Designing table-based objects

Introduction

WinCC offers various properties for designing display and operating objects.

You change the properties of table-based objects in the Inspector window of the relevant object.

Designing colors

Select the colors for font, selections and areas under "Properties > Properties > Appearance".

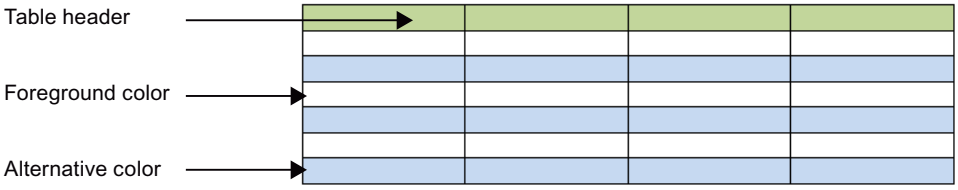


Figure 12-2 Selecting table properties

Designing the border of the table header

1. In the Inspector window, open "Properties > Properties > Table header border".
2. Select "Solid", for example, as the style.
3. Select 5, for example, as the "Width".
4. Select red, for example, as a foreground color.

The border of the table header is shown with a red border and width of 5 pixels. With "Solid" border style, only the foreground color is visible.

Colors and color gradient in the table header

1. Select the table-based object in the screen.
2. In the Inspector window, open "Properties > Properties > Table header fill pattern".
3. Select the fill pattern, for example "Horizontal gradient".
4. Select a background color, for example, blue, under "Gradient".
5. Activate "Gradient 1".
6. Select a color for "Gradient 1", for example, white.
7. Select a "Width" for the color gradient, for example, 12.
8. Activate Gradient 2.
9. Select a color for Gradient 2, for example, yellow.
10. Select a "Width" for the color gradient, for example, 10

The table header is displayed with a color gradient.

12.1.2.21 Defining color gradients

Introduction

For the objects in WinCC, color gradients can be specified as background for various surfaces.

Change the category in the Inspector window, depending on which surface you fill with a color gradient. The procedure remains the same.

The following section describes how to configure the color gradient of buttons.

Horizontal color gradient with two colors

1. Select an object with buttons, for example, a button.
2. In the Inspector window, select "Properties > Properties > Fill Pattern".
3. Select "Fill Pattern > Horizontal gradient".
4. Select a background color for the vertical color gradient, for example, orange.
5. Activate "Gradient 1".
6. Select a "Color" for Gradient 1, for example, red.
7. Select a "Width" for Gradient 1, for example, "10".

The background of the button is displayed in orange.

For the horizontal color gradient, Gradient 1 is shown from the left border. Gradient 1 is 10 pixels wide.

Vertical color gradient with three colors

1. Select a button in the screen.
2. In the Inspector window, select "Properties > Properties > Fill Pattern".
3. Select the "Vertical gradient" background.
4. Select a background color under "Gradient", for example, orange.
5. Activate "Gradient 1".
6. Select a color for "Gradient 1", for example, red.
7. Select a "Width" for the color gradient, for example, 8.
8. Activate Gradient 2.

9. Select a color for Gradient 2, for example, yellow.
10. Select a "Width" for the color gradient, for example, 10.

	← Color Gradient 1
	← Background color
	← Color Gradient 2

For the vertical color gradient, Gradient 1 is shown from top to bottom. Gradient 1 is 8 pixels wide.

The background of the button is displayed in orange.

Gradient 2 is shown at the bottom border. Gradient 2 is 10 pixels wide.

12.1.2.22 Using predefined styles

Introduction

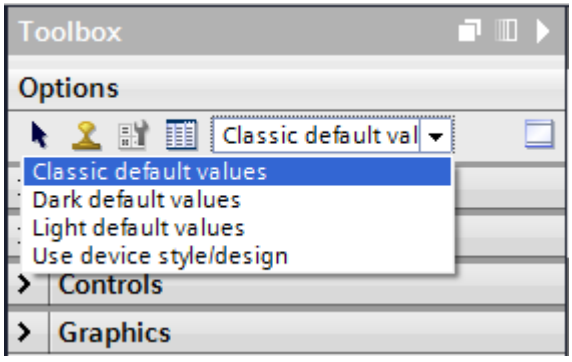
You can assign predefined styles to objects and faceplates in Runtime. You can change the background color of the display and operating elements using predefined styles. In this way, you harmonize the display in Runtime.

Requirement

The "Tools" task card is open.

Setting predefined styles

1. Select the object that you want to insert in the "Toolbox" task card.
2. Select one of the following options in the "Toolbox" task card:
   - "Classic default values" to use the default style
   - "Dark default values" to use the dark color scheme
   - "Light default values" to use the light color scheme
   - "Use device style/design" to use the settings from the current device design



3. Insert the required object in the work area.
   The object is displayed in the selected style.
4. The objects are created in the selected style as long as the style remains activated in the toolbar.
   To return to the default style, select the "Default" option in the toolbar.

See also

Working with styles (Page 3916)

12.1.2.23 Inserting multiple objects of the same type (stamping)

Introduction

WinCC offers the possibility to "stamp" several objects of the same type directly one after the other, i.e. paste without having to reselect the object every time. In addition you have the possibility of multiplying an object that has already been inserted.

Requirement

The "Tools" task card is open.

Inserting several objects of one type

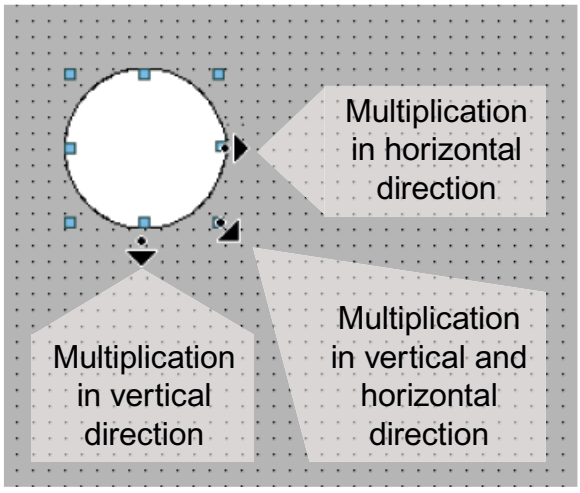
1. Select the object that you want to insert in the "Tools" task card.
2. Click the icon in the toolbar of the "Tools" task card.
   The "Stamp" function is activated.
3. To insert the object with its standard size, click the relevant insertion position in the work area.
   To insert the object with another size, position the mouse pointer at the desired location in the work area. Press the left mouse button and drag the object to the required size.
   The object is inserted in the work area as soon as you release the mouse button.
4. Repeat step 3 to insert further objects of the same type.
5. Click the icon again.
   The "Stamp" function is deactivated.

Note

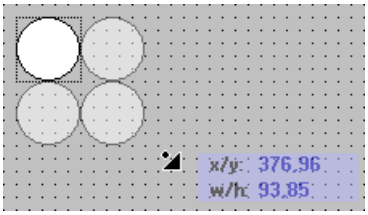
You can copy existing objects using the drag-and-drop +<CTRL> function. The existing object is not moved in this case. You paste a copy of this object into the new position instead.

Inserting and multiplying an object

1. Insert the desired object from the "Tools" task card.
2. Press the <Ctrl> key and position the cursor on one of the handles displayed in the figure shown below.



3. Drag the handles to the right and/or down while keeping the left mouse button pressed.
4. The object is multiplied depending on available space if you keep moving the cursor.



Result

You have pasted and stamped an object in a screen.

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.24 Selecting multiple objects

Introduction

Select all objects you want to align with each other or to change global properties. This procedure is called "multiple selection."

The Inspector window shows all the properties of the selected objects.

You now have several options of selecting multiple objects:

- Draw a selection frame around the objects.
- Hold down the <Shift> key, and click the required objects.

Selection frame of a multiple selection

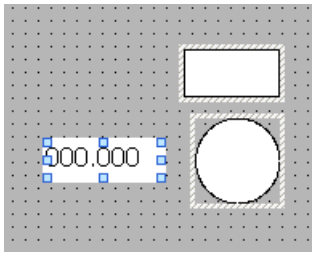
The selection frame surrounds all objects of a multiple selection. The selection frame is comparable with the rectangle that surrounds an individual object.

The selection frame is not visible. When you have made your multiple selection, the following frame is displayed:

- The reference object is indicated by the rectangle around it.
- The other selected objects are indicated by a dashed-line frame.

Specifying a reference object

The reference object is the object upon which the other objects are oriented. The reference object is framed by a rectangle with handles. The following figure shows a reference object with two other selected objects:



You have the following options to specify the reference object:

- Select the objects via multiple selection. The object selected first is then the reference object.
- Draw a selection frame around the objects. The reference object compiled automatically. If you wish to specify a different object within the selection as the reference object, click on the desired object. This action does not cancel your multiple selection.

Requirement

You have opened the work area containing at least two objects.

Selecting multiple objects with a selection frame

1. Position the mouse pointer in the work area close to one of the objects to be selected.
2. Hold down the mouse button, and draw a selection frame around the objects to be selected.

Or:

1. Hold down the <Shift> key.
2. Click the relevant objects, working in succession.
   All the selected objects are identified by frames.
   The object selected first is identified as reference object.

Note

To remove an object from the multiple selection, press <SHIFT>, hold it down and then click the relevant object once again.

Result

Multiple objects are selected. One of those is identified as the reference object. You can now perform the following steps:

- Changing the object properties of all the objects
- Resizing all the objects by the same ratio, by dragging the selection frame to increase or reduce the size
- Moving all the objects in one group
- Aligning the objects to the reference object

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.25 Repositioning and resizing multiple objects

Possible modifications

After you have selected multiple objects, you edit them:

- Shift using the mouse
  - To change the absolute position of the marked objects, position the mouse pointer over an object, and shift the multiple selection with the mouse button pressed.
  - To resize all the objects by the same ratio, grab the resizing handles of the reference object.
- Move over the work area with the icons of the toolbar
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects
- Moving with the shortcut menu commands of the work area
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects

See also

Overview of objects (Page 3935)

Designing an object (Page 3952)

12.1.2.26 Managing own controls

Introduction

Add standardized or user-defined controls in the "Toolbox > My Controls" task card. All the ActiveX controls registered in the operating system and the .Net controls on your system are available for use in WinCC.

- ActiveX controls
  ActiveX controls are operator controls from any providers, which can be used by other programs over a defined OLE-based interface.
- .Net controls
  .Net controls are operator controls from any providers based on the Microsoft .Net Framework.
- Specific .Net controls
  .Net controls are operator controls from any providers based on the Microsoft .Net Framework.

When you intend to run external controls in your WinCC project, you need to verify that these run on the HMI device you are configuring. If you are using controls from third parties, you

must first of all add these to the "My controls" palette before you copy the project to different PCs.

Note

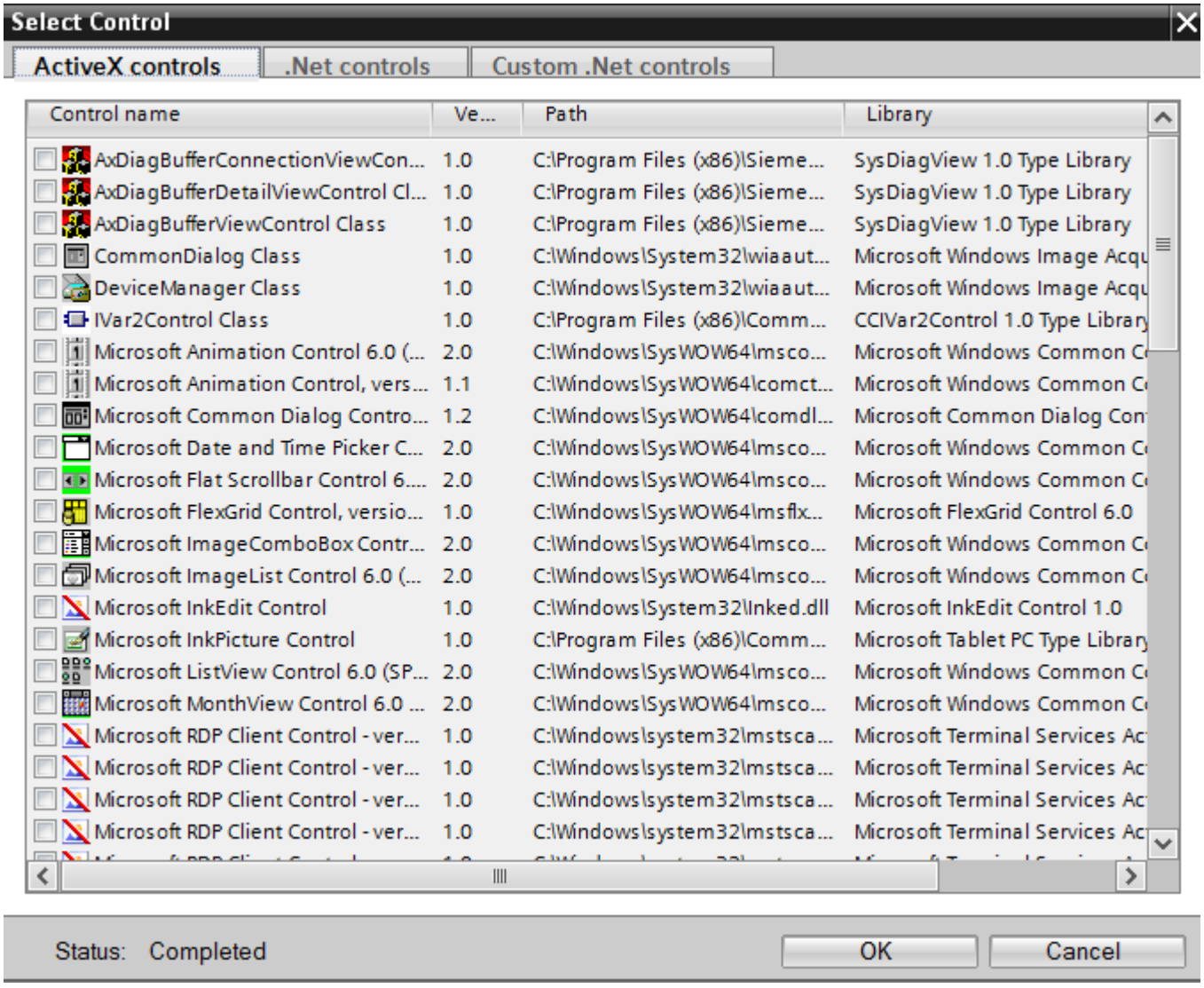
If you use any controls that are not included in the product package (for example, from third parties), errors may occur, for example, reduced performance or system crashes. The user of such control software is alone responsible for any problems caused by its use. We therefore advise you to closely examine such elements prior to their use.

Requirement

- The required ActiveX control is installed in the PC's operating system.
- The required or specific .Net control is available on the PC in an assembly.
- A screen is open.
- The "Toolbox" task card is open.

Adding a new control to the "My controls" group

1. Open "Toolbox > My controls" in the "Toolbox" task card.
2. Select "Select objects" from the shortcut menu.
   A dialog opens. This contains all the controls available on the system. Controls that already exist in the "Toolbox" task card are identified by a check mark.



3. Select the desired tab, e.g. ActiveX controls.
4. Activate the required control.
5. Confirm your selection with "OK".

Inserting a specific .Net control in the "My controls" group

1. Select the "Specific .Net controls" tab. The list is empty the first time you select the tab.
2. Click the "…" button.

3. Select the folder in which the assembly is contained.
4. Select a file.
5. Click "Open." All the available controls are displayed in the "Specific .Net controls" tab.
6. Activate the required .Net control.
7. Confirm your selection with "OK".

Note

.Net-Controls in Runtime Advanced

If you have incorporated a .Net Control in your project as "Specific .Net-Control", you have to copy the files belonging to these controls to the installation directory of WinCC Runtime, e.g. "C:\ProgramFiles\Siemens\Automation\WinCC RT Advanced". Otherwise, the control cannot be loaded in Runtime.

Result

The control now appears in the "Toolbox > My controls" task card and can be configured in a screen.

Removing a control from the "My controls" group

1. Select "Delete" from the shortcut menu for the control concerned.
   The control is removed from the "My controls" group.

See also

Designing an object (Page 3952)

12.1.2.27 External graphics

Introduction

You can use graphics created with an external graphic program in WinCC. To use these graphics you store them in the graphic browser of the WinCC project.

You can save graphics in the graphic browser:

- When you drag-and-drop graphics objects from the "Graphics" pane into the work area, these are stored automatically in the graphic browser. The graphic names are numbered in the order of their creation, for example, "Graphic_1." Use the <F2> function key to rename the graphic.
- As a graphic file with the following formats:
  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tif, *.png, *.jpeg or *.jpg
- As an OLE object that is embedded in WinCC and is linked to an external graphic editor. In the case of an OLE link, you open the external graphic editor from WinCC. The linked object is edited using the graphic editor. An OLE link only works if the external graphic editor is installed on your PC, and supports OLE.

Use of graphics from the graphic browser

Graphics from the graphic browser are used in your screens:

- In a Graphic view
- In a graphics list
- As labeling for a button/function key

Transparent graphics

In WinCC you also use graphics with a transparent background. When a graphic with a transparent background is inserted into a graphic object of WinCC, the transparency is replaced by the background color specified for the graphic object. The selected background color is linked firmly with the graphic. If you use the graphic in another graphic object of WinCC, this object is displayed with the same background color as the graphic object that was configured first. If you want to use the graphic with different background colors, include this graphic in the graphic browser again under a different name. The additional background color is configured when the graphic is used at the corresponding graphic object of WinCC.

Managing graphics

An extensive collection of graphics, icons and symbols is installed with WinCC, for example:

In the Toolbox window of the "Graphic" pane the graphic objects are structured by topic in the "WinCC graphics folder." The link to the WinCC graphics folder cannot be removed, edited or renamed.

The "Graphics" pane is also used to manage the external graphics. The following possibilities are available:

- Creating links to graphics folders
  The external graphic objects in this folder, and in the subfolders, are displayed in the toolbox and are thus integrated in the project.
- Editing folder links
- You open the program required for editing of the external graphic in WinCC.

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.28 Managing external graphics

Introduction

External graphics that you want to use in WinCC are managed in the "Screens" editor by using the "Tools" task card in the "Graphics" pane.

Requirement

- The "Screens" editor is open.
- The "Tools" task card is open.
- The graphics are available.
- The graphics have the following formats:
  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tif, *.jpeg or *.jpg

Creating a folder link

1. Click "My graphics folder."
2. Select "Link" in the shortcut menu.
   The "Create link to folder" dialog is opened. The dialog suggests a name for the folder link.
3. Edit the name as required. Select the path containing the graphic objects.
4. Click "OK" to confirm your input.
   The new folder link is added to the "Graphics" object group. The external graphics that are located in the target folder and in sub-folders are displayed in the toolbox.

Editing folder links

1. Select the folder link to edit.
2. Select the "Edit link..." command from the shortcut menu.
   The "Create link to folder" dialog is opened.
3. Edit the name and path of the folder link as required.
4. Click "OK" to confirm your input.

Renaming the folder link

1. Select the folder link to rename.
2. Select "Rename" from the shortcut menu.
3. Assign a name to the new folder link.

Removing a folder link

1. Select the folder link you want to delete.
2. Select "Remove" in the shortcut menu.

Edit external graphics

1. Select the graphic you want to edit.
2. Select the "Edit graphic" command from the shortcut menu.
   This opens the screen editor associated with the graphic object file.

Editing graphics folders from WinCC

1. Select the graphic you want to edit.
2. Select the "Open parent folder" command from the shortcut menu.
   The Windows Explorer opens.

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.29 Storing an external image in the graphics library

Introduction

To display graphics that have been created in an external graphics program in your screens, you will first have to store these graphics in the graphics browser of the WinCC project.

Requirement

- A screen has been created.
- A Graphic View is inserted in the screen.
- The Inspector window of the Graphic view is open.

To store an external graphic in the image browser:

- A graphic is available.

To store an OLE object in the browser:

- An OLE-compatible graphics program is installed on your configuration computer.

Save graphics file

1. Open the Windows Explorer.
2. Select the graphic that you want to store.
3. Drag-and-drop the graphic into the graphic browser.

Creating and saving a new graphic as an OLE object

1. Select the Graphic view on your screen.
2. In the Inspector window, select "Properties > Properties > General":
3. Open the graphic selection list.
4. Click .
5. The "Insert object" dialog box opens.

   Note

   In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

6. From the "Insert object" dialog box, select "New" and an object type. The settings in "Settings > "OLE settings" determine which object types are shown.
7. Click "OK." The associated graphic program is opened.
   When you are finished creating graphics, end the graphic programming software with "File > Close" or "File > Close & return to WinCC."
   The graphic will be stored in the graphic programming software standard format and added to the graphic browser.

Inserting created graphics in WinCC

Note

A new graphic object created as OLE object may not be directly accepted in WinCC when you save it to an external graphics program.

1. Reopen the dialog for inserting a graphic.
2. From the "Insert object" dialog box, select "Create from file."
3. Click the "Browse" button.
4. Navigate to the created graphic and select it.

Saving an existing graphic object as an OLE object

1. In the Inspector window, select "Properties > Properties > General":
2. Open the graphic selection list.
3. Click .
4. The "Insert object" dialog box opens.

Note

In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

5. From the "Insert object" dialog box, select "Create from file."
6. Click the "Browse" button.
7. Use the dialog to help you navigate to the folder in which the graphic file is saved.

Note

To import graphics files, note the following size restrictions:

*.bmp, *.tif, *.emf, *.wmf ≤4 MB

*.jpg, *.jpeg, *.ico, *.gif "*≤1 MB

Result

The image file is now stored in your image browser. It is shown in a screen with a Graphic view , or is added as a list element in an image list.

You can double-click OLE objects in your library to open them for editing in the corresponding graphic editor. When you have finished editing graphics, end the graphic programming software with "File > Close" or "File > Close & return to WinCC." The changes are applied to WinCC.

See also

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Designing an object (Page 3952)

12.1.2.30 Working with object groups

Basics on groups

Introduction

Groups are several objects that are grouped together with the "Group" function. You edit a group in the same way as any other object.

Overview

WinCC offers the following methods for editing multiple objects:

- Multiple selection
- Grouping objects

Editing mode

To edit an individual object in a group, select the object in the "Layout > Layers" task card.

Alternatively select "Group > Edit group" in the object group's shortcut menu.

Hierarchical groups

You add further objects or groups to extend a group. The group is enlarged by the new objects and is structured hierarchically in main and sub-groups. Such hierarchical groups must be broken up in stages. You also break up the group in the same order in which you grouped the objects or groups. It takes exactly the same number of steps to break up these hierarchical groups as it did to create them.

Rectangle surrounding the object

Only one surrounding rectangle is now displayed for the whole group. The surrounding rectangles of all objects are displayed for a multiple selection on the other hand.

Layers

All objects of a group are located in the same layer.

Properties of a group

A group has its own properties, such as authorization. If you set a property at a group, all objects of the group inherit this setting. If the same property is configured differently at an object in the group, the value of the property at the object and not the value in the group applies in Runtime.

See also

Grouping objects (Page 3978)

Ungroup (Page 3979)

Adding objects to a group (Page 3980)

Removing objects from the group (Page 3981)

Editing an object in a group (Page 3982)

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Grouping objects

Introduction

The "Group" command combines multiple objects to form a group.

You can change the size and position of the group. The following rules apply:

- The system automatically adapts the position coordinates of the grouped objects when you reposition the group. The relative position of the grouped objects to the group is not affected.
- The system automatically adapts the height and width of the grouped objects in proportion to a change of the group size.
- To change the size of the group proportionately, hold down the <Shift> key and drag the rectangle around the object until has the required size.

Note

To create a hierarchical group, organize the individual groups like objects.

Requirement

- You have opened a screen which contains at least two objects.

Creating object groups

1. Select all the objects you want to organize in a group.
2. Select the command "Group > Group" from the shortcut menu.

The objects of the group are displayed with a rectangle around the objects.

Grouping objects within a group

1. Select the group you want to edit.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group that you are editing is highlighted by a red frame.

3. Select the objects of a group that you want to combine into a subgroup.
4. Select the command "Group > Group" from the shortcut menu.

A subgroup with the objects is created.

Adding objects into an existing group

1. Select the group to which you want to add objects.
2. Press the <Shift> key and select the object you want to add to the group.
3. Select the "Group > Add to group" command from the shortcut menu.

The object is added to this group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

Result

The selected objects are combined in a group. The multiple selection rectangle becomes the rectangle surroundings the objects in the group. The handles are shown only for the group. The group in in the active layer.

See also

Basics on groups (Page 3977)

Example: Configuring a rectangle (Page 3987)

Ungroup

Introduction

Select the "Ungroup" command to split a group into the original object fractions.

Requirement

- You have opened a screen that contains a group.

Ungroup

1. Select the group.
2. Select the "Group > Ungroup" command from the shortcut menu.

Ungrouping a group within a group

1. Select the higher-level group.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group that you are editing is highlighted by a red frame.
3. Select the lower-level group.
4. Select the "Group > Ungroup" command from the shortcut menu.

Result

The lower-level group is ungrouped. The objects are assigned to the next higher group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

See also

Basics on groups (Page 3977)

Example: Configuring a rectangle (Page 3987)

Adding objects to a group

Introduction

The "Add objects to group" command is used to add objects to a group, without ungrouping it first.

Requirements

A screen with one group and at least one other object is opened.

Procedure

1. Select the group.
2. Press the <Shift> key and select the object you want to add to the group.
3. Select the "Group > Add to group" command from the shortcut menu.

Result

The group consists of the original objects, and the newly-added objects. The added objects are arranged at the front of the group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag&drop you can also easily edit hierarchical groups in the "Layers" palette.

See also

Basics on groups (Page 3977)

Example: Configuring a rectangle (Page 3987)

Removing objects from the group

Introduction

You use the "Remove objects from group" command to remove individual objects from a group, without ungrouping it first.

You do not have to remove the object from the group to edit an object in a group. You can edit the objects of a group individually.

Requirement

- You have opened a screen that contains a group.

Removing objects from a group

To remove an object from a group:

1. Select the group.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group you want to edit is highlighted by a red frame.
3. Select all objects in the group that you want to remove from the group.
4. Select the "Group > Remove from group" command from the shortcut menu.

The objects are removed from the group.

Note

If there are only two objects in the group, the menu command "Remove from group" is not available.

Deleting objects from a group

To remove an object from the group, and from the screen:

1. Select the group.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group you want to edit is highlighted by a red frame.
3. Select the objects in the group that you want to delete.
4. Select "Delete" from the shortcut menu.

Note

If there are only two objects in the group, the menu command "Delete" is not available.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

See also

Basics on groups (Page 3977)

Example: Configuring a rectangle (Page 3987)

Editing an object in a group

Introduction

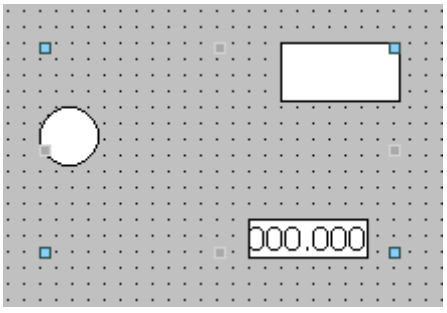
You can edit the objects of a group individually.

Requirement

You have opened a screen that contains a group.

Editing grouped objects

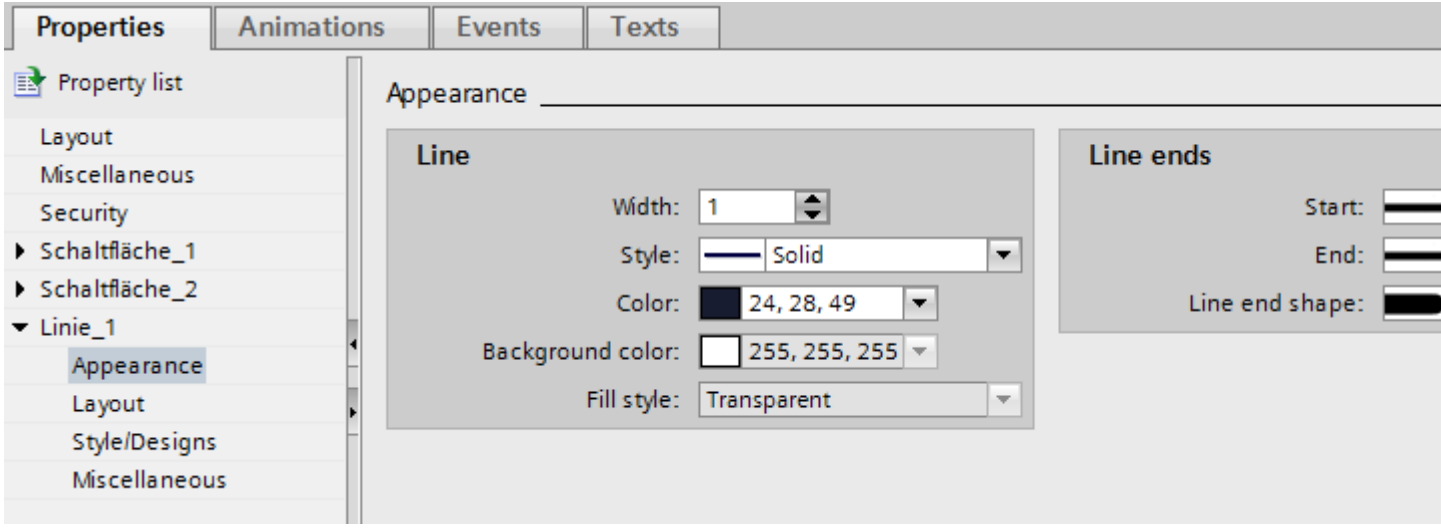
1. Select the group.



   The properties of the group are displayed in the Inspector window.
2. Change the position and size of the grouped objects in "Properties > Properties >Layout."
3. Change the name of the group in "Properties > Properties > Miscellaneous."

Modifying the properties of an object within a group

1. Select the group.
2. Select the object whose properties you want to change in the Inspector window.



   The properties of the object are displayed.
3. Change the object properties.

Note

The dynamization of properties for all objects of the group which have these properties is not possible in a group. You can only dynamize the properties of the objects associated with a group for each object itself.

Result

Although you have edited the object, it is still an element of the group. These changes do not affect the other objects of the group.

See also

Basics on groups (Page 3977)

Example: Configuring a rectangle (Page 3987)

12.1.2.31 Configuring the keyboard access

Overview of keyboard access

Introduction

For keyboard units without a mouse, the operator activates control objects using the <Tab> key. You can set up keyboard input to make the process easier to run, and to make sure that the operator enters all the necessary values. If you are using the keyboard, use the <Tab> key to activate the objects in a certain order, and to enter the necessary values.

For HMI devices without key, you can simulate the <Tab> key by configuring the "SimulateSystemKey" system function to a function key.

Operator authorizations and operator control enables

If you configure an object for operator input with the <Tab> key, the object must have both an operator authorization, and an operator control enable.

Editing the tab sequence

The tab sequence is determined automatically when the control objects are created. The numbers of the tab sequence are assigned in the sequence in which the libraries are created.

It makes sense to change the tab sequence in the following cases:

- The operator changes directly to a specific control object.
- The screen requires a specific sequence

Change to the tab sequence mode to change the tab sequence. In this mode, the tab sequence number is displayed at the top left of the control objects. The tab sequence numbers of hidden objects are also shown. The distribution of these numbers is edited using the mouse.

Note

Other functions are not available in the tab sequence mode.

See also

Setting the tab sequence order (Page 3986)

Defining the operator authorization and operator control enable for an object (Page 3985)

Overview of objects (Page 3935)

Example: Configuring a rectangle (Page 3987)

Defining the operator authorization and operator control enable for an object

Introduction

If you configure an object for operator input with the <Tab> key, the object must have both an operator authorization, and an operator control enable.

Requirement

You have opened a screen which contains at least one object.

Procedure

1. Select the object.
2. Select "Properties > Properties > Security" in the Inspector window.
3. Select the operator authorization under "Authorization."
4. Activate the authorization to operate.

Result

The operator can use the <Tab> key in Runtime to select the object.

See also

Overview of keyboard access (Page 3984)

Example: Configuring a rectangle (Page 3987)

Setting the tab sequence order

Introduction

All operable objects can be reached in runtime with the <Tab> key. You use the "Tab sequence" command to define the order in which the operator can activate objects in runtime.

Note

You cannot reach objects with the "Output" or "Two states" mode in runtime with the <Tab> key.

You can operate the screen in runtime:

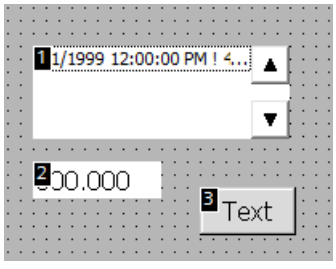
- Using the <Tab> key
- Using the mouse
- Using a configured hotkey

Requirement

- The active screen contains operable objects.
- No object is selected.
- The objects have been enabled for use in runtime, and have operator authorization.

Procedure

1. Select "Edit tab sequence" in the "Edit" menu.
   Tab sequence mode is activated. The tab sequence number is displayed for all objects that can be used. The tab sequence number is also displayed for hidden objects.
2. Use edit the tab sequence mode, click the accessible objects in the order in which you want them to be activated using <Tab> in runtime.
   The following figure shows how the tab sequence is defined in the screen. In runtime, the <Tab> key first activates the alarm view (number 1), then the I/O field (number 2), and then the button (number 3):



3. To exclude a screen object from the tab sequence, press the key combination <Shift+Ctrl> and click on the desired object.
   The tab sequence number is no longer displayed in the screen object. The screen object is now excluded from the tab sequence. The remaining tab sequence numbers are automatically decreased by 1.
4. To reenter an excluded screen object in the tab sequence, repeat step 3.
   The screen object entered as the first object in the tab sequence.

Result

The operator selects the objects in the specified order in Runtime with the <Tab> key.

See also

Overview of keyboard access (Page 3984)

Example: Configuring a rectangle (Page 3987)

12.1.2.32 Examples

Example: Configuring a rectangle

Task

In this example you configure a rectangle:

- Color = red
- Black frame 2 pixels wide
- Position = (20, 20)
- Size = (100,100)

Changing the color of the rectangle

To change the color of the rectangle:

1. Select the rectangle.
2. Define the background color in "Properties > Properties > Appearance > Background > Color" in the Inspector window.
3. Select "Solid" as the fill pattern.
4. Define the color for the border in "Properties > Properties > Appearance > Border > Color" in the Inspector window.
5. Enter the value "2" for "width".
6. Select "Solid" as the "Style".

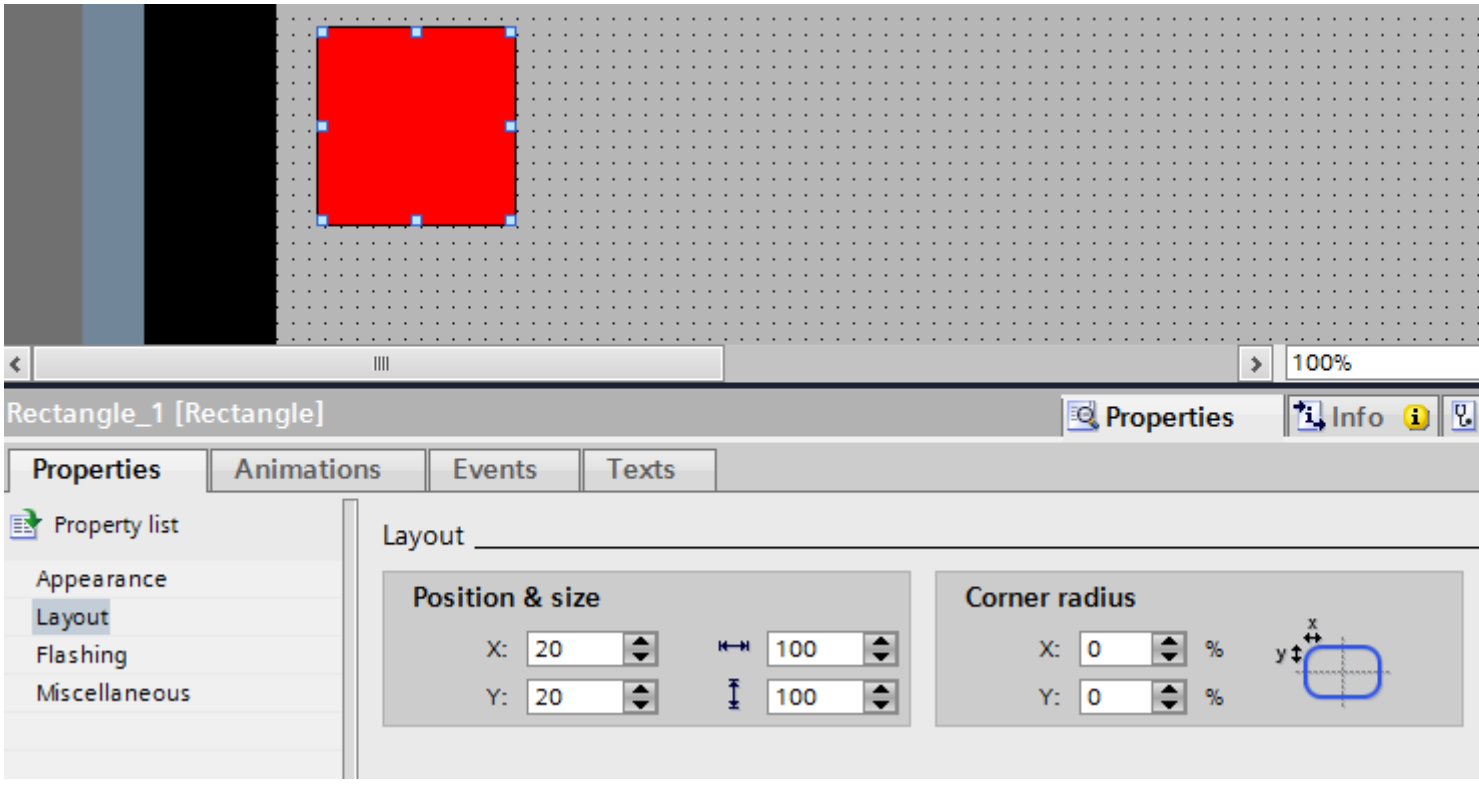
Interim result

The rectangle is red and has a black frame with a width of two pixels.

Repositioning and resizing the rectangle

To change the position and size of the rectangle:

1. Select the rectangle.
2. In the Inspector window, select "Properties > Properties > Layout".



3. Set "20" for the both the X and Y coordinates under "Position & Size".
4. Set "100" for the height and for the width.

Result

The rectangle is positioned at the coordinates (20, 20), and has a width and height of 100 pixels.

See also

Example: Inserting and configuring a rectangle (Page 3989)

Example: Inserting a rectangle (Page 3989)

Overview of objects (Page 3935)

Example: Inserting a rectangle

Task

In this example, you insert and rename a rectangle. Do not use the special characters ?, ", /, \, *, <, > for the name.

Requirement

- A screen is open.
- The Inspector window is open.
- The "Tools" task card is open.

Procedure

1. Click the "Basic objects" palette in the "Tools" task card.
2. Drag the "Rectangle" object into the screen.
3. In the Inspector window, select "Properties > Properties > Miscellaneous".
4. Type in the new name "MyRectangle".

Result

The rectangle is now inserted and named "MyRectangle". The rectangle has the default properties of the "rectangle" object.

See also

Example: Configuring a rectangle (Page 3987)

Example: Inserting and configuring a rectangle

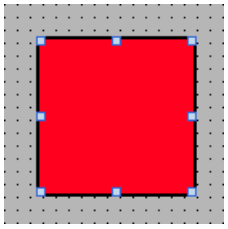
Task

In this example, you insert a rectangle in a screen. You can configure the following properties:

- Name = "MyRectangle"
- Position = (20, 20)
- Size = (100,100)
- Color = red
- Black frame 2 pixels wide

Principle

The rectangle is a closed object which can be filled with a color or pattern. The height and width of a rectangle can be adjusted to allow its horizontal and vertical alignment.



Overview

Carry out the following steps in order to create a rectangle:

- Inserting a rectangle
- Configuring a rectangle

See also

Example: Configuring a rectangle (Page 3987)

12.1.3 Working with text lists and graphics lists

12.1.3.1 Working with text lists

Basics on text lists

Introduction

Texts are assigned to the values of a tag in a text list. Assign the text list to a symbolic I/O field for example in the configuration. This supplies the text to be displayed to the object. The text lists are created in the ""Text List" editor. You configure the interface between the text list and a tag at the object that uses the text list.

The selection of objects that can have a text list assigned depends on the Runtime.

Application

The text list is used, for example, to display a drop-down list in a symbolic I/O field.

If the symbolic I/O field is a display field, the associated texts will differ according to the value of the configured tags. If the symbolic I/O field is an input field, the configured tag assumes the associated value when the operator selects the corresponding text in Runtime.

Note

Display of tag values without text

The display of tag values to which no text has been assigned depends on the Runtime:

- The display and operating element remains empty.
- Three asterisks *** are displayed.

Ranges for the text list

Three types are available for the text lists:

- Value/Range
  This setting assigns text entries from the text list to integer values or value ranges of a tag. You can select the number of text entries as needed. The maximum number of entries depends on the HMI device you are using.
  You specify a default value which is shown if the value of the tag lies outside the defined range.
- Bit (0, 1)
  This setting assigns text entries from the text list to two states of a binary tag. You can create a text entry for each state of the binary tag.
- Bit number (0 - 31)
  This setting assigns a text entry from the text list to each bit of a tag. The maximum number of text entries is 32. This form of text list can be used, for example, in a sequence control when processing a sequence in which only one bit of the used tag may be set. You influence the behavior of the bit number (0 - 31) with the set bit of the least significance and a default value.

Multilingual texts

You can configure multiple languages for the texts in a text list. The texts will then be displayed in the set language in Runtime. To this purpose you set the languages in the Project window under "Language support > Project languages."

Configuration steps

The following steps are necessary to display texts in a symbolic I/O field for example:

1. Creating the text list
2. Assignment of the texts to values or value ranges of a text list
3. Assignment of a text list in the display object, for example, to the symbolic I/O field

See also

Creating a text list (Page 3992)

Assigning texts and values to an area text list (Page 3993)

Assigning texts and values to a bit text list (Page 3995)

Assigning texts and values to a bit number text list (Page 3996)

Configuring objects with a text list (Page 3999)

Screen basics (Page 3889)

Creating a text list

Introduction

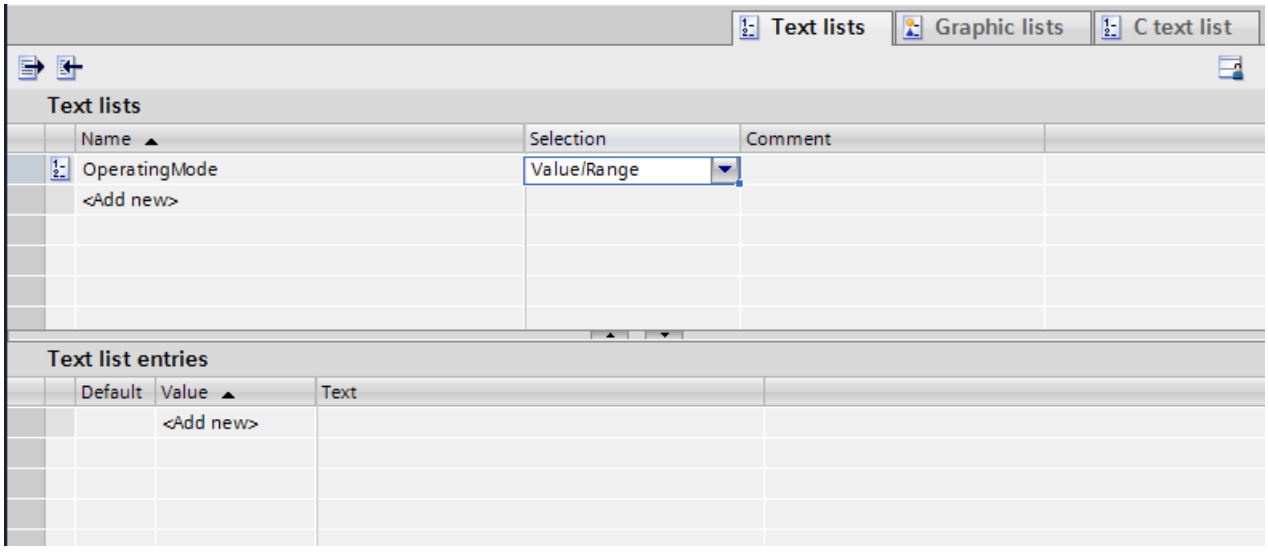
The text list allows you to assign specific texts to values and to output these in Runtime, for example in a symbolic I/O field. The type of symbolic I/O field can be specified, for example as a pure input field.

The following types of list are available:

- Value/Range
- Bit
- Bit Number

Procedure

1. Double-click "Text and graphics lists" in the project window.
2. Open the "Text lists" tab.



3. Click "Add" in the "Text lists" table.
   The Inspector window of the text list is open.

4. Assign a name to the text list that indicates its function.
5. Select the text list type under "Selection":
   - Value/Range: Text from the text list is displayed when the tag has a value that lies within the specified range.
   - Bit (0,1): A text from the text list is displayed when the tag has the value 0. A different text from the text list is displayed when the tag has the value 1.
   - Bit number (0-31): Text from the text list is displayed when the tag has the value of the assigned bit number.
6. Enter a comment for the text list.

Note

You should not use semicolons in the texts in a text list in WinCC Runtime Professional. The semicolon is a control character and is automatically deleted from a text.

Result

A text list is created.

See also

Basics on text lists (Page 3990)

Assigning texts and values to an area text list

Introduction

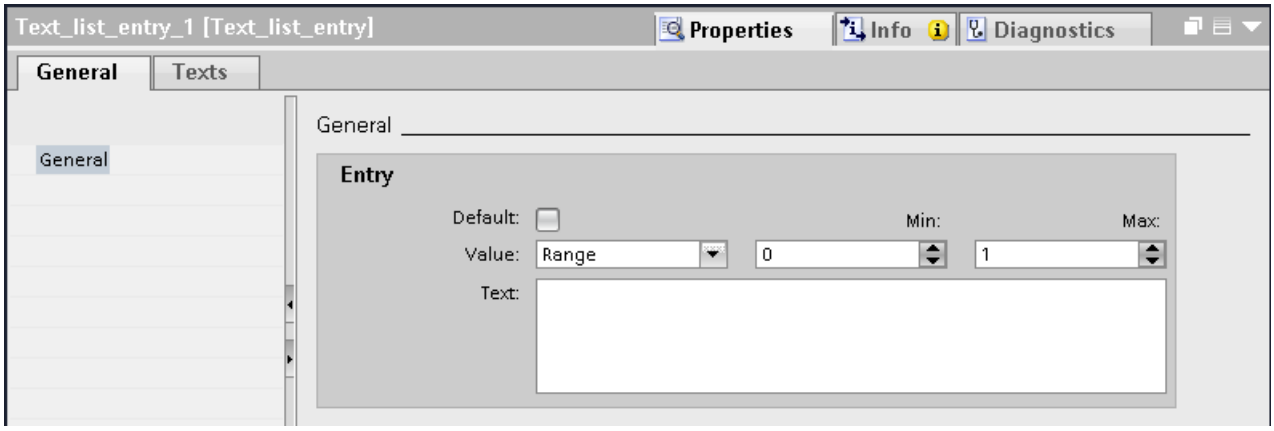
For each area text list you specify which texts are displayed at which value range.

Requirement

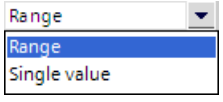
- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- An area text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.
   The Inspector window for this list entry opens.



2. Select the setting "Range" in "Properties > Properties > General > Value" in the Inspector window.



   – Enter the value "1" for "Min" for example.
   – Enter the value "20" for "Max" for example.
   – For "Text", enter the text that is displayed in Runtime if the tag is within the specified value range.

   Note

   Use a maximum of 320 characters for the text.

3. Click "Add" in the "Text list entries" table. A second list entry is created.
4. Select the setting "Range" in "Properties > Properties > General > Value" in the Inspector window.
   – Enter the value "21" for "Min" for example.
   – Enter the value "40" for "Max" for example.
   – For "Text", enter the text that is displayed in Runtime when the tag is within the specified value range.
5. If required, activate the "default entry".
   The entered text is always displayed when the tag has an undefined value. Only one default entry is possible per list.

Result

An area text list is created. Texts have been assigned to the possible value ranges.

See also

Basics on text lists (Page 3990)

Notes for bit number text list (Page 3998)

Assigning texts and values to a bit text list

Introduction

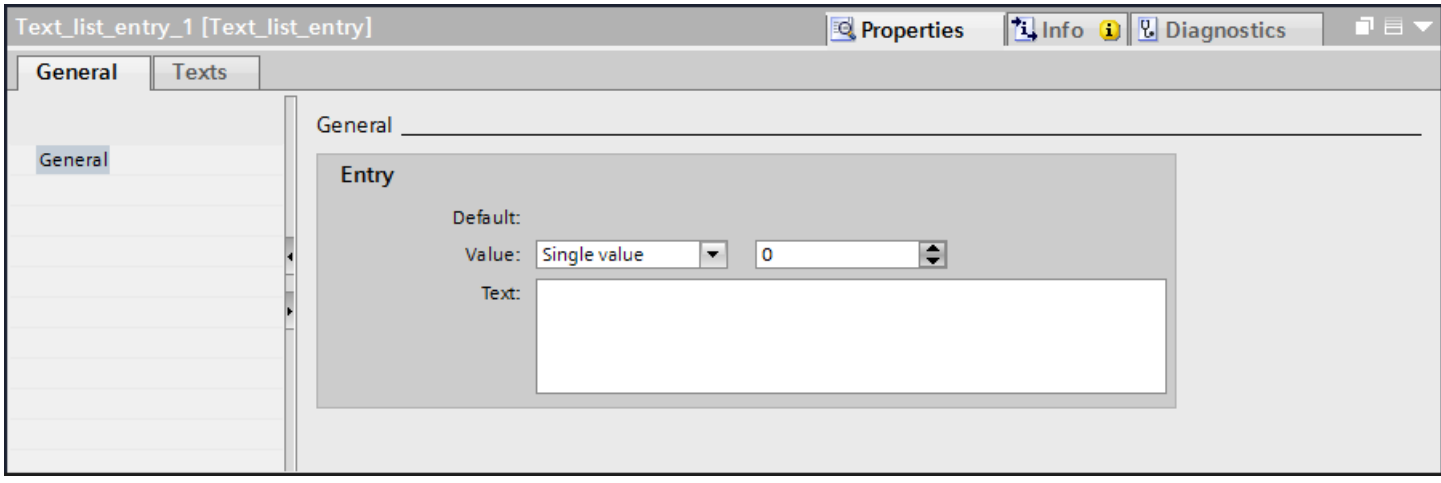
For each text list, you specify which text is displayed at which bit value.

Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- A bit text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.
   The Inspector window for this list entry opens.



2. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "0" for "Value."
   - Enter the text which is displayed in Runtime under "Text" if the bit tag is set to "0".
3. Click "Add" in the "Text list entries" table. A second list entry is created.
4. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "1" under "Value."
   - Enter the text which is to be displayed in Runtime under "Text" if the bit tag is set to "1".

Note

Use a maximum of 320 characters for the text.

Use a maximum of 255 characters and no semicolons for the text in WinCC Runtime Professional.

Result

A bit text list is created. Texts that appear in Runtime are assigned to the possible values "0" and "1".

See also

Basics on text lists (Page 3990)

Assigning texts and values to a bit number text list

Introduction

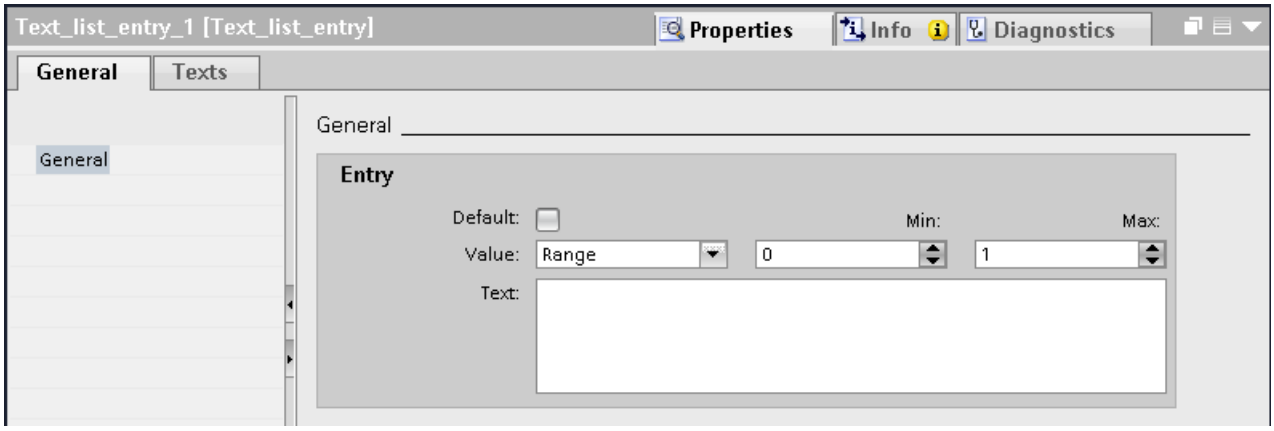
For each bit number text list you specify which texts are displayed at which bit number.

Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- A bit number text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.
   The Inspector window for this list entry opens.



2. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "10", for example, for "Value".
   - Under "Text", enter the text that is displayed in Runtime when the tag has the value "10".
3. If required, activate the "default entry".
   The entered text is always displayed when the tag has an undefined value. Only one default entry is possible per list.
4. Create further list entries for additional bit numbers of the same text list.

Note

Use a maximum of 320 characters for the text.

Use a maximum of 255 characters and no semicolons for the text in WinCC Runtime Professional.

Note

Bit selection for text lists

The output of the text list depends on the "Bit selection for text and graphics lists" option in the Runtime settings. Use this option to specify that the bit selection for text lists should be used on your HMI device. When you enable the option, the text displayed is the one configured for the set bit with the least significance. When you disable this option, only the text that has been configured for the set bit is displayed .

Result

A bit number text list is created. Texts that appear in Runtime are assigned to the specified bit numbers.

See also

Basics on text lists (Page 3990)

Notes for bit number text list

Introduction

The bit number (0 - 31) range assigns a text entry from the list to each bit of a tag. If the option "Bit selection for text and graphic lists" is disabled in the Runtime settings and no default value is set, the following standard behavior applies:

- If only 1 bit is configured of all set bits, the stored text is displayed for the configured bit. In the following example, only the set bit with significance "4" is configured. Text 2 is displayed.

Significance	7	6	5	4	3	2	1	0
Set bits	0	0	1	1	0	1	0	0
Configured	-	Text 3	-	Text 2	Text 1	-	-	-

- If no bit is set or when several configured bits are set, no text is displayed.

Default value

Define a default value to prevent an empty display. A configured default value is displayed in the following cases:

- The option "Bit selection for text and graphic lists" is disabled. No bit or several configured bits are set in the tag.
- The option "Bit selection for text and graphic lists" is enabled and no bit is set or a text is not configured for the set bit with the least significance.

Displaying the default value

1. Enable the text for the default entry in the "Default" column of the "Text list entries" table. The value "Default entry" appears in the "Value" column of the text entry.
2. You can also select the "Default" option under "Properties > General" in the inspector window.

Set bit with the least significance

If no text is configured for the set bit with the least significance and if no

default value is set, nothing is displayed. If a default value is configured, the default value is displayed.

If no text is configured for the set bit with the least significance and if no default value is set, nothing is displayed. If a default value is configured, the default value is displayed.

To only display the text for the set bit with the least significance, enable the "Bit selection for text and graphic lists" option in the "Runtime settings" editor.

This setting is deselected by default to maintain downward compatibility. The setting is valid for all text lists of the HMI device.

Multiline text list entries

Use the <SHIFT>+<RETURN> shortcut to enter a line break in the text entry. Line breaks are represented by the "¶" paragraph mark.

Multiline text list entries are only output in symbolic output fields as well as on buttons with multiple lines. In all other cases, multiline texts are displayed with the paragraph mark.

Configuring objects with a text list

Introduction

The output value and value application for text lists are specified in the display and operating object that displays the texts of the text list in Runtime. The properties of these objects are configured as required.

Requirement

- A text list is created.
- You have created a tag.
- The "Screens" editor is open.
- A screen with a symbolic I/O field is open. The object is edited.

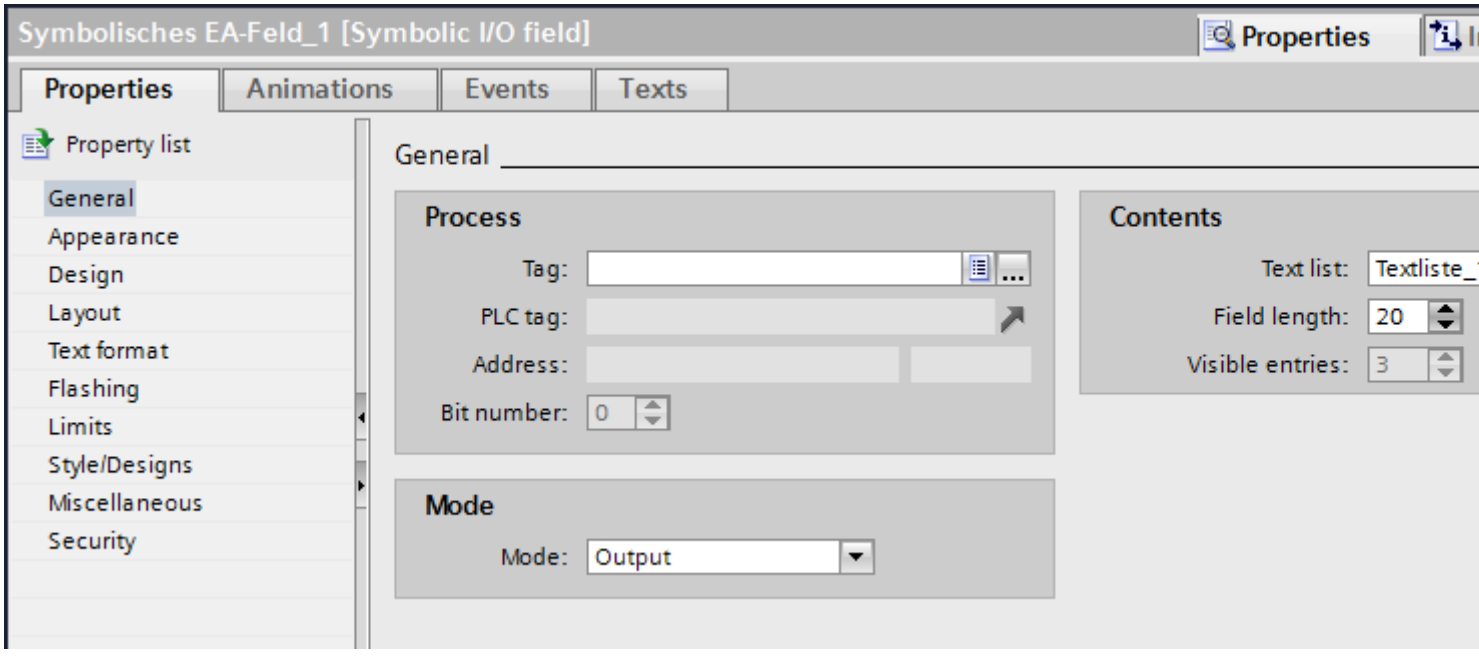
Procedure

1. Select the text list which you want to have displayed in Runtime in "Properties > Properties > General > Text list" in the Inspection window.
2. Select the setting "Output" as the "Mode".

   Note
   Runtime dependency

   Different field types are available for a symbolic I/O field depending on the Runtime.

3. Select the tag the value of which determines the display in the symbolic I/O field as "Tag".



Result

The defined texts of the text list are displayed in the symbolic I/O field in Runtime when the tag has the specified value.

See also

Basics on text lists (Page 3990)

12.1.3.2 Working with graphics lists

Basic principles of graphics lists

Introduction

The possible values of a tag are assigned to specific graphics in a graphics list. During configuration, you can create a graphics list for a button or a graphic I/O field. This supplies the graphics to be displayed to the object.

The graphics lists are created with the "Text and graphics list" editor. You configure the interface between the graphics list and a tag at the object that uses the graphics list. The availability of the graphics list is determined by the HMI device used.

Application

You can configure the graphics list for the following situations:

- Drop-down list with a graphic I/O field
- State-specific graphic for a button

The graphics in a graphics list can be configured as multilingual. The graphics will then be displayed in the set language in Runtime.

Graphic sources

Graphics can be added to the graphics list from the following sources:

- By selecting from a graphic browser
- By selecting an existing file
  You can use the following file types:
  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tiff, *.png, *.jpeg and *.jpg.
- By creating a new file

Function

If the graphic I/O field is a display field, the associated graphics will differ according to the value of the configured tags. If the graphic I/O field is an input field, the configured tag assumes the associated value when the operator selects a graphic in Runtime.

Ranges for the graphics list

Three types are available for the graphics lists:

- Value/Range
  This setting assigns graphic entries from the graphics list to integer values or value ranges of a tag. You can select the number of graphic entries as needed. The maximum number of entries depends on the HMI device you are using.
  You specify a default value which is shown if the value of the tag lies outside the defined range.
- Bit (0, 1)
  This setting assigns graphic entries from the graphics list to two states of a binary tag. You can create a graphic entry for each state of the binary tag.
- Bit number (0 - 31)
  This setting assigns a graphic entry from the graphics list to each bit of a tag. The maximum number of graphic entries is 32. This form of graphics list can be used, for example, in a sequence control when processing a sequence in which only one bit of the used tag may be set. You influence the behavior of the bit number (0 - 31) with the set bit of the least significance and a default value.

Configuration steps

The following tasks are required to display graphics, for example, in a graphic I/O field:

1. Creating the graphics list
2. Assignment of the graphics to values or value ranges of a graphics list
3. Assignment of a graphics list in the display object, for example, the graphic I/O field

See also

Creating a graphics list (Page 4002)

Assigning a graphic and values to an area graphics list (Page 4004)

Assigning graphics and values to a bit graphics list (Page 4006)

Assigning graphics and values to a bit number graphics list (Page 4007)

Configuring objects with a graphics list (Page 4010)

Screen basics (Page 3889)

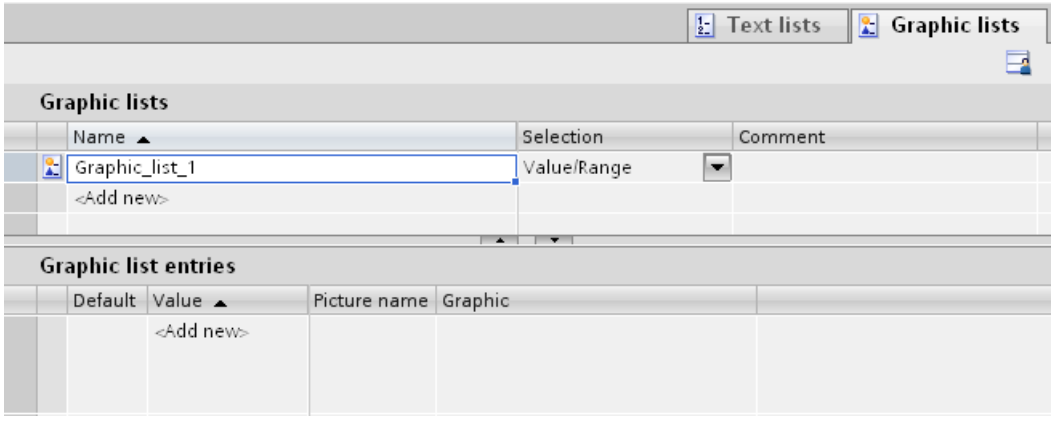
Creating a graphics list

Introduction

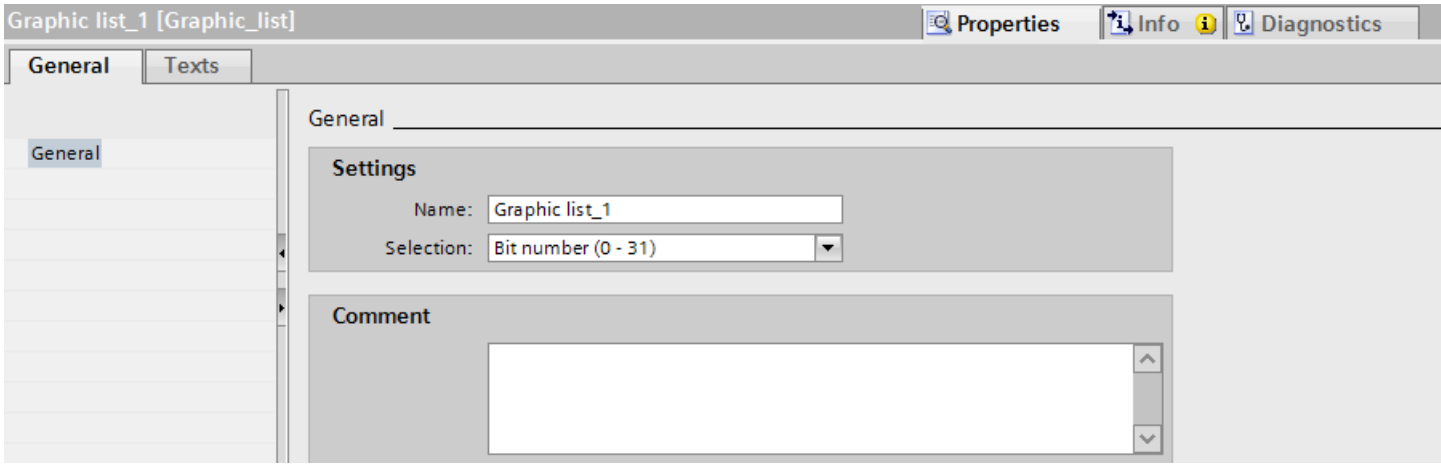
The graphics list allows you to assign specific graphics to variable values and to output these in a graphic IO field in Runtime. You can specify the type of graphic I/O field, for example as a pure output field.

Procedure

1. Double-click "Text and graphics lists" in the project window.
2. Open the "Graphics lists" tab.



3. Click "Add" in the "Graphics lists" table. The Inspector window of the graphics list will open up.



4. Assign a name to the graphics list that indicates its function.
5. Select the graphics list type "Bit number (0 - 31)" for example under "Select".
6. Enter a comment for the graphics list.

Result

A graphics list of the type "Range (0 - 31)" is created.

See also

Basic principles of graphics lists (Page 4001)

Assigning a graphic and values to an area graphics list

Introduction

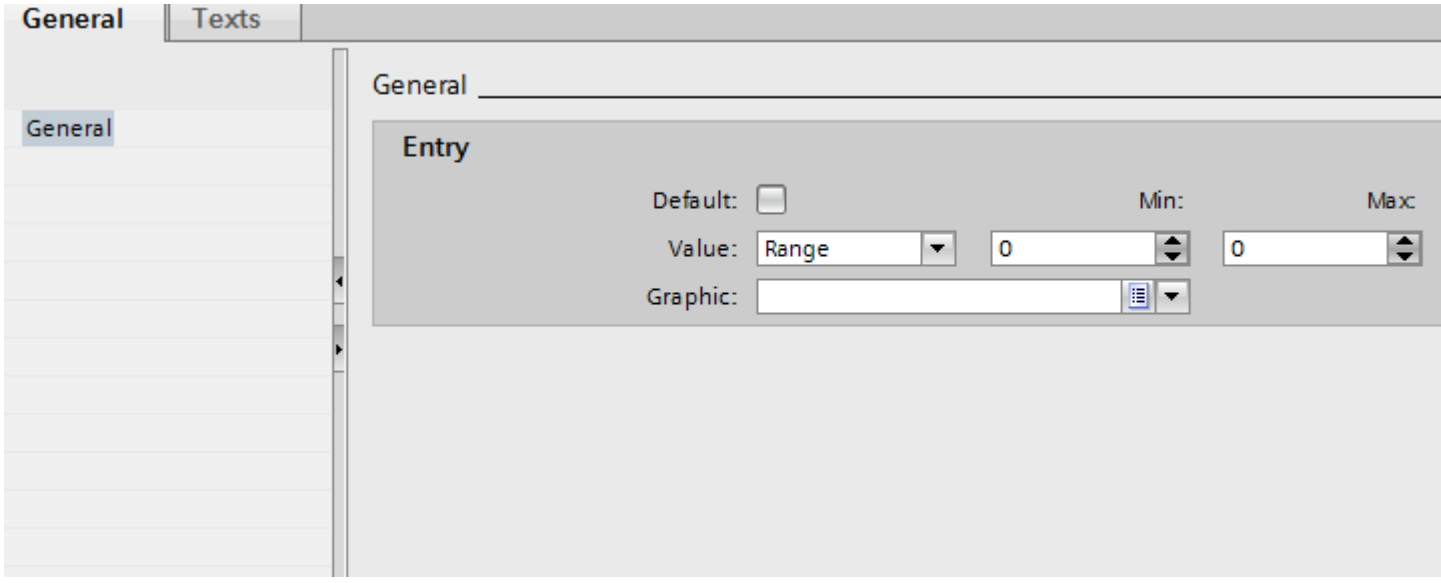
For each area graphics list you specify which graphics are displayed at which value range.

Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is open.
- An area graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.
   The Inspector window for this list entry opens.



2. Select the settings "Range" in "Properties > Properties > General > Value" in the Inspector window:
   - Enter the value "1" for "Min" for example.
   - Enter the value "20" for "Max" for example.
   - Select a graphic that is displayed in Runtime when the tag is within the specified value range.

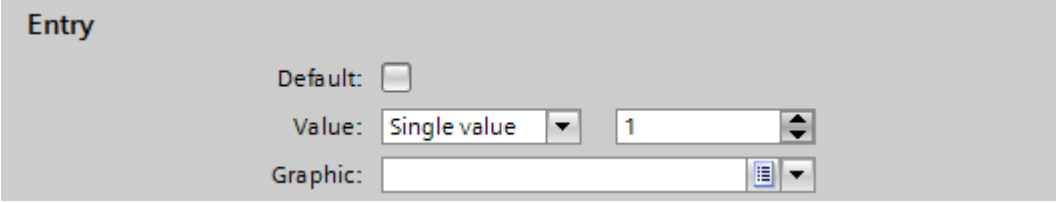
Note

As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:

1. Select a graphic in the library or in your file system.
2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. Click "Add" in the "Graphics list entries" table. A further list entry is created.

4. Select the settings "Single value" in "Properties > Properties > General > Value" in the Inspector window:
   - Enter the value "21" for example.
   - Select a graphic which is displayed in Runtime if the bit "21" is set in the tag.



5. If required, activate the "default entry".
   The graphic is always displayed when the tag has an undefined value. Only one default entry is possible per list.

Result

An area graphics list is created. Graphics that appear in Runtime are assigned to the possible values.

See also

Basic principles of graphics lists (Page 4001)

Assigning graphics and values to a bit graphics list

Introduction

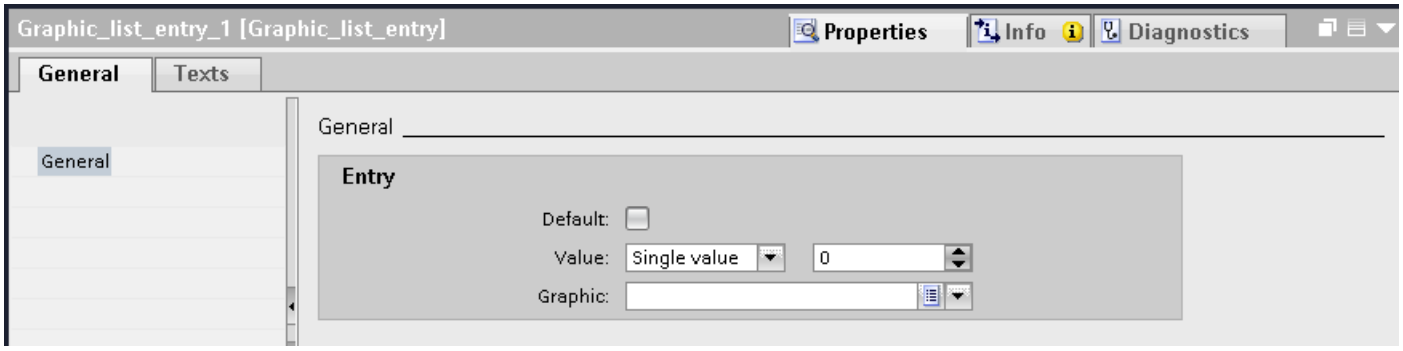
For each graphics list you specify which graphic is displayed at which bit value.

Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is opened.
- A bit graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.
   The Inspector window for this list entry opens.



2. Select the settings "Single value" in the inspector window "Properties > Properties > General > Value":
   - Enter "0" as the value.
   - Select a graphic which is displayed in Runtime if the bit "0" is set in the tag.

   Note

   As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:
   1. Select a graphic in the library or in your file system.
   2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.
3. Click "Add" in the "Graphics list entries" table. A new list entry is created.
4. Select "Properties > Properties > General > Value > Single value": in the Inspector window.
   - Enter "1" as the value.
   - Select a graphic which is displayed in Runtime if the bit "1" is set in the tag.

Result

A bit graphics list is created. Graphics that appear in Runtime are assigned to the values "0" and "1".

See also

Basic principles of graphics lists (Page 4001)

Assigning graphics and values to a bit number graphics list

Introduction

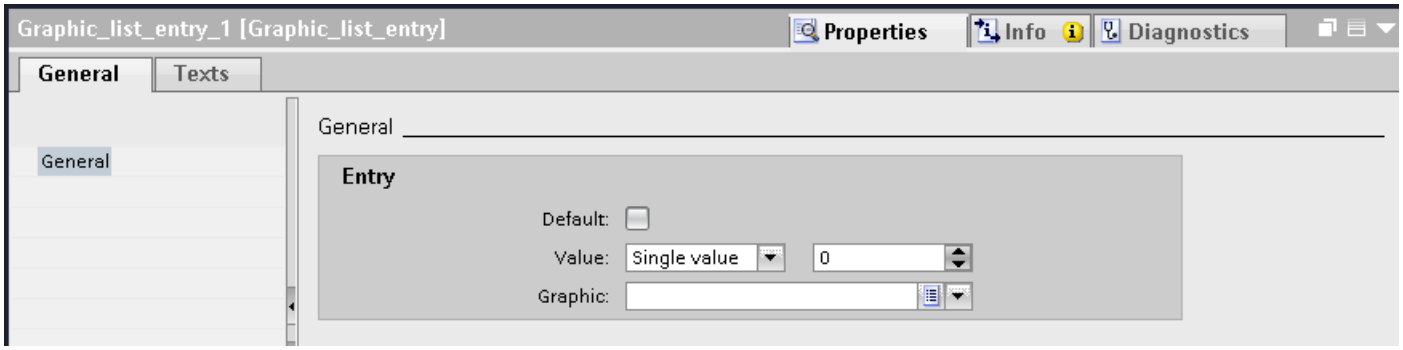
For each bit number graphics list you specify which graphics are displayed at which bit number.

Requirement

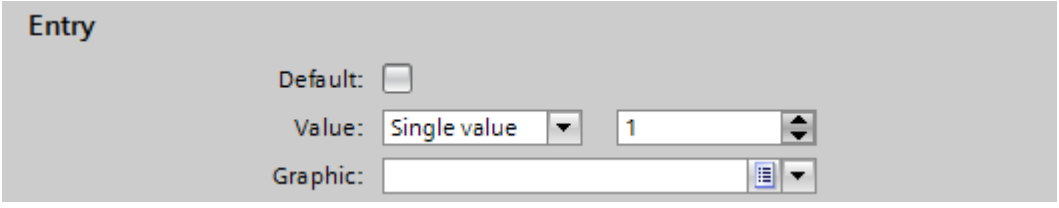
- The "Text and graphics list" editor is open.
- The "Graphics list" tab is open.
- A bit number graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.
   The Inspector window for this list entry opens.



2. Select the settings "Single value" in the Inspector window "Properties > Properties > General > Value":
   - Enter the value "1" for example.
   - Select a graphic which is displayed in Runtime if the bit "0" is set in the tag.



Note

As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:

1. Select a graphic in the library or in your file system.
2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. If required, activate the "default entry".
   The graphic is always displayed when the tag has an undefined value. Only one default entry is possible per list.
4. Create further list entries for additional bit numbers of the same graphics list.

Note

Bit selection for graphic lists

The output of the graphic list depends on the "Bit selection for text and graphics lists" option in the Runtime settings. Use this option to specify that the bit selection for graphic lists should be used on your HMI device. When you enable the option, the graphic displayed is the one configured for the set bit with the least significance. When you disable this option, the graphic that has been configured only for the set bit is displayed.

Result

A bit number graphics list is created. Graphics that appear in Runtime are assigned to the specified bit numbers.

See also

Basic principles of graphics lists (Page 4001)

Notes for bit number graphics list

Introduction

The bit number (0 - 31) range assigns a graphic entry from the list to each bit of a tag. If the option "Bit selection for text and graphic lists" is disabled in the Runtime settings and no default value is set, the following standard behavior applies:

- If only 1 bit is configured of all set bits, the stored graphic is displayed for the configured bit. In the following example, only the set bit with significance "4" is configured. Graphic 2 is displayed.

Significance	7	6	5	4	3	2	1	0
Set bits	0	0	1	1	0	1	0	0
Configured	-	Graphic 3	-	Graphic 2	Graphic 1	-	-	-

- If no bit is set or when several bits are set that are also configured, only the cactus graphic is displayed.

Default value

Define a default value to prevent an empty display. A configured default value is displayed in the following cases:

- The option "Bit selection for text and graphic lists" is disabled. No bit or several configured bits are set in the tag.
- The option "Bit selection for text and graphic lists" is enabled and no bit is set or a graphic is not configured for the set bit with the least significance.

Displaying the default value

1. Enable the graphic for the default entry in the "Default" column of the "Graphics list entries" table.
   The value "Default entry" appears in the "Value" column of the entry.
2. You can also select the "Default" option under "Properties > General" in the inspector window.

Set bit with the least significance

When "Bit selection for text and graphic lists" is enabled, the graphic displayed is the one configured for the set bit with the least significance.

If no graphic is configured for the set bit with the least significance and if no default value is set, the cactus graphic is displayed. If a default value is configured, the graphic configured for the default value is displayed.

To only display the graphic for the set bit with the least significance, enable the "Bit selection for text and graphic lists" option in the "Runtime settings" editor.

This setting is deselected by default to maintain downward compatibility. The setting is valid for all graphic lists of the HMI device.

Configuring objects with a graphics list

Introduction

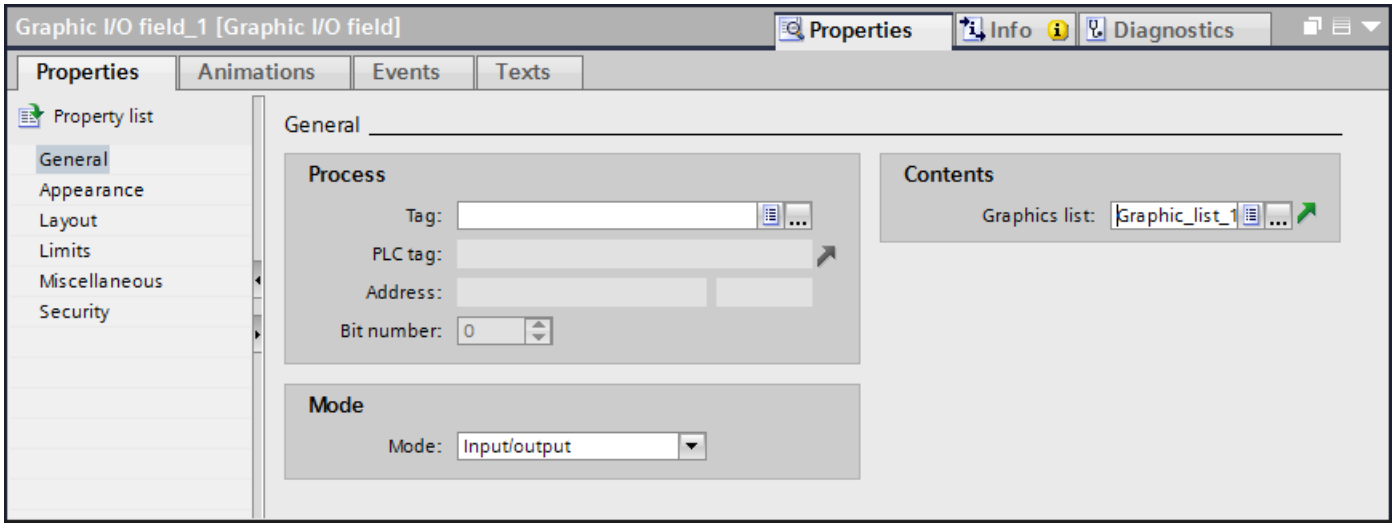
The output value and value application for graphics list are specified in the display and operating object that displays the graphics of the graphics list in Runtime. The properties of these objects are configured as required.

Requirement

- A graphics list is created. The values have been defined. Graphics have been assigned to the values.
- You have created a tag.
- The "Screens" editor is open.
- A screen with a graphics I/O field is open. The object is edited.

Procedure

1. Select the graphics list the graphics of which you want to have displayed in Runtime in "Properties > Properties > General > Graphics list" in the Inspector window.
2. Select the "Input/output" setting for "Mode".



Note

Runtime dependency

Different field types are available for a graphic I/O field depending on the Runtime.

3. As "Tag", select the tag whose values are defined by the display in the graphic I/O field.

Result

The defined graphics of the graphics list are displayed in the graphic I/O field in Runtime when the tag has the specified value.

See also

Basic principles of graphics lists (Page 4001)

12.1.4 Dynamizing screens

12.1.4.1 Basics on dynamizing screens

Dynamizing objects

In WinCC you dynamize objects to map your system and show processes on HMI devices.

You implement dynamizations by

- Animations
- Tags
- System functions

One example is the mapping of a tank, the liquid level of which rises or falls in relation to a process value.

The options for dynamization depend on the object involved. When you copy an object, its dynamization functions are included.

See also

Configuring a new animation (Page 4018)

System functions (Page 4570)

User-defined functions (Page 4572)

Runtime scripting (Page 4569)

Basics on dynamizing screens (Page 4012)

Screen basics (Page 3889)

Basics on dynamization in the property list (Page 4030)

Dynamization in the inspector window (Page 4013)

Dynamization in the inspector window (Page 4015)

12.1.4.2 Basics on dynamizing screens

Dynamizing objects

In WinCC you dynamize objects to map your system and show processes on HMI devices.

You implement dynamizations by

- Animations
- System functions
- Tags
- Local scripts

One example is the mapping of a tank, the liquid level of which rises or falls in relation to a process value.

The options for dynamization depend on the object involved. When you copy an object, its dynamization functions are included.

See also

Basics on dynamizing screens (Page 4012)

12.1.4.3 Dynamization in the inspector window

Introduction

Basically, you can dynamize all the screen objects which you have configured in a screen. Which dynamization possibilities and which events are available depends on the device and the selected object.

Animations

WinCC helps you to implement dynamization using predefined animations. If you want to animate an object, first configure the desired animation in the object's inspector window. Then adapt the animation to the requirements of your project.

The selection of the supported animations depends on the HMI device and the selected object. You choose between the following types of animation:

- Layout: Appearance, visibility
- Movements: direct, diagonal, horizontal and vertical movement
- Variable binding

You can configure the "Variable binding" type of animation several times for one object.

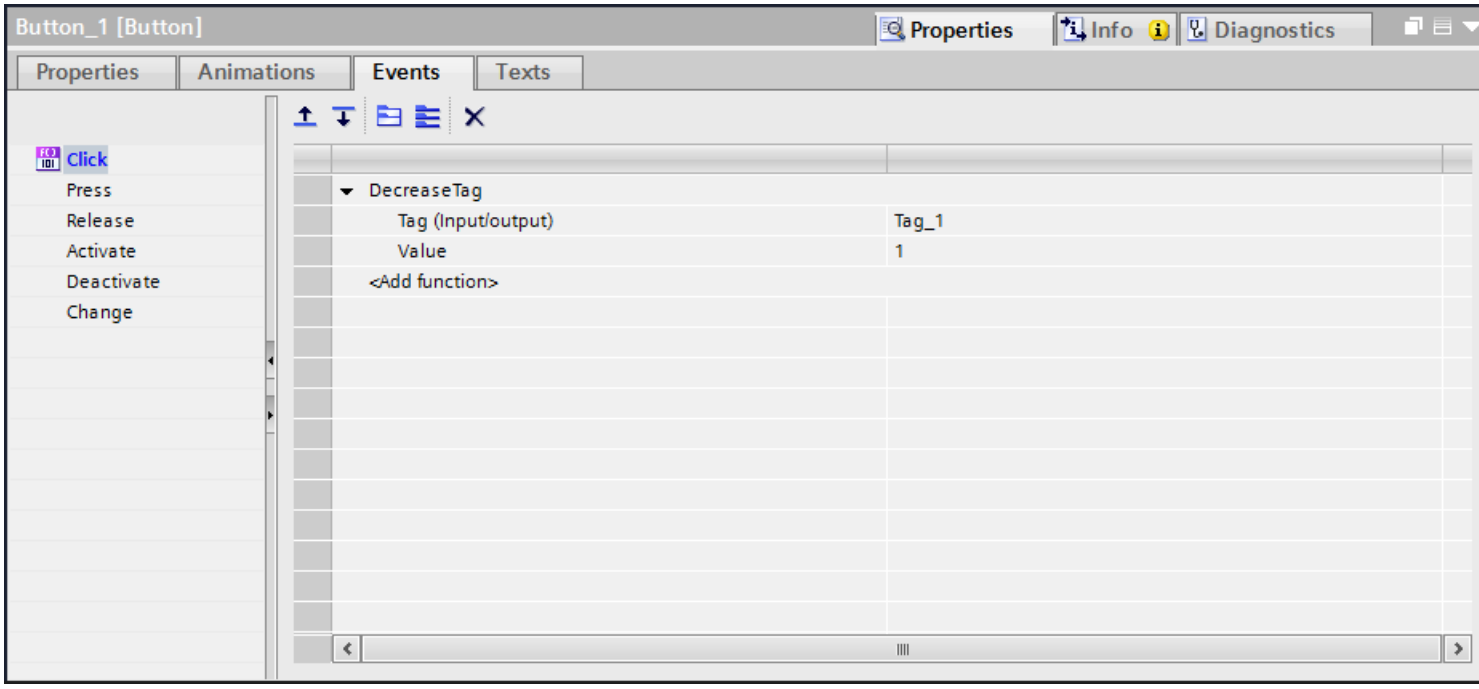
You configure animations in the "Properties > Animations" inspector window.

Events

Operable objects also react to events, such as a mouse click.

You configure a function list with system functions on an event. The system functions are processed as a reaction to the triggered event.

You configure events in the "Properties > Events" inspector window.



You will find further information in "Working with function lists".

See also

Basics on dynamizing screens (Page 4012)

12.1.4.4 Dynamization in the inspector window

Introduction

Basically, you can dynamize all the screen objects which you have configured in a screen. Which dynamization possibilities and which events are available depends on the device and the selected object.

The inspector window offers you the possibilities for dynamization of screen objects in three tabs.

Property list

With the button you go to the properties list in the "Properties > Properties" inspector window.

In the property list, you set any dynamization for the individual object properties using tags or local C and VB scripts. You can, for example, dynamically change the background color of an object.

In the "Dynamic Value" column you can dynamize the object property of a tag, a C-script or a VB-script.

Whether an object property can be dynamized is indicated in the Property list by means of the background color:

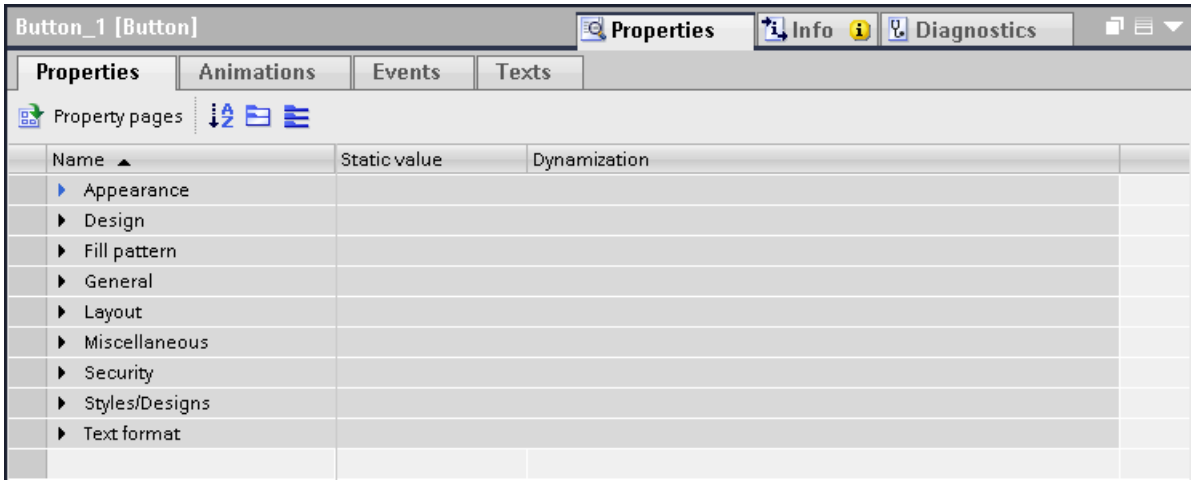
- Light gray: Can be dynamized
- Dark gray: Cannot be dynamized

Any animations you have already configured will also appear in the object property list, where you can edit the animations.

The following table shows you how to sort the property list:

	Sort by alphabet
	Sort by category

With the button you return to the property pages in the inspector window.



Animations

You have two different options for animating objects.

You configure animations

- in the inspector window under "Properties > Animations",
- in the "Animations" task card.

WinCC helps you to implement dynamization using predefined animations. If you want to animate an object, first configure the desired animation with the tools of the toolbox or in the object's inspector window. Then customize the animation in the Inspector window to suit the requirements of your project.

The selection of the supported animations depends on the device and the selected object. You choose between the following types of animation:

- Tag binding
- Layout: Appearance, control enable, visibility
- Movements: direct, diagonal, horizontal and vertical movement
- Property animation

Note

HMI device dependency

The "Control enable" animation is not available for every HMI device.

You can configure the "Tag binding" and "Property animation" types of animation several times for one object.

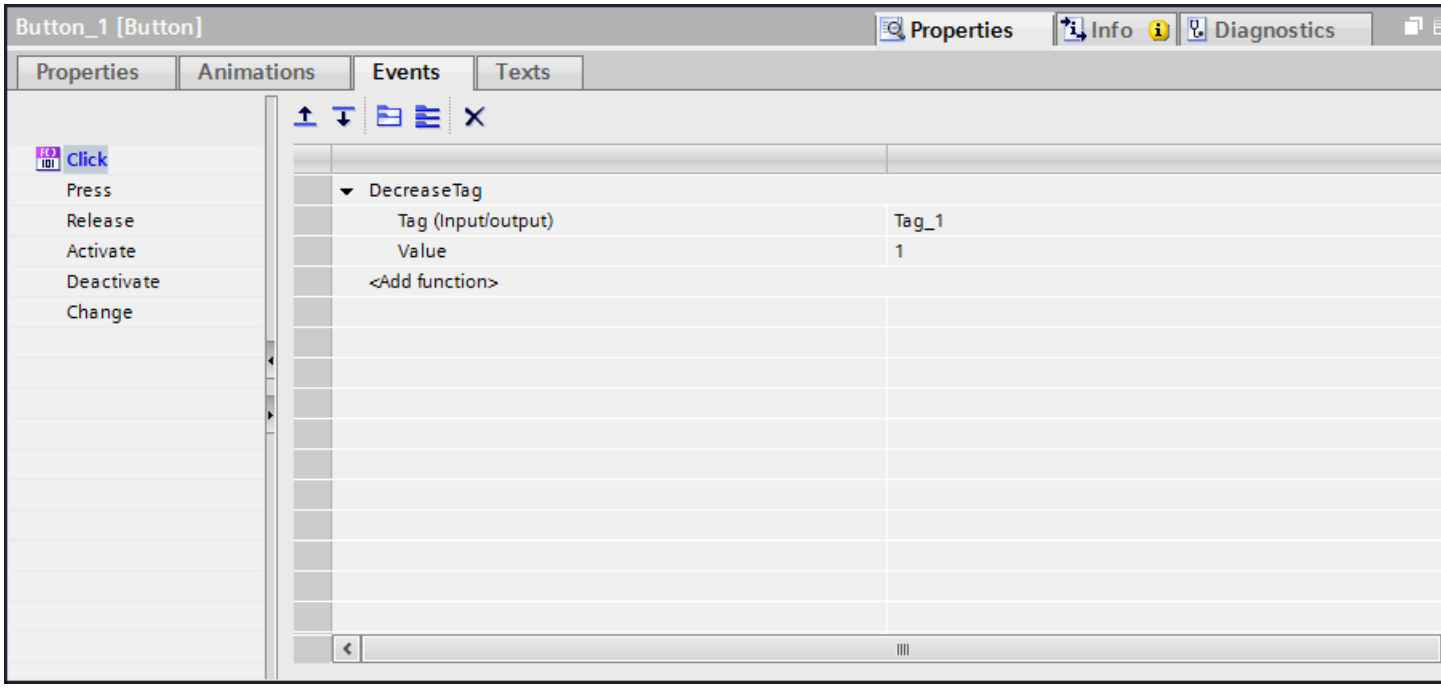
You configure animations in the "Properties > Animations" inspector window.

Events

Operator control enabled objects also react to events, such as a mouse click.

You configure a function list with system functions or local scripts on an event. The system functions or the local script are processed as a reaction to the triggered event.

You configure events in the "Properties > Events" inspector window.



You will find further information in "Working with function lists".

See also

Basics on dynamizing screens (Page 4012)

12.1.4.5 Dynamization with animations

Configuring a new animation

Introduction

Use predefined animations to dynamize screen objects.

Requirement

- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure in the inspector window

1. In the Inspector window, select "Properties > Animations".
2. Select the animation you want to use.
3. Click the button.

Procedure in the "Animations" task card.

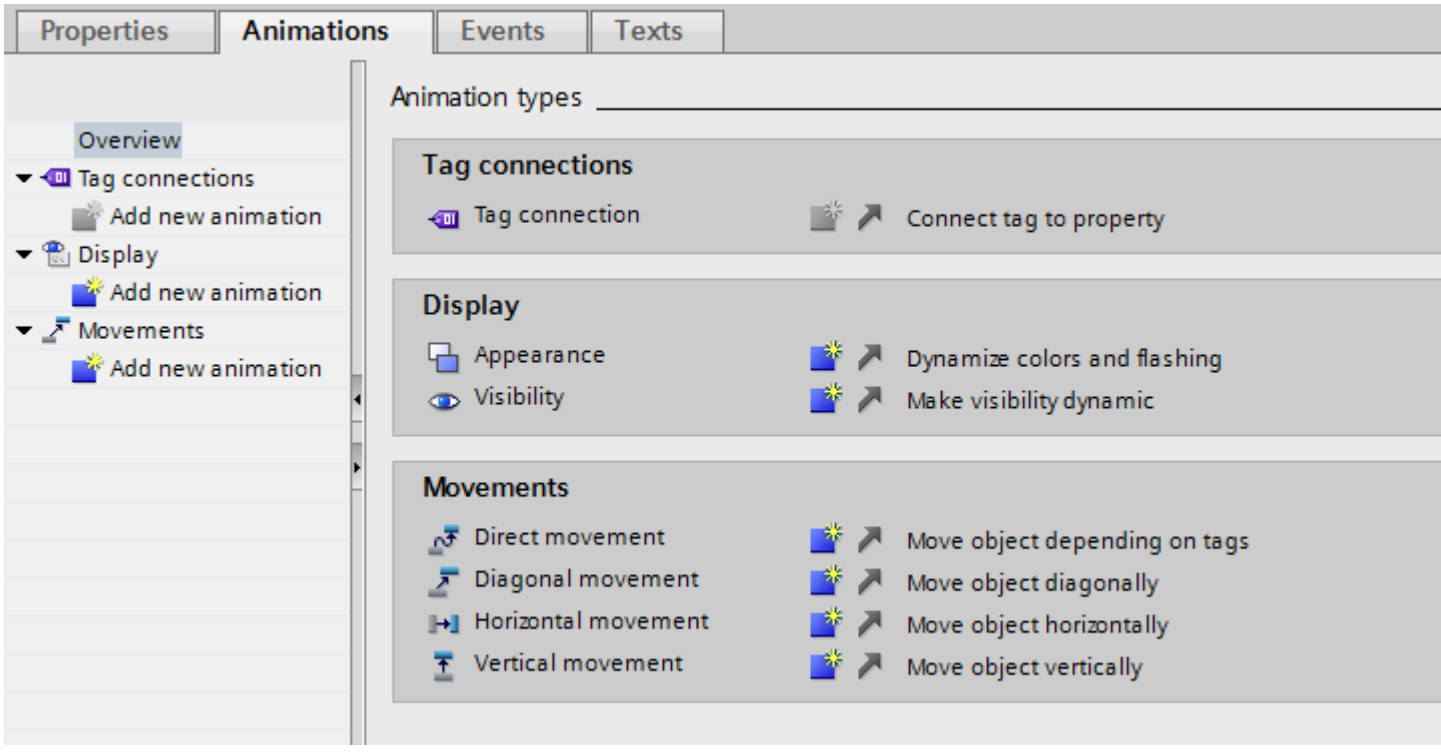
1. Open the object group containing the required animation in the "Animations" task card.
2. Drag the animation onto the object that you want to make dynamic.

Alternatively you select the object in the screen and double click the desired animation in the "Animation" task card.

Result

The animation appears in the Inspector window of the object. You configure the animation in the following steps.

In the animations overview the green arrow indicates which animation is already configured. The configured animation opens in the inspector window when you click the green arrow.



See also

Basics on dynamizing screens (Page 4012)

Animations in multiple selection (Page 4030)

Animations of object groups (Page 4029)

Animations of object groups (Page 4028)

Configuring animation with tag connection (Page 4026)

Dynamizing the visibility of an object (Page 4025)

Dynamization of the control enable state of an operating element (Page 4024)

Configuring direct movement (Page 4022)

Configuring movements (Page 4021)

Dynamizing the appearance of an object (Page 4019)

Dynamizing the appearance of an object

Introduction

The appearance of a screen object is controlled by changing the value of a tag in runtime. When the tag adopts a certain value, the screen object changes its color or flashing characteristics according to the configuration.

Type

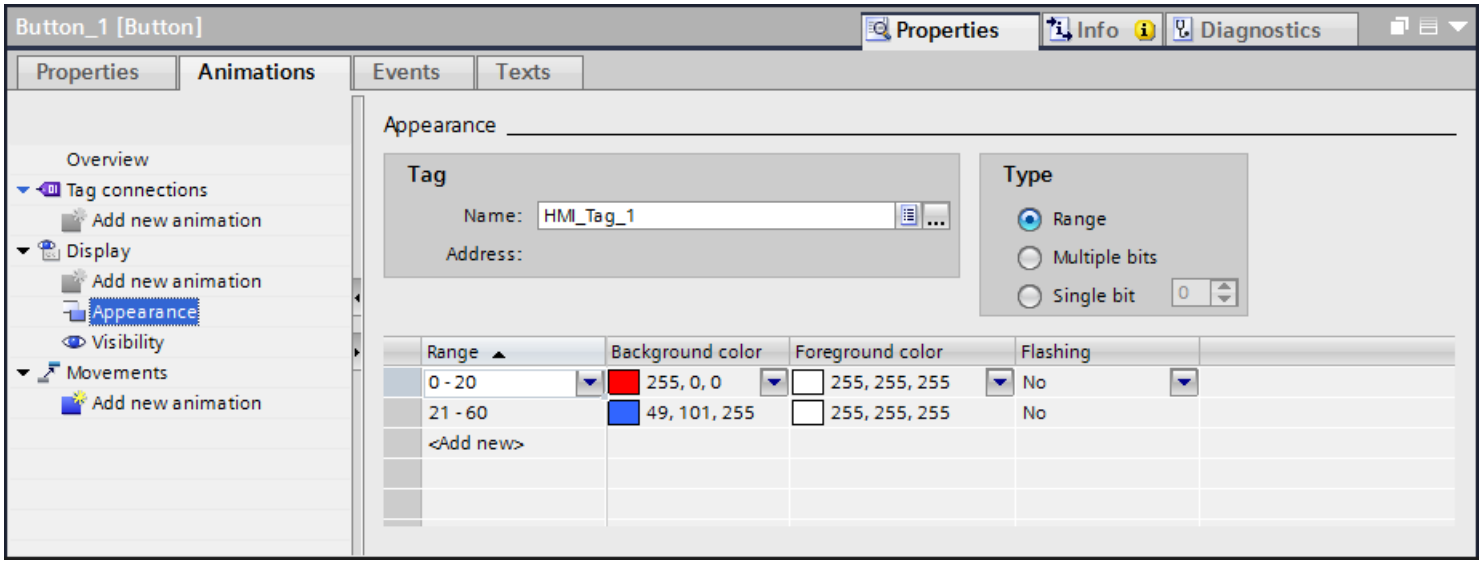
Areas or single values of the tag are observed in Runtime depending on the selection. The appearance of the object changes according to the configuration.

Requirement

- A screen is open.
- A dynamic object is contained and selected in the screen.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure

1. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.
2. Select the animation "Appearance" and click the button.
   The parameters of the animation are displayed.
3. Select a tag in "Tag > Name".
4. Select "Type > Range" for example.
5. Click "Add" in the table.
6. Enter the tag interval "0 - 20" in the "Range" column for example.
7. For "Foreground color" and "Background color", select the color the object is to acquire in Runtime when the tag reaches the interval.
8. Select a flashing mode for the object from the "Flashing" list.
9. Repeat steps 5 to 8 to create another tag interval, e.g. "21 - 60".



Result

In Runtime, the flashing response, and color of the object change dynamically according to the process value returned in the tag.

See also

Configuring a new animation (Page 4018)

Configuring movements

Introduction

You can configure dynamic objects in such a way that they move on a certain track. The movement is controlled via tags. The object moves every time the tag is updated.

You can only program one type of movement for each object.

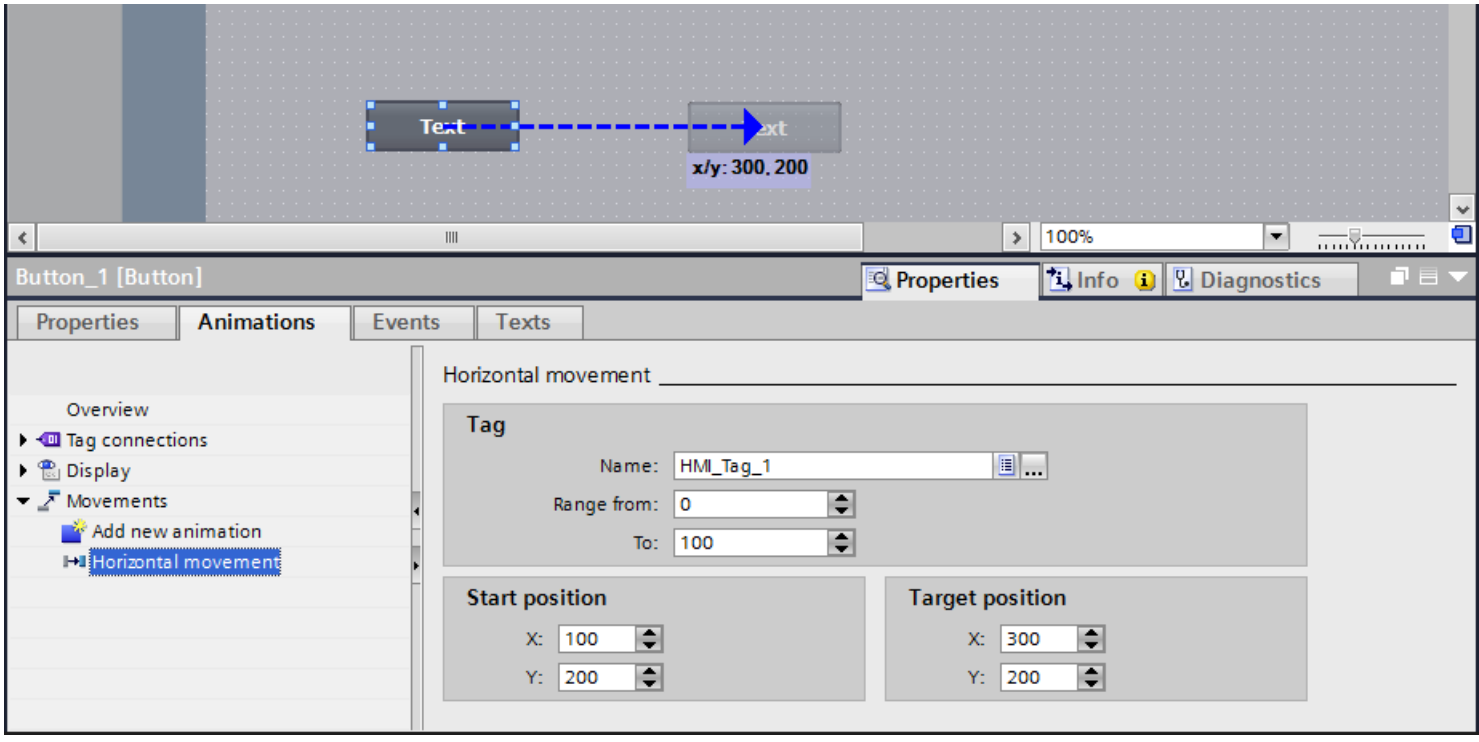
Requirement

- You have created a tag.
- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure

1. Select the screen object you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.
3. Select "Horizontal movement" and click the button.
   The parameters of the animation are displayed.
   A transparent copy of the object is shown in the work area, which is connected to the source object by means of an arrow.
4. Select a tag for control of movement.

5. Move the object copy to the relevant destination. The system automatically enters the pixel values of the final position in the Inspector window.
6. Customize the range of values for the tag as required.



Result

In Runtime, the object moves in response to every change in value of the tag that is used to control the movement. The direction of movement corresponds to the configured type of movement "horizontal".

Note

You configure vertical and diagonal movements similar to horizontal movements

See also

Configuring a new animation (Page 4018)

Configuring direct movement

Introduction

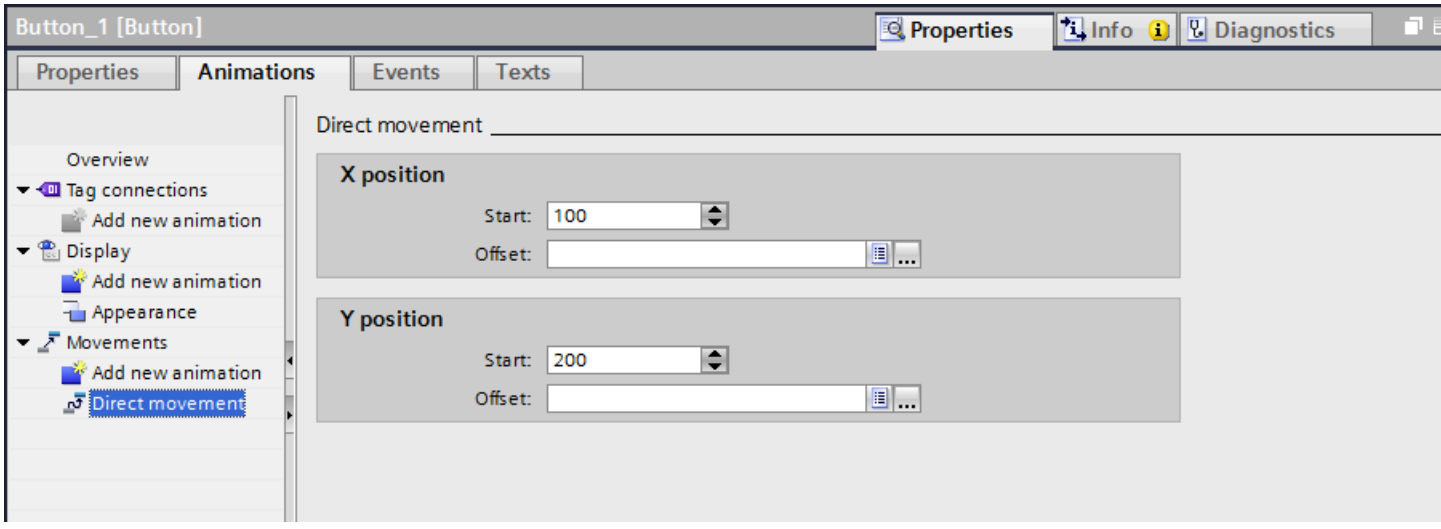
The object is moved respectively in x direction and y direction with "Direct movement". Two tags define the number of pixels by which the object moves from its original static start position.

Requirement

- Two tags are set up.
- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Configuring "Direct movement"

1. Select the screen object you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
3. Select "Direct movement" and click the button.
   The parameters of the animation are displayed.
4. Select a tag for the X position with which the movement in x direction is controlled.
5. Select a tag for the Y position with which the movement in y direction is controlled.



Result

In Runtime, the object moves in response to every change in value of the tag that is used to control the movement.

See also

Configuring a new animation (Page 4018)

Dynamization of the control enable state of an operating element

Introduction

You can dynamically change the usability of an operating element. The object either can, or cannot be used in Runtime, depending on the value of a tag.

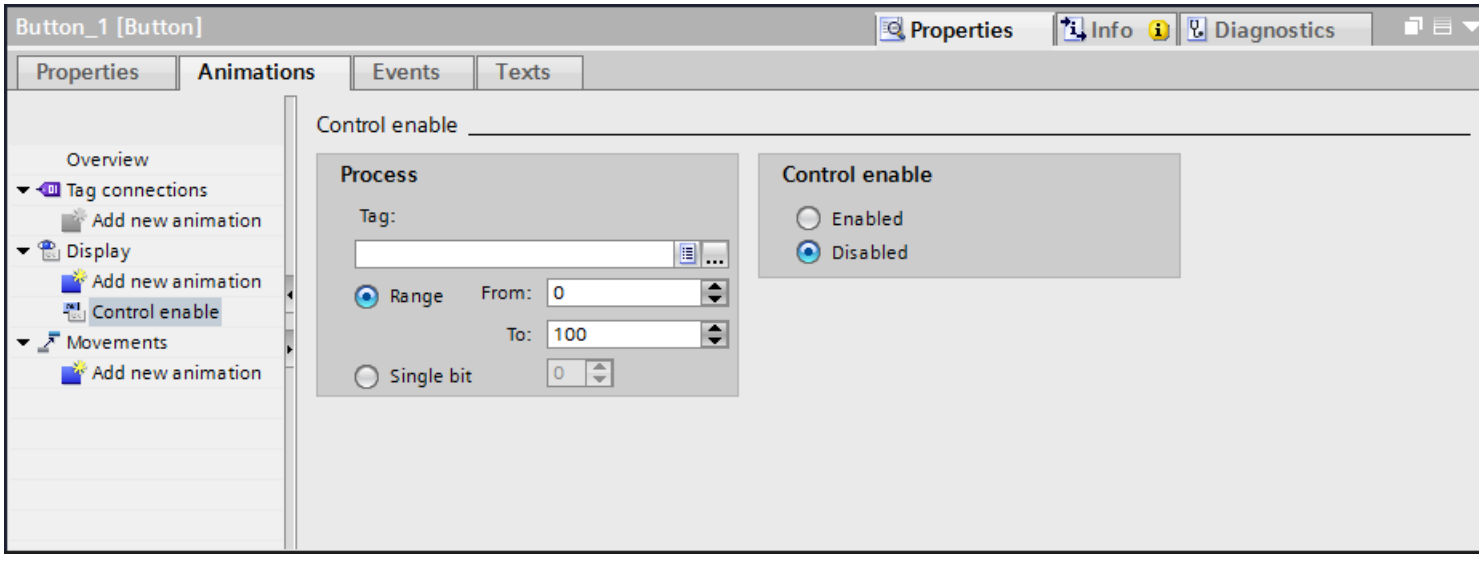
You can use this dynamic control to restrict operator access to an operating element to specific situations, such as for maintenance.

Requirement

- You have created a tag.
- A screen with an operating element is open.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure

1. Select the operating element in the screen that you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.
3. Select "Control enable" and click the button.
   The parameters of the animation are displayed.



4. Select a tag.

5. Activate "Range".
   - For "From:" enter e.g. the value "0"
   - For "To:" enter e.g. the value "100"
6. Activate "Control enable > Activated".

Result

The screen object is operator control enabled in Runtime depending on the tag value:

- The object becomes operator control enabled when the tag value is between 0 and 100.
- The object is not operator control enabled if the tag value is outside the range.

See also

Configuring a new animation (Page 4018)

Dynamizing the visibility of an object

Introduction

Dynamization of the "Visibility" property allows you to output an alarm to your screen, which is triggered when the tag value exceeds a critical range, for example. The alarm is cleared when the tag value returns to the non-critical range.

The "Simple recipe view" and "Simple alarm view" objects are always visible.

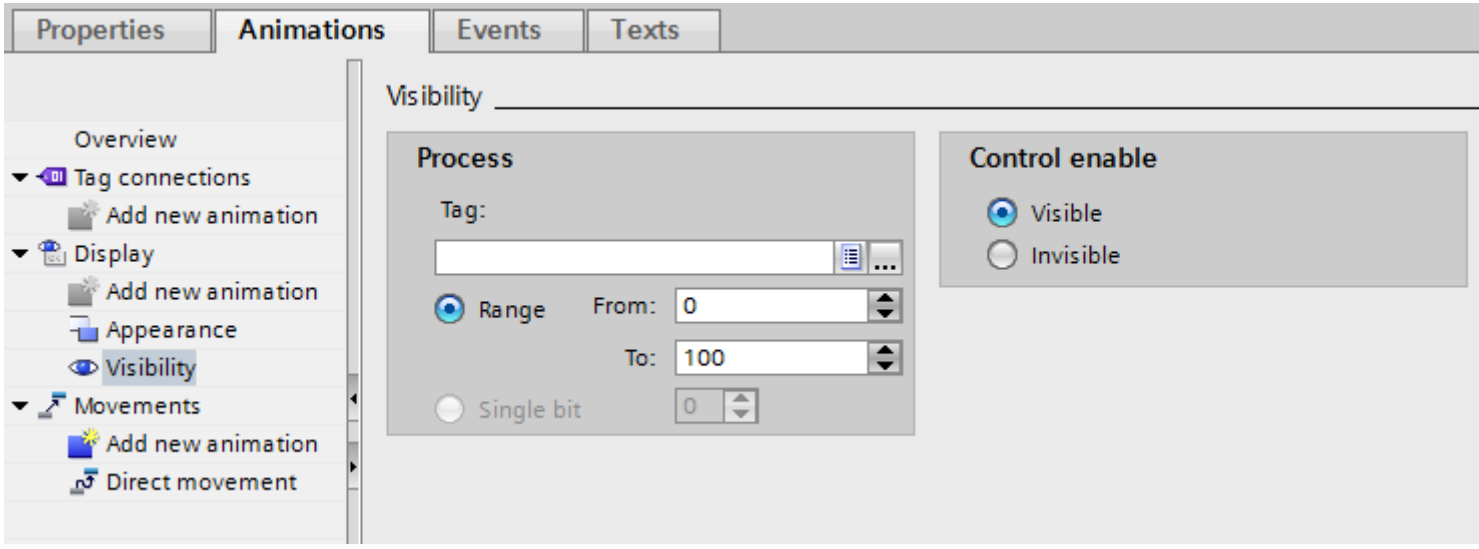
Requirement

- You have created a tag.
- You have opened a screen containing an object that you want to show or hide in Runtime.
- The Inspector window is open.

Procedure

1. Select the screen object you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.
3. Select "Visibility" and click the button.
   The parameters of the animation are displayed.
4. Select a tag.
5. Activate "Range".

6. Select, for example, "from 20" and "to 40"
7. Activate "Visible".



Result

The screen object is shown or hidden in Runtime depending on the tag value.

- If the tag value matches the configured range between 20 and 40, the screen object is shown.
- If the tag value is outside the configured range, the screen object is hidden.

See also

Configuring a new animation (Page 4018)

Configuring animation with tag connection

Introduction

When you object property an object property with a tag, the object property is changed in Runtime depending on the tag value.

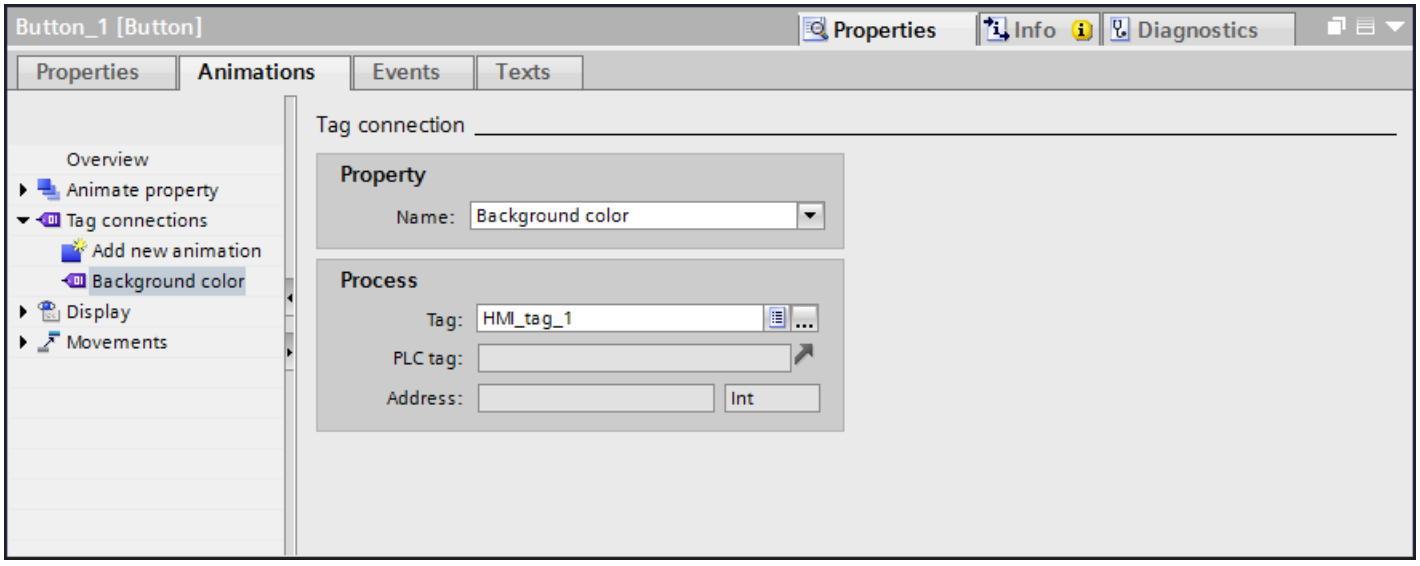
The following example shows the tag binding in the "Properties > Animations" inspector window. If you have configured tag binding it is also visible in the property list or in the property pages in the inspector window.

Requirement

- An object with the property to be dynamized is selected.
- You have created a tag.
- The Inspector window is open.

Procedure

1. Select the screen object whose properties you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animation".
   The animations available for the selected object are displayed.
3. Select "Tag connection" and click the button.
   The parameters of the animation are displayed.
4. Select the object property that you want to dynamize in "Properties > Animation > Tag binding > Property > Name", e.g. "Background color".
5. Select a tag.



Result

You have dynamized the "Background color" property with a tag. The background color of the screen object changes in Runtime depending on which value the tag adopts.

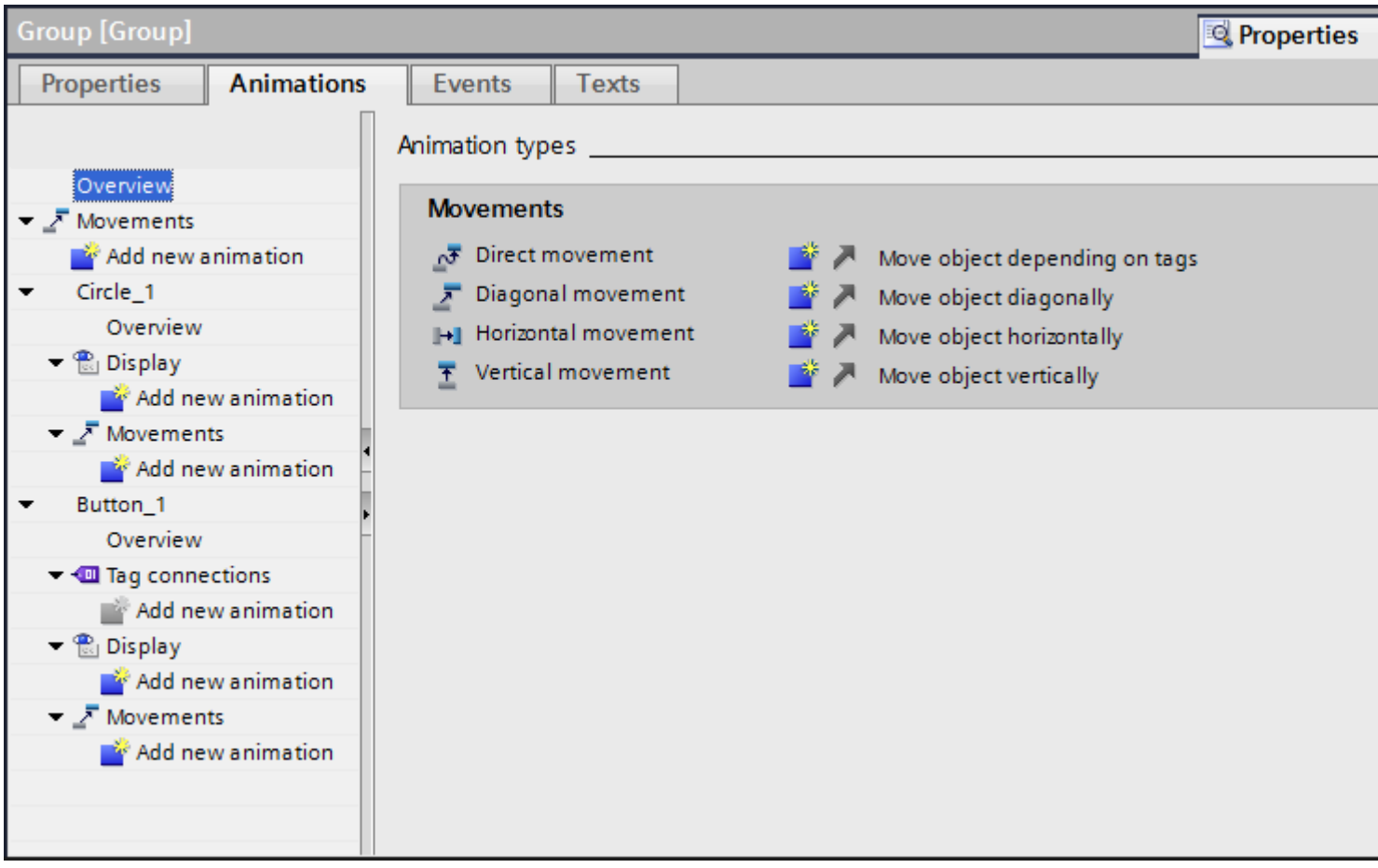
See also

Configuring a new animation (Page 4018)

Animations of object groups

Applying animations to object groups

The inspector window shows all objects of a group and their possible animations. The animation types which are supported by all objects in the group are also listed separately.



If you configure an animation for an object group, this animation will apply to all individual objects that support this animation.

Application example

The "horizontal movement" animation is configured for the object of an object group. The "direct movement" animation is configured for the whole object group. Only the animation of the object group i.e. "direct movement" is executed in Runtime. This also applies for object groups within object groups. Only the animation of the group on the top layer is listed.

See also

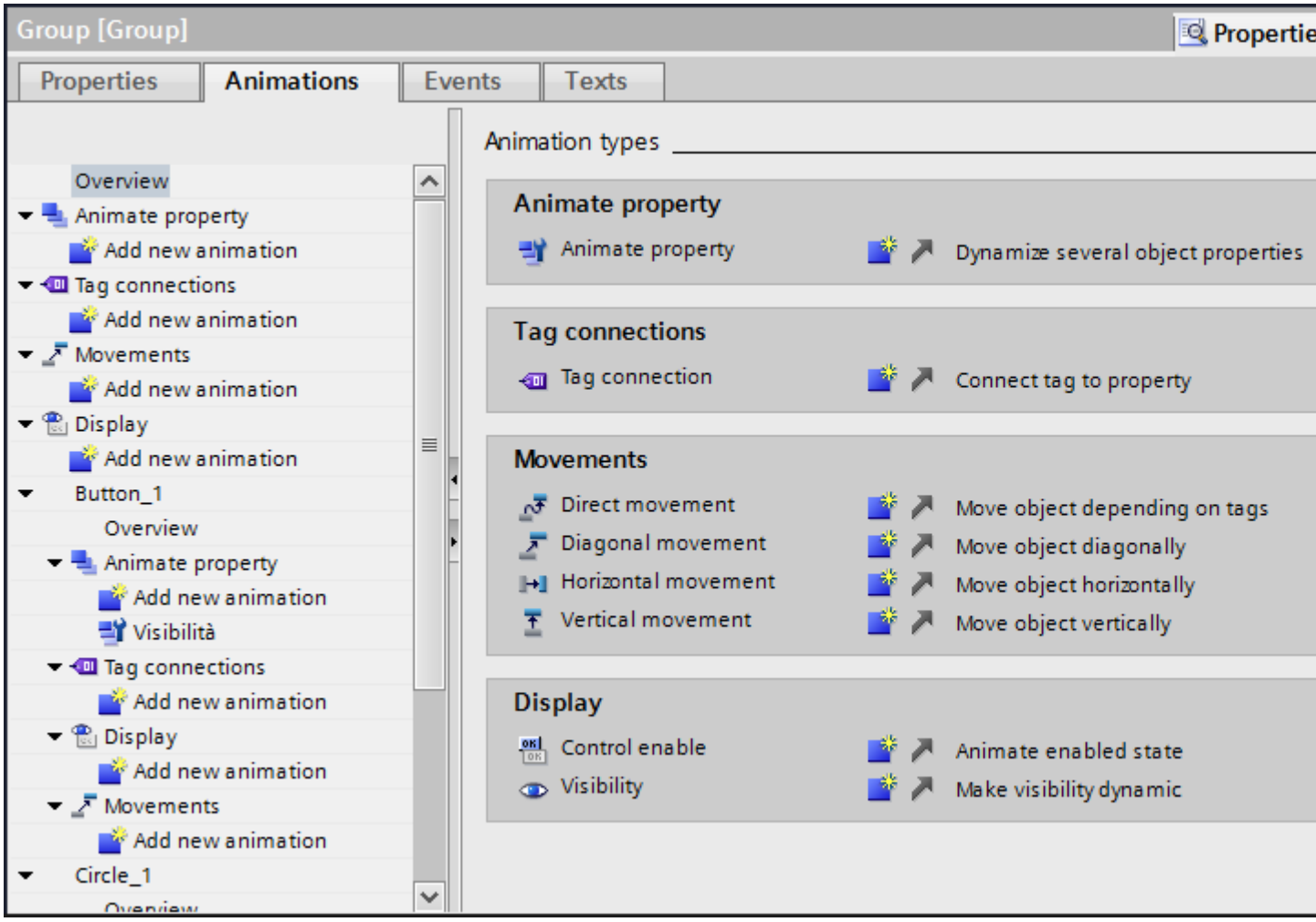
Configuring a new animation (Page 4018)

Animations of object groups

Changing animations of object groups

The inspector window shows all objects of a group and their possible animations. The animation types which are supported by all objects in the group are also listed separately.

If you configure an animation for an object group, this animation will apply to all individual objects that support this animation.



Application example

The "horizontal movement" animation is configured for the object of an object group. The "direct movement" animation is configured for the whole object group. Both animation types are executed in runtime. This also applies for object groups within object groups.

See also

Configuring a new animation (Page 4018)

Animations in multiple selection

Changing animations for multiple objects

For a multiple selection, the animations that are configured for the reference object appear in the Inspector window. You can change the animations as usual. The changes will apply to all the objects in the multiple selection that support the configured animation. This means that even objects for which you have not yet configured an animation will have the reference object's animation.

Application example

Select a button, and a circle at the same time. The button is the reference object. The "Appearance" animation is already configured for the button, so it appears in the Inspector window of the multiple selection. When you activate "Properties > Animations >Appearance > Flashing" in the inspector window, the settings of the "Appearance" animation apply for the button and for the circle.

Configuring new animations for multiple objects

If you configure a new animation for the objects of a multiple selection, this animation will apply to all selected objects that support the configured animation. If the new animation replaces an existing animation, a security prompt appears.

Application example

You select a circle, and a rectangle. The "Diagonal movement" animation is already configured for the circle. You configure the "Horizontal movement" animation for the multiple selection. This animation applies to the rectangle since no animation of the Movement type is yet configured for it. For the circle, you are asked to confirm that you want to replace the existing "Diagonal movement" animation with the new "Horizontal movement" animation.

See also

Configuring a new animation (Page 4018)

12.1.4.6 Dynamization in the property list

Basics on dynamization in the property list

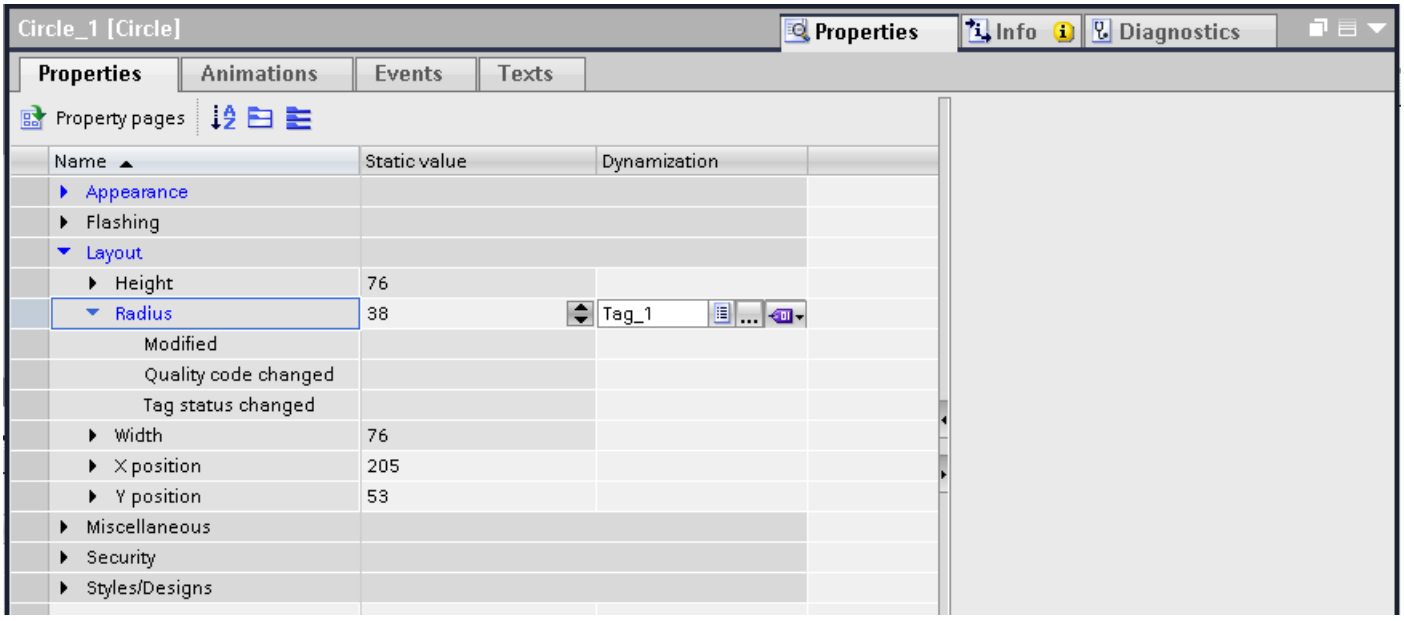
Basic principle of dynamization

Select a tag from the object list for a dynamizable object property in the property list. The object is dynamized if the tag value changes in Runtime.

Dynamization with tags

You dynamize an object property with a tag and configure a function list. The tags are updated in the update cycle set for the screen.

The figure below shows the "Radius" object property dynamized with a tag. When the value of the tag changes, a function list is executed additionally:



You can also convert the function list into a local script.

You can find additional information under "Basics on the function list".

See also

Basics on dynamizing screens (Page 4012)

Programming a dynamization method for a tag change (Page 4033)

Dynamizing an object property with a tag (Page 4031)

Dynamizing an object property with a tag

Introduction

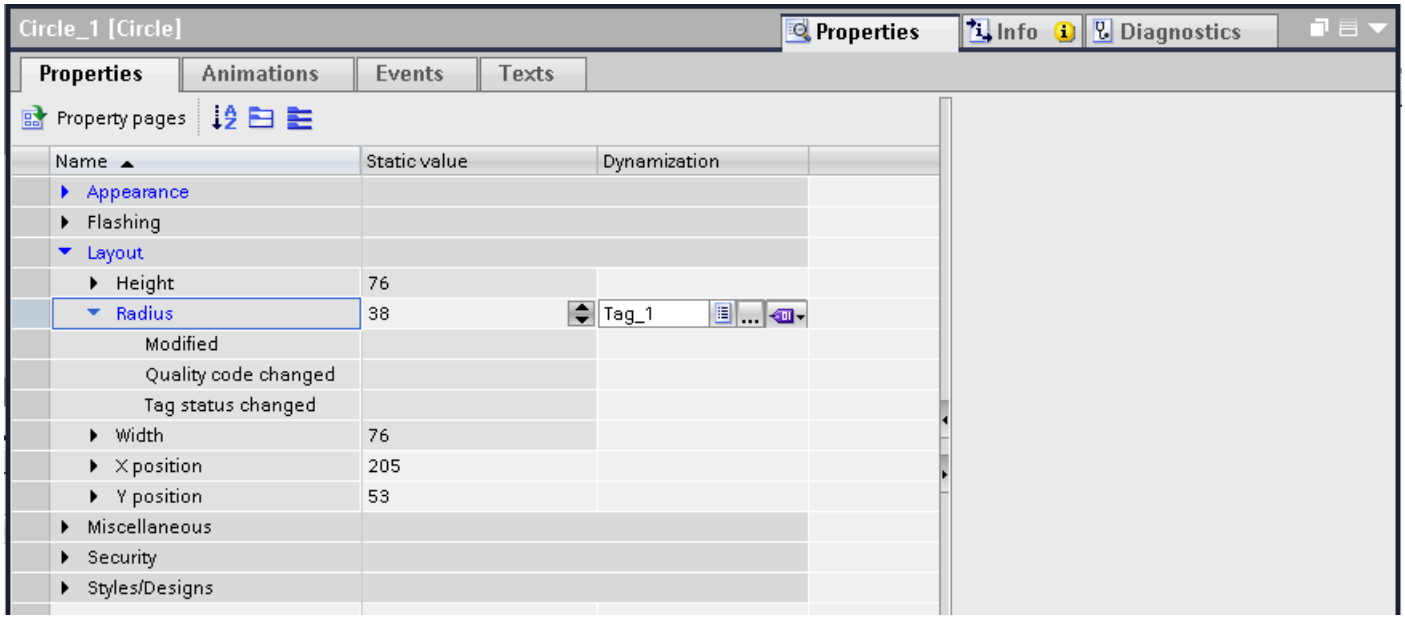
When you dynamize an object property with a tag, the object property is changed in Runtime depending on the tag value. When you indirectly dynamize the object property, you connect the property to a tag that sets the property value in Runtime.

Requirement

- An object with the property to be dynamized is selected.
- You have created a tag.
- The Inspector window is open.

Procedure

1. Select the screen object whose properties you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. Click the button in the Inspector window.
   The property list is displayed with the properties of the selected object.
3. Select the property to be dynamized.
4. In the "Dynamization" column, select the entry "HMI_Tag".
   A dialog opens.
5. Select a tag.
6. Activate "Use indirect addressing" to dynamize the object property.



Result

The object property changes in Runtime according to how the property list is configured.

Representation of a multiple selection in the property list

If you selected multiple objects at the same time, the "Object type" column is added to the property list. The objects that support the property displayed in the row are displayed in this column. Changes to, or dynamization applied to a property affect all the objects displayed in the "Object type" column.

See also

Basics on dynamization in the property list (Page 4030)

Programming a dynamization method for a tag change

Introduction

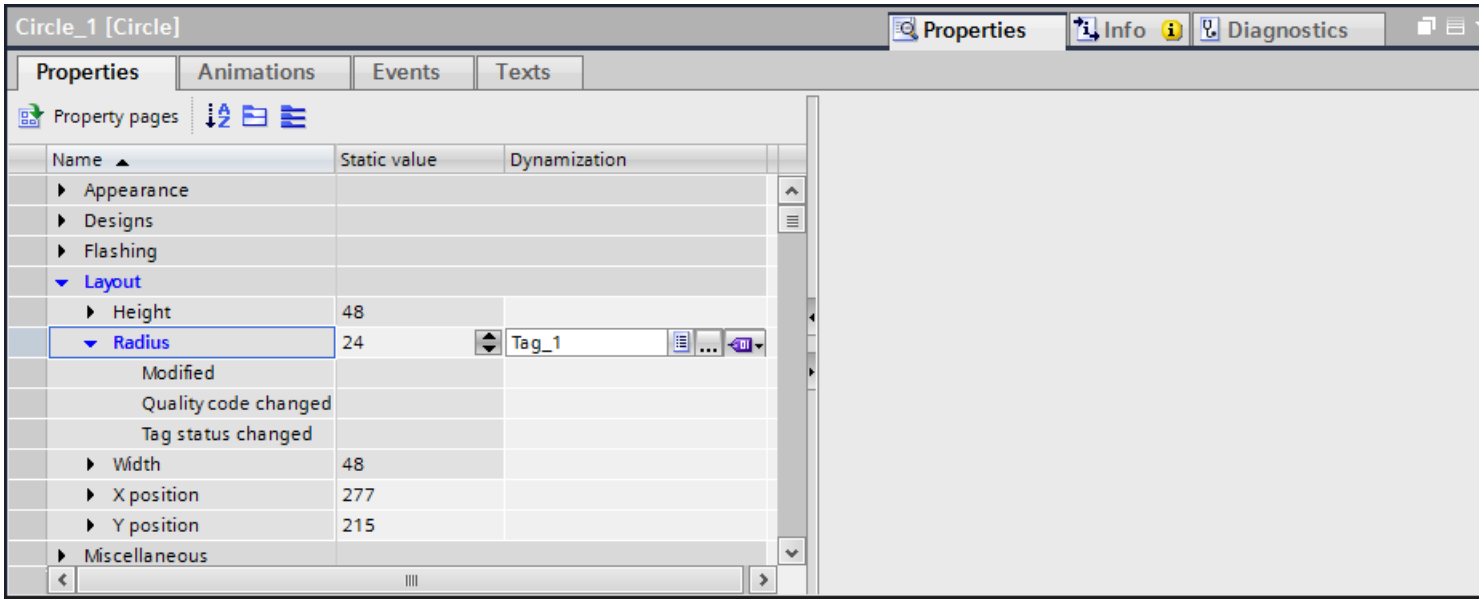
If you have entered a tag in the "Dynamization" column, you can configure a function list additionally on different events.

Requirement

- An object with the property to be dynamized is selected.
- You have created a tag.
- The property list is open in the inspector window.
- An object property is connected with the tag.

Procedure

1. Extend the property list by clicking next to the property which you have dynamized with a tag.
2. Three other lines with the following events are displayed below the property.
   - Changed
   - Tag status changed
   - Quality code changed
3. Configure a function list for the required event.
   Alternatively you can convert the function list into a local script.



Result

The object property changes in Runtime according to how the property list is configured.

See also

Basics on dynamization in the property list (Page 4030)

12.1.4.7 Dynamizing with system functions

Basic on events

Introduction

Screen objects react to events. You configure a function list with system functions on the events of an object.

Events

Which events and system functions are available depends on the object used.

If the operator activates a screen object for example, the configured system function is executed.

You can find additional information under "Basics on the function list".

See also

Basics of the function list (Page 4574)

Basics on dynamizing screens (Page 4012)

Example: Configuring a button for language switching (Page 4036)

Configure system function on the "Click" event (Page 4035)

Configure system function on the "Click" event

Introduction

You configure a function list on an object event. The linked system function is executed when the event occurs in runtime.

Requirements

A screen is open.

A button has been created in the screen.

The inspector window is open.

Procedure

1. Select the button.
2. Click "Properties> Events" in the Inspector window.
3. Select the "Click" event.
4. Click "Add function" in the table.
5. Select the "ShowAlarmWindow" system function.

Result

The alarm window opens in the screen if the operator clicks the button in runtime.

See also

Basics of the function list (Page 4574)

Basic on events (Page 4034)

Example: Configuring a button for language switching

Introduction

In this example, you configure a button that can be used to toggle between multiple runtime languages during runtime.

Requirements

- You have completed the "Configuring a button in multiple languages" example.
- The "Screen_1" screen is open.
- The button on the screen has been selected.

Procedure

1. In the Inspector window, select "Properties > Events > Press".
2. Click on "Add function" in the table.
3. Select the "SetLanguage" system function and the "Switch" setting.

Result

You have assigned the button the function "SetLanguage". Pressing the button during runtime will switch the runtime language. The runtime languages are switched in the order specified by the number sequence in the "Languages and fonts" editor.

See also

Basic on events (Page 4034)

Button (Page 4161)

12.1.5 Working with function keys

12.1.5.1 Working with function keys

Introduction

The function key is a physical key on your HMI device and its functions can be configured. A function list can be configured for the events "Key pressed" and "Release key".

A function key can be assigned global or local functions.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Global function keys

Global function keys always trigger the same action, regardless of the displayed screen.

Global function keys are configured in the "Global Screen" editor. The global assignment applies for all the screens of the set HMI device.

Global function keys reduce programming considerably, because there is no need to assign these global keys to each individual screen.

Local function keys in screens

Local function keys in screens can trigger a different action in each screen. This assignment applies only to the screen in which you have defined the function key.

Within a screen, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Note

If a screen with local function keys is overlapped by an alarm view or an alarm window, then the function keys are still active in runtime. This may happen in particular on HMI devices with a small display.

Local function keys in templates

Local functions keys that are assigned in templates are valid for all the screens based on this template. They can trigger a different action in every screen. Function keys for templates are assigned in the template of the "Screens" editor. Within a template, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Hotkey assignment

You can assign hotkeys, such as buttons, to control objects. The available hotkeys depend on the HMI device.

Note

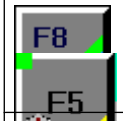
The function key has a local or global action assigned to it. If the function key also has a hotkey assigned to it, then the hotkey function will be executed in Runtime.

Graphics

When a function key is placed directly next to the display, you can assign a graphic to it to make the function of the function key more clear.

Display of assignment

Table 12-1 The following table shows which symbols display the assignment of the function keys:

Function key	Description
	Not assigned
	Global assignment
	Assigned locally in the template
	Local assignment
	Local assignment (local assignment of the template overwrites global assignment)
	Local assignment (local assignment overwrites global assignment)
	Local assignment (local assignment overwrites local assignment of the template)
	Local assignment (local assignment overwrites local assignment of the template, which already overwrites global assignment)
	Assigning buttons with screen navigation

Note

Basic Panels

The "Screen Navigation" editor is not available for Basic Panels.

12.1.5.2 Assigning function keys globally

Introduction

You define the global assignment of a function key in the "Global Screen" editor. The global assignment applies to all screens of the set HMI device.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

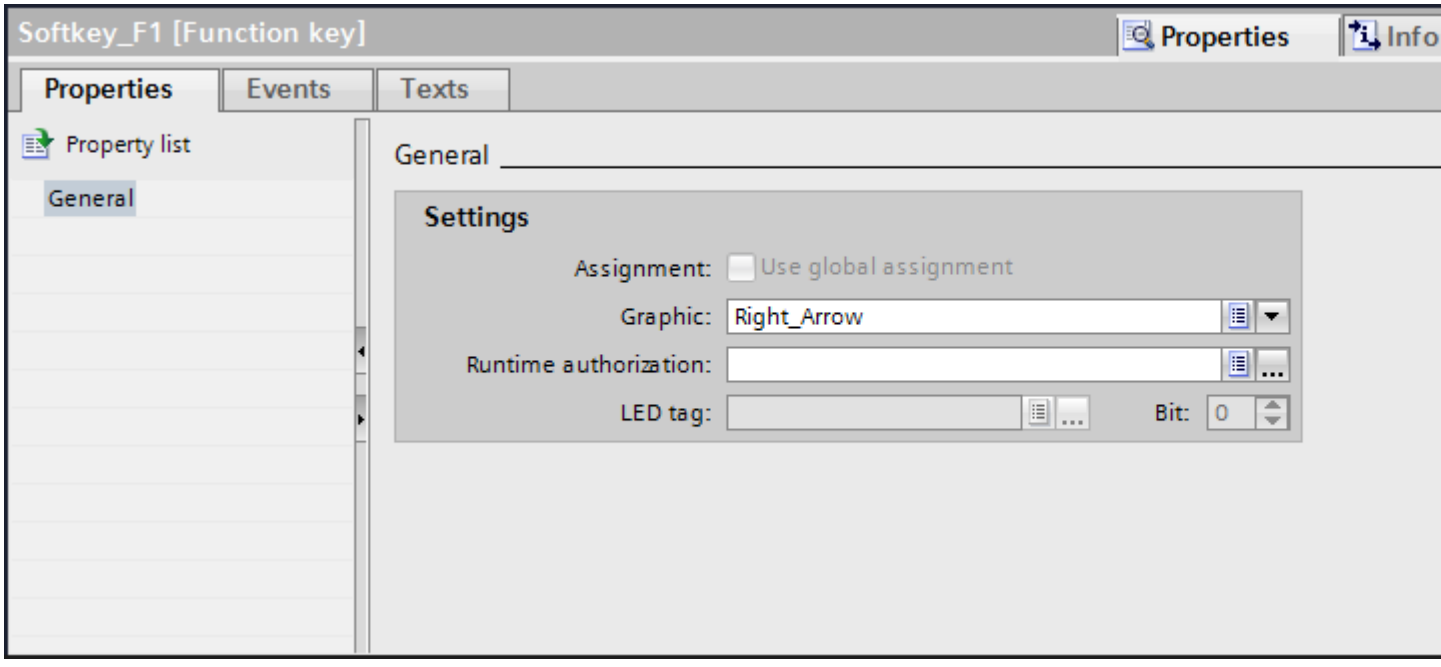
Requirement

- You have opened the project.
- The Inspector window is open.

Procedure

Proceed as follows to assign a screen-based function to a function key:

1. To open the "Global Screen" editor, double-click "Global Screen" in the "Screen management" group of the Project window.
2. Select the desired function key.
   The properties of the function key are shown in the Inspector window.



3. Under "General", configure a graphic and an operator authorization for the function key.
4. Configure a function list for the required event under "Events".

Result

If a local assignment does not overwrite the global assignment, the assignment of the function key changes in all the screens of the set HMI device in accordance with your entry.

See also

Working with function keys (Page 4036)

12.1.5.3 Local assignment of function keys

Introduction

Function keys are assigned globally and locally. A local assignment of the function keys is only valid for the screen or the template in which it was defined. The following local function keys are available:

- Local function keys of a screen
  Different functions are assigned to the function key for each screen. This assignment applies only to the screen in which you have defined the function key.
- Local function keys of a template
  You assign the function keys in a template. The assignment applies to all the screens that are based on this template and are not overwritten by a local assignment in a screen.

A local assignment overwrites the global assignment of a function key.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Using existing assignments

The option for using existing assignments is referred to as follows in the Inspector window:

- In a template: "Use global assignment"
- In a screen:
  - Screen based on a template: "Use local template"
  - Screen not based on a template: "Use global assignment"

The "Use local template" option includes the use of the local assignment in the template and the global assignment.

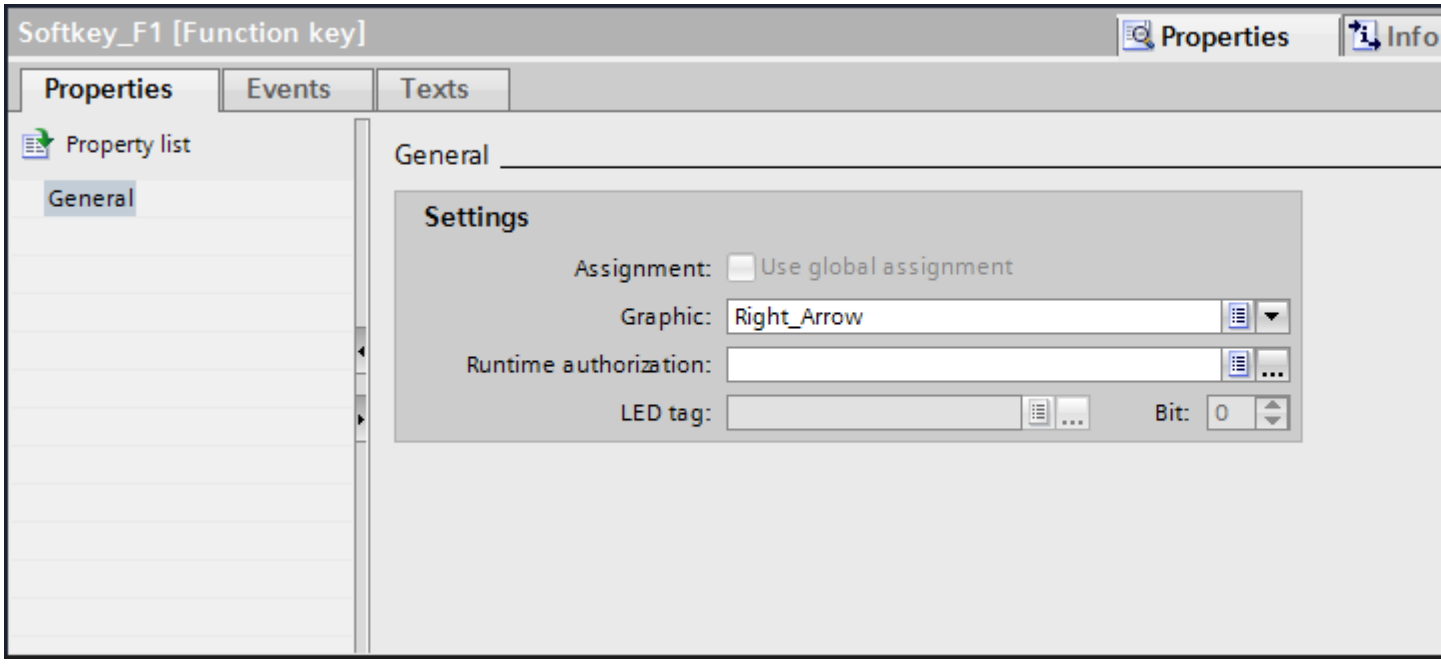
Requirement

- You have opened the screen or the template in which you want to assign a function key locally.
- The Inspector window is open.

Procedure

Proceed as follows:

1. Select the desired function key in the screen or in the template.
   The properties of the function key are shown in the Inspector window.
2. Click "General" in the Inspector window.



3. Disable the "Use local template" or "Use global assignment" option.
4. Under "General", configure a graphic and an operator authorization for the function key.
5. Configure a function list for the required event under "Events".

Result

The function key is assigned the configured function in the screen or in the template.

See also

Working with function keys (Page 4036)

12.1.5.4 Assigning a function to a function key

Introduction

A function key can have two states:

- Pressed: Defined by the "Key pressed" event.
- Released: Defined by the "Release key" event.

Both of these events are configured in the Inspector window of the function key. You can assign any event a function list which contains system functions or scripts. Execution of this function list is event-driven in runtime.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Note

Basic Panels

Scripts are not available for Basic Panels.

Requirement

To assign the function key a global function:

- The "Global Screen" editor is open.

To assign the function key a local function:

- The screen in which you want to assign a function key is open.

If you want to assign a function key locally in a template:

- The template in which you want to assign a function key is open.
- The Inspector window is open.

Procedure

Proceed as follows:

1. Select the function key you want to define.
   The properties of the function key are shown in the Inspector window.
2. Configure the function list for the desired result in the Inspector window under "Properties" in the "General" group.

Result

The function list is executed in runtime when the operator presses or releases the function key.

See also

Working with function keys (Page 4036)

12.1.5.5 Assigning operator authorization for a function key

Introduction

In WinCC you can assign an operator authorization for a function key in runtime. This allows you to restrict access to the function keys to specific persons or operator groups when you configure your project. Only authorized personnel can then change important parameters and settings in runtime.

You configure access protection in order to avoid operator errors and increase plant or machine security.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

- The user groups have been defined.

To protect a global function key:

- The "Global Screen" editor is open.

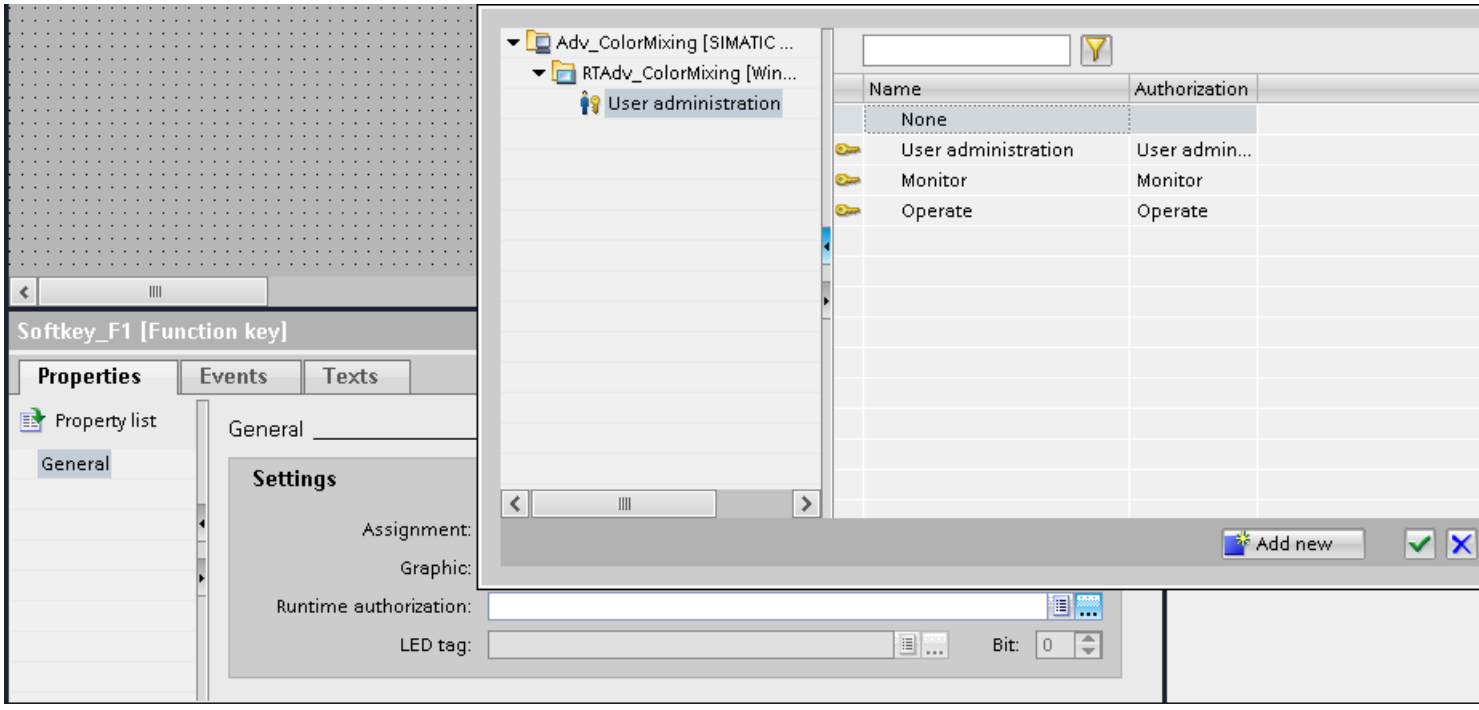
If you want to protect a local function key of a screen or of a template:

- The screen or the template which contains the function key is open.
- The Inspector window is open.

Procedure

Proceed as follows:

1. Select the relevant function key.
   The properties of the function key are shown in the Inspector window.
2. Click "General" in the Inspector window.



3. In the "Authorization" list, select the user group you want to allow runtime access to the function key.

Result

The operator authorization is configured.

See also

Working with function keys (Page 4036)

12.1.5.6 Assigning a graphic to a function key

Introduction

In order to make the function of a key more clear, you can insert a graphic in the screen alongside the function key. Graphics can only be assigned to function keys that border the screen margin of the HMI device.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

To assign a graphic to a global function key:

- The "Global Screen" editor is open.

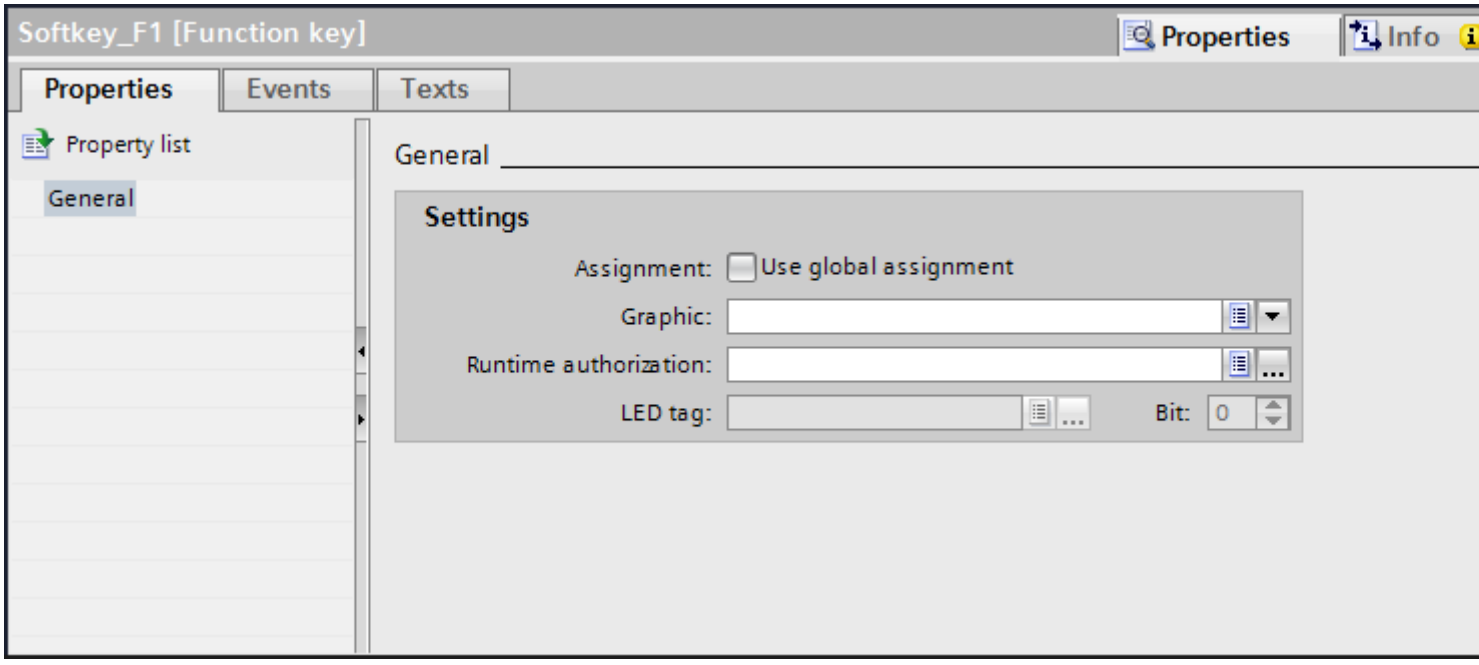
If you want to assign a graphic to a local function key in a screen or template:

- The screen or the template that contains the corresponding function key is open.
- The Inspector window is open.
- You have created the graphic for the function key.

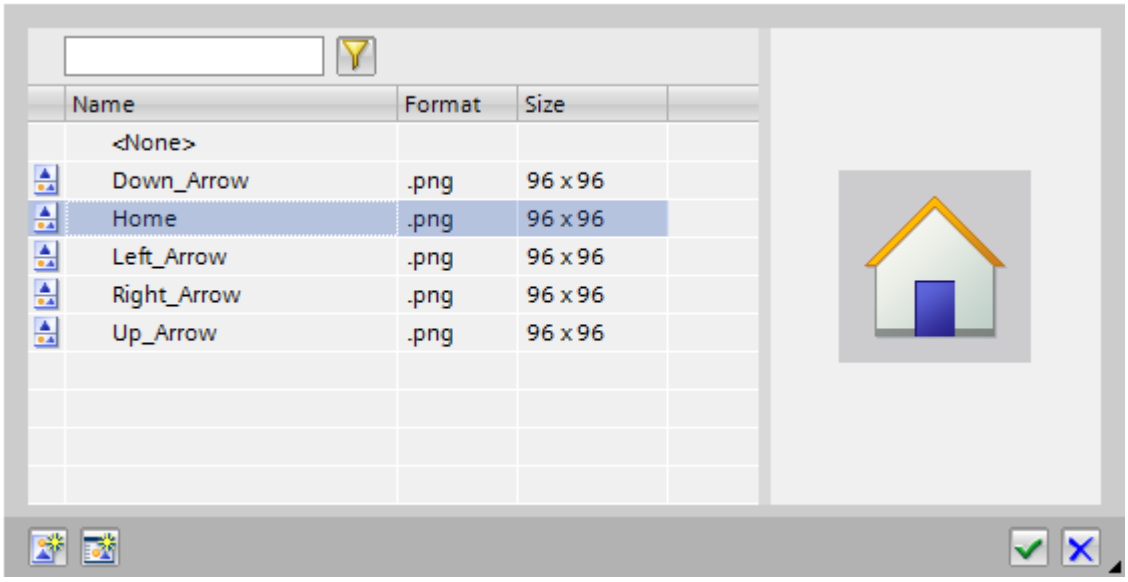
Procedure

Proceed as follows:

1. Select the relevant function key.
   The properties of the function key are shown in the Inspector window.
2. Click "General" in the Inspector window.



3. Click in the "Graphic" area of the list.
   The graphic browser of your WinCC project appears. The pane on the left side shows the external graphics which are stored in the browser. The pane on the right side shows you a preview of the graphic you have selected in the browser.



   Using the and icons, you can display the collection either in form of thumbnails or as a list.
   In order to open and edit OLE objects with the associated graphics program, double-click on the object.

4. In the browser click the desired graphic or store the relevant graphic in the graphic browser.
   The graphic preview is shown in the right pane.
5. Click "Select" to add the graphic to the screen.
   Click "Clear" to remove the graphic from the screen.

Result

The graphic is displayed next to the function key.

See also

Working with function keys (Page 4036)

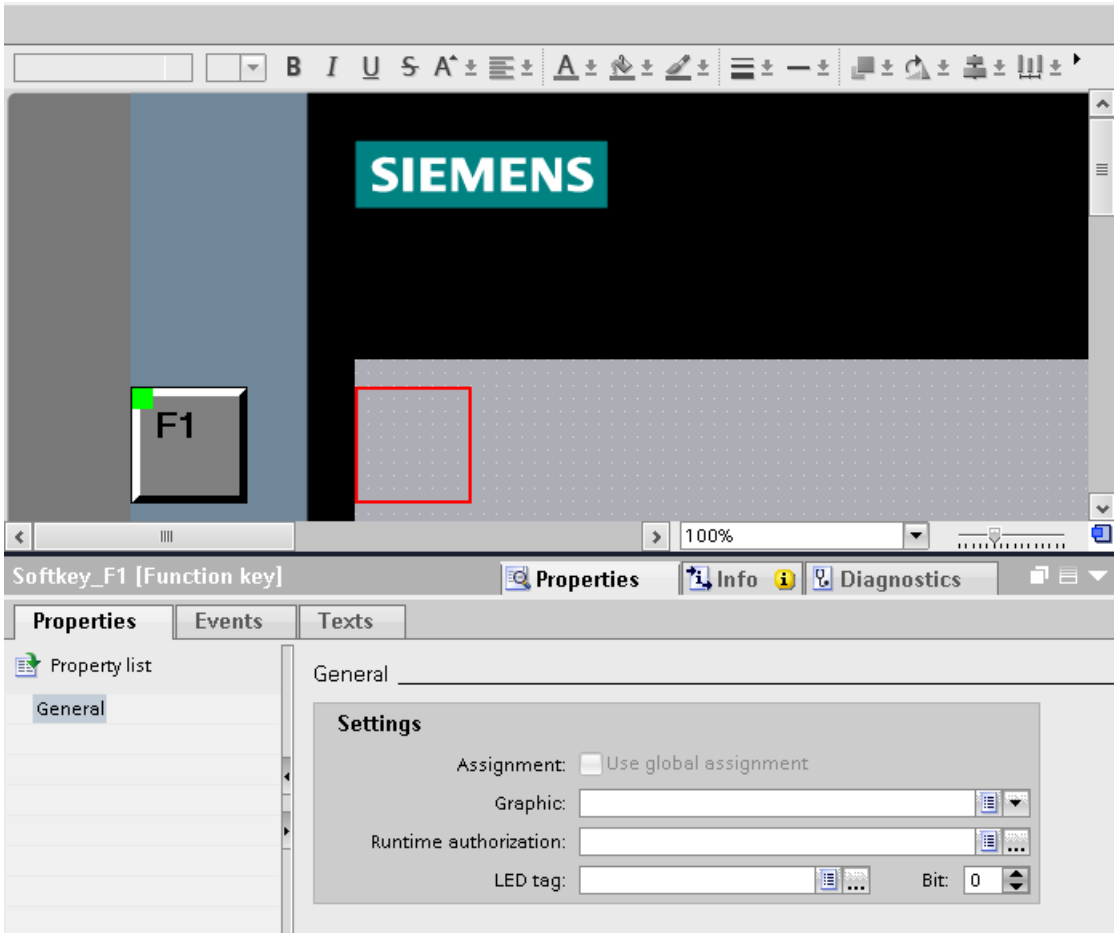
12.1.5.7 Configuring LED tags

Requirements

- An HMI device with key operation has been created.
- You have created an LED tag.
- The global screen is open.

Procedure

1. Click an F-key of the HMI device.
2. In the Inspector window, click "Properties > Properties > General".



3. Select a tag under "LED tag" in the "General > Settings" area.
4. Under "Bit" enter the correct bit number.
   The correct bit number depends on the HMI device and the input and output assignments on the HMI device.

Assignment of inputs and outputs

The exact assignment of inputs and outputs can be found under:

- PROFINET IO direct keys: Auto-Hotspot
- PROFIBUS DP direct keys: Auto-Hotspot

12.1.5.8 Example: Using function keys for screen navigation

Task

In this example you create a local function key in a screen. When the operator presses this function key, a screen change to a predefined screen is triggered, for example "Boiler 2".

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

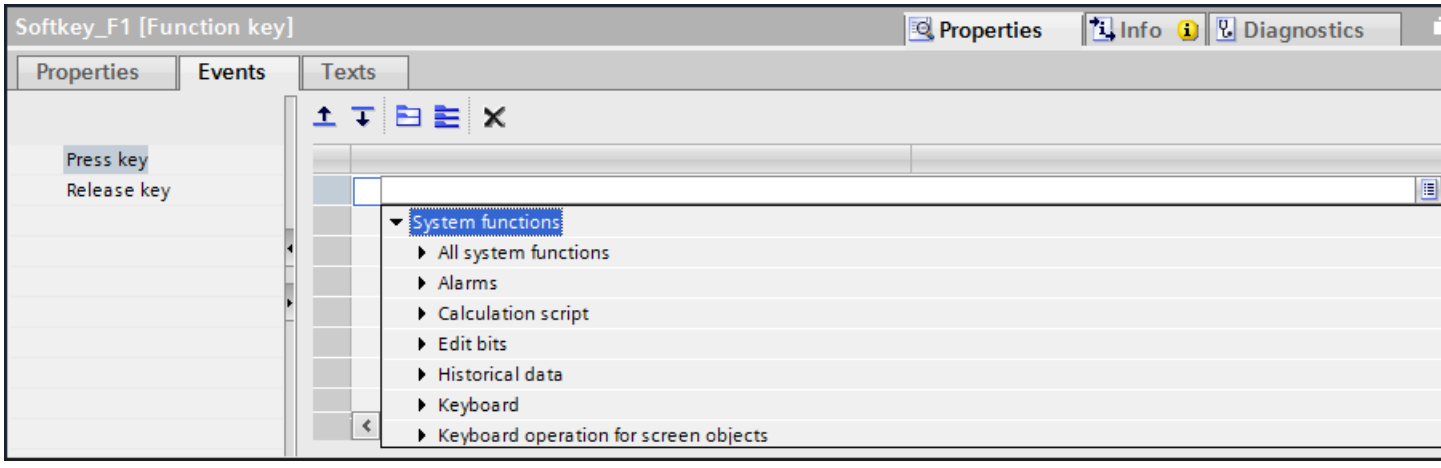
Requirement

- The screen in which you want to assign the function key is open.
- You have created the "Boiler 2" screen.
- The Inspector window is open.

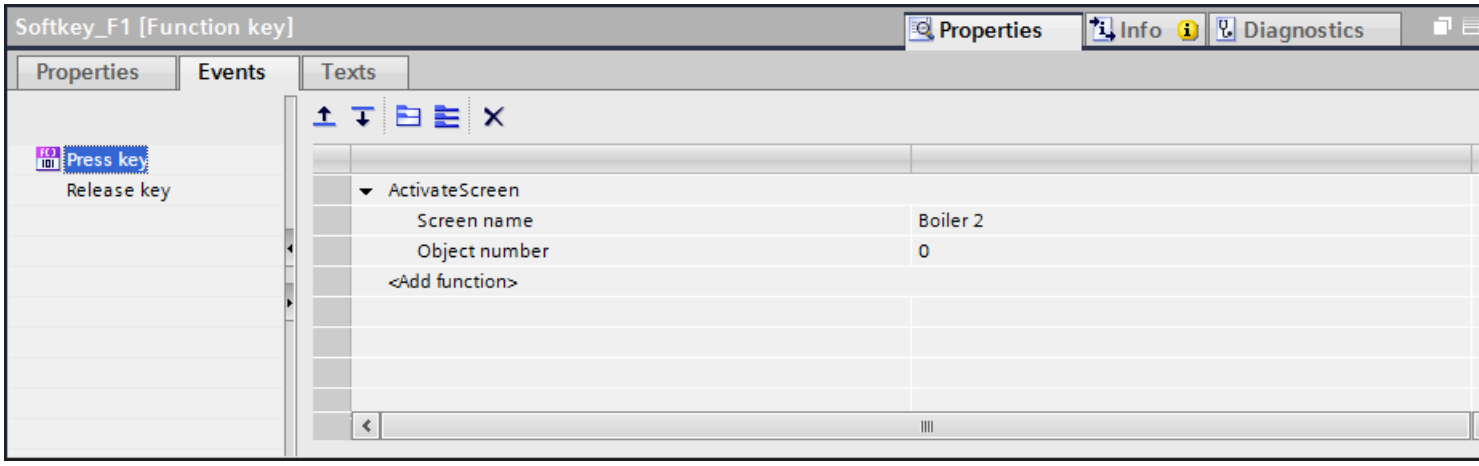
Procedure

Proceed as follows to use the "ActivateScreen" function:

1. Select the desired function key.
   The properties of the function key are shown in the Inspector window.
2. Click "General."
3. To overwrite a global assignment, disable the "Use local template" option.
4. Click "Key pressed" under "Events".



5. Select the "ActivateScreen" system function from the list.
   The "ActivateScreen" function appears in the "Function list" dialog box, including the "Screen name" and "Object number" parameters.



6. Select the "Boiler 2" screen from the "Screen name" list.

Result

The operator changes to the "Boiler 2" screen in runtime by pressing the selected function key.

See also

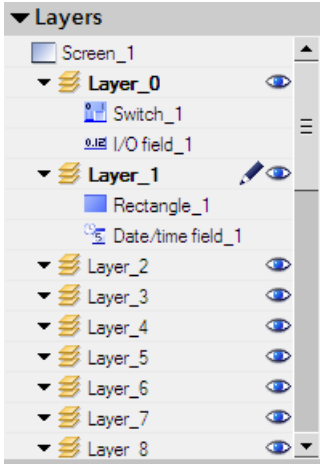
Working with function keys (Page 4036)

12.1.6 Working with layers

12.1.6.1 Basics on working with layers

Layers

Use layers in order to achieve differentiated editing of the objects in a screen. A screen consists of 32 layers that you can give any names. If you assign objects to the layers, you thereby define the screen depth. Objects of layer 0 are located at the screen background, while objects of layer 31 are located in the foreground.



The objects of a single layer are also arranged hierarchically. When you create a screen, the object inserted first is located at the rear within the layer. Each further object is placed one position towards the front. You can shift objects forwards and backwards within a layer.

Principle of the layer technique

Always one layer of the 32 layers is active. New objects you add to the screen are always assigned to the active layer. The number of the active level is displayed in the inspector window of the screen and in the "Layout > Layers" task card.

When you open a screen, all 32 layers of the screen are displayed. You can hide all the layers except for the active layer in the inspector window of the screen and in the "Layout > Layers" task card. You then explicitly edit objects of the active layer.

In the tree view of the "Layers" palette in the "Layout" task card, you administer layers and objects with drag-and-drop and the context menu.

Application examples

Use layers, for example, in the following cases:

- To hide the labeling of objects when editing,
- To hide objects, e.g. alarm windows, while configuring other objects

See also

Screen basics (Page 3889)

Renaming layers (Page 4055)

Showing and hiding layers (Page 4054)

Setting the active layer (Page 4053)

Moving objects on layers (Page 4052)

12.1.6.2 Moving objects on layers

Introduction

By default, a new object is inserted on the active layer. You can, however, assign an object to another layer at a later time.

Requirement

- A screen with an object is open.
- The Inspector window is open.

Procedure

1. Select the object in the screen.
   The object properties are displayed in the Inspector window.
2. Enter the layer to which you want to move the object in "Properties > Properties > Miscellaneous > Layer" in the Inspector window.

Alternatively, select the object from the "Layout" task card and drag it to the required layer.

Changing the order of objects

1. Select the object in the screen.
   The object properties are displayed in the Inspector window.
2. To move the object to the front or back, select the "Order" > "Move backward" or "Move forward" command from the shortcut menu.
   Alternatively, use the or button in the toolbar.

Result

The object is assigned to the selected layer, and positioned at the top of the layer.

See also

Setting the active layer (Page 4053)

Showing and hiding layers (Page 4054)

Renaming layers (Page 4055)

Basics on working with layers (Page 4051)

12.1.6.3 Setting the active layer

Introduction

The screen objects are always assigned to one of the 32 layers. There is always an active layer in the screen. New objects you add to the screen are always assigned to the active layer.

The number of the active layer is indicated in the "Layer" toolbar. The active layer is indicated by the icon in the "Layout > Layers" task card.

Layer 0 is the active layer when you start programming. You can activate a different layer during configuration, if necessary.

Requirement

- You have opened a screen which contains at least one object.
- The Inspector window of the active screen is open.

Procedure

1. Click "Properties > Properties > Layers" in the Inspector window of the current screen.
2. Enter the layer number in "Settings > Active layer".

Alternative procedure

1. Select "Layout > Layers" in the "Layout" task card.
2. Select the "Set to active" command from the shortcut menu of a layer.

Result

The layer with the specified number is now active.

See also

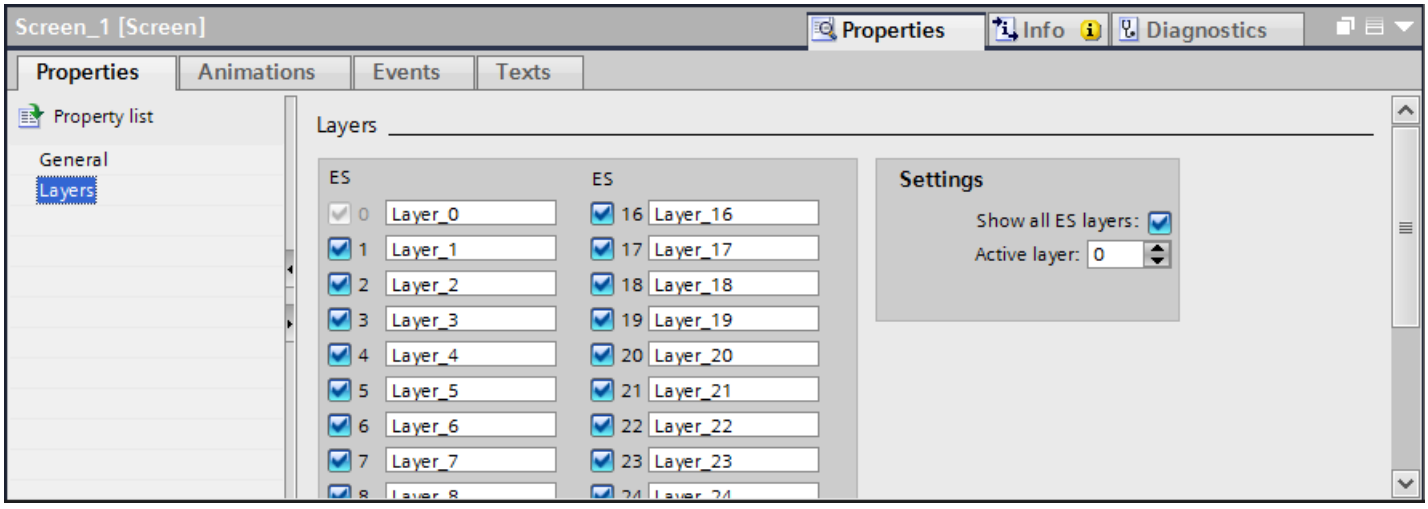
Moving objects on layers (Page 4052)

Basics on working with layers (Page 4051)

12.1.6.4 Showing and hiding layers

Introduction

You can show or hide the layers of a screen as required. You specify the layers that are shown in the Engineering System. When you open a screen, all the layers are always shown.



Requirement

- The screen is opened.
- The "Layout" task card is open.

Procedure

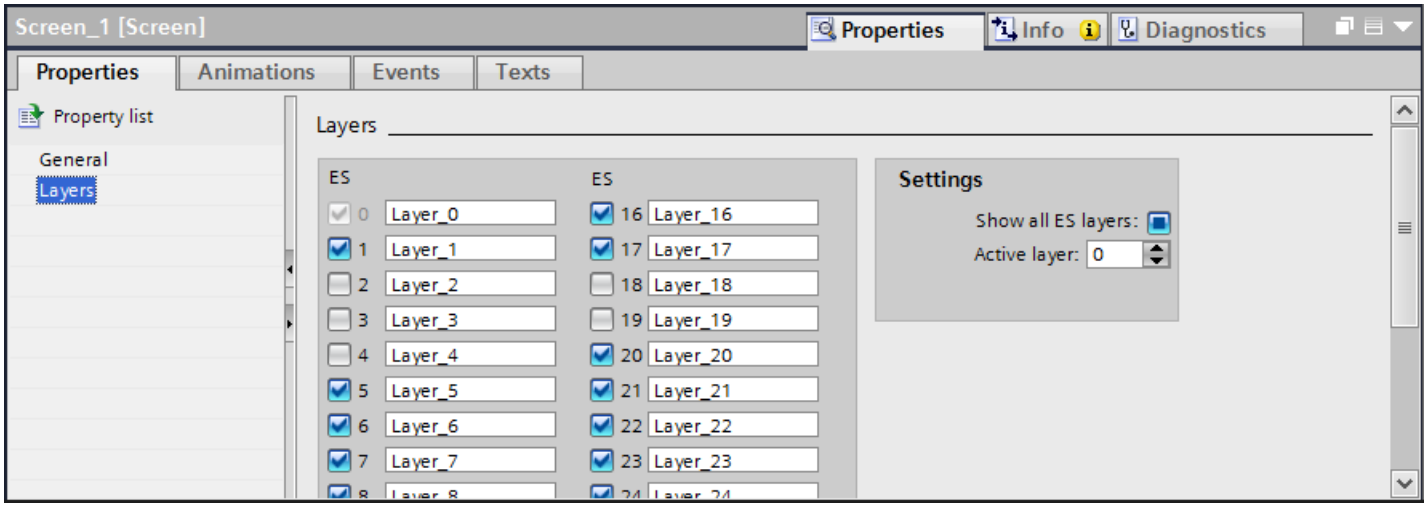
1. Select the layer that you want to hide or show in the "Layout > Layers" task card.
2. Click one of the icons next to the corresponding layer:
   - A shown layer is hidden
   - A hidden layer is shown

   Note

   The active layer cannot be hidden.

Alternative procedure

1. Click in an area of the screen that does not contain an object.
   The screen properties are shown in the Inspector window.
2. In the Inspector window, select "Properties > Properties > Layers":



3. In the list, disable the levels you wish to hide.
   If you activate "All ES layers" for a layer, the objects in this layer will be shown in the Engineering System.

Result

The layers are shown according to your settings.

See also

Moving objects on layers (Page 4052)

Basics on working with layers (Page 4051)

12.1.6.5 Renaming layers

Introduction

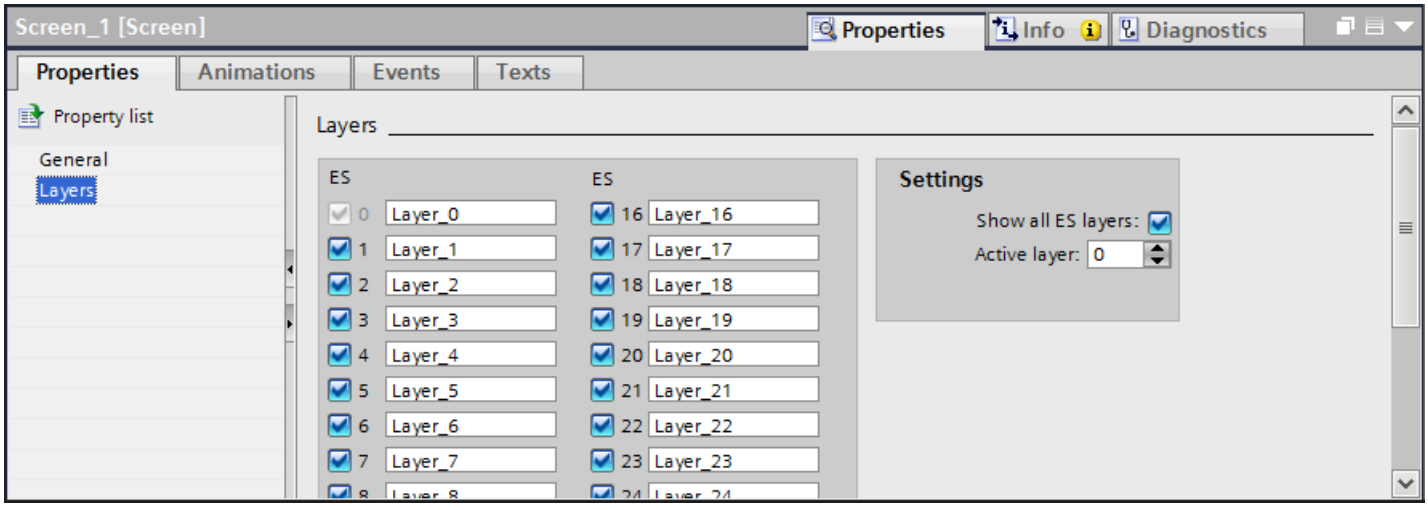
When you create a screen, the 32 layers are numbered consecutively by default. To improve clarity, you can rename the layers to suit your requirements.

Requirement

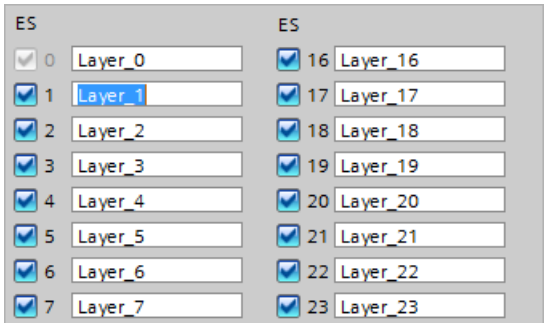
- The screen is opened.

Procedure

1. Click in an area of the screen that does not contain an object.
   The screen properties are shown in the Inspector window.
2. In the Inspector window, select "Properties > Properties > Layers".



3. Enter the new layer name.



Result

The layer is displayed with the new name.

See also

Moving objects on layers (Page 4052)

Basics on working with layers (Page 4051)

12.1.7 Working with faceplates

12.1.7.1 Basics on faceplates

Introduction

Faceplates are a configured group of display and operating objects that you manage and change centrally in a library. You can use a faceplate in several projects as required. The faceplates are stored in the project library.

Use

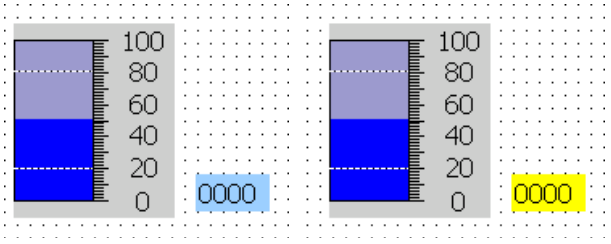
You use faceplates to create individually configured display and operating objects. You can use faceplates several times in the project or in different projects. All instances of a faceplate in the project are changed centrally. This reduces the configuration effort.

Types and instances

Faceplates are based on a type-instance model to support the central changeability. You create central properties for an object in types. The instances represent local points of use of the types.

- Faceplate type
  You create a display and operating object according to your requirements and store it in the project library. The instances are bound to the respective faceplate type. If you change a property of a faceplate type, the property is saved centrally and also changed in all instances. In the faceplate type, you define the properties that can be changed on the faceplate. You edit a faceplate type in the "Faceplates" editor.
- Faceplate
  The faceplate is an instance of the faceplate type. You use a faceplate in screens as a display and operating object. You configure the variable properties of the faceplate type on the instance. You assign the tags of your project to the faceplate, for example. If you configure properties on the faceplate, you overwrite the properties of the faceplate type. The changes on the faceplate are saved at the point of use and have no effect on the faceplate type.

The figure below shows the relationships between the faceplate type and the faceplate. The background color of the I/O field is configured in the faceplate type in such a way that every instance can change the property. Blue is used as the background color in the faceplate type. You use yellow as the background color in the faceplate.



When you change the background color in the faceplate type, this has no effect on the faceplate because the "Background color" property is assigned on an instance-specific basis.

See also

Faceplate editor (Page 4059)

Creating a faceplate type (Page 4062)

Screen basics (Page 3889)

Basics of dynamizing faceplates (Page 4079)

Example: Configuring a faceplate (Page 4081)

12.1.7.2 Device dependency of faceplates

Device dependency of faceplates

Not all HMI devices support every display and operating object. The screen objects that are not available in the respective HMI device are not displayed when using the faceplate.

The following devices support faceplates:

Runtimes

- WinCC Runtime Advanced
- WinCC Runtime Professional

Comfort Panels

- KTP 400
- KP 400
- TP 700
- KP 700
- TP 900
- KP 900
- TP 1200
- KP 1200
- TP 1500
- KP 1500
- TP 1900
- TP 2200

Panels

- TP 277
- OP 277

Mobile Panels

- Mobile Panel 277
- Mobile Panel 277 IWLAN V2
- Mobile Panel 277F IWLAN V2
- Mobile Panel 277F IWLAN (RFID Tag)

Multi Panels

- MP 277
- MP 377

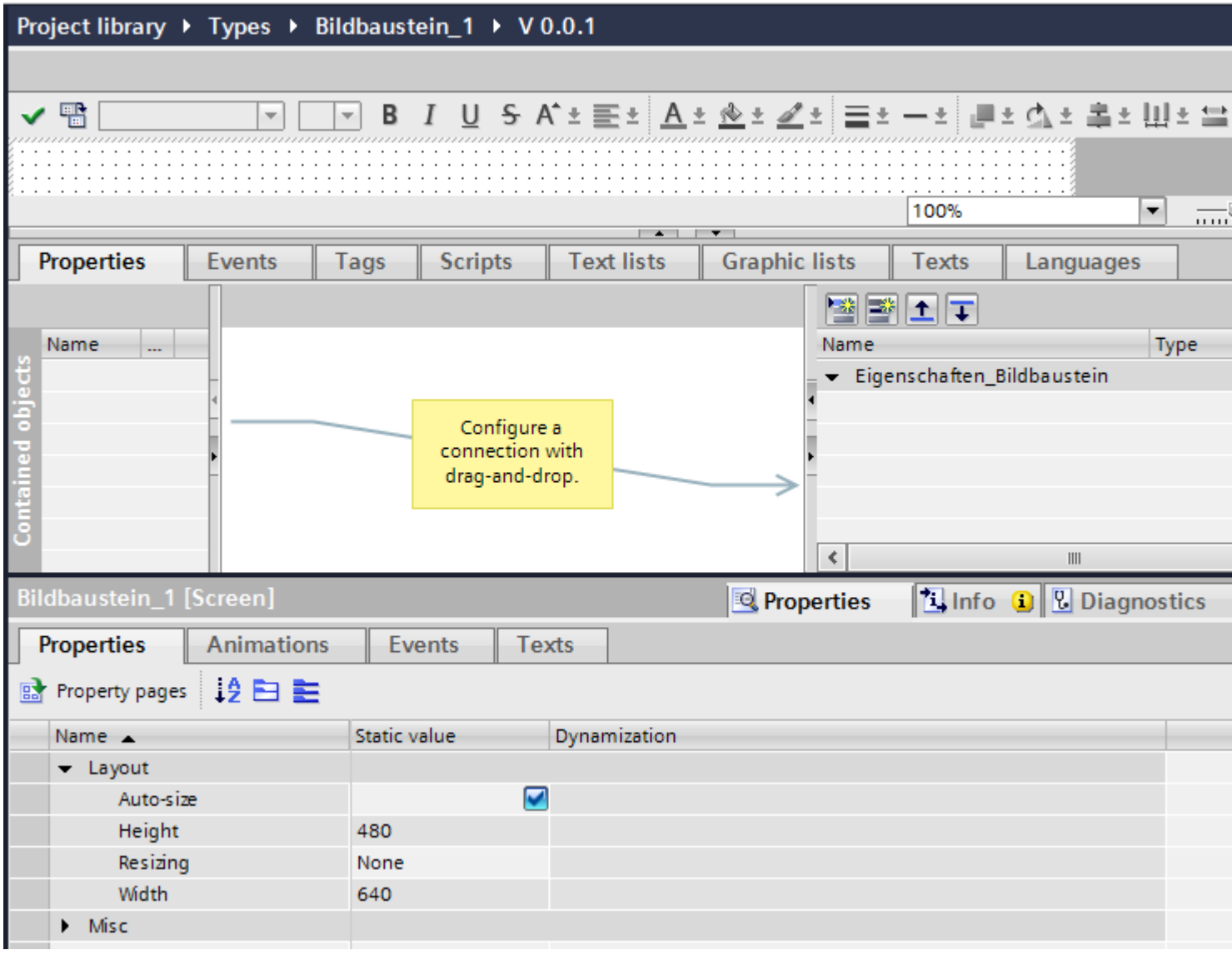
12.1.7.3 Faceplate editor

Open

Create faceplate types and edit them in the library view.

Layout

The library view for faceplate types consists of several editors, e.g. "Screens" and "Tags" which are arranged in the work area and the configuration area.



Work area

In the work area, place the objects contained in the faceplate type as you usually do in the "Screens" editor. You can remove objects, or use the "Toolbox" task card to add new objects.

Configuration section

- Properties
  The static and dynamic properties are identified as follows:
  - The symbol identifies a dynamic property. Dynamic implementation of this property, using tags or text and graphic lists.
  - The symbol identifies a static property. You change only the values of a static property.

    Note

    Device dependency of static properties

    Static properties are only available for panels and RT Advanced.

  You define the faceplate type properties in the "Properties" tab.
  You use the following two lists for this:
  - The "Contained objects" list
    This list includes the properties of the objects contained in the faceplate type. These properties can only be configured in the faceplate type in the "Faceplate" editor.
  - The "Interface" list contains the properties and tags of the faceplate type. The "Interface" list consists of the pre-defined category "Dynamic properties" and user-defined categories.
- Events
  You define the faceplate type events in the "Events" tab. You use the following two lists for this:
  - The "Contained objects" list contains the events of the objects embedded in the faceplate type.
  - The "Interface" list contains the events of the faceplate.
    You create links between the two lists using a drag and drop operation.
- Tags
  You can also create faceplate tags on the "Tags" tab. The tags are only available within the faceplate type. You can interconnect the faceplate type tags directly with an object contained in the faceplate type.
- Scripts
  In the "Scripts" tab, you can configure scripts for the faceplate type. In the script, you can call up system functions or program new functions, for example to convert values. The scripts are only available within the faceplate type. They work with VB Script code.

  Note

  The "Scripts" tab is not available for Runtime Professional.

- Text lists
  You can also create and edit texts lists for the faceplate type in the "Text lists" tab. These text lists are only available within the faceplate type. You can interconnect the text lists of the faceplate type directly with an object contained in the faceplate type, such as a symbolic I/O field.

- Graphics lists
  If necessary, you can also create graphics lists for the faceplate type in the "Graphics lists" tab. These graphic lists are only available within the faceplate type. You can interconnect the graphic lists of the faceplate type directly with an object contained in the faceplate type, e.g. a graphic I/O field.
- User texts
  The "User text" tab shows the contained user text of the opened faceplate type, for example entries from text lists. You can enter the compilations for the individual project languages directly in the "User text" tab.

See also

Basics on faceplates (Page 4057)

Example: Configuring a faceplate (Page 4081)

12.1.7.4 Creating and managing faceplates

Creating a faceplate type

Introduction

Faceplate types are display and operating objects which are made up of several objects, e.g. controller modules.

Requirement

You have opened a screen which contains multiple objects.

Procedure in the project view

1. Select all the objects that you need for the faceplate type.
2. Select the "Create faceplate type" command in the shortcut menu of the multiple selection. A dialog opens.
3. Enter a name for the faceplate.
4. If necessary, add a comment or change the version number.

Result

The library view opens. Two versions of the type are displayed in the "types" folder. Version 0.0.1 has been released and contains the currently selected objects.

The 0.0.2 version has the status "in progress". Edit the faceplate type in version 0.0.2 as required.

Procedure in the library view

1. The library view is open.
2. Select the "Add new type" command in the shortcut menu of the project library. A dialog opens.
3. Select the runtime for which the faceplate type is to be available. An empty faceplate type is created.
4. Drag-and-drop the objects from the "Toolbox" task card to the work area of the faceplate type.

Result

The new faceplate type is created and displayed under the selected name in the project library.

The faceplate type is assigned the status "in progress" and the version 0.0.1.

See also

Configuring a faceplate type (Page 4064)

Configuring a tag in the faceplate type (Page 4068)

Configuring an event in the faceplate type (Page 4069)

Configuring scripts in the faceplate type (Page 4071)

Creating an instance of the faceplate type (Page 4077)

Editing a faceplate type (Page 4075)

Basics on faceplates (Page 4057)

Example: Configuring a faceplate (Page 4081)

Editing the category and property of a faceplate type (Page 4067)

Configuring a graphics list in the faceplate type (Page 4073)

Configuring text lists in the faceplate type (Page 4072)

Removing faceplate from faceplate type (Page 4078)

Resizing a faceplate (Page 4077)

Configuring a faceplate type

Introduction

In the configuration area of the "Faceplates" editor you define which properties and process values of the objects contained in the faceplate can be configured in the "Properties" tab.

Note

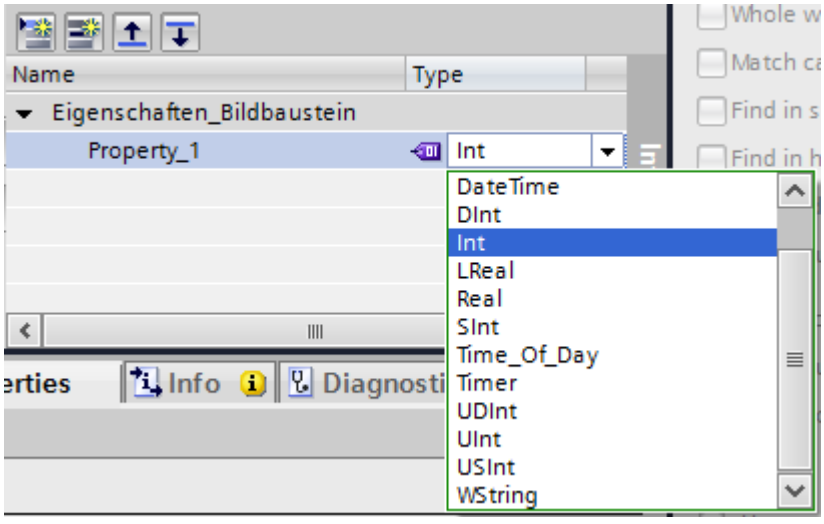
For tooltip texts configured for objects in the faceplate to be displayed in Runtime, insert the property "Tooltip text" as faceplate property.

Requirement

- The faceplate is generated and is displayed in the "Faceplate" editor.
- The "Properties" tab is open in the configuration area.

Procedure

1. To create a new property, click the "Add property" button in the interface list.
   A new property is displayed in the "Interface" list.
2. To create a new category, click the "Add category" button in the interface list.
   A new category including a new property is displayed in the "Interface" list.
3. Click the name of the property and assign a name, for example, "Color".
4. Select the data type.



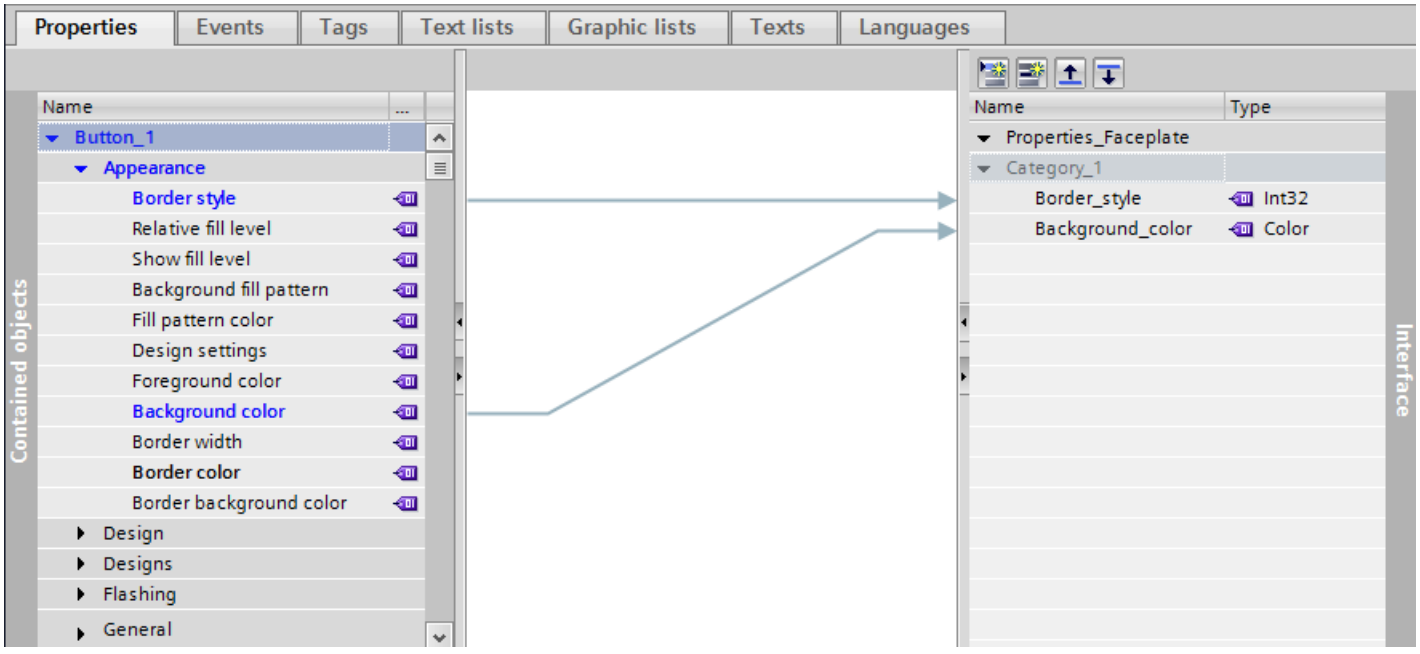
Interim result

You have created a new property.

1. Click an object in the "Contained objects" list.
2. Select a property, e.g. "Appearance > Background color".
3. Move the property from the "Contained objects" list to the desired property in the "Interface" list by drag&drop.
   The connection is displayed as a colored line.
   The system assigns the same data type to the property in the "Interface" list as that of the property in the "Contained objects" list.
   To change the data type, select the data type from the data types of the connected property.
4. Repeat step 3 to connect other properties of the contained properties with the interface.

Note

The property from the "Contained objects" list must be in concordance with the data type of the already connected property of the "Interface" list.



Deleting a connection

1. Click the connection that you wish to delete.
2. In the shortcut menu select the "Delete" command or use the <DEL> key.
   The connection is deleted.

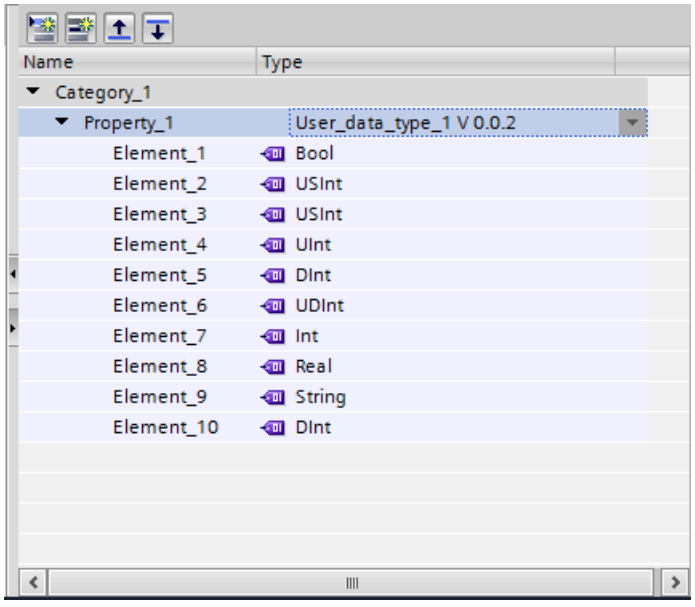
Interconnecting user data types with faceplates

As of V13 SP1, you can use the HMI user data types and the PLC user data types as properties of the faceplates.

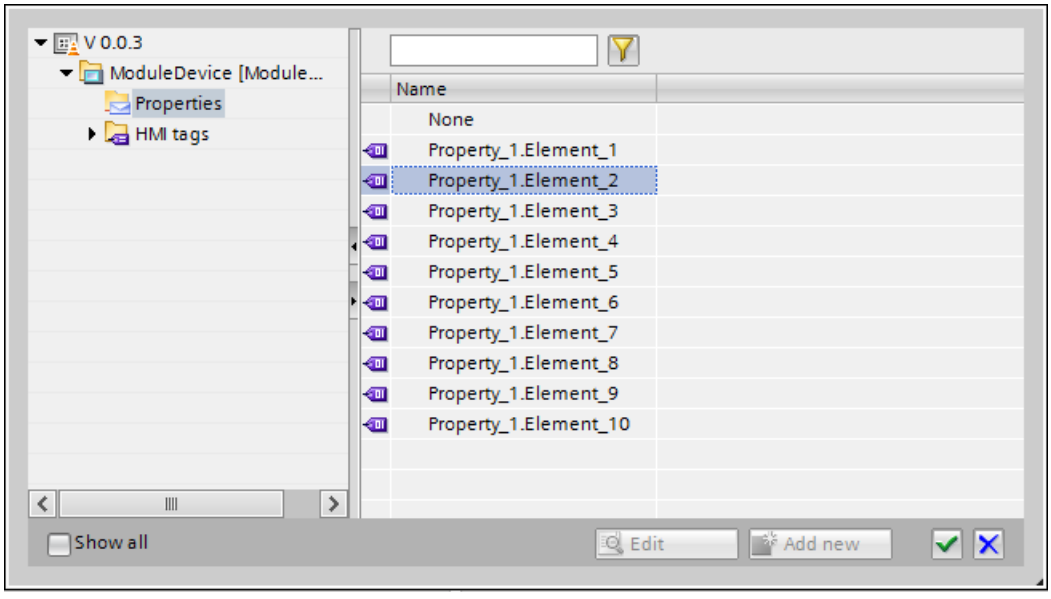
You interconnect the HMI user data types and the PLC user data types with the faceplates using interface properties by connecting the faceplates with the structural elements of the interface properties.

To set the HMI user data types and the PLC user data types as properties of the faceplates, follow these steps:

1. Select the user data type in the selection list under "Type".



2. In the Inspector window under "Properties > Properties > General", select the interface property of the faceplate in the "Tag" selection list.



The faceplate is interconnected with the user data type.

Result

You have created a new property or a new category with a property. You have connected the property of the interface with a property of the contained objects. If you use an instance of the faceplate type in a screen, you configure the properties of the faceplate which you have created in the "Interface" list.

Note

Device dependency of static properties

Static properties are only available for panels and RT Advanced. You work with dynamic properties in RT Professional.

See also

Creating a faceplate type (Page 4062)

Example: Configuring a faceplate (Page 4081)

Editing the category and property of a faceplate type

Deleting a property in the "Interface" list

1. Select "Edit faceplate" in the shortcut menu. The "Faceplates" editor is open.
2. Select the property which you want to delete in the "Interface" list.
3. Select "Delete" in the shortcut menu.

The property is deleted from the "Interface" list. Existing connections to the contained objects are deleted. All linked faceplates lose their instance-specific property.

Deleting a category in the "Interface" list

1. Select "Edit faceplate" in the shortcut menu. The "Faceplates" editor is open.
2. Select the category which you want to delete in the "Interface" list.
3. Select "Delete" in the shortcut menu.

The category including the properties is deleted from the "Interface" list. Existing connections to the contained objects are deleted. All linked faceplates lose their instance-specific properties.

Moving a property to another category

1. Select a property in the "Interface" list.
2. Move the property to a new category by "drag&drop".

You have moved the property to a new category.

Changing the data type of a property

1. Select a property in the "Interface" list.
2. Click the second column of the drop-down list.
3. Select a data type.

You have changed the data type of the property in the "Interface" list.

See also

Creating a faceplate type (Page 4062)

Configuring a tag in the faceplate type

Introduction

You connect a tag in the faceplate type directly with the properties of the objects contained in the faceplate type. The tags in the faceplate type are a central means of defining dynamic properties in the faceplate type.

The tags of a faceplate type have limited functionality. The "Tags" tab of the configuration area has the same structure as the "Tags" editor.

Array elements for faceplates

In the following situations you may not assign an array element or a multiplex tag that is interconnected with a faceplate to the property of a screen object.

- The property is configured in a faceplate as parameter of a system function.
- System functions or scripts are assigned in the faceplate to events of the property.

Requirements

- The library view is open.
- The faceplate type is created and has the "in progress" status.
- The "Tags" tab is open in the configuration area.

Procedure

1. Click "Add" in the "HMI tags" table. A new tag is created in the faceplate type.
2. Open "Properties > Events" in the inspector window if necessary. You configure the "Value change" event on the tag for example.

3. In the work area, select the object to which you want to assign this tag, e.g. an I/O field.
4. Select the tag in the inspector window of the I/O field "Properties > Properties > General > Process > Tag".

Note

Only tags of the faceplate type are displayed in the object list in the "Faceplates" editor.

Result

You have created a tag in the faceplate type. You are using the tag in the faceplate for scripting, for example.

See also

Creating a faceplate type (Page 4062)

Example: Configuring a faceplate (Page 4081)

Configuring an event in the faceplate type

Introduction

In the configuration area of the "Faceplates" editor you define which events of the objects contained in the faceplate can be configured in the "Events" tab. You configure a function list on the events of the faceplate in the "Screens" editor.

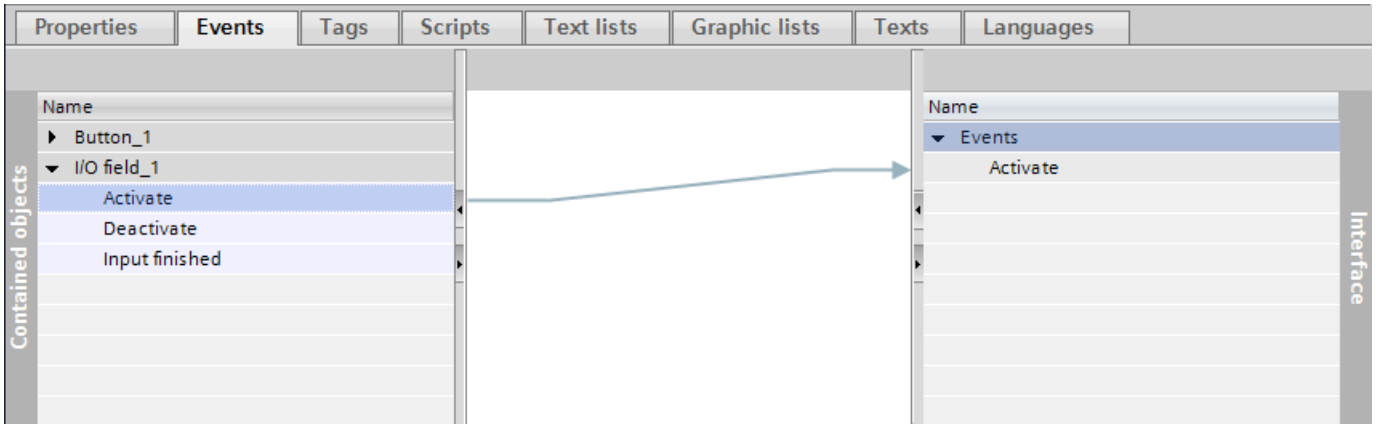
Requirement

- The library view is open.
- The faceplate type is created and has the "in progress" status.
- The "Events" tab of the configuration area is open.

Procedure

1. Click an object in the "Contained objects" list.
2. Select an event, e.g. "Activate".

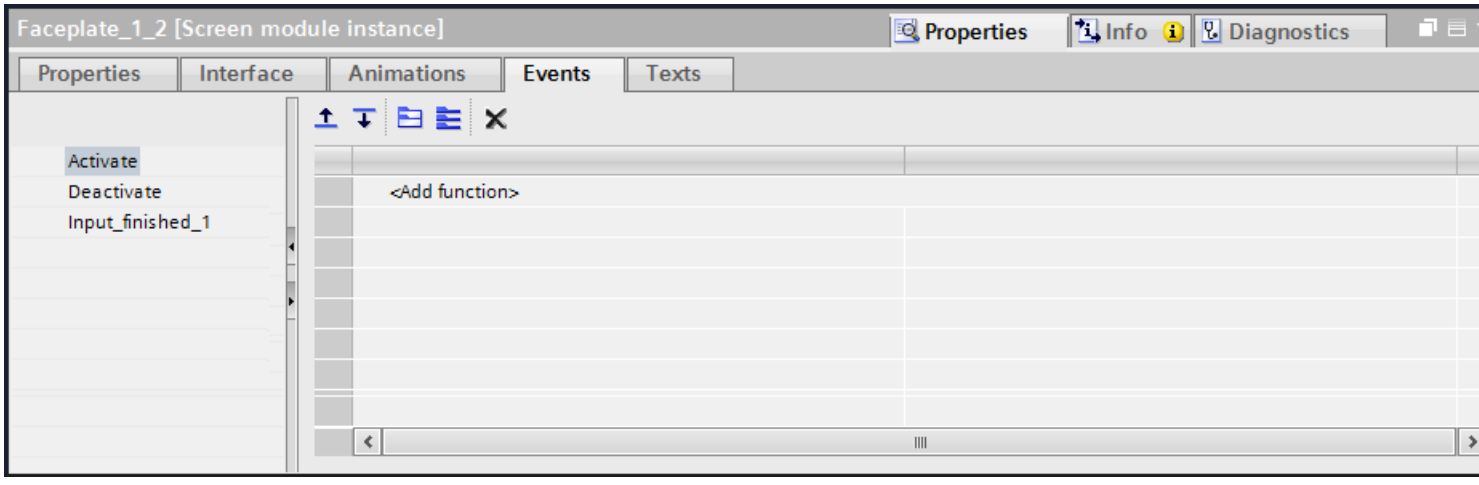
3. Move the event from the "Contained objects" list to a new category in the "Interface" list.
   The new event is displayed in the "Interface" list.
   The connection appears as a colored line.



4. Double-click the event and, if required, change the name of the event.

Result

You have created an event in the faceplate type. if you use an instance of the faceplate type, you will only be offered the configured events in the inspector window. You configure a function list on the event in the faceplate.



See also

Creating a faceplate type (Page 4062)

Example: Configuring a faceplate (Page 4081)

Configuring scripts in the faceplate type

Introduction

In the configuration area of the "Faceplates" editor you create scripts which you only use within a faceplate type. You can only refer to tags of the faceplate type or properties of the contained objects within the script. The script is used as a copy in the instance of the faceplate type.

Requirement

- The library view is open.
- The faceplate type is created and has the "in progress" status.
- Faceplate tags or dynamic properties have been created.

Procedure

1. Click the "Scripts" tab in the configuration area of the faceplate.
2. Double click "Faceplate scripts > Add VB script".
3. Write the program code.
4. Press the shortcut keys <CTRL+J> to Include tags of the faceplate type in the script. The object list opens.
5. Click "HMI tags".
   All tags of the faceplate type are displayed in the object list.
6. Select a tag and confirm the selection.
7. Press the shortcut keys <CTRL+J> to use properties of the faceplate type in the script. The object list opens.
8. Click "Properties".
   All dynamic properties of the "Interface" list for the faceplate type are displayed in the object list.

Result

You have created a script in the faceplate type. The instance of the faceplate type uses a copy of the script. The script is executed in runtime depending on the configuration.

See also

Creating a faceplate type (Page 4062)

Example: Configuring a faceplate (Page 4081)

Configuring text lists in the faceplate type

Introduction

In a faceplate type, you connect a text list with an object included in the faceplate type. Text is assigned to the values of a tag in a text list. The text list is used, for example, to display a selection list in a symbolic I/O field contained in the faceplate type. The interface between the text list and a tag of the faceplate type is configured at the object that uses the text list.

Requirements

- The library view is open.
- The faceplate type is created and has the "in progress" status.
- The "Text list" tab is open in the configuration range.
- A tag has been created in the faceplate type.

Procedure

1. Click "Add" in the "Text lists" table. A new text list is created in the faceplate type.
2. Assign a name to the text list that indicates its function.
3. Select the text list type, e.g. "Value/Range" under "Select".
4. Click "Add" in the "Text list entries" table.
5. Define the values or range of values for the text list.
6. Enter a text for every value range which is displayed in Runtime when the tag is within the specified value range.
7. Select an object, for example a button in the work area of the faceplate type.
8. In the Inspector window of the object, enable "Properties > Properties > General > Label > Text list".
9. Select the "Text list" from the selection list.
10. Select a tag of the faceplate type under "Process > Tag".

Result

You have created a text list in the faceplate type. This text list is linked to an object contained in the faceplate type. You have configured a faceplate type tag at the object.

See also

Creating a faceplate type (Page 4062)

Configuring a graphics list in the faceplate type

Introduction

In a graphics list, specific graphics are assigned to possible values of a tag. In a faceplate type, you connect a graphics list with an object included in the faceplate type. Graphics are assigned to the values of a tag in a graphics list. For example, the graphics list is used to display a selection list in a graphic I/O field contained in the faceplate type. The interface between the graphics list and a tag of the faceplate type is configured at the object that uses the graphics list.

Requirements

- The library view is open.
- The faceplate type is created and has the "in progress" status.
- The "Graphics list" tab is open in the configuration range.
- A tag has been created in the faceplate type.

Procedure

1. Click "Add" in the "Graphics lists" table. A new graphics list is created in the faceplate type.
2. Assign a functional name to the graphics list.
3. Select a graphics list type, e.g. "Range (... - ...)", under "Selection".
4. Click "Add" in the "Graphics list entries" table. A new entry is created in the list.
5. Define the values or range of values for the graphics list.
6. Select a graphic for every value or value range which is displayed in runtime when the tag is within the specified value range.
7. Select an object, e.g. a graphic I/O field, in the work area of the faceplate type.
8. Activate "Properties > Properties > General > Contents > Graphics list" in the object's Inspector window.
9. Select the "Graphics list" from the selection list.
10. Select a tag of the faceplate type in "Process > Tag".

Result

You have created a graphics list in the faceplate type. This graphics list is linked to an object contained in the faceplate type. You have configured a faceplate type tag at the object.

See also

Creating a faceplate type (Page 4062)

Translating texts directly in the faceplate type

Translating texts

If you use several languages in your project, you can translate individual texts directly. As soon as you change the language of the software user interface, the translated texts are available in the selected language.

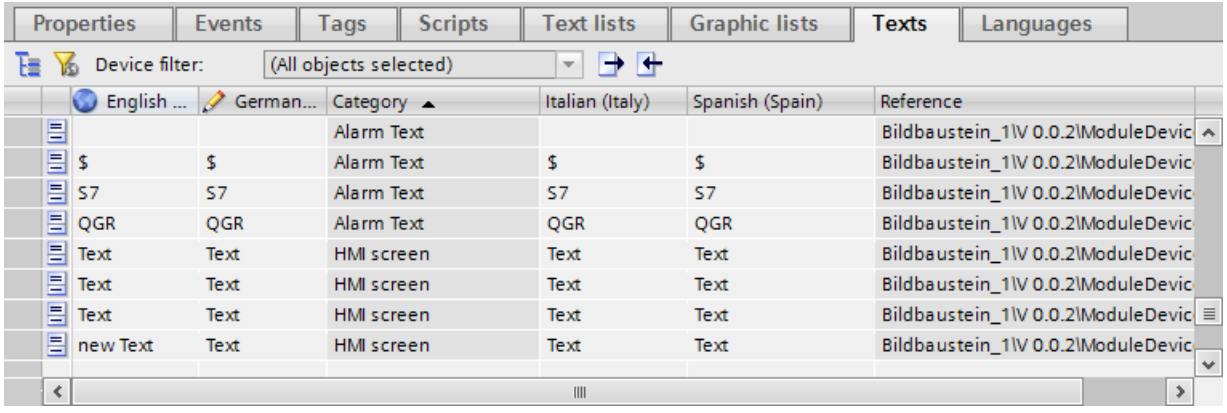
Requirements

- A project is open.
- You have opened the "Faceplates" editor.
- You have created a faceplate type.
- The faceplate type contains at least one object with text, e.g. a text field.

Procedure

Proceed as follows to translate individual texts:

1. Open the "Tasks" task card.
2. Click the button under "Languages & Resources > Editing language". The "Project languages" editor opens.
3. Activate at least two project languages.
4. Open the "Texts" tab in the "Faceplates" editor.
   A list with the texts in the project is displayed in the work area. There is a separate column for each project language.



5. Click on the button in the toolbar to group identical texts together.
6. To hide texts that do not have a translation, click on the button in the toolbar.
7. Click on an empty column and enter the translation.

Result

You have translated individual texts in the "Texts" tab of the faceplate type. The texts will then be displayed in the runtime language.

Editing a faceplate type

Introduction

All faceplate types in the project library have a status. You create a faceplate type in the "in progress" status. You can edit the faceplate types as required in this status. When editing is complete, release the faceplate type.

Requirement

- A faceplate type has been created.
- The faceplate type has the version 0.0.1. and the status "in progress".
- The "Library" task card or the library view is open.

Enable faceplate type

1. Select version 0.01 of the faceplate type in the project library.
2. Select "Release version" in the shortcut menu.

You have released version 0.0.1 of the faceplate type.

Edit faceplate type

1. Select, for example, the released version 0.0.1 of a faceplate type in the project library.
2. Select "Edit faceplate type" in the shortcut menu.

The library view opens. The new version 0.0.2 of the faceplate type has been created.

The faceplate type has the "In progress" status.

Restoring the last version of the faceplate type

The last released version of the faceplate type is version 0.0.2.

Edit the faceplate type. A new version 0.0.3 is created and the "in progress" status is assigned to it.

1. Select the faceplate type from the project library.
2. Select "Discard changes and restore type" in the shortcut menu.

All changes to the faceplate type since the last enabling operation are rejected. The faceplate type is enabled again and has the version 0.0.2.

See also

Creating a faceplate type (Page 4062)

Example: Configuring a faceplate (Page 4081)

Duplicating faceplate type

Introduction

You can duplicate a released faceplate type and a released version of the faceplate type in the project library.

Requirement

- A faceplate type has been created.
- The faceplate type is released and no version of the faceplate type has the "in progress" status.
- The "Libraries" task card or the library view is open.

Duplicating faceplate type

1. Select a released faceplate type from the project library.
2. Select "Duplicate faceplate type" in the shortcut menu.
   A dialog opens.
3. Enter a name for the new faceplate type.
4. If necessary, add a comment or change the version number.
   The new faceplate type is in the same folder as the original.

Duplicating a version of a faceplate type

1. Select a released version of a faceplate type in the project library.
2. Select "Duplicate faceplate type" in the shortcut menu.
   A dialog opens.
3. Enter a name for the new faceplate type.
4. If necessary, add a comment or change the version number.
   The new faceplate type is in the same folder as the original.

See also

Editing a faceplate type (Page 4075)

Creating a faceplate type (Page 4062)

Resizing a faceplate

Introduction

In the faceplate type, specify the response of the faceplate when it is resized.

Requirements

- A faceplate type has been created.
- You have opened the "Faceplates" editor.
- The Inspector window is open.

Specifying the response to resizing

1. Activate "Properties > Properties > Layout > Fit to size > Auto-size".
2. Select the "Fixed aspect ratio" setting, for example.
   This setting retains the aspect ratio when you resize the faceplate.
3. Click the faceplate in the "Libraries" task card.
4. Select "Release version" in the shortcut menu.

Result

You have specified how the faceplate responds when its size in a screen is changed. You can change the response for a particular faceplate.

Select a faceplate in the screen. Select the required setting in "Properties > Properties > Layout > Characteristics" in the Inspector window.

See also

Creating a faceplate type (Page 4062)

Creating an instance of the faceplate type

Introduction

The faceplate type is stored in the project library. If you use the faceplate type in a screen, you create an instance of the faceplate type.

Note

Note that a faceplate is always configured for a particular class of HMI devices. For example, you cannot use a faceplate type that is configured for "RT Professional" in a screen on an "RT Advanced" HMI device.

Note

The number of faceplate instances per screen is not limited. However, you should be aware of the fact that the number of faceplate instances used, or the use of scripts in the faceplate instances, will have an impact on performance.

Requirement

- A screen is open.
- The "Libraries" task card is opened.
- The project library is open and it contains at least one faceplate type.

  Note

  Faceplate types are also stored in global libraries. When you add a faceplate type from the global library to the screen, the system automatically saves a copy of it to the project library.

- The Inspector window is open.

Procedure

1. Move the desired faceplate type from a library to the screen by drag&drop.

Result

You have created an instance of the faceplate type. The instance is given the version number of the most recently released type.

Configure the properties of the faceplate in the Inspector window as required. To dynamize the faceplate, open the interface in the Inspector window.

See also

Creating a faceplate type (Page 4062)

Example: Configuring a faceplate (Page 4081)

Removing faceplate from faceplate type

Introduction

Cancel application, to remove specified instances from the update of the faceplate type.

Requirement

- A faceplate type has been created.
- The faceplate must be used in a minimum of one screen.

Disconnect the faceplate object from faceplate type.

1. Select a faceplate on the screen.
2. In the shortcut menu, select "Disconnect faceplate object from faceplate type".

Result

The faceplate is independent of the associated faceplate type. Changes to the faceplate type are not updated in the faceplate.

See also

Creating a faceplate type (Page 4062)

12.1.7.5 Dynamizing faceplates

Basics of dynamizing faceplates

Application

You can dynamically control events and properties of faceplates in two ways:

- Dynamically controlling faceplates
  You configure the events or dynamic properties individually for the application point on the faceplate. To do this, define in the "Faceplates" editor that these properties and events are configurable in the faceplate. To dynamize the faceplates in the "Screens" editor, use the scripts and tags that are created in the project.
- Dynamizing the faceplate type
  You dynamize the objects which are contained in the faceplate type in the "Faceplates" editor. You configure the individual objects as in the "Screens" editor. Tags and scripts are available in the "Faceplates" editor to dynamize properties and events. You do not have access to the tags and scripts of the project within the faceplate.
  Every faceplate created with the faceplate type has the same preconfigured dynamization. You can edit this dynamic control only in the "Faceplates" editor.

Animation

WinCC provides preconfigured dynamic control in the "Animations" task card. You use animations to dynamize faceplates and faceplate types.

See also

Basics on faceplates (Page 4057)

Example: Configuring a faceplate (Page 4081)

Dynamizing faceplates

Introduction

You dynamize a faceplate in exactly the same way as you dynamize an object from the "Tools" task card.

You connect the dynamic properties of the faceplate in the "Screens" editior with a tag or a script which supplies values to the property in runtime.

You Include tags and scripts which you have created in the project for the faceplate.

Requirement

- A faceplate is inserted in the screen.

Procedure

1. Select the faceplate.
2. In the Inspector window, open "Properties > Animation".
   The animations available for the selected object are displayed.
3. Select "Horizontal movement" and click the button.
   The parameters of the animation are displayed.
   A transparent copy of the object is shown in the work area, which is connected to the source object by means of an arrow.
4. Select a tag for control of movement.
5. Move the object copy to the relevant destination. The system automatically enters the pixel values of the final position in the Inspector window.
6. Customize the range of values for the tag as required.

Result

In Runtime, the object moves in response to every change in value of the tag that is used to control the movement. The direction of movement corresponds to the configured type of movement "horizontal".

12.1.7.6 Examples of faceplates

Example: Configuring a faceplate

Introduction

In this example you create a faceplate in which you can convert the value of a length measuring system from kilometers into meters. The length in meters is displayed in an output field.

Procedures overview

The example is divided into the following steps:

1. Creating a faceplate type
2. Configuring a faceplate type
3. Creating a script in the faceplate type
4. Connecting a script with the contained objects of the faceplate type
5. Creating a faceplate and connecting with a tag

See also

Basics on faceplates (Page 4057)

Example: Creating an instance of the faceplate type (Page 4086)

Example: Configuring included objects (Page 4085)

Example: Creating a script in the faceplate type (Page 4083)

Example: Configuring a faceplate type (Page 4082)

Example: Creating a faceplate type (Page 4081)

Example: Creating a faceplate type

Task

You create a faceplate type.

Requirement

- A screen is open.
- The toolbox window is displayed.

Settings

For the example you require objects with the following settings:

Object	Object name	Properties
Text field	Label_Meter	Text: Meter
I/O field	Output_Meter	Mode: Output
Button	KM_to_Meter	Text OFF: Convert

Procedure

1. In the toolbox, click the individual objects and drag&drop the objects into the screen.
2. Set the properties as shown above.
3. Select all objects.
4. Select the "Create faceplate type" command in the shortcut menu of the multiple selection. The "Add type" window opens.
5. Enter "KMtoMeter" as type name.

Result

The faceplate type appears in the project library under the name "KMtoMeter". The faceplate type is assigned the "in progress" status.

See also

Example: Configuring a faceplate (Page 4081)

Example: Configuring a faceplate type

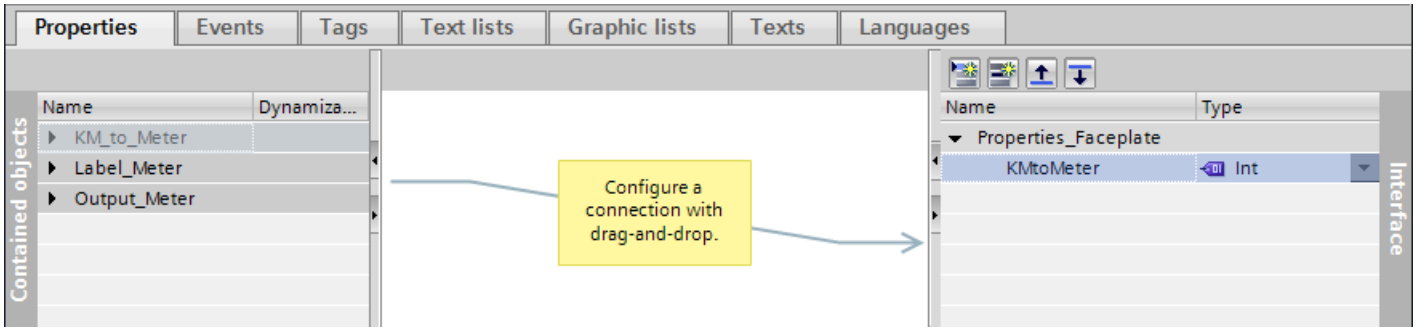
Job

You create the dynamic property and tag of the faceplate type.

Creating a new property

1. Click "Properties" in the configuration area of the faceplate.
2. In the "Interface" list, click the icon "Add property".
   A new property is added to the "Interface" list.

3. Double click the name of the property and enter "KMToMeter".
4. Click the selection list and select the data type "Int".



Interim result

The "KMToMeter" property is displayed in the "Interface" list. The symbol identifies the dynamic property. The property will be linked to a tag in the next step.

Creating a tag in the faceplate type

1. Click "Tags" in the configuration area.
2. Click "Add" in the "HMI tags" table. A new tag is added to the table. The inspector window of the tag is opened.
3. In the Inspector window, select "Properties > Properties > General":
4. Enter "BB_Tag" as the name. Select the "Int" data type.

Result

The "KMToMeter" property and the "BB_Tag" of the faceplate type are created in the configuration area of the "Faceplates" editor.

See also

Example: Configuring a faceplate (Page 4081)

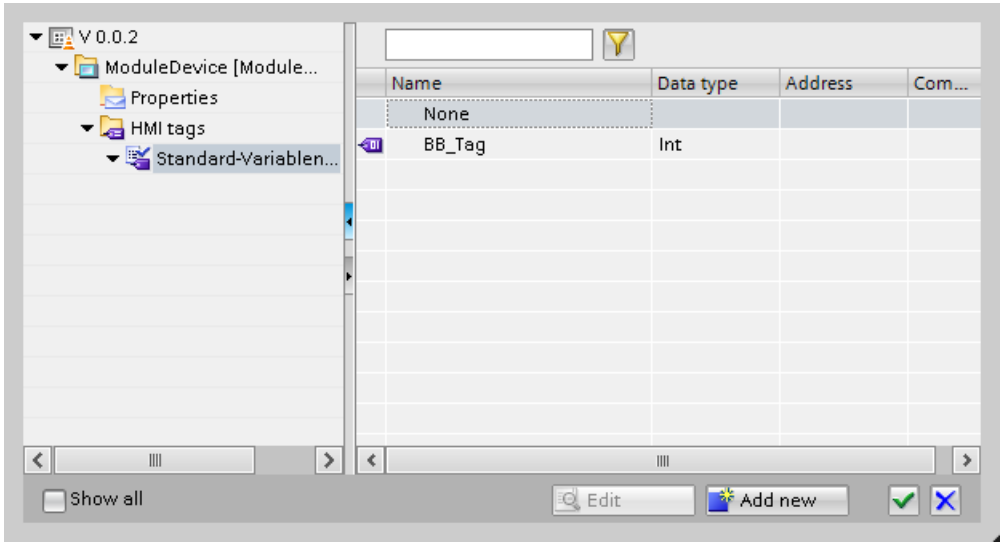
Example: Creating a script in the faceplate type

Task

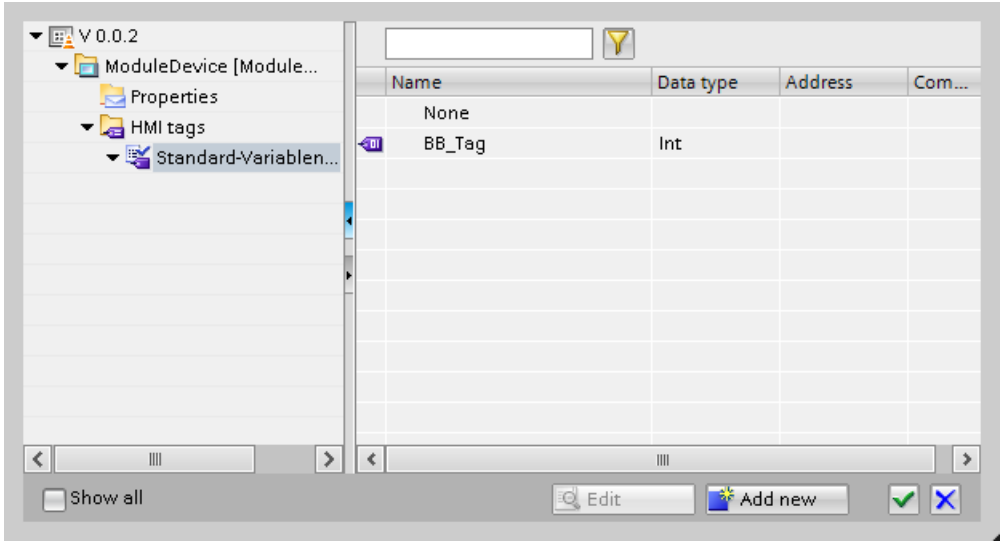
You create a script that converts the value of a length measuring system from kilometers to meters.

Procedure

1. Click "Scripts > Faceplate scripts" in the configuration area.
2. Double click "Add VB script".
3. Press the shortcut keys <CTRL+J>.
   The object list opens.
4. Click "HMI tags".
   The faceplate tag "BB_Tag" is displayed in the object list.

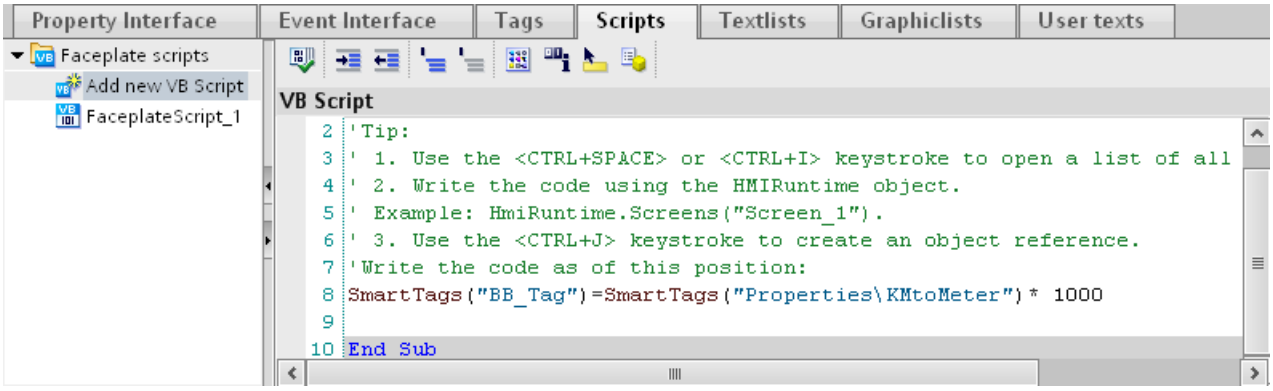


5. Select the faceplate type tag "BB_Tag". Confirm the selection.
6. Insert an "=" sign.
7. Press the shortcut keys <CTRL+J>. The object list is opened.
8. Click "Properties."
   The dynamic property "KMtoMeter" is displayed in the object list.



9. Select the "KMToMeter" dynamic property. Confirm the selection.

10. Enter "*1000" at the end of the code. This corresponds to the conversion factor from kilometers to meters.
11. Enter the name "Script_1" under "Properties > Properties > General >Setting > Name" in the Inspector window.



Result

The script has been created in the faceplate type. If you use the "KMtoMeter" faceplate type in a screen, you assign a tag to the "KMtoMeter" property. The value of this tag is multiplied by factor 1000 and assigned to the tag of the faceplate type "BB_Tag" as a new value. In this way you supply values to the tags of the faceplate type.

See also

Example: Configuring a faceplate (Page 4081)

Example: Configuring included objects

Task

You connect the included objects in the faceplate "KMtoMeter" to a faceplate tag and a faceplate script.

You connect the I/O field with the "BB_Var" tag of the faceplate type. You connect the "Click" event for a button to the script "Script_1".

Requirement

The faceplate is displayed in the "Faceplates" editor.

Interconnecting I/O field with a tag

1. Select the "Output_Meter" I/O field in the faceplate.
2. Connect the I/O field with the faceplate tag "BB_Tag" in the "General > Process > Tag" inspector window.

Connecting an event to a faceplate script

1. Select the "KM_to_Meter" button in the faceplate.
2. Select "Script_1" in the inspector window under "Events > Click".

Result

The I/O field is connected with the tag of the faceplate type. The button is connected with the script of the faceplate type.

If you click the button of the faceplate during runtime, the script is executed. The value of the tag of the faceplate is output in the I/O field.

See also

Example: Configuring a faceplate (Page 4081)

Example: Creating an instance of the faceplate type

Task

You insert the faceplate type "KMtoMeter" in a screen and assign a tag to the dynamic property "KMtoMeter".

Requirement

- The "Screens" editor is open.
- A new screen has been created.

Settings

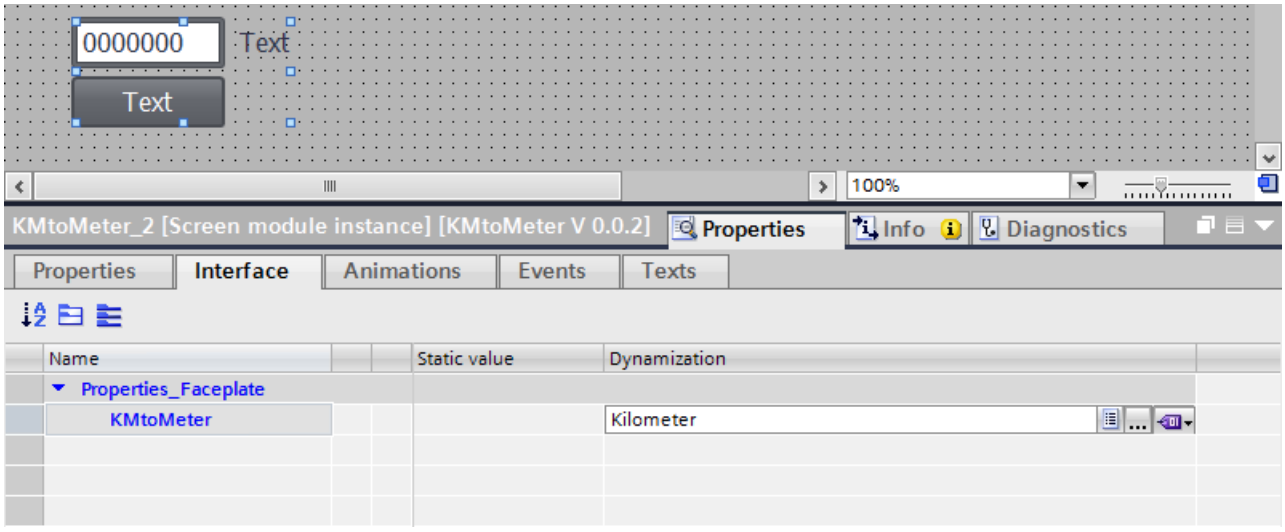
For the example you require a tag with the following settings:

Name	PLC connection	Type
Kilometer	No	ULong

Procedure

1. Create the HMI tag "Kilometer" with the settings named above.
2. Open the "project library" in the "Libraries" task card.
3. Move the "KMtoMeter" faceplate type into the screen by drag&drop.

4. Click "Properties > Properties > Interface" in the Inspector window.
   The table shows all the properties which you can dynamize in the faceplate.



5. Click the "Dynamization" column in the "KMtoMeter" line.
6. Select "HMI tag".
7. Click the "..." button.
8. Select the tag "Kilometer" from the object list.
9. Confirm the selection.

Result

The dynamic property "KMtoMeter" is linked with the "Kilometer" tag. During runtime the dynamic property is supplied with values of the "Kilometer" tag. The values are multiplied by factor 1000 in the script of the faceplate type and assigned to the tag in the faceplate as a new value.

See also

Example: Configuring a faceplate (Page 4081)

12.1.8 Display and operating elements

12.1.8.1 Device dependency of the objects

Objects for Basic Panels

Availability of display and operating objects for Basic Panels

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and operating objects for the Basic Panels.

On devices with the device version earlier than V13, configure simple display objects. On devices with the device version V13 or later, configure table-based display objects.

Overview

	KP300 Basic KP400 Basic	KTP400 Basic KTP600 Basic KTP1000 Basic TP1500 Basic	KTP700 Basic PN KTP900 Basic	KTP700 Basic DP KTP1200 Basic DP	KTP1200 Basic PN
Bar	Yes	Yes	Yes	Yes	Yes
User view	Yes	Yes	Yes [1]	Yes [1]	Yes [1]
Date/time field	Yes	Yes	Yes	Yes	Yes
I/O field	Yes	Yes	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes	Yes	Yes
Graphic view	Yes	Yes	Yes	Yes	Yes
Graphic I/O field	Yes	Yes	Yes	Yes	Yes
Help indicator	Yes	No	No	No	No
HTML Browser	No	No	Yes	No	Yes
Circle	Yes	Yes	Yes	Yes	Yes
Trend view	Yes	Yes	Yes [1]	Yes [1]	Yes [1]
Line	Yes	Yes	Yes	Yes	Yes
Alarm view Alarm window	Yes	Yes	Yes [1]	Yes [1]	Yes [1]
Alarm indicator	Yes	Yes	Yes	Yes	Yes
Rectangle	Yes	Yes	Yes	Yes	Yes
Recipe view	Yes	Yes	Yes [1]	Yes [1]	Yes [1]
Button	Yes	Yes	Yes	Yes	Yes
Switch	Yes	Yes	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes	Yes	Yes
System diagnostics view	Yes	Yes	Yes [1]	Yes [1]	Yes [1]
Text field	Yes	Yes	Yes	Yes	Yes

1) The objects are displayed on the panels as table-based objects.

See also

Objects for Panels (Page 4089)

Objects for Comfort Panels (Page 4090)

Objects for Multi Panels (Page 4092)

Objects for Mobile Panels (Page 4094)

Objects for WinCC Runtime Advanced (Page 4096)

Objects for Panels

Availability of display and operating objects for Panels

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and operating objects for the Panels.

Overview

	OP 73	OP 77A OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Bar	Yes	Yes	Yes	Yes	Yes
User view	Simple	Simple	Simple	Yes	Yes
Date/time field	Yes	Yes	Yes	Yes	Yes
I/O field	Yes	Yes	Yes	Yes	Yes
Ellipse	No	No	Yes	Yes	Yes
f(x) trend view	No	No	No	No	No
Function key	only on keyboard units				
Graphic view	Yes	Yes	Yes	Yes	Yes
Graphic I/O field	Yes [1)]	Yes [1)]	Yes [1)]	Yes	Yes
Handwheel	No	No	No	No	No
Help indicator	Yes	No	No	No	No
HTML browser	No	No	No	No	No
Circle	No	No	Yes	Yes	Yes
Trend view	No	No	Yes	Yes	Yes
Charging condition	No	No	No	No	No
Illuminated pushbutton	No	No	No	No	No
Line	No	No	Yes	Yes	Yes
Media Player	No	No	No	No	No
Alarm view Alarm window	Simple	Simple	Simple	Yes	Yes

	OP 73	OP 77A OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Alarm indicator	Simple	No	Simple	Yes	Yes
Polygon	No	No	No	Yes	Yes
Polyline	No	No	No	Yes	Yes
Rectangle	No	No	Yes	Yes	Yes
Recipe view	No	Simple	Simple	Yes	Yes
Button	Yes	Yes	Yes	Yes	Yes
Switch	Yes [2)]	Yes [2)]	Yes [2)]	Yes	Yes
Slider	No	No	No	Yes	Yes
Key switch	No	No	No	No	No
Sm@rtClient view	No	No	No	Yes	Yes
Status/Force	No	No	No	Yes	Yes
Symbol library	No	No	No	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes	Yes	Yes
System diagnostics view	No	No	No	No	No
System diagnostics window	No	No	No	No	No
Text field	Yes	Yes	Yes	Yes	Yes
Clock	No	No	No	No	No
Effective range signal	No	No	No	No	No
Effective range name	No	No	No	No	No
Effective range name (RFID)	No	No	No	No	No
WLAN reception	No	No	No	No	No
Gauge	No	No	Yes	Yes	Yes
Zone name	No	No	No	No	No
Zone signal	No	No	No	No	No

1) Only two different graphics can be used.
2) Only the "Switch with text" and "Switch with graphic" types can be used.

See also

Objects for Basic Panels (Page 4088)

Objects for Comfort Panels

Availability of display and operating elements for Comfort Panels

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and control objects on Comfort Panels.

Overview

	KP400 Comfort KTP400 Comfort	KP700 Comfort TP700 Comfort	KP900 Comfort TP900 Comfort	KP1200 Comfort TP1200 Comfort	TP1900 Comfort TP2200 Comfort
Bar	Yes	Yes	Yes	Yes	Yes
User view	Yes	Yes	Yes	Yes	Yes
Date/time field	Yes	Yes	Yes	Yes	Yes
I/O field	Yes	Yes	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes	Yes	Yes
f(x) trend view	Yes	Yes	Yes	Yes	Yes
Function key	only on keyboard units				
Graphic view	Yes	Yes	Yes	Yes	Yes
Graphic I/O field	Yes	Yes	Yes	Yes	Yes
Handwheel	No	No	No	No	No
Help indicator	No	No	No	No	No
HTML Browser	No	Yes	Yes	Yes	Yes
Camera view	Yes	Yes	Yes	Yes	Yes
Circle	Yes	Yes	Yes	Yes	Yes
Trend view	Yes	Yes	Yes	Yes	Yes
Charge condition	No	No	No	No	No
Illuminated push-button	No	No	No	No	No
Line	Yes	Yes	Yes	Yes	Yes
Media Player	Yes	Yes	Yes	Yes	Yes
Alarm view Alarm window	Yes	Yes	Yes	Yes	Yes
Alarm indicator	Yes	Yes	Yes	Yes	Yes
PDF viewer	Yes	Yes	Yes	Yes	Yes
Polygon	Yes	Yes	Yes	Yes	Yes
Polyline	Yes	Yes	Yes	Yes	Yes
Rectangle	Yes	Yes	Yes	Yes	Yes
Recipe view	Yes	Yes	Yes	Yes	Yes
Button	Yes	Yes	Yes	Yes	Yes
Switch	Yes	Yes	Yes	Yes	Yes
Slider	Yes	Yes	Yes	Yes	Yes
Key switch	No	No	No	No	No
Sm@rtClient view	Yes	Yes	Yes	Yes	Yes
Status/Force	Yes	Yes	Yes	Yes	Yes
Symbol library	Yes	Yes	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes	Yes	Yes
System diagnostics view	Yes	Yes	Yes	Yes	Yes
System diagnostics window	Yes	Yes	Yes	Yes	Yes

	KP400 Comfort KTP400 Comfort	KP700 Comfort TP700 Comfort	KP900 Comfort TP900 Comfort	KP1200 Comfort TP1200 Comfort	TP1900 Comfort TP2200 Comfort
Text field	Yes	Yes	Yes	Yes	Yes
Clock	Yes	Yes	Yes	Yes	Yes
Effective range signal	No	No	No	No	No
Effective range name	No	No	No	No	No
Effective range name (RFID)	No	No	No	No	No
WLAN reception	No	No	No	Yes	Yes
Gauge	Yes	Yes	Yes	Yes	Yes
Zone name	No	No	No	No	No
Zone signal	No	No	No	No	No

See also

Objects for Basic Panels (Page 4088)

Objects for Multi Panels

Availability of display and operating elements for Multi Panels

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and control objects for the Multi Panels.

Overview

	MP 177	MP 277	MP 377
Bar	Yes	Yes	Yes
User view	Yes	Yes	Yes
Date/time field	Yes	Yes	Yes
I/O field	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes
f(x) trend view	No	No	No
Function key	only on keyboard units		
Graphic view	Yes	Yes	Yes
Graphic I/O field	Yes	Yes	Yes
Handwheel	No	No	No
Help indicator	No	No	No

	MP 177	MP 277	MP 377
HTML browser	No	No	No
Circle	Yes	Yes	Yes
Trend view	Yes	Yes	Yes
Charging condition	No	No	No
Illuminated pushbutton	No	No	No
Line	Yes	Yes	Yes
Media Player	No	No	Yes
Alarm view, Alarm window	Yes	Yes	Yes
Alarm indicator	Yes	Yes	Yes
Polygon	Yes	Yes	Yes
Polyline	Yes	Yes	Yes
Rectangle	Yes	Yes	Yes
Recipe view	Yes	Yes	Yes
Button	Yes	Yes	Yes
Switch	Yes	Yes	Yes
Slider	Yes	Yes	Yes
Key switch	No	No	No
Sm@rtClient view	Yes	Yes	Yes
Status/Force	Yes	Yes	Yes
Symbol library	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes
System diagnostics display	No	No	No
System diagnostics window	No	No	No
Text field	Yes	Yes	Yes
Clock	Yes	Yes	Yes
Effective range signal	No	No	No
Effective range name	No	No	No
Effective range name (RFID)	No	No	No
WLAN reception	No	No	No
Gauge	Yes	Yes	Yes
Zone name	No	No	No
Zone signal	No	No	No

See also

Objects for Basic Panels (Page 4088)

Objects for Mobile Panels

Availability of display and operating elements for Mobile Panels

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and control objects for the Mobile Panels.

Overview

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP700 Mobile KTP700F Mobile KTP900 Mobile KTP900F Mobile
Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes
User view	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time field	Yes	Yes	Yes	Yes	Yes	Yes	Yes
I/O field	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes	Yes	Yes	Yes	Yes
f(x) trend view	No	No	No	No	No	No	Yes
Function key	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Graphic view	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Graphic I/O field	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Handwheel	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]
Help indicator	No	No	No	No	No	No	No
HTML Browser	No	No	No	No	No	No	Yes
Camera view	No	No	No	No	No	No	Yes
Circle	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Trend view	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Charge condition	No	No	No	Yes	Yes	Yes	No
Illuminated push-button	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]
Line	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Media Player	No	No	No	No	No	No	Yes
Alarm view Alarm window	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Alarm indicator	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PDF viewer	No	No	No	No	No	No	Yes
Polygon	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Polyline	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Rectangle	Yes	Yes	Yes	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP700 Mobile KTP700F Mobile KTP900 Mobile KTP900F Mobile
Recipe view	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Button	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Switch	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Slider	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key switch	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]
Sm@rtClient view	No	Yes	Yes	Yes	Yes	Yes	Yes
Status/Force	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Symbol library	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes	Yes	Yes	Yes	Yes
System diagnostics view	No	No	No	No	No	No	Yes
System diagnostics window	No	No	No	No	No	No	Yes
Text field	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Clock	No	No	Yes	Yes	Yes	Yes	Yes
Effective range signal	No	No	No	No	Yes	Yes	No
Effective range name	No	No	No	No	Yes	No	
Effective range name (RFID)	No	No	No	No	No	Yes	No
WLAN reception	No	No	No	Yes	Yes	Yes	No
Gauge	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Zone name	No	No	No	Yes	Yes	No	No
Zone signal	No	No	No	Yes	Yes	No	No

1) optional operator control

See also

Objects for Basic Panels (Page 4088)

Objects for WinCC Runtime Advanced

Availability of display and operating elements for WinCC Runtime Advanced

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and control objects for WinCC Runtime Advanced.

Overview

	WinCC Runtime Advanced
Bar	Yes
User view	Yes
Date/time field	Yes
I/O field	Yes
Ellipse	Yes
f(x) trend view	Yes
Function key	No
Graphic view	Yes
Graphic I/O field	Yes
Handwheel	No
Help indicator	No
HTML Browser	Yes
Camera view	No
Circle	Yes
Trend view	Yes
Charge condition	No
Illuminated pushbutton	No
Line	Yes
Media Player	No
Alarm view Alarm window	Yes
Alarm indicator	Yes
PDF view	Yes
Polygon	Yes
Polyline	Yes
Rectangle	Yes
Recipe view	Yes
Switch	Yes
Button	Yes
Slider	Yes
Key switch	No
Sm@rtClient view	Yes

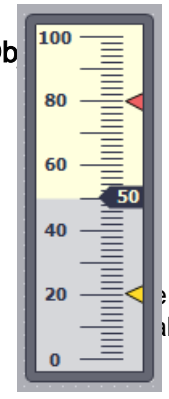
	WinCC Runtime Advanced
Status/Force	Yes
Symbol library	Yes
Symbolic I/O field	Yes
System diagnostics display	Yes
System diagnostics window	Yes
Text field	Yes
Clock	Yes
Effective range signal	No
Effective range name	No
Effective range name (RFID)	No
WLAN reception	No
Gauge	Yes
Zone name	No
Zone signal	No

See also

Objects for Basic Panels (Page 4088)

PDF view (Page 4146)

12.1.8.2 Ob



Bar

Application

displayed graphically using the "Bar" object. The bar graph can be labeled with lues.

Layout

In the Inspector window, you customize the settings for the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Color transition: Specifies the change in color display when limit values are exceeded.
- Displaying the limit lines / limit markers: Shows the configured limit as a line or marker.
- Define bar segments: Defines the gradations on the bar scale.
- Define scale gradation: Defines the subdivisions, scale markings and intervals of a bar scale.

Color transition

You define how the color change is represented in "Properties > Properties > Appearance" in the Inspector window.

Color transition	Description
"Segmented"	If a particular limit was reached, the bar changes color segment by segment. With segment by segment representation, you visualize, for example, which limits are exceeded by the displayed value.
"Entire bar"	If a particular limit was reached, the entire bar changes color.

Displaying limit lines and limit markers

You display the configured limit in the bar as a line or marking in Runtime using the "Lines" and "Markings" property:

1. In the Inspector window, select "Properties > Properties > Appearance":
2. Activate "Lines" and "Markings".

Define bar segments

Use the "Subdivisions" property to define the number of segments into which the bar is divided by the main gradations on the scale.

Use the "Interval" property to divide the distance between the main gradations. The value appears as the difference in value between two adjacent main gradations:

1. In the Inspector window, select "Properties > Properties > Scales":
2. Activate "Show scale."
3. Select the corresponding value for "Settings > Subdivisions".
4. Select the corresponding value for "Settings > Marks label".
5. Select the corresponding value for "Large interval > Interval".

See also

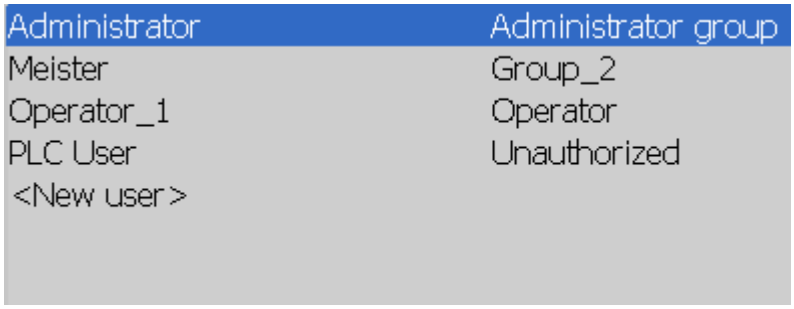
Device dependency of the objects (Page 4088)

User view

Simple user view

Application

The "User view" object is used to set up and administer users and authorizations.



Note

Do not use the user view in a group.

Note

The "Simple user view" object cannot be operated dynamically with a script.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Number of lines: Specifies the maximum number of visible entries.

Number of lines

The number of lines in the user view displayed in Runtime is specified in the Inspector window. The setting for the number of lines is only effective if the property "Fit object to contents" is active.

1. In the Inspector window, select "Properties > Properties > View".
2. Enter an integer value under "Number of lines".
3. In the Inspector window, activate "Properties > Properties > Layout".
4. Select "Fit object to contents".

Display in Runtime

The appearance depends on the authorizations:

- All users on the HMI device are displayed in the User view to the administrator or to a user with administrator authorizations.
- For a user without administrator rights, only the user's own entry is shown.

Operation

Depending on the configuration you can:

- Manage users, e.g. create, delete.
- Change existing user data.
- Export or import user data.

Note

The number is limited to 100 users and one PLC user on an HMI device. This restriction does not apply to PCs. On a PC, the maximum number of users is restricted by the physical memory.

Advanced user view

Application

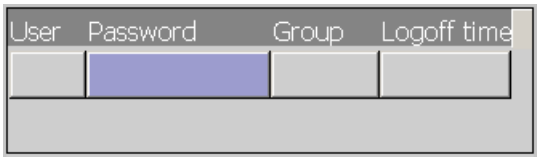
The "User view" object is used to set up and administer users and authorizations. The user view is used, for example, to create new users in Runtime and assign them to a user group.

Note

Configure only one user view at a time in one screen on the HMI devices TP 177A 6", TP 177A 6" (Portrait), OP 73 and OP 77A. Otherwise, a corresponding error message appears during generation.

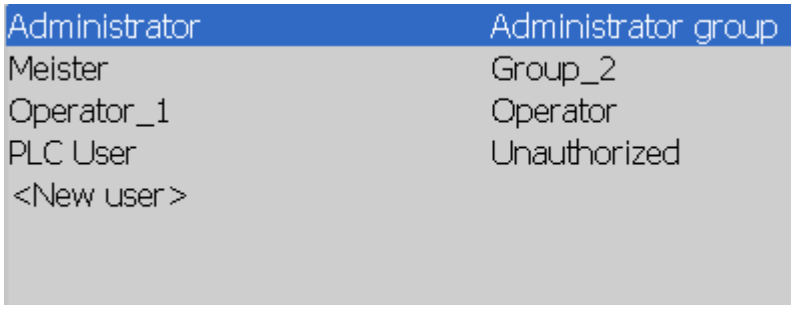
Advanced user view

Depending on the HMI devices, the extended or simple user view is available for administration of users and authorizations.



Changing the view type

On HMI devices with a display size larger than 6 ", you can configure both the advanced user view and a simplified user view. The view type is only available for configuration on devices up to device version V13.



Proceed as follows to configure the appropriate user view:

1. Click "Properties > Properties > Layout > Mode" in the Inspector window.
2. Select "Simplified" or "Advanced".

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Number of lines: Specifies the maximum number of visible entries.
- Columns moveable: Specifies whether the operator can change the sequence of columns in Runtime.

Number of lines

The number of lines in the user view displayed in Runtime is specified in the Inspector window. The setting for the number of lines is only effective if "Fit object to contents" is active.

1. Click "Properties > Properties > View" in the Inspector window.
2. Enter an integer value under "Number of lines".
3. Click "Properties > Properties > Layout" in the Inspector window.
4. Select "Fit object to contents".

Columns moveable

The operator can change the sequence of columns in Runtime with the "Columns movable" property.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Activate "Columns moveable".

This option is only available at the complex user view.

See also

Device dependency of the objects (Page 4088)

Configuring a user view (Page 4549)

User view (Page 4548)

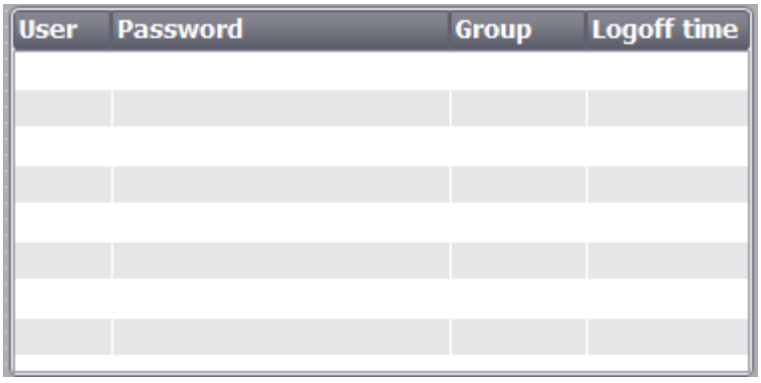
Creating users (Page 4550)

User view (as of V13)

Application

The "User view" object is used to set up and administer users and authorizations. The user view is used, for example, to create new users in Runtime and assign them to a user group.

On HMI devices with device version V13 or later, the table-based user view is available for managing users and authorizations.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Number of lines: Specifies the maximum number of visible entries.
- Columns moveable: Specifies whether the operator can change the sequence of columns in Runtime.

In addition, you set the borders, the fill pattern and the colors of the table header in the Inspector window.

Number of lines

The number of lines in the user view displayed in Runtime is specified in the Inspector window. The setting for the number of lines is only effective if "Fit object to contents" is active.

1. Click "Properties > Properties > View" in the Inspector window.
2. Enter an integer value under "Number of lines".

3. Click "Properties > Properties > Layout" in the Inspector window.
4. Select "Fit object to contents".

Columns moveable

The operator can change the sequence of columns in Runtime with the "Columns moveable" property.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Activate "Columns moveable".

Column width

In the Inspector window, you can change the width of columns displayed in runtime.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Enter integer values for the column width under "Columns".

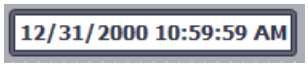
Note

The "Password" column is a dynamic column: it adapts to the outer dimensions of the object so that the object content occupies the entire width. The column width you assign is applied as the minimum size. In a dynamic column, the minimum size is changed; however, the width is displayed dynamically to fill the display.

Date/time field

Application

The "Date/time field" object shows the system time and the system date. The appearance of the "Date/time field" depends on the language set on the HMI device.



Layout

In the Inspector window, you customize the position, style, colors and font types of the object. You can adapt the following properties in particular:

- Display system time: Specifies that the system time is displayed.
- Include tag: Specifies that the time of the connected tag is displayed.
- Long date/time format: This setting defines the format displayed for the data and time.

Display system time

The time displayed in the "Date/time field" on the HMI device is specified in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Select "Format > System time".

Using tags

The time of the interconnected tag is displayed in the date/time field.

1. In the Inspector window, select "Properties > Properties > General".
2. In the "Format" area, select a tag with the "DateTime" data type, e.g. an internal tag.

Long date/time format

Visualization of the date and time is specified in the "Format" area under "General" in the Inspector window.

Option	Description
"Enabled"	Date and time are displayed in full, e.g. "Sunday, December 31, 2000 10:59:59 AM"
"Disabled"	Date and time are displayed in short form, e.g. "12/31/2000 10:59:59 AM"

Behavior during operation

When the operator ignores the syntax when entering values, or enters illegal values, the system rejects these. Instead, the original value (plus the time that has elapsed in the meantime) appears in the date/time field and a system event is displayed on the HMI device.

See also

Device dependency of the objects (Page 4088)

Example: 2. Configuring a Date/time field (Page 5993)

I/O field

Application

The "I/O field" object is used to enter and display process values.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Display format: Specifies the display format in the I/O field for input and output of values.
- Hidden input: Specifies whether the input value is displayed normally or encrypted during input.

Note

Reports

In reports, I/O fields only output data. "Output" mode is preset. Properties for configuring input are not available, e.g. "hidden input".

Mode

The response of the I/O field is specified in the Inspector window in "Properties > Properties > General > Type".

Mode	Description
"Input"	Values can only be input into the I/O field in Runtime.
"Input/output"	Values can be input and output in the I/O field in Runtime.
"Output"	The I/O field is used for the output of values only.

Layout

The "display format" for the input and output of values is specified in "Properties > Properties > General > Format" in the Inspector window.

Layout	
"Binary"	Input and output of values in binary form
"Date"	Input and output of date information. The format depends on the language setting on the HMI device.
"Date/time"	Input and output of date and time information. The format depends on the language setting on the HMI device.
"Decimal"	Input and output of values in decimal form
"Hexadecimal"	Input and output of values in hexadecimal form
"Time"	Input and output of times. The format depends on the language setting on the HMI device.
"Character string"	Input and output of character strings.

Note

Data formats

Not all data formats are available for selection for Runtime Professional.

Hidden input

In Runtime the input can be displayed normally or encrypted, for example for hidden input of a password. A "*" is displayed for every character during hidden input. The data format of the value entered cannot be recognized.

1. In the Inspector window, select "Properties > Properties > Response":
2. Select "Hidden input".

Avoid overlaps in output fields

If several I/O fields are configured as output fields with a transparent background in a screen, these I/O fields may overlap. The transparent part of the one field covers the digits of the other field. This may cause display problems in Runtime. In order to avoid such overlaps, set the borders of the I/O fields to zero in the object properties under "Properties > Properties > Appearance". Select "Properties > Properties > Layout > Fit object to contents."

Limits

Under "Properties > Properties > Limits" in the Inspector window, set the colors for the values that violate the high or low limits.
You define the limits via the properties of a tag.

In Runtime Professional, you can also define the limit range for the entry in the I/O field via "Properties > Properties > Limits".

If you enter a numeric value outside this limit, it is not accepted; for example, 80 with a limit of 78. In this case, the HMI device generates a system event, if an alarm window is configured. The original value is displayed again.

Decimal places for numerical values

The configuration engineer can define the number of decimal places for a numerical input field. The number of decimal places is checked when you enter a value in this type of I/O field. Decimal places in excess of the limit are ignored. Empty decimal places are filled with "0".

Behavior when switching between input fields

When you change to another input field within the same screen, and the screen keyboard appears, the "Exit field" event is not executed for the previous field unless you close the screen keyboard.

See also

Using multiple tags simultaneously in a screen (Page 4214)

Ellipse

Application

The "Ellipse" is an enclosed object that can be filled with a color or pattern.

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Horizontal radius: Specifies the horizontal radius of the elliptical object.
- Vertical radius: Specifies the vertical radius of the elliptical object.

Horizontal radius

The horizontal radius of the "Ellipse" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 under "Horizontal."

Vertical radius

The vertical radius of the "Ellipse" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 at "Vertical."

See also

Device dependency of the objects (Page 4088)

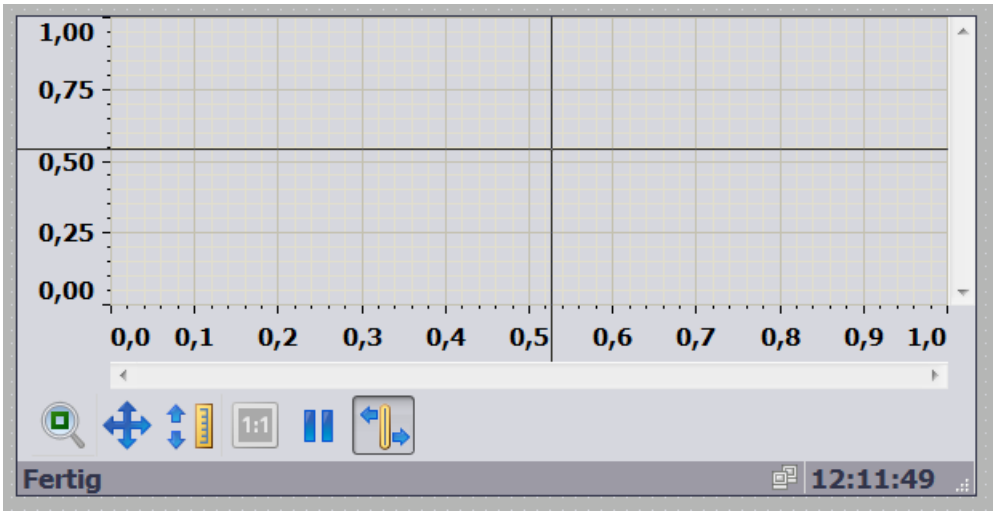
Rotating objects (Page 3949)

Flipping objects (Page 3951)

f(x) trend view

Application

You use the "f(x) trend view" object to represent the values of a tag as a function of another tag. This means that you can present temperature trends as a function of the pressure, for example.



Layout

In the Inspector window, you customize the position, shape, style, colors and font types of the object. You can adapt the following properties in particular:

- Adding and configuring trends
- Configuring a diagram
- Window settings
- Persistence
- Toolbar

Configuring trends

You create and configure trends in the Inspector window "Properties > Properties > Properties > Trend".

1. Select the data supply for the trend in the "Data source" column.
   - "Log tag": The trend window is supplied with values from a data log.
   - "Tags": The trend view is supplied with current values from the PLC. The values are constantly updated.
2. Configure the area of the trend display under "Data area".
   - "Time span": You define the time range using a starting time and a following time span.
   - "End time": You define the time range using a starting time and an end time.
   - "Measuring points": You define the time range using a starting time and a number of measuring points.
3. Configure the value range of the trend display under "X axis" and "Y axis".
4. Under "Limits", configure the colored marking of specific values, .e.g.. high and low limit, uncertain status.

Configuring a diagram

Configure the display of several trends under "Properties > Properties > Appearance":

- Common diagram or separate diagrams
- Common or separate axes
- Writing direction of all trends

Toolbar

The operator controls of the f(x) trend display are defined in the "Properties > Properties > Toolbar" Inspector window. The following operator controls are available for the f(x) trend view:

Button	Name	Function
	Zoom +/-	Increases or decreases the size of the trends in the trend window. Increase the size of the trends by left-clicking with the mouse. Hold down <Shift> and left-click to decrease the size of the trends.
	Zoom area	Increases the size of any section of the trend window. Specify an area in the trend window by using the mouse. This area of the trend window is enlarged.
	Zoom X axis	Increases or decreases the size of the X axis in the trend window. Enlarge the X axis by left-clicking with the mouse. Hold down <Shift> and left-click to decrease the size of the X axis.
	Zoom Y axis	Increases or decreases the size of the Y axis in the trend window. Enlarge the Y axis by left-clicking with the mouse. Hold down <Shift> and left-click to decrease the size of the Y axis.
	Trend area	Shift the trends in the trend window along the X axis and Y axis by using this button.

Button	Name	Function
	Axis area	Shift the trends in the trend window along the value axis by using this button.
	Original view	Switches from the magnified trend view back to the normal view
	Start/stop	Stops and starts the trend update. The values are buffered and updated as soon as you start trend update again.
	Ruler	Determines the coordination of a point of the trend.

The order of the buttons is fixed. The operator can set up individual key assignments, and shortcuts for every button on the toolbar.

See also

Bar (Page 4097)

Device dependency of the objects (Page 4088)

Trend view (Page 4123)

Graphic view

Application

The "Graphic view" object is used to display graphics.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Graphic: Specifies the graphic file that is displayed in the object.
- Adjust graphic: Specifies the automatic size stretching for objects with graphics.
- Transparent color: Specify whether or not the transparent color is used for the graphic.

Inserting graphics

The following graphic format is used in the "Graphic view" object: *.bmp, *.tif, *.png, *.ico, *.emf, *.wmf, *.gif, *.jpg or *.jpeg. You may also use graphics as OLE objects in the Graphic view .

1. In the Inspector window, select "Properties > Properties > General".
2. Select the graphic that you wish to insert.
   The graphic preview is shown in the right pane.
3. Click "Apply" to insert the graphic in the Graphic view .

Adjust graphic

Whether a graphic displayed in a Graphic view is stretched to the size of the Graphic view in Runtime is specified in the Inspector window.

Option	Description
"No auto-sizing"	The size is not automatically adapted in Runtime.
"Fit graphic to object size"	The graphic is automatically adapted to the object size in Runtime.
"Fit object size to graphic"	Graphic display is adapted to the largest used graphic and other graphics are stretched.
"Adjust object size to graphic"	Graphic display is adapted to the largest used graphic but the other graphics retain their original size.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select the required size adjustment for the graphic.

Transparent color

This property defines whether the transparent color is used for the graphic to be displayed.

1. Click "Properties > Properties > Appearance" in the Inspector window:
2. Select "Background > Transparent".
3. Select a transparent color.

Note

When using bitmaps in WinCC screens the "Transparent color" setting demands a high character performance in the layout on the panel. Visualization performance is enhanced by disabling the "Transparent" setting in the properties of the relevant display object. This restriction applies in particular when bitmaps are used as background image.

Note

Basic Panels

The "Transparent" property is not available for Basic Panels.

See also

Device dependency of the objects (Page 4088)

Options for editing objects (Page 3939)

Storing an external image in the graphics library (Page 3974)

Graphic I/O field

Application

The "Graphic I/O field" object can be used to configure a list for display and selection of graphic files.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Scroll bar type: Specifies the graphic layout of the scroll bar.

Note

Scroll bar

The scroll bar is available for Panels and Runtime Advanced up to device version V12.

Note

Reports

Graphic I/O fields output exclusively graphics in reports. "Output" mode is preset. Properties for configuring the selection of graphics are not available, e.g. "scroll bar".

Note

Border

You can configure the width and the style of the border of a graphic I/O field in the "Two states" mode.

Mode

The response of the "Graphic I/O field" object is specified under "Properties > Properties > General > Type > Mode" in the Inspector window.

Mode	Description
"Input"	The "Graphic I/O field" object is only used to select graphics.
"Input/output"	The "Graphic I/O field" object is used to select and display graphics.
"Output"	The "Graphic I/O field" object is used to display graphics only.
"Two states"	The "Graphic I/O field" object is only used to display graphics and can have a maximum of two states. You use no graphics list but insert one graphic each for the "ON" and "OFF" state.

Adjust graphic

Whether a graphic displayed in a graphic I/O field is stretched to the size of the view in Runtime is specified in the Inspector window.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select the required size adjustment for the graphic.

Scroll bar type

The response for the graphic representation of the scroll bar is specified under "Properties > Properties > Appearance > Scroll Bar > Type" in the Inspector window.

Type	Description
"Permanent"	The scroll bar is always visible.
"No scrollbar"	The scroll bar is not visible.
"Visible after clicking"	The scroll bar is made visible by a mouse click.

Behavior during operation

If the graphic I/O field displays a cactus image, a graphic to be output for a specific value has not been defined in the project. The contents of the graphic I/O field change color to show that it is now activated. The frame in 3D is only shown graphically in an output field.

See also

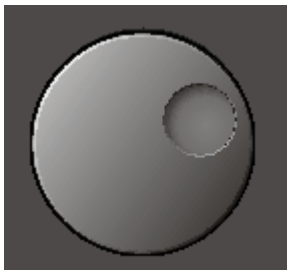
Device dependency of the objects (Page 4088)

Symbolic I/O field (Page 4174)

Handwheel

Application

The object "Handwheel" is a control element for some SIMATIC Mobile Panels. You can enter incremental values in Runtime using the "HandWheel" object.



Layout

You can set the following property in the Inspector window:

- Tag: Specifies the tags to which the "HandWheel" object is linked.

Tag

Global assignment

The "Tag" property is specified in the template for the global assignment of the "HandWheel" object. There is only one template for each HMI device.

1. In the Inspector window, select "Properties > Properties > General":
2. Select a tag from the selection list under "Settings > Tag".

Local assignment

The "HandWheel" object is assigned to another tag. To do so you configure a function that assigns the tag to the handwheel in Runtime.

Example for configuration to a button:

1. Open the screen in which the "HandWheel" object is assigned to another tag.
2. Drag the "Button" object from the toolbox to the screen. Select the button on your screen.
3. Click "Properties > Events > Click" in the Inspector window.

4. The "Function list" dialog box opens. Click the first line of the function list. The list of system functions and scripts available in the project appears.
5. Select the system function "SetTagToHandWheel" from the "Other functions" group. Select a tag from the combo box at "Value".

Note

The Mobile Panel may ignore the change of the value in the PLC made at another location while the handwheel is being operated. The value is the PLC is overwritten again.

See also

Device dependency of the objects (Page 4088)

Illuminated pushbutton (Page 4126)

Help indicator

Application

The object "help indicator" is available for the HMI devices OP 73 and KP300 Basic. If a a tooltip exists for the selected object, the help indicator is displayed during runtime. If a tooltip was configured for the opened screen, the help indicator always remains visible.

You configure the object "help indicator" exclusively in the global screen.

Layout

You can adapt the following properties in the Inspector window:

- Position: Determines the position of the object "Help indicator."

Position

You can use this property to set the position of the object "Help indicator."

1. Select the object "Help indicator" in the template.
2. In the Inspector window, select "Properties > Properties > Layout".
3. Enter a value for X and Y. You can also use the cursor keys to position the selected object.

If you have configured a screen object at this position, the visible help indicator covers the screen object. The help indicator is covered only by incoming system alarms and dialogs.

See also

Device dependency of the objects (Page 4088)

HTML Browser

Use

The "HTML Browser" object is designed for the visualization of simple HTML pages. This function allows you to draw up machine-specific descriptions which are stored centrally and which can then be displayed from different HMI devices.

Note

Switching the functionality of the HTML Browser to that of a file explorer by one of the following methods, for example, is not authorized in the context of WinCC:

- Entering a folder or drive, for example "\" or "C:" or
- Connecting to an FTP server, for example "ftp://"

One reason is to prevent inadvertent changes to files, their deletion or execution.

When configuring, ensure that the operator can only enter valid Internet addresses, for example, by using symbolic I/O fields. Configure a password-protected input for service purposes.

Note

Only one instance of the HTML Browser is supported per screen in Panels and RT Advanced to preserve memory space.

If you move quickly from a slide-in-screen to a pop-up screen and an HTML browser is configured in both screens, the second HTML browser will not open. Wait two seconds after closing the slide-in screen until the handle is no longer visible and then open the pop-up screen.

Layout

Customize the object position and size in the Inspector window. In particular, you can customize the following property:

- URL: Specifies which Internet address is opened in the HTML Browser.
- Tag for URL: Specifies the tag which sets the Internet address.

Address

The Internet address is specified in "Properties > Properties > General >URL" in the Inspector window.

Browser type (RT Advanced as of V13 SP1)

You can select the browser type in Runtime Advanced as of version V13 SP1.

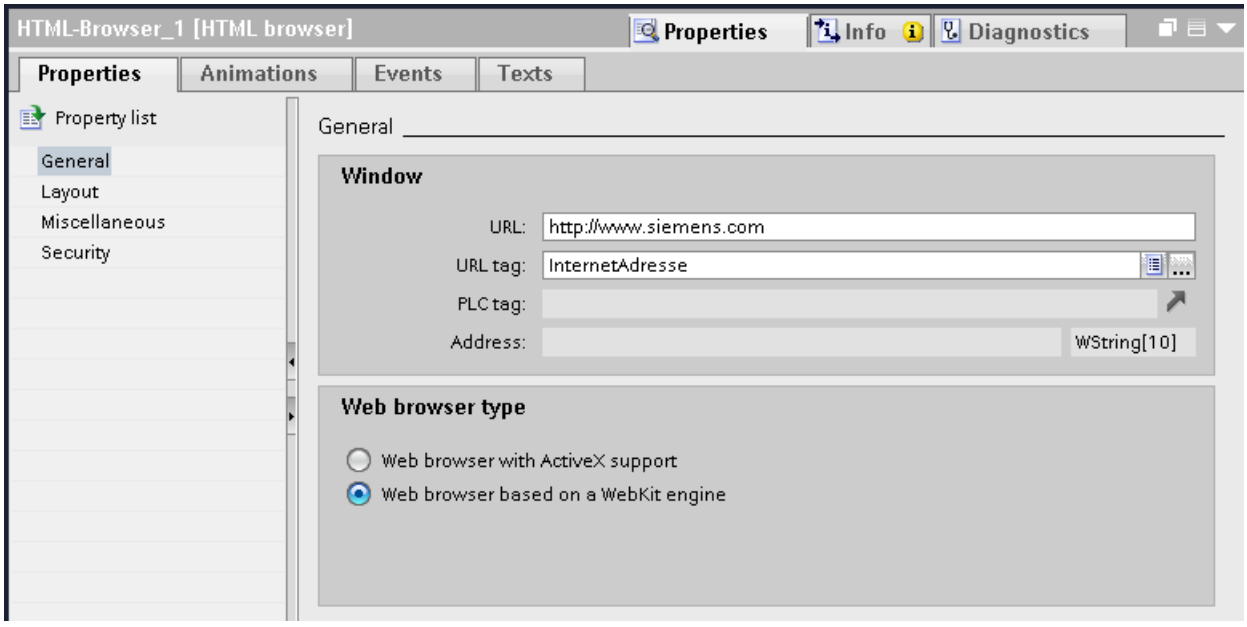
Specify the type of the web browser under "Properties > Properties > General > HTML Browser type" in the Inspector window.

Two options are available:

- Web browser based on WebKit engine without Active X support
- Web browser based on Internet Explorer with Active X support

Basic procedure

1. Create an internal tag of the "String" data type, for example, InternetAddress, in the "Tag" editor.
2. Insert the "HTML Browser" object in the screen in the "Screens" editor.
3. In the Inspector window, select "Properties > Properties > General".
4. At "Tag for the URL", select the "InternetAddress" tag from the selection list.
5. Insert an I/O field in the screen in the "Screens" editor.
6. In the Inspector window of the IO field "Properties > Properties > General > Tag for URL", select the "InternetAddress" tag from the selection list.



The HTML Browser opens the relevant page after the operator has entered an address in the I/O field.

Operator controls

Operator controls are configured buttons in the process screen or softkeys on the HMI device.

The "HTML Browser" object does not have its own operator controls. Assign the system functions used to control the "HTML Browser" object to your configured operator controls, such as buttons.

1. For example, drag the "Button" object to the screen from the toolbox.
2. Input a text of any length or select a graphic, for example Update.
3. In the Inspector window, select "Properties > Events > Click".
   The "Function list" dialog opens.
4. The system functions for controlling the "HTML Browser" object can be found in the list of system functions in the "Keyboard operation for screen objects" group.
   Select a function from the list, for example, HTMLBrowserRefresh.
5. Select the object name of the HTML Browser in which the command will be executed from the "Screen object" list.

Defining the website selection

To define the website selection that is provided in the HTML browser, you can define a number of allowed URL addresses.

You define a String or Wstring type tag or for the "Tag for URL" property of the HTML browser that can be filled with the predefined URL addresses in the following ways:

- Using the "SetTag" system function, which you configure on a button
- Using an I/O field

  Note

  Any website can be accessed by entering the address in an I/O field.

  To restrict the selection of allowed websites, use the provided pre-defined URL addresses via a system function or a script.

- Via a script (panels and RT Advanced only)
- Via the PLC

Comments

The functional scope of the HTML Browser is limited in comparison to the Internet Explorer:

- The HTML Browser will only show pure HTML pages. VBScript, Java, JavaScript, Flash and ActiveX controls are not supported. Set up the HTML pages for display in the HTML Browser by using a text editor or using a simple HTML editor.
- Links to embedded files, for example *.pdf or *.xls, are not supported.
- Queries and dialogs that are conducted during the access of, for example protected pages are not supported. When accessing protected pages enter your user name and password in the URL: <http://User_Name:Password@Server_Name> (for example, "http://Administrator:Admin123@192.168.0.199").

Note

The object "HTML Browser" supports dynamization.

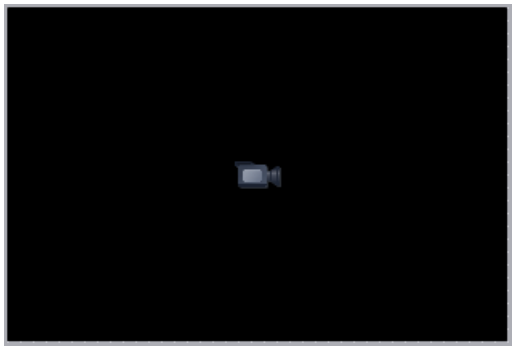
If you select the HTML browser type "Web browser, based on a WebKit En", only the variable connection is available for dynamization under "Address".

Camera view

Use

You use the "Camera view" object to configure the camera view which outputs screens for the operator in runtime from a connected network camera.

Monitoring with the help of the network camera lends itself for situations in which the operator has limited visibility of the plant or cannot get close to the machines. The operator can then respond to any problem much faster to remedy its cause.



The camera view can be configured in the following screens:

- In a screen
- In a template
- In a slide-in screen
- In a pop-up screen

Note

Device-dependency of the "Camera view" object

The "Camera view" object is available for Comfort Panels and KTP Mobile Panels (as of device version V13).

Note

Only one camera view may be in operation in runtime at the time.

Layout

You enter the information on the source of the video stream as well as position and size of the camera view in the Inspector window. You can adapt the following properties in particular:

- URL for the source of the video stream:
  - Camera URL: Specifies a static URL which is only entered during configuration.
  - Camera URL tag: Specifies a dynamic URL which the operator can change later in runtime.
- Transmission protocol: Specifies the transmission protocol by which the data exchange between the panel and the network camera takes place.
- Position and size of the camera view: Specifies position and size of the camera view as well as the aspect ratio and size of the displayed video stream.

Specifying URL for the source of the video stream

1. Insert the camera view in the screen in the "Screens" editor. The camera view is found in the task card "Tools" > "Controls".
2. Click "Properties > Properties > General > Media" in the Inspector window.
3. Enter the static camera URL in the "Camera URL" field, for example, "rtsp://user:password@192.168.56.208" for a Siemens camera.
   The address of the network camera is available in the network camera manual.

   Note

   The camera URL always starts with "rtsp://" and must only include permitted characters. If you enter characters that are not permitted or if the URL does not start with "rtsp://", the camera cannot be operated in runtime. The operator will receive an error message in the "Camera view" to indicate the problem.

   The following characters of the US ASCII character set are permitted:

   0123456789

   A...Z a...z

   !#$%&()*+,'-_./:;<=>?@ [\]_{|}~^(")`
4. Assign a tag to the camera view under "Camera URL tag" in the Inspector window. The tag must be of the data type "String" for external tags and "WString" for internal tags.
   When you assign a tag to the camera view, the user can change the source of the video stream in runtime.

Note

If you leave the "Camera URL tag" field blank, the static URL is used in runtime.

If you enter information in the "Camera URL" and "Camera URL tag" field, the dynamic URL is used in runtime.

Specifying the transmission protocol

1. Click "Properties > Properties > General > Media" in the Inspector window.
2. Specify the transmission protocol with the "Use UDP instead of TCP" check box.
   The selection of the protocol depends on the protocol types which your network camera supports. Information on supported protocols is available in the network camera manual.

Specifying position and size of the camera view

1. Click "Properties > Properties > General > Layout" in the Inspector window.
2. Enter the size of the object under "Camera view" in the "Position & Size" area. To achieve the best picture quality of the video stream, enter the physical resolution of the network camera as size of the camera view.
   The recommended setting "Keep video size" is selected by default in the "Fit to size" area. The "Keep aspect ratio" check box is disabled and cannot be selected.
3. To keep the aspect ratio of the video stream, disable "Keep video size".
   The "Keep aspect ratio" check box is automatically selected.

Note

The screen is not scaled when you keep the size of the video stream. The screen size of the network camera is output 1:1 by the camera view. If the camera view is smaller than the selected resolution of the network camera, only a corresponding section of the camera screen is output by the camera view. If the camera view is greater than the selection solution of the network camera, any blank spaces are displayed as black horizontal or vertical margins.

Any blank spaces are output as black horizontal or vertical margins by the camera view when you keep the aspect ratio of the video stream.

Note

When you disable the "Keep video size" and "Keep aspect ratio" check boxes, the camera screen may be distorted.

Result

You have configured a camera view which can be used for monitoring the plant in runtime.

Note

You cannot configure system functions for the camera view.

Note

When you test a project with the simulator, a black screen with a camera symbol is output.

See also

Device dependency of the objects (Page 4088)

Circle

Application

The "Circle" object is a closed object which can be filled with a color or pattern.

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Radius: Specifies the size of the circle.

Radius

The radius of the "Circle" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 in the "Radius" area.

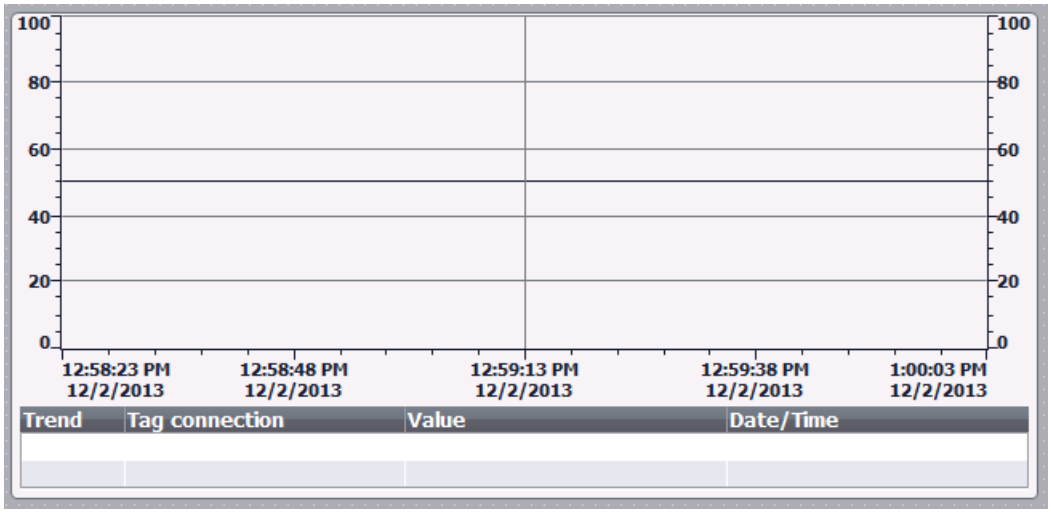
See also

Device dependency of the objects (Page 4088)

Trend view

Use

The trend view is meant for the graphical representation of tag values from the current process or from a log in form of trends.



Layout

In the Inspector window, you customize the position, geometry, style, color, and font types of the object. You can adapt the following properties in particular:

- Display value table, ruler and grid: Specifies whether a value table, a ruler or a grid is displayed in addition to the coordinate system to improve legibility.
- Toolbars: Defines the display of the operator controls.

Display value table, ruler and grid

For improved legibility a value table, a ruler and a grid can be displayed in Runtime.

1. Activate "Properties > Properties > Appearance > Show ruler".
2. Activate "Properties > Properties > Table > Show table".
3. Activate "Properties > Properties > Table > Show grid".

Toolbars

The layout of the operator controls is defined in the "Properties > Properties > Toolbar" inspector window.

Note

Basic Panels

Because archiving with Basic Panels is only possible for devices as of the 2nd Generation, the operator controls are not available on all Basic Panels.

Toolbar button	Brief description	Description
	"Go to start"	Scroll to the current value of the trend.
	"Zoom in"	Reduces the displayed time slice.
	"Zoom out"	Zooms into the displayed time section.
	"Ruler backward"	Moves the ruler back.
	"Ruler forward"	Moves the ruler forward.
	"Backward"	Scrolls back one display width.
	"Forward"	Scrolls forward one display width.
	"Ruler"	Shows or hides the ruler. The ruler displays the X-value associated with a Y-value.
	"Start/stop"	Stops trend recording or continues trend recording.

Configuration behavior

Displaying column headers

The layout of the table in the trend view depends on the view settings in the Control Panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Appearance" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

Consistency test

If warnings or errors are displayed in the output window during a consistency check in connection with trend views, clicking "Go to Error/Tag" on the shortcut menu will not always take you to the exact error position. In some cases only the trend view is shown as cause of error.

Adding, configuring, and removing trends

The trends of the trend view are managed in the Inspector window under "Properties > Properties > Trend". You can copy trends between different trend views.

See also

Device dependency of the objects (Page 4088)

Configuring trend displays for values from the PLC (Page 4273)

Trends (Page 4270)

Outputting tag values in screens (Page 4272)

Configuring trend display for values from the PLC (Basic) (Page 4268)

Charge condition

Application

The object "Charging condition" is a control element for some SIMATIC Mobile Panels. The "Charging condition" object indicates the charging status of the battery. Charge the battery in good time. Alternatively, you can replace the battery with a spare battery.

Layout

Symbol	Description	Charging condition
	Battery is charged	>20 %
	Battery must be charged	<20 % and >6 %
	Battery is weak	< 6 %

Operation

The object is for display only and cannot be controlled by the operator.

See also

Device dependency of the objects (Page 4088)

Effective range name (Page 4184)

Effective range name (RFID) (Page 4186)

Effective range signal (Page 4187)

WLAN reception (Page 4189)

Zone name (Page 4192)
Zone signal (Page 4193)

Illuminated pushbutton

Application

The object "Illuminated pushbutton" is a control element for some SIMATIC Mobile Panels. The LEDs can be activated by the PLC. The illuminated LED signals the operator in Runtime that the illuminated pushbutton has to be pressed. The illuminated pushbutton is configured in the global screen.

Layout

You can adapt the following properties in the Inspector window:

- Tag: Specifies the tag in which the status of the illuminated pushbutton is written.
- LED assignment: Specifies the assignment of the LED to the bit in the LED tag.

Tag

The "Tag" property is used to specify the global assignment of the "Illuminated pushbutton" object in the global screen.

Note

Information for configuring

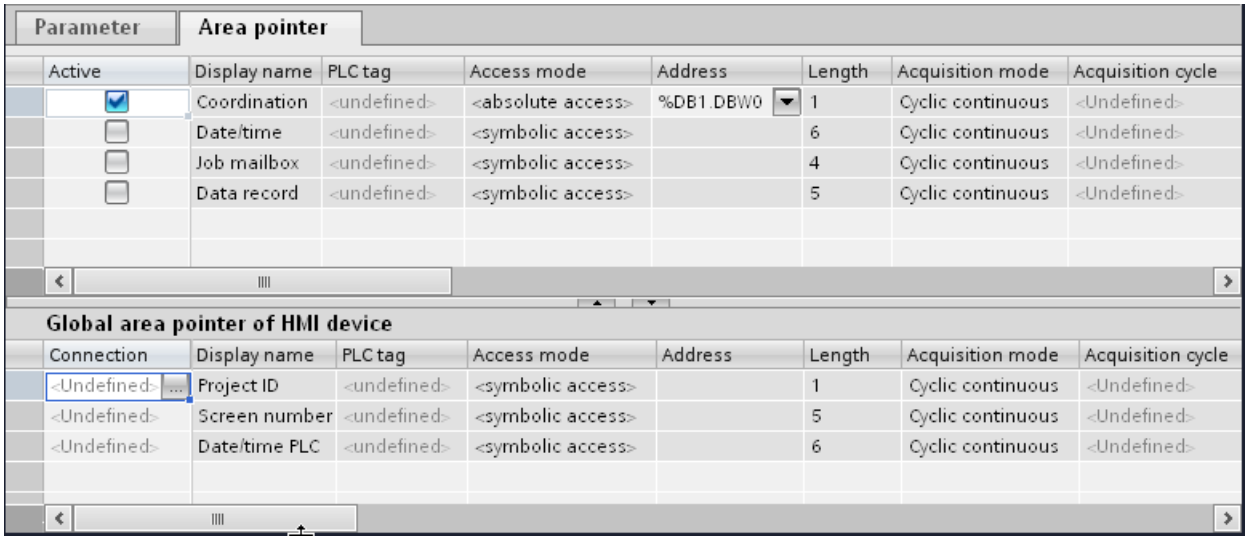
The current status of the illuminated pushbutton is written to the tag when Runtime is started and whenever the button is pressed. This may cause inconsistencies between the status of the illuminated pushbutton and the illuminated pushbutton tags.

1. If the tag has a connection to the PLC, the current value is transferred from the PLC to the tag after establishment of communication. The tag that contains the current status of the key is overwritten by this process. The illuminated pushbutton tag may therefore no longer reflect the current value of the illuminated pushbutton, for example if the Mobile Panel is switched off while the illuminated pushbutton is pressed.
2. If the illuminated pushbutton is pushed before communication to the PLC could be established (for example after a device start-up), it may not be possible to send the value of the illuminated pushbutton tag to the PLC. In this case the value of the tag in the PLC is different from the current status of the key.

Carry out the following configuration steps so that the illuminated pushbutton always reflects the actual state.

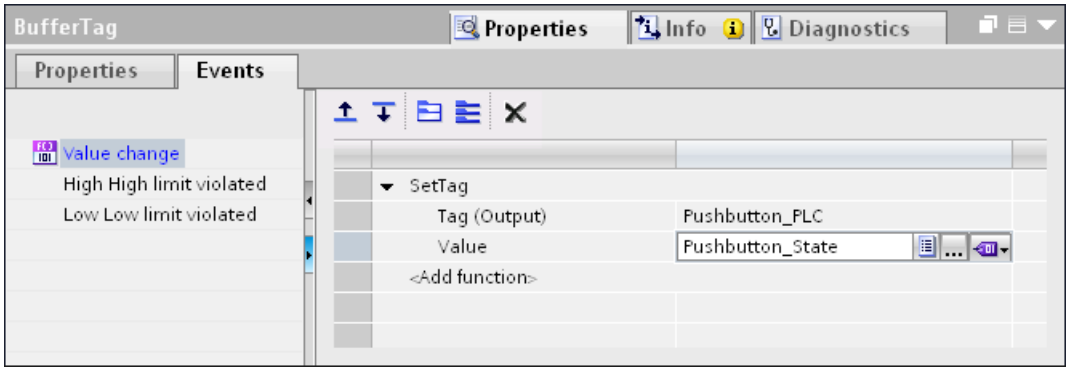
Basic procedure

1. Specify the connection to the PLC in the "Connections" editor. Activate the "Coordination" area pointer to ensure that the life bit is available to the PLC side.



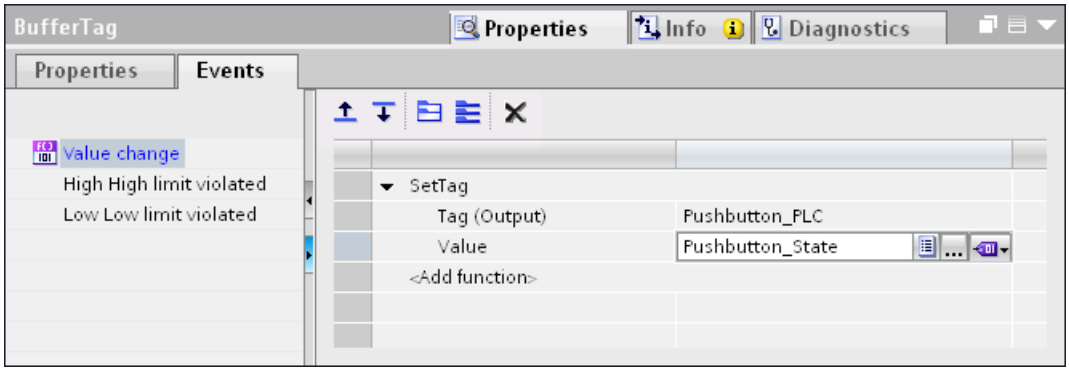
2. Open the "HMI tags" editor and create three tags.
   - Internal tag: Pushbutton_State
   - External tag: BufferTag
   - External tag: Pushbutton_PLC

3. Open the Inspector window of the "Pushbutton_State" tag.
4. In the Inspector window, select "Properties > Events > Value change". Click the first line of the function list. The list opens, showing the system functions available in the project.
5. Select the "SetTag" system function.
   - Select the "Pushbutton_PLC" tag at "Tag (output)".
   - Select the "Pushbutton_State" tag at "Value".



   The value of the "Pushbutton_PLC" tag is written to the PLC with the "Pushbutton_PLC" tag. A program in the PLC evaluates the life bit. If communication is established, the current value is written to the "Pushbutton_PLC" tag from the PLC. The "BufferTag" is required to transfer the current position of the pushbutton to the PLC.

6. Open the Inspector window of the "BufferTag" tag.
7. In the Inspector window, select "Properties > Events > Value change".
8. Click the first line of the function list. The list opens, showing the system functions available in the project.
9. Select the "SetTag" system function.
   - Select the "Pushbutton_PLC" tag at "Tag (output)".
   - Select the "Pushbutton_State" tag at "Value".



After establishing communication the current value is written to the "BufferTag" from the PLC. The "SetTag" system function is executed by the value change in the auxiliary tag. The system function re-allocates the value of the "Pushbutton_State" tag to the "Pushbutton_PLC" tag.

10. Open the global screen in the "Screen navigation" editor.
11. Select the illuminated key on the global screen.
12. Select the tag "Pushbutton_State" in "Properties > Properties > General > Settings >Tag" in the Inspector window.
    If you press the illuminated pushbutton, the value is written to the "Pushbutton_PLC" tag.

LED assignment

To actuate the LED an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration. The LED tag is specified at "LED tag" in the "Settings" area in the "General" group. With "LED bit" the allocated bit is given. When the bit is set the following states of the LED can be realized in runtime.

Bit n+1	Bit n	LED function
0	0	Off
0	1	Fast flash
1	0	Slow flash
1	1	On permanently

See also

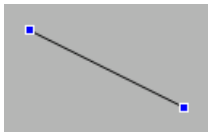
Device dependency of the objects (Page 4088)

Handwheel (Page 4114)

Line

Application

The "Line" object is an open object. The line length and gradient slope are defined by the height and width of the rectangle enclosing the object.



Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Line style
- Line start and end

Line style

The representation of the line is specified under "Properties > Properties > Appearance" in the Inspector window. The line is shown without interruption if you select "Solid", for example.

Note

The line styles available depend on the selected HMI device.

Line start and end

The start and end points of the line are specified under "Properties > Properties > Appearance > Line ends" in the Inspector window.

Use arrow point, for example, as start and end point. The available start and end points depend on the device.

See also

Device dependency of the objects (Page 4088)

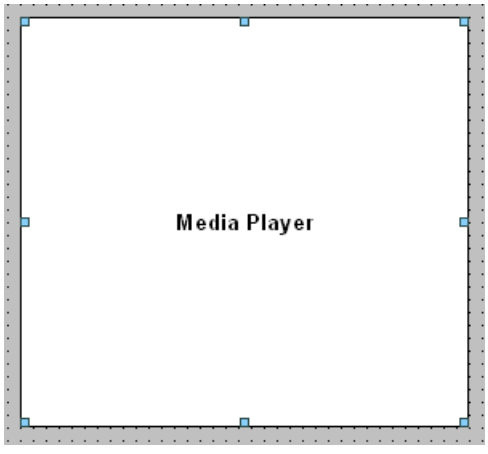
Media Player

Application

The Media Player is used in Runtime to play back video or audio files.

The Media Player is only available for the following devices:

- MP 377
- Comfort Panels



Layout

You can set the following properties in the Inspector window:

- Display status bar: Determines whether to display the status bar.
- Display operator controls: specifies the operator controls in Runtime.
- Show seek: establishes, whether a slider is available for the operation.

Operator controls

The operator controls that can be used to control the Media Player in Runtime are specified in the Inspector window under "Properties > Properties > Appearance > Elements".

Supported file formats

The following audio formats are supported by Media Player:

- Moving Picture Experts Group standard 1, Layer 1, 2, 3 (.mpa, .mp2, .mp3)
- Windows Media Audio (.wma)
- Waveform Audio (.wav)

The following video formats are supported by Media Player:

- Moving Picture Experts Group standard 1 (.mpg, .mpeg, .mpv, .mpe)
- Advanced Streaming Format (.asf)
- Windows Media Video Format (.wmv)
- Audio-Video Interleaved (.avi)

Note

Media Player does not support the creation and management of favorites. If you create favorites before closing the Media Player, they will not reappear when you open it again.

Note

The Media Player cannot be operated using the keyboard.

You cannot simulate the Media Player. A static screen appears in the simulation window instead of the Media Player.

See also

Device dependency of the objects (Page 4088)

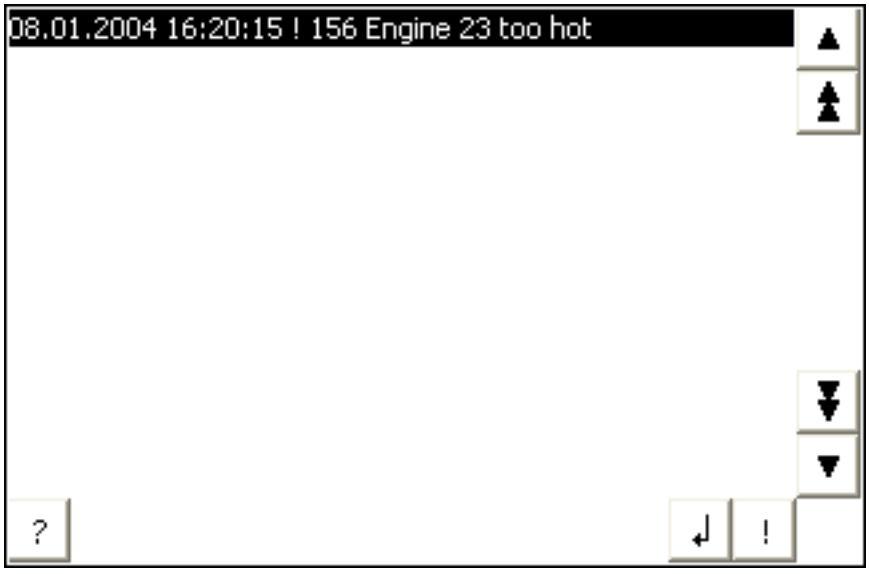
Alarm view

Simple alarm view

Application

Alarms are indicated in the Alarm view or in the Alarm window of the HMI device.

The following screen contains a simple alarm view:



Note

The "Single alarm view" object cannot be operated dynamically with a script.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object.

Note

The fonts available for selection depend on the fonts you have configured in the Runtime settings under "Language & font" and the fonts supported by your HMI device.

You can adapt the following properties in particular:

- Operator controls: Defines the operator controls of the alarm view.
- Alarm classes: This setting defines which alarm classes are displayed in the alarm view.
- Columns: Specifies the displayed columns in Runtime.

Note

If you have different alarm classes output, these will be initially sorted into alarm classes in Runtime, and then by the time at which the alarm occurred.

Operator controls

The operator controls that can be used to control the alarm display in Runtime are specified in the Inspector window under "Display > Settings". The following table shows the operator controls in the alarm view, and what they do:

Button		Function
"Info text"		Displays info text for an alarm.
"Acknowledge"		Acknowledges an alarm.
"Loop-In-Alarm"		If a screen change is configured, the display switches to a screen containing information about the alarm.

Select alarm classes

1. Click "Properties" in the Inspector window.
2. Under "Alarm classes", activate the alarm classes to be displayed in the alarm view in Runtime.

Define columns

Define the columns to be displayed in the alarm view in Runtime in the Inspector window.

1. In the Inspector window, click "Properties > Columns".
2. Activate the columns that are to be displayed in Runtime under "Columns".

Displaying column headers

The layout of the alarm view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Layout tab" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

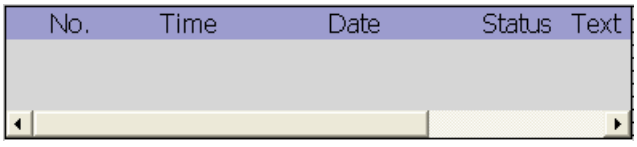
Note

In the engineering system you can dynamically control the visibility of an object, for example, in the "Animations" group of the Inspector window. In Runtime, the "Simple alarm view" does not support animations. If you have configured an animation and, for example, wish to perform a consistency check of the project, then an error alarm is issued in the Output window.

Extended alarm view

Alarm view

Alarms are indicated in the Alarm view or in the Alarm window of the HMI device. The following screen contains an alarm view with no content:



Alarm line

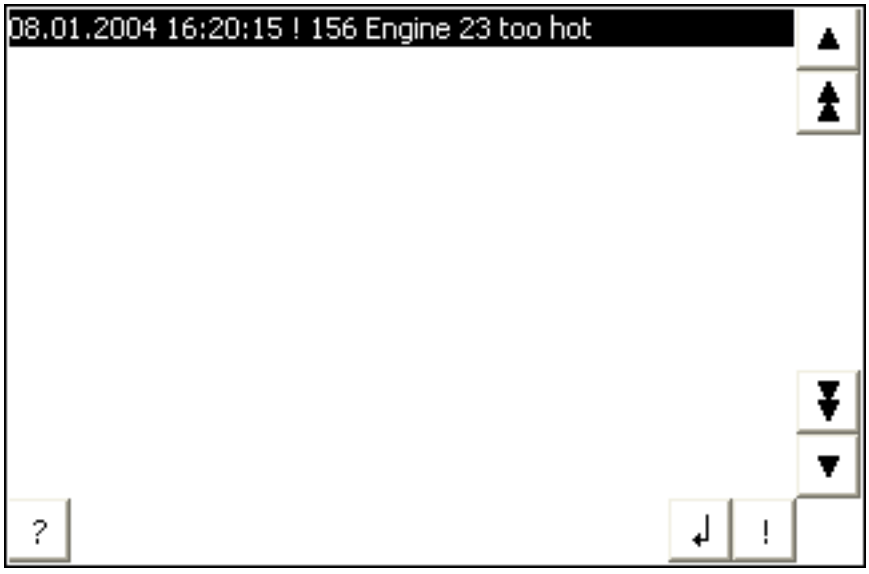
The alarm line allows display of the most current pending alarm in Runtime. The following figure shows an alarm line:

How to configure the alarm line:

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select "Mode > Alarm line".

Changing the view type

On HMI devices with a display size larger than 6 ", you can configure both the advanced alarm view and a simplified alarm view. The view type is only available for configuration on devices up to device version V13.



Proceed as follows to configure the simplified user view:

1. Click "Properties > Properties > Layout > Mode" in the Inspector window.
2. Select "Simplified".

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object.

Note

The fonts available for selection depend on the "Language and fonts" you have configured in the device settings and the fonts your HMI device supports.

You can adapt the following properties in particular:

- Operator controls: Defines the operator controls of the alarm view.
- Alarm classes: This setting defines which alarm classes are displayed in the alarm view.
- Columns: Specifies the displayed columns in Runtime.
- Sequence of columns: Specifies whether the sequence of columns can be changed in Runtime.
- Identification of alarm classes: To be able to distinguish various alarm classes, they are identified in the first column of the alarm view.
- Filter: Defines that only alarms that contain a specific character string will be displayed in the alarm text.

- Enable sorting: Specifies whether the alarms can be sorted in Runtime by date and time.
- Define display area: This setting specifies the date from which alarms are displayed in the alarm view.

Note

If you have different alarm classes output, these will be initially sorted into alarm classes in Runtime, and then by the time at which the alarm occurred.

Operator controls

The operator controls that can be used to control the alarm view in Runtime are specified in the Inspector window under "Properties > Properties > Display > Settings". The following table shows the operator controls in the alarm view, and what they do:

Button		Function
"Tooltip"		Displays a tooltip for an alarm.
"Acknowledge"		Acknowledges an alarm.
"Loop-In-Alarm"		If a screen change is configured, the display switches to a screen containing information about the alarm.

Select alarm classes

1. Click "Properties > Properties > General" in the Inspector window.
2. Under "Alarm classes", activate the alarm classes to be displayed in the alarm view in Runtime.

Access protection in Runtime

Configure access protection in the "Properties > Security" group in the properties of the alarm view. If a logged-on user has the required authorization, he can acknowledge and edit alarms using the operator controls in the alarm view. If the logged-on user does not have the required authorization, or if no user is logged in, pressing the "Acknowledge" or "Edit" buttons or double-clicking an alarm line opens the logon dialog.

Define columns

Define the columns to be displayed in the alarm view in Runtime in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Activate the columns that are to be displayed in Runtime under "Columns".

Sequence of columns

If this property is enabled, the column sequence in the alarm view can be changed in Runtime.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Column properties > Column order".

Enable sorting

If this property is enabled, the alarms displayed in the alarm view in Runtime can be sorted by date and time.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Column properties > Sorting by date/time possible".

Filter alarms

This property is used to define that only alarms that contain a configured string in the alarm text will be displayed in Runtime for the extended alarm view.

1. In the Inspector window, select "Properties > Properties > Filter.
2. Enter the required term for filtering in the field "Filter string".
   Alternatively, you configure a filter tag using the field "Filter tag". The contents of the filter tag serve as the filter criterion in Runtime.

Identify alarm classes

An icon is displayed in the first column of the alarm view. This symbol allows the alarm to be allocated to an alarm class.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Columns > Alarm class".
3. Open the "Alarms" editor, and click the "Alarm classes" tab.
4. Specify an icon for use to identify the alarms of this alarm class for an alarm class in the "Display name" column.

Note

The "alarm view" object cannot be grouped.

Define display area

You select a tag which defines the time from which alarms should be displayed. The following data types are permitted:

- External tags: Date, Date_and_Time, Time_of_Day
- Internal tags: DateTime

To define the display area, proceed as follows:

1. Click "Properties > Properties > Display" in the Inspector window.
2. Define the tag in which the time is specified under "Control tag for the display area".

Configuration behavior

Displaying column headers

The layout of the alarm view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Layout tab" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

Displaying the buttons for operation

The alarm view in the HMI devices OP 73, OP 77A and TP 177A has a button which displays the alarm text in a separate window. This button is not shown during configuration.

The display of the buttons for using the alarm view depends on the configured size. You should check on the HMI device whether all necessary buttons are available.

See also

Device dependency of the objects (Page 4088)

Alarm window (Page 4142)

Alarm indicator (Page 4144)

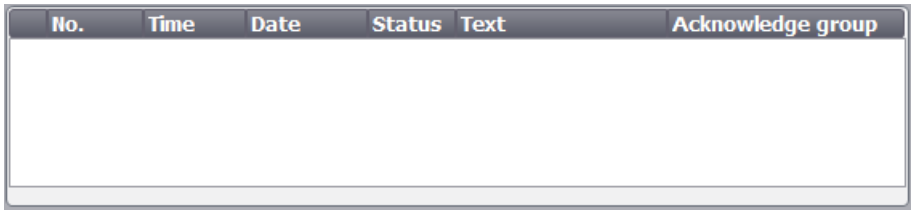
Configuring an alarm view for active alarms (Page 4318)

Alarm view (as of V13)

Use

Advanced alarm view is used to display alarms on the HMI device.

On HMI devices with device version V13 or later, the advanced table-based alarm view is available for managing alarms.



Alarm line

The alarm line allows display of the most current pending alarm in Runtime. The following figure shows an alarm line:

How to configure the alarm line:

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select "Mode > Alarm line".

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object.

Note

The fonts available for selection depend on the fonts you have configured in the Runtime settings under "Language & font" and the fonts supported by your HMI device.

You can adapt the following properties in particular:

- Toolbar: Defines the operator controls of the alarm view.
- Alarm classes: This setting defines which alarm classes are displayed in the alarm view.
- Columns: Specifies the displayed columns in Runtime.
- Columns moveable: Specifies whether the sequence of columns can be changed in Runtime.
- Identification of alarm classes: To be able to distinguish various alarm classes, they are identified in the first column of the alarm view.
- Filter: Defines that only alarms that contain a specific character string will be displayed in the alarm text.
- Define display area: This setting specifies the date from which alarms are displayed in the alarm view.
- Sorting by date/time possible: Specifies whether the alarms can be sorted in Runtime by date and time.

  Note

  If you have different alarm classes output, these will be initially sorted into alarm classes in Runtime, and then by the time at which the alarm occurred.

Operator controls

The operator controls that can be used to control the alarm view in Runtime are specified in the Inspector window under "Properties > Properties > Toolbar". The following table shows the operator controls in the alarm view and their function:

Button	Designation	Function
	"Tooltip"	Displays a tooltip for an alarm.
	"Acknowledge"	Acknowledges an alarm.
	"Loop-in-alarm"	If a screen change is configured, the display switches to a screen containing information about the alarm.

Select alarm classes

1. Click "Properties > Properties > General" in the Inspector window.
2. Under "Alarm classes", activate the alarm classes to be displayed in the alarm view in Runtime.

Access protection in Runtime

You can configure access protection in the "Properties > Security" group in the properties of the alarm view. A logged-on user with the required authorization can acknowledge and edit alarms using the operator controls in the alarm view. If the logged-on user does not have the required authorization, or if no user is logged in, pressing the "Acknowledge" or "Edit" buttons or double-clicking an alarm line opens the logon dialog.

Define columns

Define the columns to be displayed in the alarm view in Runtime in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Activate the columns that are to be displayed in Runtime under "Visible columns".

Sequence of columns

If this property is enabled, the column sequence in the alarm view can be changed in Runtime.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Column properties > Columns moveable".

Sorting

If this property is enabled, the alarms displayed in the alarm view in Runtime can be sorted by date and time.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Column properties > Sorting by date/time possible".

Filter alarms

This property is used to define that only alarms that contain a configured string in the alarm text will be displayed in Runtime for the extended alarm view.

1. In the Inspector window, select "Properties > Properties > Filter.
2. In the "Filter string" field, enter the desired term for filtering.
   Alternatively, you can configure a filter tag using the "Filter tag" field. The contents of the filter tag serve as the filter criterion in Runtime.

Identify alarm classes

An icon is displayed in the first column of the alarm view. This symbol allows the alarm to be allocated to an alarm class.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Visible columns > Alarm class name".
3. Open the "Alarms" editor, and click the "Alarm classes" tab.
4. Specify an icon for use to identify the alarms of this alarm class for an alarm class in the "Display name" column.

Note

The "alarm view" object cannot be grouped.

Define display area

You select a tag which defines the time from which alarms should be displayed. The following data types are permitted:

- External tags: Date, Date_and_Time, Time_of_Day
- Internal tags: DateTime

To define the display area, proceed as follows:

1. Click "Properties > Properties > Display" in the Inspector window.
2. Define the tag in which the time is specified under "Control tag for display area".

Displaying column headers

The layout of the alarm view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Layout tab" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

Alarm window

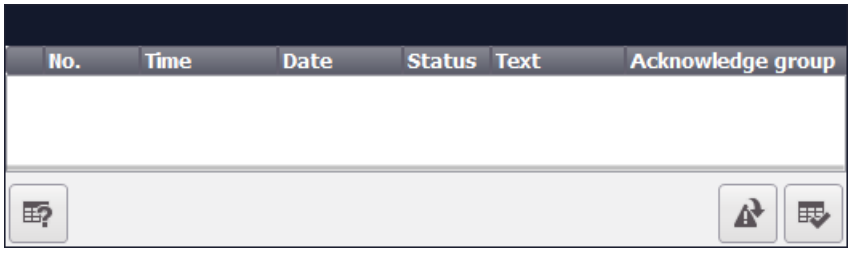
Use

Alarms are indicated in the Alarm view or in the Alarm window of the HMI device. The layout and operation of the Alarm window are similar to that of the Alarm view. The Alarm window has the following characteristics that are the same as in the Alarm view:

- Alarm window
- Alarm line: The alarm line is not available on Basic Panels.

The Alarm window is configured in the "Global screen" editor.

The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active. Depending on the configuration, it is not closed until the alarm is acknowledged.

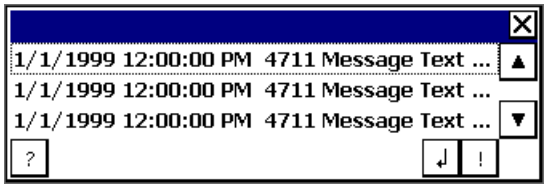


Note

In the engineering system you dynamize, for example, the visibility of an object in "Properties > Animations" in the inspector window. In Runtime, the "Simple alarm window" object does not support animations. If you have configured an animation and, for example, wish to perform a consistency check of the project, then an error alarm is issued in the Output window.

Simple alarm window

The simple alarm window is available on Basic Panels with a device version prior to V13 to display the alarms.



Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You configure the Alarm window in the same way as the Alarm view. In addition you adapt the following properties:

- Fixed alarm window: Specifies that the Alarm window retains the focus after a screen change.
- Window: You define the operator input and response of the Alarm window in Runtime.

Note

If you have different alarm classes output, these will be initially sorted into alarm classes in Runtime, and then by the time at which the alarm occurred.

Operator controls

The operator controls that can be used to control the alarm view in Runtime are specified in the inspector window under "Properties >Display > Settings". The following table shows the operator controls in the Alarm window, and what they do:

Button		Function
"Tooltip"		Displays a tooltip for an alarm.
"Acknowledge"		Acknowledges an alarm.
"Loop-In-Alarm"		If a screen change is configured, the display switches to a screen containing information about the alarm.

Access protection in Runtime

Configure access protection under "Properties > Properties > Security" in the inspector window of the alarm view. If a logged-on user has the required authorization, he can acknowledge and edit alarms using the operator controls in the alarm view. If the logged-in user does not have the required authorization, or if no user is logged in, clicking the "Acknowledge" or "Edit" buttons or double-clicking an alarm line opens the login dialog box.

Note

Basic Panels

Access protection is not available for Basic Panels.

Activating the focus of the Alarm window

Select the following option so that the Alarm window does not lose the focus after a screen change:

1. In the Inspector window, select "Properties > Properties > Mode".
2. Enable "Label".

Window

Define the response of the Alarm window under "Properties > Properties > Mode > Window" in the inspector window. The following table shows the possible properties:

Option	Function
Automatic display	The Alarm window is automatically displayed when a system alarm occurs, for example.
Closable	The window closes again after a set time has elapsed. You define the display duration in the alarm settings.
Modal	The Alarm window is linked to a confirmation, such as: Alarm must be acknowledged. If the modal alarm window has the focus, the buttons in the screen behind it cannot be used. The functions configured on a function key are carried out.
Sizeable	You can change the size of the Alarm window in Runtime.

See also

Device dependency of the objects (Page 4088)

Extended alarm view (Page 4134)

Alarm indicator (Page 4144)

Configuring an alarm window (Page 4325)

Alarm indicator

Application

The alarm indicator is a graphic symbol that shows current errors or errors that need to be acknowledged, depending on the configuration. The alarm indicator is configured in the "Global screen" editor. The following figure shows an alarm indicator:

Alarm indicator OP 73

A "simple" alarm indicator is available for the HMI OP 73. The following diagram shows the alarm indicator for the OP 73 HMI devices:

The "simple" alarm indicator shows alarms to be acknowledged or alarms which have already been acknowledged and have not yet gone. Only the position can be defined for the "simple" alarm indicator. The alarm indicator is displayed on the device at the selected position. If you have configured a screen object at this position, the visible alarm indicator covers the screen object. The alarm indicator is covered by system dialogs, such as the login dialog, Help dialog, and alarm windows.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Alarm classes: Establishes the alarm classes where the alarm indicator is displayed.
- Operator control in Runtime: Defines the operator actions in Runtime that cause the Alarm window to open.

Alarm classes

You define which alarm classes are shown with an alarm indicator in "General > Alarm classes" in the Inspector window. Alarm classes, such as "Warnings" or "Errors".

Define operator control in Runtime

1. Select the alarm indicator in the screen.
2. Click "Events > Click" or "Click when flashing" in the Inspector window.
3. The "function list" opens. Click the first line of the function list. The list of system functions, and scripts available in the project opens.
4. Select the "ShowAlarmWindow". system function under "Alarms."
5. Under "object name" select the name of the Alarm window from the selection list. Under "Layout", define whether the Alarm window should be visible, hidden, or should toggle between the two states.

See also

Device dependency of the objects (Page 4088)

Extended alarm view (Page 4134)

Alarm window (Page 4142)

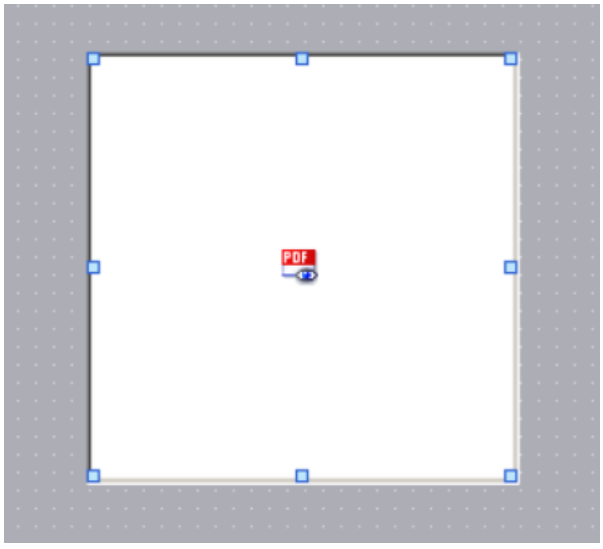
Configuring an alarm indicator (Page 4326)

PDF view

Application

Using the object "PDF view" you project a PDF view on which the operator can open and display documents in PDF format in runtime. The documents in PDF format are located in the internal or external memory of the respective HMI device in this case.

The PDF view supports the PDF version 1.7.



You can configure the PDF view in the following:

- In a screen
- In a template
- In a slide-in screen
- In a pop-up screen

Note

The "PDF view" object may be configured multiple times in a project. However, a maximum of one PDF view may be active at the same time.

Note

Device dependency of the "PDF view" object

The "PDF view" object is only available for KTP Mobile Panels, Comfort Panels and RT Advanced.

Note

Only one "PDF view" object may be inserted per screen.

Layout

You provide information about the position and size of the PDF view in the Inspector window. You can adapt the following properties in particular:

- File name: Specifies the PDF file that opens by default in the PDF view.
- File name tag: Specifies the tag that opens any PDF file.

Basic procedure

1. Create an internal tag of data type "String" in the "Tag" editor, for example, "PDFFile".
2. Insert the PDF view in the screen of the "Screens" editor. The PDF view is available in the "Tools > Controls" task card.
3. In the Inspector window, select "Properties > Properties > General".
4. Enter the file name of the PDF file that opens by default in runtime in "File name". You can also leave the "File name" field blank.
5. For "File name tag", select the "PDFFile" tag from the selection list.
6. Insert an I/O field in the screen in the "Screens" editor.
7. In the Inspector window of the I/O field "Properties > Properties > General > Tag" select the "PDFFile" tag from the selection list.
8. Assign the PDF view the "PDFFile" tag under "File name tag" in the Inspector window.
9. Create a text field for the label in the configured I/O field, for example, "PDF path".

If the operator enters the file name along with the path in the I/O field, the PDF view opens the file in question.

Operator controls

Operator controls are configured buttons in the process picture or softkeys on the HMI device for operating the object.

The "PDF view" object does not have its own operator controls. Configure operator controls yourself, for example, buttons, and link the operator controls with system functions.

1. For example, drag the "Button" object to the screen from the toolbox.
2. Enter a text of any length.
3. In the Inspector window, select "Properties > Events > Click". The "Function list" dialog box is opened.
4. The system functions for operation of the PDF view are available in the list of system functions in the group "Keyboard operation for screen objects". From the list select a function, for example, "PDFZoomIn".
5. From the "Screen object" list, select the object name of the PDF view in which the command is executed.

System functions

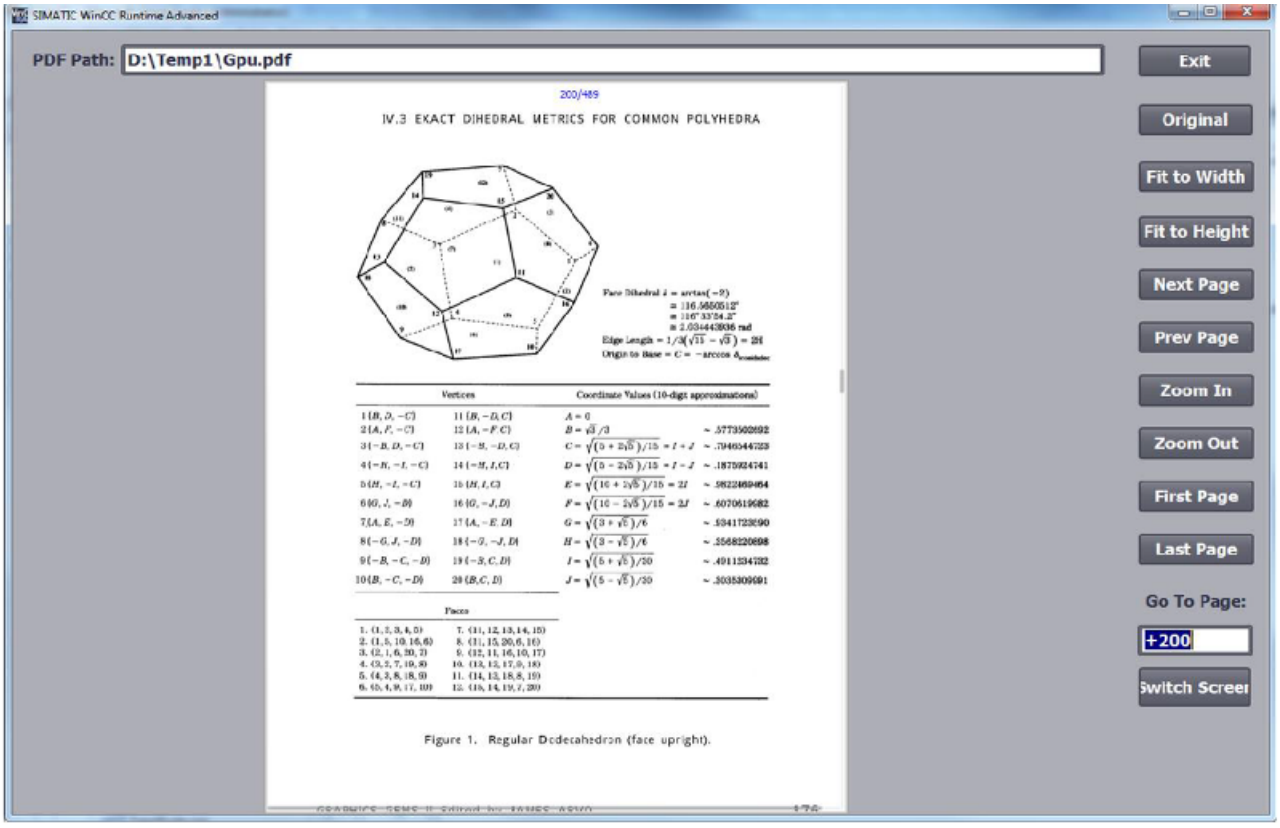
The following system functions are available for the "PDF view" object:

System function	Description
"PDFZoomIn"	Zooms in one zoom level
"PDFZoomOut"	Zooms out one zoom level
"PDFGotoNextPage"	Moves to the next page
"PDFGotoPrevious-Page"	Moves to the previous page
"PDFGotoFirstPage"	Moves to the first page
"PDFGotoLastPage"	Moves to the last page
"PDFFitToWidth"	Fits the display to fit the width of the PDF view window
"PDFFitToHeight"	Fits the display to the height of the PDF view window
"PDFZoomOriginal"	Changes the display to the original size
"PDFScrollUp"	Scrolls up
"PDFScrollDown"	Scrolls down
"PDFScrollLeft"	Scrolls to the left
"PDFScrollRight"	Scrolls to the right
"PDFGotoPage"	Moves to the specified page

Note

Scrolling on all HMI devices also works with multi-touch operation.

The figure below shows a PDF view with the configured operator controls:



Comments

Compared to Adobe Acrobat, the PDF view has limited functionality:

- The PDF view does not support links.
- The PDF view supports zoom levels 25% to 200%.

Note

The "PDF view" object cannot be addressed by using VB scripting.

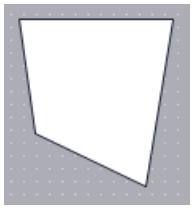
See also

Device dependency of the objects (Page 4088)

Polygon

Application

The "Polygon" is a closed object which you can fill with a background color.



Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. In particular, you can customize the following property:

Geometry: Modifies, deletes or adds corners.

Geometry

The corner points are numbered in the order of their creation. You can change, delete, or add more corner points:

1. In the Inspector window, select "Properties > Properties > Layout".
2. Select the required corner point in the "Geometry" area. Input a value at "X position " and "Y position ".
3. To add a corner point, click the button.
4. To delete a corner point, click the button.

Alternative procedure

You can also change, delete, or add new corner points using the mouse:

1. Select the object. Position the mouse cursor on the desired corner. The mouse cursor changes to a crosshair.
2. Drag the corner to the desired position while holding down the mouse button.
3. Right-click the position at which you want to insert the corner point. Select "Add point" from the shortcut menu.

Configuring Rotation in runtime

You configure the "Polygon" object so that it rotates about a reference point in runtime. The rotation in runtime is only possible for devices with Runtime Professional.

Enter the values for the angle of rotation using Degrees as the unit.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter values for the following attributes in the "Rotation" area.
   - X
   - Y
   - Angle

See also

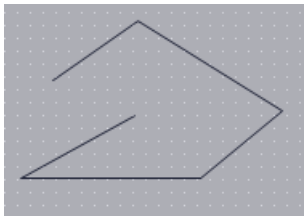
Device dependency of the objects (Page 4088)

Polyline (Page 4151)

Polyline

Application

The "Polyline" is an open object. Use the "Polygon" object if you want to fill the object with color.



Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Line start and end: Specifies the type of line start and line end.
- Geometry: Modifies, deletes or adds corners.

Line start and end

You define the start point and the end point of the line under "Properties > Properties >"Appearance" in the Inspector window. Use arrow point, for example, as start and end point. The available start and end points depend on the device.

Geometry

The corner points are numbered in the order of their creation. You can change, delete, or add more corner points:

1. In the Inspector window, select "Properties > Properties > Layout".
2. Select the required corner point in the "Geometry" area. Input a value at "X position" and "Y position".
3. To add a corner point, click the button.
4. To delete a corner point, click the button.

Alternative procedure

You can also change, delete, or add new corner points using the mouse:

1. Select the object. Position the mouse cursor on the desired corner. The mouse cursor changes to a crosshair.
2. Drag the corner to the desired position while holding down the mouse button.

Configuring rotation in Runtime

You configure the "Polyline" object so that it rotates about a reference point in Runtime. The rotation in Runtime is only possible for devices with Runtime Professional.

Enter the values for the angle of rotation using Degrees as the unit.

1. Click "Layout" in the Inspector window.
2. Enter values for the following attributes in the "Rotation" area.
   - X
   - Y
   - Angle

Further information can be found in "Auto-Hotspot".

See also

Polygon (Page 4150)

Rectangle

Application

The "Rectangle" is a closed object which you can fill with a color.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Corner radius: Specifies the horizontal and vertical distance between the corner of the rectangle and the start point of a rounded corner.

Corner radius

The corners of the "Rectangle" object can be rounded to suit your requirements. When the properties "X" and "Y" are set to the 100 % value, the rectangle is displayed as an ellipse. As soon as one of the properties has the value 0%, a normal rectangle without a rounded corner is shown.

1. Click "Properties > Properties > Layout" in the inspector window.
2. Enter a value for "X" in the "Corner radius" area.
   The input value is the percentage proportion of half the width of the rectangle.
3. Enter a value for "Y" in the "Corner radius" area.
   The input value is the percentage proportion of half the height of the rectangle.

See also

Device dependency of the objects (Page 4088)

Rotating objects (Page 3949)

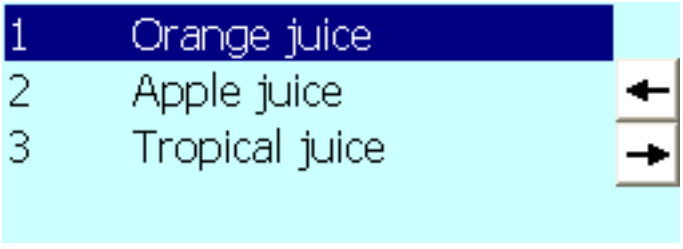
Flipping objects (Page 3951)

Recipe view

Simple recipe view

Application

The "Recipe view" object is used to display and modify recipes.



Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Toolbar: Specifies the menu commands of the recipe view.

Toolbar

The menu items with which the recipe view is operated in Runtime are configured under "Properties > Properties > Toolbar" in the Inspector window.

Menu command	Description
"Tooltip"	Calls up the configured tooltip for the selected recipe.
"New record"	Creates a new recipe record in the recipe.
"Delete record"	Deletes the selected record.
"Saving"	Saves the modified record with its current name.
"Save as"	Saves the modified record with a new name.
"Write to PLC"	Sends the current value to the PLC.
"Read from PLC"	Reads the current value from the PLC.
"Rename"	Specifies that the "Rename" button is shown.
"Forward"	Specifies that the menu buttons are visible.
"Back"	Specifies that the "Back" button is shown.

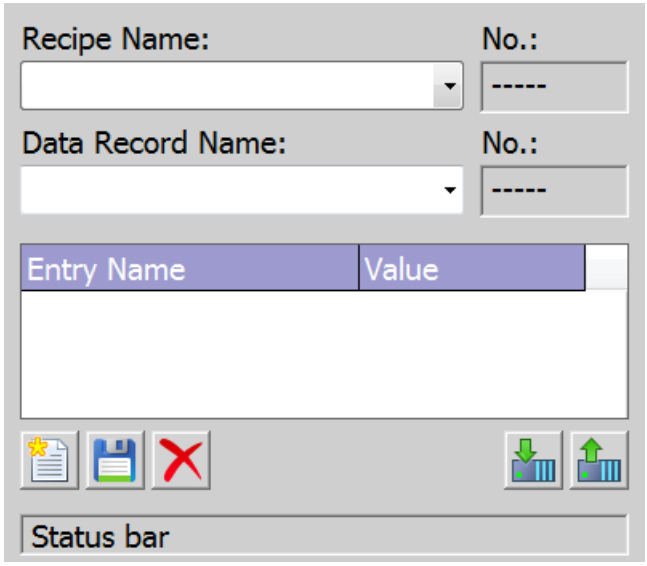
See also

Objects for Basic Panels (Page 4088)

Advanced recipe view

Advanced recipe view

The "Recipe view" object is used to display and modify recipes. The advanced or the simple user view is available depending on the HMI device

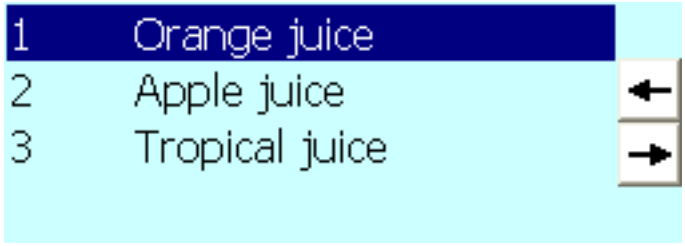


Changing the view type

On HMI devices with a display larger than 6", both the advanced and the simplified recipe view can be used to manage and process recipes. The view type is only available for configuration on devices up to device version V13.

Note

Do not use the simple recipe view in a group.



Configuring the simple recipe view:

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select "Mode > Display type > Simplified".

Note

Behavior for Runtime Professional, Runtime Advanced and Panels

In the Engineering System you can dynamically control the visibility of an object, for example, in the "Animations" tab of the Inspector window. In Runtime, the "Simple recipe view" does not support animations. If you have configured an animation and, for example, want to check the consistency of the project, an error alarm is displayed in the Output window.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Operator controls: Defines the operator controls of the recipe view.
- Display number: Defines whether the number of the recipe and the number of the recipe data record is displayed.

Operator controls

The operator controls with which the recipe view is operated in Runtime are configured under "Properties > Properties > Buttons" in the Inspector window. In the simple recipe view, the operator controls are based on the functions of the menu.

Operator control		Description
	"Tooltip"	Calls up the configured tooltip for the selected recipe.
	"Add data record"	Creates a new recipe data record in the recipe.
	"Delete data record"	Deletes the selected data record.
	"Save"	Saves the modified record with its current name.
	"Save as"	Saves the modified record with a new name.
	"Write to PLC"	Sends the current value to the PLC.
	"Read from PLC"	Reads the current value from the PLC.
	"Synchronize recipe tags"	Compares the values of the selected data record with the values on the PLC.

Display number

Specifies whether the number of the recipe and the number of the recipe data record are displayed in Runtime. The number of the recipe uniquely identifies the recipe in the project.

1. Click "Properties > Properties > View" in the Inspector window.
2. Select "Display > Display numbers".

Configuration behavior

Displaying column headers

The layout of the recipe view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Appearance" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

See also

Device dependency of the objects (Page 4088)

Simple user view (Page 4099)

Advanced recipe view (Page 4444)

Display of recipes (Page 4427)

Configuring the Advanced Recipe View (Page 4464)

Recipe view (as of V13)

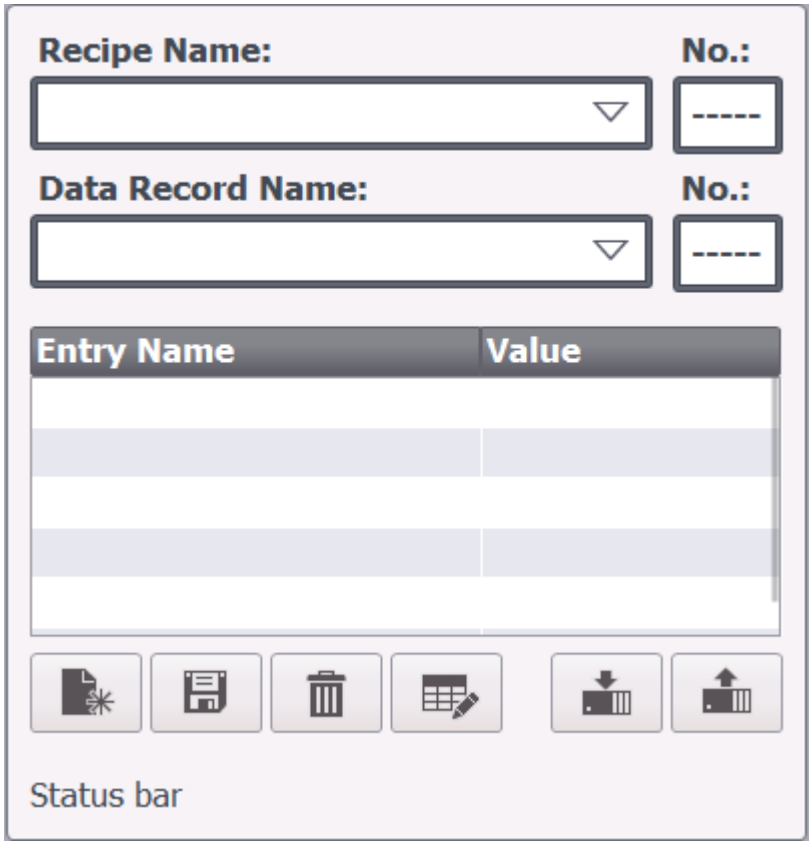
Application

The "Recipe view" object is used to display recipes on the HMI device.

Note

Device dependency of the "Advanced recipe view" object

The "Advanced recipe view" object is available on the HMI devices Basic Panels 2nd Generation, Comfort Panels, Mobile Panels with the device version V13.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Toolbar: Defines the operator controls of the recipe view.
- Display number: Defines whether the number of the recipe and the number of the recipe data record is displayed.
- Label: Sets the label for the name of the recipe and the recipe data record.

Operator controls

The operator controls with which the recipe view is operated in Runtime are configured under "Properties > Properties > Buttons" in the Inspector window. In the simple recipe view, the operator controls are based on the functions of the menu.

Operator control		Description
	"Tooltip"	Calls up the configured tooltip for the selected recipe.
	"Add data record"	Creates a new recipe data record in the recipe.
	"Delete data record"	Deletes the selected data record.
	"Rename data record"	Changes the name of selected data record.
	"Save"	Saves the modified record with its current name.
	"Save as"	Saves the modified record with a new name.
	"Write to PLC"	Sends the current value to the PLC.
	"Read from PLC"	Reads the current value from the PLC.
	"Synchronize recipe tags"	Compares the values of the selected record with the values on the PLC.

Display number

Specifies whether the number of the recipe and the number of the recipe data record are displayed in Runtime. The number of the recipe uniquely identifies the recipe in the project.

1. Click "Properties > Properties > View" in the Inspector window.
2. Select "Display > Display numbers".

Display label

Use this property to specify the name to be shown for recipes and data records in the recipe view.

1. Click "Properties > Properties > Label" in the Inspector window.
2. Select "Display labels".
3. Enter the labels for the recipe and the recipe data record.

Behavior during configuration

Displaying column headers

The layout of the recipe view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Appearance" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

Switch

Application

The "Switch" object is used to configure a switch that is used to switch between two predefined states in runtime. The current state of the "Switch" object can be visualized with either a label or a graphic.

The following figure shows a "Switch" type switch.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. In particular, you can customize the following property:

- Type: Defines the graphic representation of the object.

Type

Button visualization is specified at "Properties > Properties > General >Type" in the Inspector window.

Type	Description
"Switch"	The two states of the "Switch" are displayed in the form of a switch. The position of the switch indicates the current state. The state is changed in runtime by sliding the switch. You specify the direction of movement of the switch in "Switch orientation" with this type.
"Switch with text"	The switch is shown as a button. The current state is visualized with a label. In runtime click on the button to actuate the switch.
"Switch with graphic"	The switch is shown as a button. The current state is visualized with a graphic. In runtime click on the button to actuate the switch.

Note

Basic Panels

The "Switch" type is not available for Basic Panels.

See also

Device dependency of the objects (Page 4088)

Elements (Page 4838)

Overview of objects (Page 3935)

Switch (Page 5167)

Button

Application

The "Button" object allows you to configure an object that the operator can use in Runtime to execute any configurable function.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Defines the graphic representation of the object.
- Text / Graphic: Defines whether the Graphic view is static or dynamic.
- Define hotkey: Defines a key, or shortcut that the operator can use to actuate the button.

  Note

  You can only define a hotkey for HMI devices with keys.

Mode

The button display is defined in "Properties > Properties > General >Mode" in the Inspector window.

Mode	Description
"Invisible"	The button is not visible in Runtime.
"Text"	The button is displayed with text. This text explains the function of the button.

Mode	Description
"Graphic"	The button is displayed with a graphic. This graphics represents the function of the button.
"Graphics or text"	The button is displayed with text or graphics. If the graphics cannot be displayed, the corresponding text is displayed.
"Graphics and text"	The button is displayed with text and graphics.

Different options are available depending on the device.

Text / Graphic

The "Mode" property settings are used to define whether the display is static or dynamic. The display is defined in "Properties > Properties > General >Text" or "Graphic" in the Inspector window.

You can, for example, select the following options for the "Graphic" or "Text" type.

Type	Option	Description
"Graphic"	"Graphic"	Use "Graphic when button is "not pressed"" to specify a graphic displayed in the button for the "OFF" state. Use "Graphic when button is "pressed"" to specify a graphic for the "ON" state.
	"Graphics list"	The graphic in the button depends on the state. The corresponding entry from the graphics list is displayed depending on the state.
"Text"	"Text"	Use "Text when button is "not pressed"" to specify the text displayed in the button for the "OFF" state. Use "Text when button is "pressed"" to specify text for the "ON" state.
	"Text list"	The text in the button depends on the state. The entry from the text list corresponding to the state is displayed.

Define hotkey

In the Inspector window, a key or key combination is defined that the operator can use to control the button in Runtime.

1. In the Inspector window, select "Properties > Properties > General".
2. Select a key or key combination from the selection list in the "Hotkey" area.

See also

Device dependency of the objects (Page 4088)

Example: Configuring a button for language switching (Page 4036)

Example: Configuring a button with logon dialog box (Page 4559)

Example: Configuring a button with access protection (Page 4564)

Slider

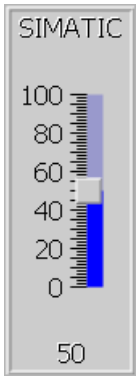
Application

Process values are monitored and adapted within a defined range with the "Slider" object. The monitored range is visualized in the form of a slider. By adjusting the slider, you intervene in the process and correct the displayed process value.

Note

Do not configure a system function at the "Change" event for a regulated project if it is used to execute a GMP-relevant action.

Instead configure the system functions at the "Value change" event of the tags in order to reduce the number of user actions.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can, in particular, adapt the following properties as required:

- Maximum Value and Minimum Value: Specifies the top and bottom values of the scale.
- Display current value: Specifies whether the current position of the controller appears below the slider.
- Display of bars: The sliders above and below the bar can be hidden.

Maximum Value and Minimum Value

The top and bottom end values of the scale are specified in the Inspector window.

1. In the Inspector window, activate "Properties > Properties > General".
2. Enter a number at "Maximum value" and "Minimum value". If you select a tag as the end value of the scale, the number will be no longer available.

Display current value

Specify that the value of the current position is displayed below the slider in the Inspector window.

1. In the Inspector window, activate "Properties > Properties > Layout".
2. Select "Show current value."

Display bar

The bar can be hidden.

1. In the Inspector window, activate "Properties > Properties > Layout".
2. Deactivate "Display bar".

Behavior during operation

The displayed value on the slider control may deviate from the actual value of the associated tag in the following circumstances:

- The value range (minimum and maximum value) configured for the slider control does not correspond to the configured limits for the slider control tag.
- An invalid password has been entered for a password-protected slider control.

See also

Device dependency of the objects (Page 4088)

KeySwitch

Application

The object "KeySwitch" is an optional control element for the SIMATIC Mobile Panel. The key switch is used to lock functions that can be triggered from the Mobile Panel.

You configure the object "key switch" object in the "global screen" or in a template.

Layout

You can set the following properties in the Properties view:

- Tag: Specifies the tag in which the position of the key switch is written.

Tag

The "Tag" property is used to specify the "Key switch" object assignment in the global screen or template.

Note

Notes on configuring

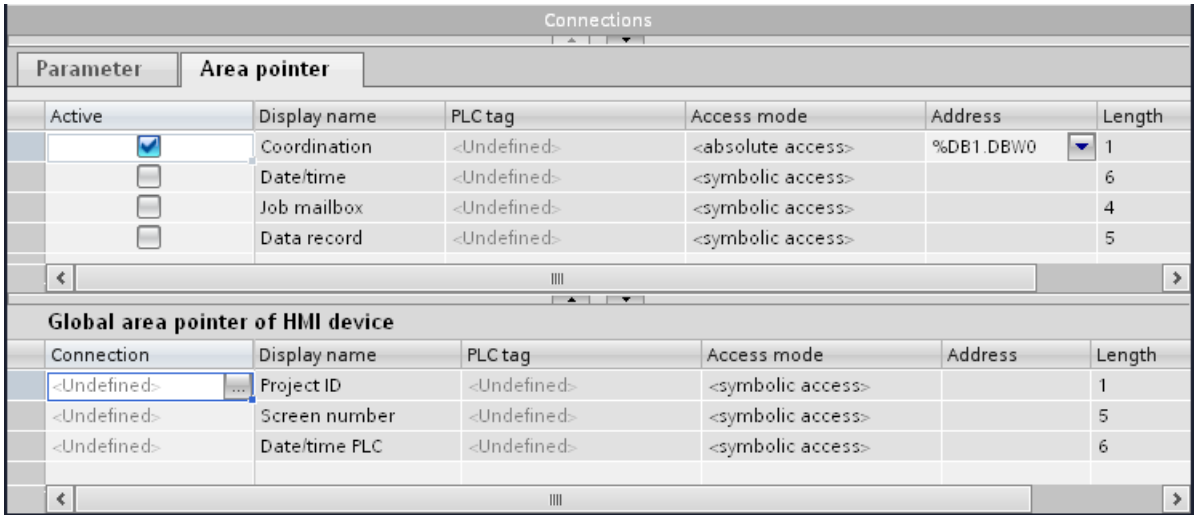
When starting runtime and every time the key is pressed the current position of the key switch is written to the tag. In doing so it can cause inconsistencies between the position of the key switch and the key switch tags.

1. If the tag has a process connection, the current value is transferred from the PLC to the tag after establishment of communication. The tag that contains the current position of the key is overwritten by this process. It can be that the key switch tag no longer reflects the current value of the key switch, e.g. if the Mobile Panel is switched off while the key is being turned.
2. If the key switch is pressed before communication with the PLC has been established (e.g. after a device start-up), it may not be possible to send the value of the key switch tag to the PLC. In this case the value of the tag in the PLC is different from the current position of the key.

A specific configuration is required to make sure that the key switch tag reflects the actual position of the key switch at all times. See the instructions below for the basic procedure.

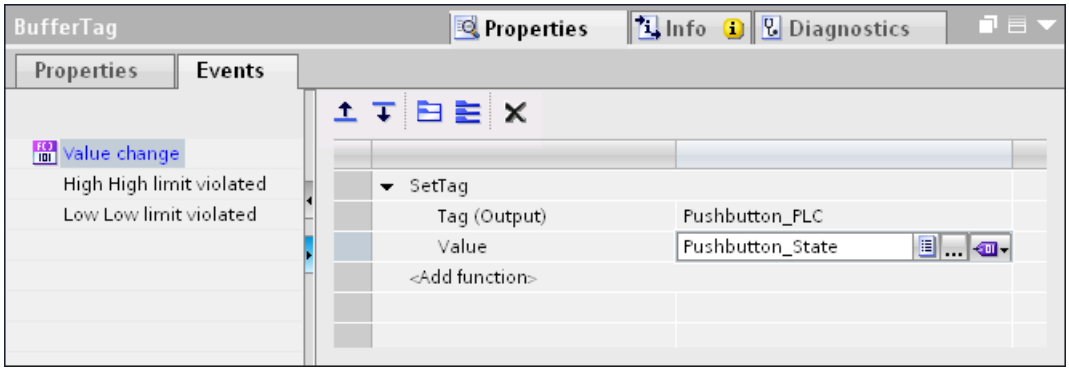
Basic procedure

1. Specify the connection to the PLC in the "Connections" editor. Activate the "Coordination" area pointer to ensure that the life bit is available to the PLC side.



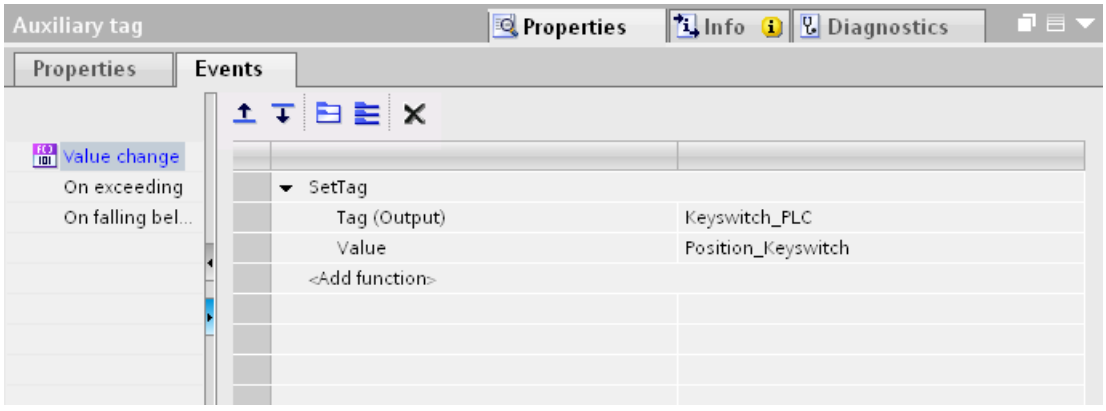
2. Create three tags in the "Tag" editor.
   - Internal tag: Position_Keyswitch
   - External tag: Auxiliary tag
   - External tag: Keyswitch_PLC

3. Open the Inspector window of the "Position_KeySwitch" tag.
4. In the Inspector window, select "Properties > Events > Value change".
5. Click the first line of the function list. The list of system functions available in the project appears.
6. Select the "SetTag" system function.
   - Select the "Keyswitch_PLC" tag at "Tag (output)".
   - Select the "Position_Keyswitch" tag at "Value".



   The value of the "Keyswitch_PLC" tag is written to the PLC with the "Position_Keyswitch" tag. A program in the PLC evaluates the life bit. If communication is established, the current value is written to the "Keyswitch_PLC" tag from the PLC.
7. Open the Inspector window of the "Auxiliary tag" tags.
8. In the Inspector window, select "Properties > Events > Value change".
9. Click the first line of the function list. The list of system functions available in the project appears.

10. Select the "SetTag" system function.
    - Select the "Keyswitch_PLC" tag at "Tag (output)".
    - Select the "Position_Keyswitch" tag at "Value".



    After establishing communication the current value is written to the "Auxiliary Tag" from the PLC. The "SetTag" system function is executed by the value change in the auxiliary tags. The system function re-allocates the value of the "Position_Keyswitch" tag to the "Keyswitch_PLC".

11. Open the global screen or a template in the "Screen navigation" editor.
12. Select the key switch in the screen.
13. Select the "Position_KeySwitch" tag in the Inspector window under "Properties > Properties > General > Settings > Tag".
    If you press the illuminated pushbutton, the value is written to the "Position_KeySwitch" tag.

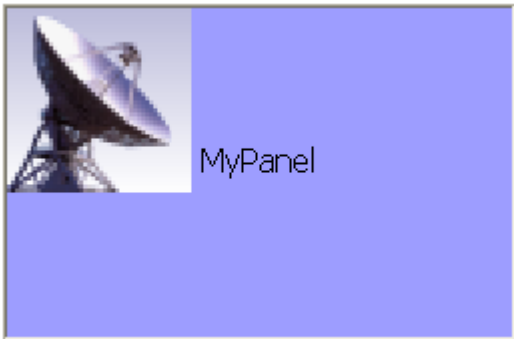
See also

Field of application of the Mobile Panel Wireless (Page 5944)

Sm@rtClient view

Application

The "Sm@rtClient View" object is used to configure a network connection to the remote monitoring and remote maintenance of a connected unit.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Connect on start: Establishes whether when opening the Sm@rtClient view the connection to the HMI device is automatically created which should be remote-controlled.
- Shared use_ Specifies that an HMI device is shared by several Sm@rtClient Views.
- Local cursor: Specifies whether the cursor data is transferred separately in order to increase the performance.
- Use cursor key scroll: For keyboard devices places the control of the horizontal and vertical scroll with the cursor keys.
- Show controls: Establishes whether additional fields exist for entering the address and password.
- Allow menu: Specifies whether the shortcut menu is enabled to control the Sm@rtClient view.
- Connection type: Establishes the expected transfer speed for the remote monitoring.
- Scaling: Specifies whether the Sm@rtClient view can be zoomed in or out.

Connect on start

Establishes whether when opening the Sm@rtClient view the connection to the HMI device is automatically created which is remote-controlled.

1. In the Inspector window, select "Properties > Properties > General".
2. Select "View > Connect on start".

Shared use

Define whether the HMI device can be used by several Sm@rtClient views under "Properties > Properties > General > View > Shared".

Shared use	
Enabled	HMI devices on which remote control is activated can access the remote HMI device and control the process. Only one HMI device can be active at a time in this case. A different HMI device can only assume control when there has been no activity on the active HMI device for a specified period of time.
Disabled	Only one HMI device can use the remote control at any given time. The activities can be followed on the other HMI devices. Depending on the settings on the Sm@rtServer the following options are available if another HMI device logs on: • The HMI device is rejected. • The active HMI device is disconnected and the new HMI device is connected.

Local cursor

If this property is set, then the cursor data is transferred separately.

1. In the Inspector window, select "Properties > Properties > General".
2. Select "View > Local cursor".

Display extended objects

If this property is set then there are additional fields for inputting the address and password. For the fields to be shown the "Connect on start" option is not activated.

1. In the Inspector window, select "Properties > Properties > General".
2. Select "View > Show enhanced objects".

Allow menu

If this property is set, then a shortcut menu to control the Sm@rtClient view is opened in Runtime.

When operating using the mouse, the left mouse key must remain depressed for a few seconds for the shortcut menu to open.

1. In the Inspector window, select "Properties > Properties > General".
2. Select "View > Allow Menu".

Connection type

The "Connection type" property establishes the expected transfer speed for the remote monitoring.

Set the connection type at "Connection type" under "Properties > Properties > General > Connection type" in the Inspector window.

Connection type	
LAN	Connection via network
Slow	Connection with a low transfer speed
Medium speed	Connection with a medium transfer speed
Modem	Connection via an analog modem
Very slow	Connection with a very low transfer speed

Scaling

If this property is set, then via "Client scale factor" and "Server scale" you can reduce or enlarge the Sm@rtClient view on the HMI device.

1. In the Inspector window, select "Properties > Properties > Scaling".
2. Select "Scaling > Scaling".
3. Enter the values for the scaling under "Client scale factor" and "Server scale factor".

Behavior during operation

If the program cannot resolve the dynamic address of a Sm@rtClient view after a screen change, the system connects to the configured server using the static address of the server. If a tag is now set using the dynamic address, you can make a second connection to this server. As a remedy preset the tags of the dynamic address for example by means of a script.

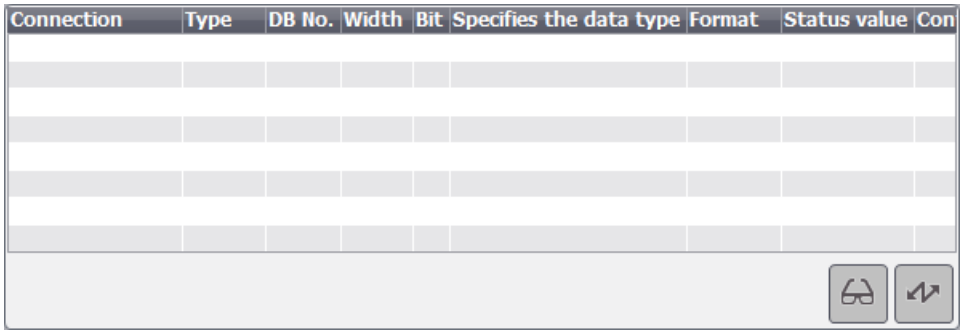
See also

Device dependency of the objects (Page 4088)

Status/Force

Application

The "Status/Force" object is used to configure an editor that will allow you to process single address areas of a SIMATIC S7 in runtime.



Note

The "Status/Force" object is enabled for the following controllers:

- SIMATIC S7-300
- SIMATIC S7-400

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- PLC type: Specifies the PLC type in the object for which the address area is output in runtime.
- Control elements: Specifies the controls of the object.
- Visible columns: Specifies the displayed columns in runtime.
- Columns moveable: Specifies whether the operator can change the sequence of columns in runtime.

Control elements

Set the control elements that you can use to control the "Status/Force" object in runtime at "Properties > Properties > Display" in the "Settings" area of the Inspector window.

	Function
	This button refreshes the display in the status value column.
	Use this button to accept the new value in the control value column. The control value is then written to the PLC.

Visible columns

The columns that are displayed in runtime are specified in the Inspector window in the "General" group in the "Visible columns" area.

Column	Description
"Connection"	The PLC of which the address areas are displayed.
"Type" "DB number" "Offset" "Bit"	The address area of the operand.
"Data type" "Format"	The data type of the operand.
"Status value"	The value that was read from the specified address of the operand.
"Control value"	The value written to the specified address of the operand.

Sequence of columns

The operator can use the "Column ordering" property to change the sequence of columns in runtime.

1. Select "Properties > Properties > Columns" in the Inspector window.
2. Activate "Properties of columns > Sequence of columns".

Display of the column headers and buttons

The layout of the "Status/Force" object is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Appearance" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style. This behavior only occurs during configuration. The column headers are displayed correctly in runtime.

If you change the layout, for example from "Windows Classic" to "Windows XP Style", the button in the "Status/Force" object may no longer be displayed. Change the size of the object to make the buttons in the engineering system visible again.

See also

Device dependency of the objects (Page 4088)

Symbol library

Application

The "Symbol library" object contains a large collection of ready-to-use icons. These icons are used to represent systems and plant areas in screens.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Select icon: Specifies which library object will be used.
- Fill style: Specifies the graphic design of the object.
- Flip: Specifies the flip type of the library object.

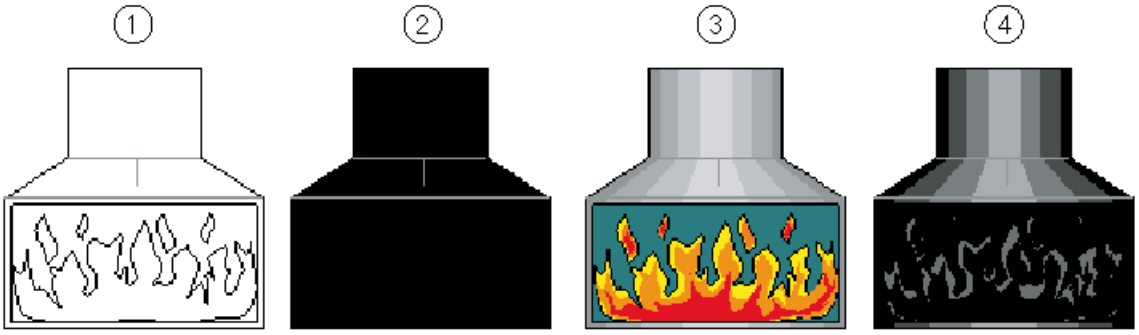
- Rotate: Specifies the angle around which the library object is rotated.
- Fixed aspect ratio Defines whether the aspect ratio is maintained if the size is changed.

Select symbol

Select the icon at "Properties > Properties > General" in the Inspector window.

Fill style

Specify the design of the library object at "Properties > Properties > Appearance > Style" in the Inspector window.



1	"Hollow"	The library object is displayed in outline.
2	"Solid"	Black lines remain as outlines. Elements of the icon with a different color are displayed as a main color.
3	"Original"	The screen object will not be changed.
4	"Shaded"	Black lines remain as outlines. Elements of the symbol in other colors are displayed as brightness levels of a main color.

At the TP 177B mono and OP 177B mono HMI devices the library object is displayed as an outline. "Hollow" is set as the default value for the fill style.

If you change the setting from "hollow" to "original" or "shaded" then the color of the display on the configuration computer will deviate from that on the HMI device. This deviation is caused by the different color resolutions. You may also use a graphic object from the "Graphics" object group. The graphic objects are structured by topic and by depth of color in the "WinCC graphics folder."

Flip and rotate

The content of the screen flips around the horizontal or vertical central axis of the icon. The icon can be flipped horizontally, vertically or simultaneously horizontally and vertically.

The screen content rotates around the central point of the icon. The symbol is rotated clockwise in increments of 90, 180, or 270 degrees.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select the mirroring mode at "Orientation > Mirror image".
3. Select the angle of rotation at "Orientation > Rotate".

Fixed aspect ratio

The smaller page of the library object defines the maximum size of the symbol. If you change the size of the library object unproportionally, the symbol is still scaled proportionally. Proceed as follows to configure a fixed aspect ratio:

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select "Form > Fixed aspect ratio".

Behavior during operation

The possibility of operating the mouse is indicated in Runtime by the cursor symbol changing, depending on the configuration. Operation feedback, e.g. by means of a color change, does not take place.

See also

Device dependency of the objects (Page 4088)

Symbolic I/O field

Application

The "Symbolic I/O field" object can be used to configure a selection list for input and output of texts in Runtime.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Text list: Specifies the text list that is linked to the object.
- Button for selection list: Specifies that the object has a button to open the selection list.

Note

Reports

In reports, symbolic I/O fields only output data. "Output" mode is preset. Properties for configuring the selection of graphics are not available, e.g. "button for selection list".

Mode

The response of the symbolic I/O field is specified in the Inspector window in "Properties > Properties > General > Type".

Mode	Description
"Output"	The symbolic I/O field is used to output values.
"Input"	The symbolic I/O field is used to input values.
"Input/output"	The symbolic I/O field is used for the input and output of values.
"Two states"	The symbolic I/O field is used only to output values and has a maximum of two states. The field switches between two predefined texts. This is used, for example, to visualize the two states of a valve: closed or open.

Note

The behavior possible for the symbolic I/O field depends on the Runtime.

Text list

In the Inspector window, you specify which text list is linked to the symbolic I/O field.

1. In the Inspector window, select "Properties > Properties > General".
2. Under "Contents" open the selection list for "Text list".
3. Select a text list.

Button for selection list

The "Button for selection list" property is used to display a button for opening the selection list.

1. Select "Properties > Properties > Layout" in the Inspector window.
2. Select "Response > Button for selection list".

Note

Basic Panels

The "Button for selection list" option is not available for Basic Panels.

Behavior during operation

The selection list of the symbolic I/O field displays an empty text line if a corresponding entry was not defined in the project. The active state is indicated on the HMI device by changing the color of contents of the symbolic I/O field.

See also

Device dependency of the objects (Page 4088)

Graphic I/O field (Page 4112)

Connecting tags and text lists in the text (Page 3958)

System diagnostics display

Introduction

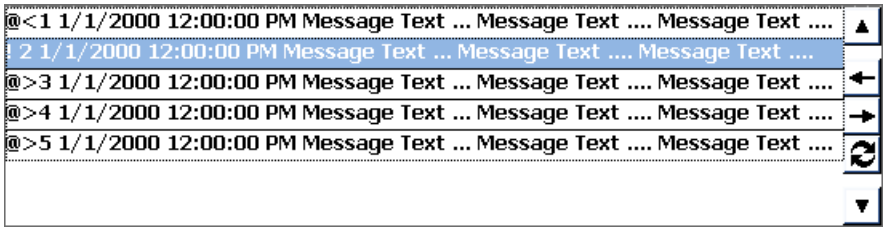
The system diagnostics view offers you an overview of all the available devices in your plant. You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

Use

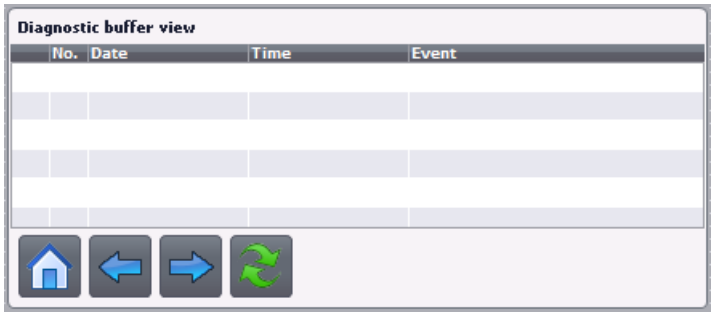
The system diagnostics view enables you to achieve the maximum level of detail of the diagnostics data. A precise diagnosis is possible, as all the available data is displayed. You have the system status of the entire plant at one glance.

Views in the system diagnostics view

A simple system diagnostics view is available on Basic Panels with a device version prior to V13.



The advanced system diagnostics view is available on Basic Panels with a device version of V13 or later.



Three different views are available in the system diagnostics view.

- Device view
- Diagnostic buffer view
- Detail view

Device view

The device view of the system diagnostics view shows all the available connections in tabular form. Double-clicking a connection opens the detail view. The device view is only displayed if more than one connection has been created in the "Devices & Networks" editor.

Diagnostic buffer view

In the diagnostic buffer view, the current data from the diagnostic buffer is displayed.

Detail view

The detail view shows detailed information on the selected connection. You cannot sort error texts in the detail view. The detail view is only available if there is an integrated connection to an S7 1200 or S7 1500.

Layout

You change the settings for the position, geometry, style, color, and font of the object in the Inspector window. You can adapt the following properties in particular:

- Lines per entry: Specifies the number of lines that are shown for an entry.

Configuring the system diagnostics view

1. Drag-and-drop the system diagnostics view from the toolbox.
2. In the Inspector window, select "Properties > Layout".
3. Enter a number under "Lines per entry", i.e. 5.
4. Select an authorization for operation in "Properties > Properties > Security".

See also

System diagnostics display (Page 4177)

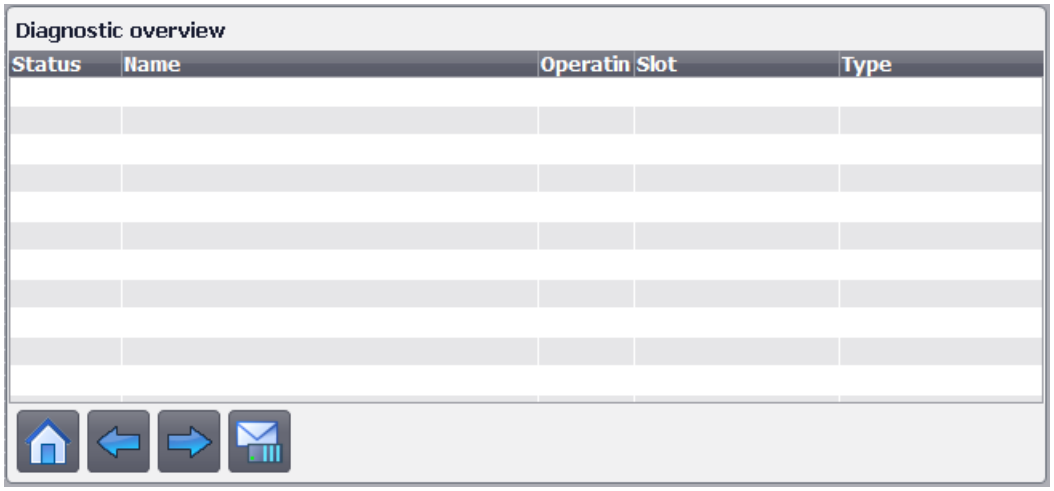
System diagnostics display

Introduction

The system diagnostics view offers you an overview of all the available devices in your plant. You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

Use

The system diagnostics view enables you to achieve the maximum level of detail of the diagnostics data. A precise diagnosis is possible, as all the available data is displayed. You have the system status of the entire plant at one glance.



Three different views are available in the system diagnostics view.

- Device view
- Detail view
- Diagnostic buffer view
- Matrix view (only for master systems, PROFIBUS)

Layout

You change the settings for the position, geometry, style, color, and font of the object in the Inspector window. You can adapt the following properties in particular:

- Columns: Defines the elements of the device view and detail view.
- Show split view: Specifies that the device view and the detail view are shown simultaneously.

Configuring the system diagnostics view

1. Drag-and-drop the system diagnostics view from the toolbox.
2. In the Inspector window, select "Properties > Properties > Columns".
3. Select the columns that you require in the device view for Runtime, for example, Operating mode, Name, Slot.
4. Select the columns which you need in the detail view for runtime, e.g., Operating mode, Name, Address.
5. Select the columns that you require in the diagnostic buffer view, e.g., Status, Name, Rack.
6. Customize the column headers, if necessary.

7. Enter "IP" for "Address", for example.
   The column header "IP" is displayed in the device view in runtime.
8. Rename other elements if necessary.
9. Select an authorization for operation in "Properties > Properties > Security".

Showing device view and detail view in a single display

You can split the system diagnostics view and allow the device and detail view to be shown in the same window.

1. In the Inspector window, click "Properties > Properties > Layout".
2. Activate "Show split view".

Symbols in the system diagnostics view

Symbol	Function
	Device in operation
	Device cannot be accessed
	Errors in device
	Input/output data are not available
	Device deactivated
	Maintenance required
	Maintenance recommended
	Superimposed symbol Shows the subordinate status
	Current configuration
	Stop e.g. Update, Boot, Own initialization
	Stop

See also

Device dependency of the objects (Page 4088)

System diagnostics window (Page 4180)

System diagnostics display (Page 4176)

System diagnostics window

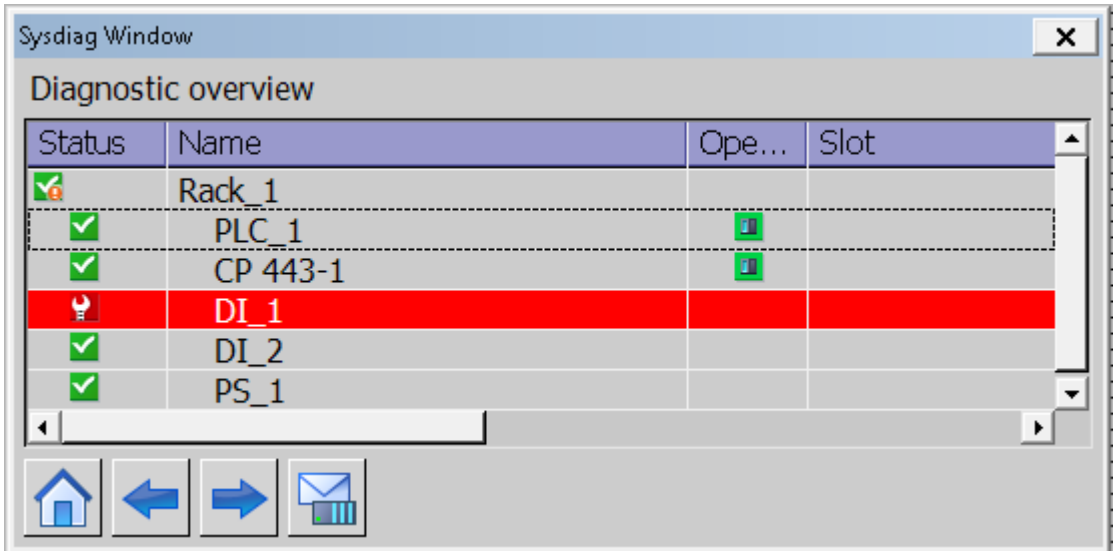
Introduction

The system diagnostics window offers you an overview of all the available devices in your plant. You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

The system diagnostics window is only available in the global screen.

Use

The system diagnostics window enables you to achieve the maximum level of detail of the diagnostics data. A precise diagnosis is possible, as all the available data is displayed. You have the system status of the entire plant at one glance.



Three different views are available in the system diagnostics window.

- Device view
- Detail view
- Diagnostic buffer view
- Matrix view (only for master systems, PROFIBUS)

Layout

You change the settings for the position, geometry, style, color, and font of the object in the Inspector window. You can adapt the following properties in particular:

- Columns: Defines the elements of the device view and detail view.
- Table headers: Specifies the table headers of the view.
- Window: Specifies whether the system diagnostics window can be closed, for example.

Configuring the system diagnostics window

1. Drag and drop the system diagnostics view from the toolbox into the global screen.
2. In the Inspector window, select "Properties > Properties > Columns".
3. Select the columns that you require in the device view for Runtime, e.g., Operating mode, Name, Slot.
4. Select the columns that you require in the detail view for Runtime, e.g., Operating mode, Name, Plant designation.
5. To change a column header, click in the field and (for example) enter "IP" for "Address". The column header "IP" is displayed in the device view in runtime.
6. Rename other elements if necessary.
7. To be able to close the system diagnostics window in runtime, select "Properties > Properties > Window > Closable" in the Inspector window.

Icons in the system diagnostics window

Button	Function
	Device in operation
	Device cannot be accessed
	Errors in device
	Device deactivated
	Input/output data are not available
	Maintenance required
	Maintenance recommended
	Superimposed symbol Shows the subordinate status
	Current configuration
	Stop e.g. Update, Boot, Own initialization
	Stop

See also

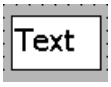
System diagnostics display (Page 4177)

Device dependency of the objects (Page 4088)

Text field

Application

The "Text field" is a closed object which you can fill with a color.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Text: Specifies the text for the text field.
- Size of text field: Defines whether the size of the object is adapted to the space required by the largest list entry.

Text

Specify the text for the text field in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Enter a text.
   For texts over several lines you can set a line break by pressing the key combination <Shift + Enter>.

Size of text field

In the Inspector window, you can define whether the size of the object is adapted to the space required by the largest list entry.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Activate "Resize > Fit to contents".

See also

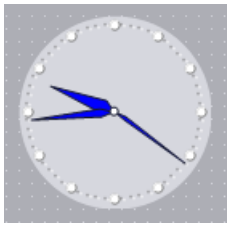
Device dependency of the objects (Page 4088)

Connecting tags and text lists in the text (Page 3958)

Clock

Application

The "Clock" object displays the date and time.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Analog display: Specifies whether the clock is shown as an analog clock or digital clock.
- Display clock dial: Specifies whether hour marks of the analog clock will be displayed.
- Width and length of hands: Specifies the width and length of the hands.

Analog display

In the Inspector window you can specify whether the clock is displayed as an analog or digital clock. The digital clock shows both the time and the date. The display format depends on the language set on the HMI device.

1. In the Inspector window, select "Properties > Properties > General":
2. Activate "Analog".

Display dial

In the Inspector window you can specify whether hour marks will be displayed.

1. In the Inspector window, select "Properties > Properties > General":
2. Activate "Analog".
3. Activate "Show dial".

Width and length of hands

The width of the second hand, minute hand and hour hand is set for the analog display.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter values for "width" and "length".
   The values for the length are interpreted as a percentage of the size of the clock. The value for the length of the second hand influences the radius of the dial at the same time.

See also

Device dependency of the objects (Page 4088)

Effective range name

Use

The "Effective range name" object is a operator control for some SIMATIC Mobile Panels. The "Effective range name" object shows the names of the effective range in which the HMI device is located and also the logon status.

Layout

Symbol	Meaning	Description	Logon status
	Acknowledgement button has no effect	Within the "MixingAxisControl" effective range	It is not possible to log on to the effective range
	Acknowledgement button functions	Within the "MixingAxisControl" effective range	Logged on to effective range
	Acknowledgement button has no effect	Within the "MixingAxisControl" effective range	Logon to the effective range is rejected because a different HMI device is already logged on. Note: When using the "Override" mode: Although no other HMI device is still logged on to the effective range, logon is rejected because the override switch is still set.
	Acknowledgement button has no effect	Outside of each effective range	Note: You can only log on within the effective range

In the Inspector window "Properties > Properties > Security", assign an authorization to log on to a machine in runtime.

Logon requirements

- The HMI device is located within the effective range.
- No HMI device is logged on to this effective range.

Logon procedure

1. Click on the "Effective range name" object.
   The "Effective range logon" dialog box opens.
2. Read the ID from the machine or plant to be operated.
3. Enter the ID you read.
4. Confirm your entries by clicking "OK".

Result

If the ID and effective range match, the HMI device is logged on to the effective range. Successful logon is signaled by an icon on green background. No other HMI device can log on to this effective range.

Requirement for logging off

The HMI device is logged on to this effective range.

Logoff procedure

Before you leave the effective range or runtime, you must log off from the effective range first. If you leave the effective range without logging off, a warning appears.

1. Click on the "Effective range name" object. The "Effective range logoff" dialog box opens.
2. Confirm logoff with "Yes".

Result

The HMI device is no longer logged into the effective range. Other HMI devices can now log on to this effective range.

Note

Do not deactivate the "Override" mode again unless you logon a different HMI device.

See also

Device dependency of the objects (Page 4088)

Charge condition (Page 4125)

Effective range signal (Page 4187)

WLAN reception (Page 4189)

Zone name (Page 4192)

Zone signal (Page 4193)

Configuring the effective range name (Page 5973)

Effective range name (RFID)

Use

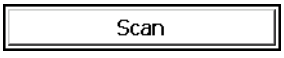
The "Effective range name (RFID)" object is a operator control for some SIMATIC Mobile Panels. The "Effective range name (RFID)" object shows the name of the effective range (RFID) in which the HMI device is located and also the logon status.

Note

RFID tags in the plant area

RFID tags may only be attached in demarcated areas which are secured by additional protective measures, e.g. protective fencing.

Layout

Symbol	Meaning	Description	Logon status
	Acknowledgement button has no effect	Within the effective range (RFID) "MixingAxisControl"	It is not possible to log on to the effective range (RFID)
	Acknowledgement button functions	Within the effective range (RFID) "MixingAxisControl"	Logon to effective range (RFID) running
	Acknowledgement button functions	Within the effective range (RFID) "MixingAxisControl"	Logged on to the effective range (RFID)

In the "Properties > Properties > Security" category of the Inspector window, you can assign an authorization for the user group that logs on to the effective range (RFID) during runtime.

Logon requirements

- The HMI device is located within the effective range (RFID).
- The machine is demarcated by additional protective measures.
- No HMI device is logged on to this effective range (RFID).

Logon procedure

1. Click on the "Effective range name (RFID)" object.
   The "Effective range logon" dialog box opens.
2. Read the ID from the machine or plant to be operated.

3. Enter the ID you read.
4. Confirm your entries by clicking "OK".

Result

If the ID and effective range (RFID) match, the HMI device is logged on to the machine. Successful logon is signaled by an icon on green background. No other HMI device can log on to this effective range (RFID) or the machine.

Logoff requirements

The HMI device is logged on to this effective range (RFID).

Logoff procedure

Before you exit the effective range (RFID) or runtime, you must first log off from the effective range (RFID). If you exit the effective range (RFID) without logging off, a local rampdown is executed.

1. Click on the "Effective range name (RFID)" object. The "Effective range (RFID) - logoff" dialog box opens.
2. Confirm logoff with "Yes".

Result

The HMI device is no longer logged into the effective range (RFID) or machine. Other HMI devices can now log on to this effective range (RFID).

See also

Device dependency of the objects (Page 4088)

Charge condition (Page 4125)

WLAN reception (Page 4189)

Configuring the effective range name (RFID) (Page 5976)

Effective range signal

Application

The "Effective range signal" object is a control element for some SIMATIC Mobile Panels. The "Effective range signal" object indicates how accurately the Mobile Panel Wireless is positioned within an effective range.

Layout

Symbol	Description
	Within an effective range
	When leaving an effective range When entering an effective range
	Outside of each effective range
	Override function is activated: Within a special security range of the effective range

The "Effective range name" object indicates which effective range this concerns.

Override function

An effective range, e.g. "robots" is formed outwardly through transponders. However, within a special protection zone, e.g. "robot cells", the effective range is not formed by transponders, but by alternative restrictions e.g. by railings and a door.

The operator initially logs on to the effective range. An operator who enters the restricted and secure zone "robot cells" must press an additional switch to activate the override function:

- The effective range is allocated and can no longer be established by the transponder.
- Therefore, the quality does not play a role and is no longer displayed.
- The HMI device remains logged into the effective range.
- Other HMI devices can no longer logon to this effective range.

Calculating the signal quality

The quality within an effective range depends, as follows, on the measured distance:

- In the center of the effective range the quality is 100%.
- On the transponder and on the border of the effective range, the quality is 0%.

Operation

The object is for display only and cannot be controlled by the operator.

See also

Device dependency of the objects (Page 4088)

Charge condition (Page 4125)

Effective range name (Page 4184)

WLAN reception (Page 4189)

Zone name (Page 4192)

Zone signal (Page 4193)

Configuring additional Mobile Wireless objects (Page 5974)

WLAN reception

Application

The "WLAN reception" object is a control element for some SIMATIC Mobile Panels. The "WLAN reception" object displays the signal strength of the WLAN wireless connection. In addition, the signal strength is measured and displayed by means of 5 bars.

Layout

Symbol	Meaning	Signal strength
	No wireless connection	No signal
	Very poor wireless connection	=<20 %
	Poor wireless connection	>20 % =<40 %
	Wireless connection OK	>40 % =<60 %
	Good wireless connection	>60 % =<80 %
	Very good wireless connection	>80 %

Operation

The object is for display only and cannot be controlled by the operator.

See also

Device dependency of the objects (Page 4088)

Charge condition (Page 4125)

Effective range name (Page 4184)

Effective range name (RFID) (Page 4186)

Effective range signal (Page 4187)

Zone name (Page 4192)

Zone signal (Page 4193)

Configuring additional Mobile Wireless objects (Page 5974)

Gauge

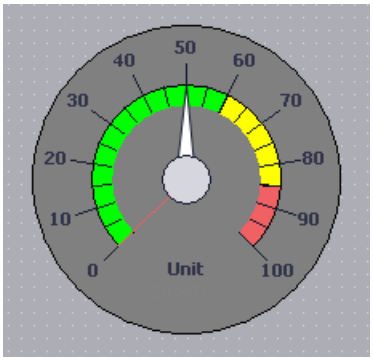
Application

The "Gauge" object shows numeric values in the form of an analog gauge. For example, a glance in Runtime is enough to note that the boiler pressure is in the normal range.

Note

The gauge is for display only and cannot be controlled by the operator.

Configure system functions that are performed at value changes in the tag properties at the "Value change" event and not at the "Change" event of the gauge. If your project is to comply with Good Manufacturing Practices, and the system function performs a GMP-relevant action. e.g. increasing a GMP-relevant tag, every value change of the tag used at the gauge is interpreted as a user action.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Display peak value pointer: Specifies whether the actual measurement range is indicated with a slave pointer.
- Maximum Value and Minimum Value: Specifies the top and bottom values of the scale.
- Start value of the danger range and start value of the warning range: Specifies the scale value from which the danger range and the warning range start.
- Display normal range: Specifies whether the normal range is shown in color on the scale.
- Color of individual ranges: Different operating modes, such as normal range, warning range and danger range, are shown in different colors so that the operator can distinguish them easily.

Display peak value

The "Display peak value" property can be used to activate a marker function for the maximum and minimum pointer movement in Runtime. The actual measurement range is shown with a slave pointer.

1. In the Inspector window, activate "Properties > Properties > Appearance > Show peak values".

Maximum Value and Minimum Value

You can set the top and bottom end values of the scale in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Enter a number under "Scale > Maximum value" and at "Minimum value".
   If you select a tag as the end value of the scale, the number will be no longer available.

Note

Ensure that the minimum value is always less than the maximum value, especially if you make the minimum value or maximum value dynamic through a tag. If the minimum value is greater than the maximum value, no adjustment is made to the scale end values.

Start value of the danger range and start value of the warning range

Define the scale value at which the danger range and the warning range start in the Inspector window.

1. Click "Properties > Properties > Areas" in the Inspector window.
2. Input the start value at "start value of danger range" and "start value of warning range".
3. Select "start value of danger range" and "start value of warning range" to display the ranges on the scale.

Normal range visible

In the Inspector window, define whether to show the normal range in color.

1. Click "Properties > Properties > Areas" in the Inspector window.
2. Select "Normal".

Color of individual ranges

You can display the normal range, the danger range and the warning range in different colors. Define the color in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Areas".
2. Select a color from the color palette for the three areas.

Note

The following restrictions apply for HMI devices with Windows CE up to version 5.0:

- The three ranges (normal / warning / danger) are not visualized in different colors in Runtime.

See also

Device dependency of the objects (Page 4088)

Zone name

Application

The "Zone name" object is a control element for some SIMATIC Mobile Panels. The "Zone name" object shows the names of the zone in which the HMI device is located.

Layout

Symbol	Meaning	Description
	The "On entry" and "On exit" events of the zone and the zone ID can be used in the configuration.	Within the "MixingPlant" zone
	–	Outside of each zone

Procedure for "On entry" event

1. In the "Zones" or "Zones and effective ranges" editor, configure the function "ActivateScreen" on the "On entry" event.

Procedure with zone ID

1. Select an internal tag as the "Zone ID / Connection point ID" in the "Runtime settings > General" editor.
2. Animate the visibility of an object, e.g. "I/O field" via the internal tag.

Operation

The object is for display only and cannot be operated.

See also

Device dependency of the objects (Page 4088)

Charge condition (Page 4125)

Effective range name (Page 4184)

Effective range signal (Page 4187)

WLAN reception (Page 4189)

Zone signal (Page 4193)

Zone signal

Application

The "Zone signal" object is a control element for some SIMATIC Mobile Panels. The "Zone signal" object indicates how well the Mobile Panel is still in a zone. Compared to the "WLAN reception" object, it is not the signal strength that is measured, it is calculated from the distance.

Layout

Symbol	Meaning	Quality
	Within one zone	>=15 %
	When leaving a zone When entering a zone	<15 % and >0 %
	Outside of each zone	=0 %

The "Zone name" object indicates which zone this concerns.

Calculating the signal quality

The quality of the signal within a zone depends, as follows, on the measured distance:

- In the centre of the zone the quality is 100%.
- To the rear towards the transponder and to the front away from the transponder, the quality starts to diminish linearly.
- On the transponder and on the border of the zone, the quality is 0%.

Operation

The object is for display only and cannot be operated.

See also

Device dependency of the objects (Page 4088)

Charge condition (Page 4125)

Effective range name (Page 4184)

Effective range signal (Page 4187)

WLAN reception (Page 4189)

Zone name (Page 4192)

12.1.9 Configuring screen navigation

12.1.9.1 Basics on screen navigation

Types of navigation for the screen change

For a production process consisting of multiple subprocesses, you will configure multiple screens. You have the following options to enable the operator to switch from one screen to the next in Runtime:

- Assign buttons to screen changes
- Configuring screen changes at local function keys

Procedure

Before you create a screen change, define the plant structure and derive from it the screen changes that you want to configure.

Create the start screen under "Runtime Settings > General > Start screen".

See also

Screen basics (Page 3889)

Assigning screen change to function key (Page 4196)

Assigning screen change to button (Page 4195)

12.1.9.2 Assigning screen change to button

Introduction

Configure a button in the screen to switch between the screens on the HMI device during operation.

Note

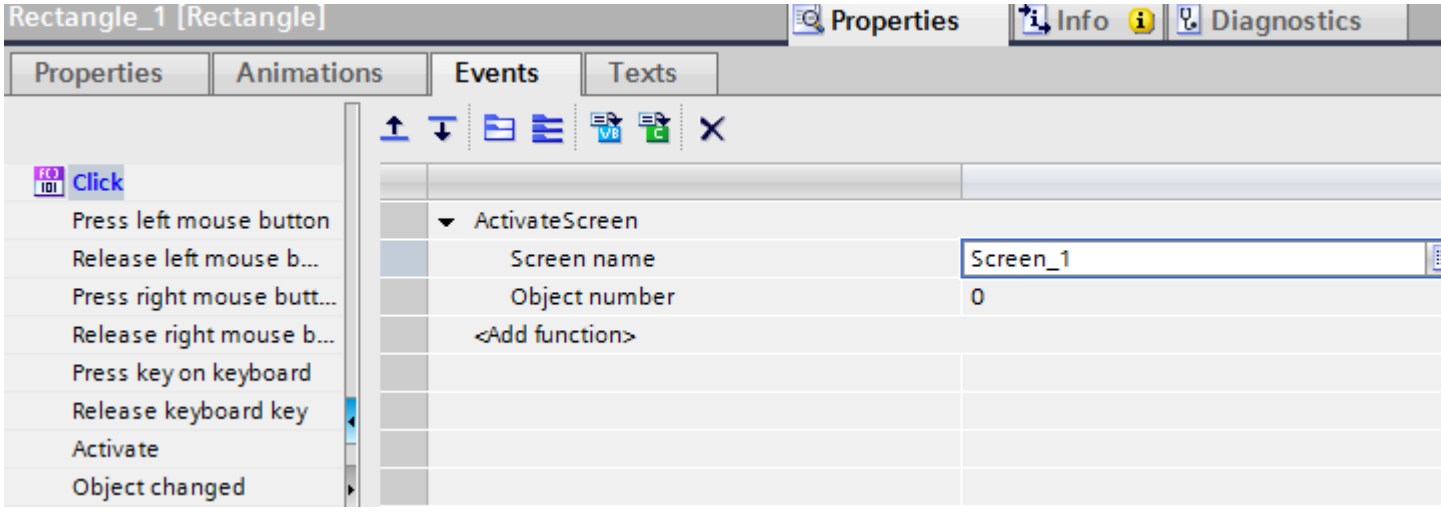
If you have set the "Visibility" of animations to "Hidden" in the Inspector window of a screen, this screen cannot be called up in Runtime.

Requirements

- You have created the project.
- You have created the "Screen_2" screen.
- "Screen_1" is created.

Procedure

1. Double-click "Screen_1" in the project navigation. The screen is displayed in the work area.
2. Move "Screen_2" from the project tree to the open screen by drag&drop.
   A button with the name "Screen_1" is inserted.
3. In the Inspector window, select "Properties > Events > Click".
   The "ActivateScreen" system function is displayed in the "Function list".



4. At the "Object number" attribute, define, if required, the tab sequence number of the object on which the focus is to be set after a screen change. You can also specify a tag that contains the object number.

Alternative procedure

1. Move a button from the "Tools" task card to "Screen2" by drag&drop.
2. In the Inspector window, select "Properties > Events > Click".
3. Select the "ActivateScreen" system function.
4. Select "Screen_2" for the "Screen number".

Result

The operator goes to "Screen_1 with the button in Runtime. If you have specified an object number, the object with this object number has the focus following a screen change.

See also

Basics on screen navigation (Page 4194)

12.1.9.3 Assigning screen change to function key

Introduction

Configure a screen change function key in the screen to switch between the screens on the HMI device during operation.

Note

If you have set the "Visibility" of animations to "Hidden" in the inspector window of a screen, this screen cannot be called up in Runtime.

Requirements

- You have created a project.
- You have created the "Screen_2" screen.
- You have created the "Screen_1" screen.

Procedure

1. Double click "Screen_1" in the project tree. The screen is displayed in the work area.
2. Move "Screen_2" from the project tree to a function key, e.g. "F2".
   The configured function key displays a yellow triangle.
3. Click "Properties > Events > Press key" in the inspector window.
   The "ActivateScreen" system function is displayed.

Result

The operator goes to the specified "Screen_2" with function key "F2" in Runtime.

See also

Basics on screen navigation (Page 4194)

12.2 Working with Tags

12.2.1 Basics

12.2.1.1 Basics of tags

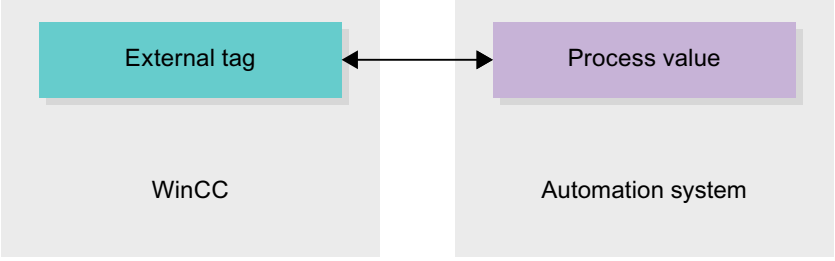
Introduction

Process values are forwarded in runtime using tags. Process values are data which is stored in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. Define external tags for processing the process values in WinCC.

WinCC works with two types of tag:

- External tags
- Internal tags

The external tags form the link between WinCC and the automation systems. The values of external tags correspond to the process values from the memory of an automation system. The value of an external tag is determined by reading the process value from the memory of the automation system. It is also possible to rewrite a process value in the memory of the automation system.



Internal tags do not have a process link and only convey values within the WinCC. The tag values are only available as long as runtime is running.

Tags in WinCC

For external tags, the properties of the tag are used to define the connection that the WinCC uses to communicate with the automation system and form of data exchange.

Tags that are not supplied with values by the process - the internal tags - are not connected to the automation system. In the tag's "Connection" property, this is identified by the "Internal tag" entry.

You can create tags in different tag tables for greater clarity. You then directly access the individual tag tables in the "HMI tags" node in the project tree. The tags from all tag tables can be displayed with the help of the table "Show all tags".

With structures you bundle a number of different tags that form one logical unit. Structures are project-associated data and are available for all HMI devices of the project. You use the "Types" editor in the project library to create and edit a structure.

See also

Creating external tags (Page 4207)

Manage tags (Page 4211)

Tag Limits (Page 4218)

Indirect addressing of tags (Page 4232)

Internal Tags (Page 4205)

Basics on arrays (Page 4236)

Basics on user data types (Page 4239)

Cycle basics (Page 4247)

Trends (Page 4270)

Overview of HMI tag tables (Page 4198)

Addressing external tags (Page 4202)

Basic principles for data logging (Page 4249)

External tags (Page 4200)

Outputting tag values in screens (Basic) (Page 4268)

12.2.1.2 Overview of HMI tag tables

Introduction

HMI tag tables contain the definitions of the HMI tags that apply across all devices. A tag table is created automatically for each HMI device created in the project.

In the project tree there is an "HMI tags" folder for each HMI device. The following tables can be contained in this folder:

- Default tag table
- User-defined tag tables
- Table of all tags

In the project tree you can create additional tag tables in the HMI tags folder and use these to sort and group tags and constants. You can move tags to a different tag table using a drag-and-drop operation or with the help of the "Tag table" field. Activate the "Tags table" field using the shortcut menu of the column headings.

Default tag table

There is one default tag table for each HMI device of the project. It cannot be deleted or moved. The default tag table contains HMI tags and, depending on the HMI device, also system tags. You can declare all HMI tags in the standard tags table or, as necessary, additional user-defined tables of tags.

User-defined tag tables

You can create multiple user-defined tag tables for each HMI device in order to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. To group tag tables, create additional subfolders in the HMI tags folder.

All tags

The "All tags" table shows an overview of all HMI tags and system tags of the HMI device in question. This table cannot be deleted, renamed or moved. This table also contains the "Tags table" column, which indicates the tag table of where a tag is included. Using the "Tags table" field, the assignment of a tag to a tags table can be changed.

With devices for Runtime Professional, the table "All tags" contain an additional tab "System tags". The system tags are created by the system and used for internal management of the project. The names of the system tags begin with the "@" character. System tags cannot be deleted or renamed. You can evaluate the value of a system tag, but cannot modify it.

Additional tables

The following tables are also available in an HMI tag table:

- Discrete alarms
- Analog alarms
- Logging tags

With the help of these tables you configure alarms and logging tags for the currently selected HMI tag.

Discrete alarms table

In the "Discrete alarms" table, you configure discrete alarms to the HMI tag selected in the HMI tag table. When you configure a discrete alarm, multiple selection in the HMI tag table is not possible. You configure the discrete alarms for each HMI tag separately.

Analog alarms table

In the "Analog alarms" table, you configure analog alarms to the HMI tag selected in the HMI tag table. When you configure an analog alarm, multiple selection in the HMI tag table is not possible. You configure the analog alarms for each HMI tag separately.

Logging tags table

In the "Logging tags" table, you configure logging tags to the HMI tag selected in the HMI tag table. When you configure a logging tag, multiple selection in the HMI tag table is not possible. You configure the logging tags for each HMI tag separately. The "Logging tags" table is only available if the HMI device used supports logging.
If WinCC Runtime Professional is used, you can also assign several log tags to a tag. With the other HMI devices, you can only assign one log tag to a tag.

See also

Basics of tags (Page 4197)

12.2.1.3 External tags

Introduction

External tags allow communication (data exchange) between the components of an automation system, for example, between an HMI device and a PLC.

Principle

An external tag is the image of a defined memory location in the PLC. You have read and write access to this storage location from both the HMI device and from the PLC.

Since external tags are the image of a storage location in the PLC, the applicable data types depend on the PLC which is connected to the HMI device.

If you write a PLC control program in STEP 7, the PLC tags created in the control program will be added to the PLC tag table. If you want to connect an external tag to a PLC tag, access the PLC tags directly via the PLC tag table and connect them to the external tag.

Data types

All the data types which are available at the connected PLC are available at an external tag in WinCC. Information about data types which are available for connection to other PLCs can be found in the documentation about the respective communication drivers.

See "Auto-Hotspot" for additional information.

Note

As well as external tags, area pointers are also available for communication between the HMI device and PLC. You can set up and enable the area indicators in the "Connections" editor.

Central tag management in STEP 7

You can connect also connect DB instances of user-defined PLC data types (UDT) to the HMI tags.

The PLC data type and the corresponding DB instances are created and updated centrally in STEP 7. In WinCC, you can use the following sources as PLC tag (DB instances):

- Data block elements that use a UDT as data type
- Data block instances of a UDT

The data type is taken from STEP 7 and is not converted to an HMI data type. The access mode is always "Symbolic access". Depending on the release for WinCC in STEP 7, elements and structured elements of the PLC data type are applied to WinCC. Elements of a structured UDT are applied and displayed in the PLC tag table if the instance-specific properties "Visible in HMI" and "Accessible from HMI" have been set.

Note

Accessing PLC data types

Access to PLC data types is available only in conjunction with SIMATIC S7-1200 or S7-1500.

Synchronization with PLC tags

A variety of options for synchronizing external tags with the PLC tags are available in the Runtime settings under "Settings for tags".

When you perform synchronization, you have the option of automatically applying the tag names of the PLC to external tags and reconnecting the existing tags.

The generated tag name is derived from the position of the data value in the hierarchical structure of the data block.

Update of tag values

For external tags, the current tag values are transmitted in Runtime via the communication connection between WinCC and the connected automation systems and then saved in the Runtime memory. Next, the tag value will be updated to the set cycle time. For use in the Runtime project, WinCC accesses tag values in the Runtime memory that were read from the PLC at the previous scan cycle checkpoint. As a result, the value in the PLC can already change whilst the value from the Runtime memory is being processed.

Note

PLC array elements in conjunction with S7-1200 or S7-1500

The index of the PLC array elements can begin with any number. In WinCC, indexing always starts with 0.

A PLC tag "Array [1..3] of Int", for example, is mapped to "Array [0..2] of Int" in WinCC.

When you access an array in a script, pay attention to the correct indexing.

See also

Basics of tags (Page 4197)

12.2.1.4 Addressing external tags

Introduction

The options for addressing external tags depend on the type of connection between WinCC and the PLC in question. A distinction must be made between the following connection types:

- Integrated connection
  Connections of devices which are within a project and were created with the "Devices & Networks" editor are referred to as integrated connections.
- Non-integrated connection
  Connections of devices which were created with the "Connections" editor are referred to as non-integrated connections. It is not necessary that all of the devices be within a single project.

The connection type can also be recognized by its icon.

	Integrated connection
	Non-integrated connection

You can find additional information in the section "Basics of communication (Page 5996)".

Addressing with integrated connections

An integrated connection offers the advantage that you can address a tag both symbolically and absolutely.

For symbolic addressing, you select the PLC tag via its name and connect it to the HMI tag. The valid data type for the HMI tag is automatically selected by the system. You have to distinguish between the following cases when you address elements in data blocks:

Symbolic addressing of data blocks with optimized access and standard access:

During the symbolic addressing of a data block with optimized access and standard access, the address of an element in the data block is dynamically assigned and is automatically adopted in the HMI tag in the event of a change. You do not need to compile the connected data block or the WinCC project for this step.
For data blocks with optimized access, only symbolic addressing is available.

For symbolic addressing of elements in a data block, you only need to recompile and reload the WinCC project in case of the following changes:

- If the name or the data type of the linked data block element or global PLC tag has changed.
- If the name or the data type of the higher level structure node of a linked element in the data block element or global PLC tag has changed.
- If the name of the connected data block has changed.

Symbolic addressing is currently available with the following PLCs:

- SIMATIC S7 1200
- SIMATIC S7 1500

Symbolic addressing is also available if you have an integrated link.

You can also use absolute addressing with an integrated connection. You have to use absolute addressing for PLC tags from a SIMATIC S7 300/400 PLC. If you have connected an HMI tag with a PLC tag and the address of the PLC tag changes, you only have to recompile the control program to update the new address in WinCC. Then you recompile the WinCC project and load it onto the HMI device.

In WinCC, symbolic addressing is the default method. To change the default setting, select the menu command "Options > Settings". Select "Visualization > Tags" in the "Settings" dialog. If required, disable the "Symbolic access" option.

The availability of an integrated connection depends on the PLC used. The following table shows the availability:

PLC	Integrated connection	Comments
S7 300/400	Yes	The linking of tags is not checked in Runtime. If the tag address changes in the PLC and the HMI device is not compiled again and loaded, the change is not registered in runtime.
S7 1200	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the following behavior applies to the S7 300/400.
S7-1500	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the following behavior applies to the S7 300/400.

Create an integrated connection in the "Devices & Networks" editor. If the PLC is contained in the project and integrated connections are supported, you can then also have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.

Addressing with non-integrated connections

In the case of a project with a non-integrated connection, you always configure a tag connection with absolute addressing. Select the valid data type yourself. If the address of a PLC tag changes in a project with a non-integrated connection during the course of the project, you also have to make the change in WinCC. The tag connection cannot be checked for validity in Runtime, an error message is not issued.

A non-integrated connection is available for all supported PLCs.

Symbolic addressing is not available in a non-integrated connection.

With a non-integrated connection, the control program does not need to be part of the WinCC project. You can perform the configuration of the PLC and the WinCC project independently of each other. For configuration in WinCC, only the addresses used in the PLC and their function have to be known.

See also

Basics of communication (Page 5996)

Basics of tags (Page 4197)

12.2.1.5 Internal Tags

Introduction

Internal tags do not have any connection to the PLC. Internal tags transport values within the HMI device. The tag values are only available as long as runtime is running.

Principle

Internal tags are stored in the memory of the HMI device. Therefore, only this HMI device has read and write access to the internal tags. You can create internal tags to perform local calculations, for example.

You can use the HMI data types for internal tags. Availability depends on the HMI device being used.

The following HMI data types are available:

HMI data type	Data format
Array	One-dimensional array
Bool	Binary tag
DateTime	Date/time format
DInt	Signed 32-bit value
Int	Signed 16-bit value
LReal	Floating-point number 64-bit IEEE 754
Real	Floating-point number 32-bit IEEE 754
SInt	Signed 8-bit value
UDnt	Unsigned 32-bit value
UInt	Unsigned 16-bit value
USInt	Unsigned 8-bit value
WString	Text tag, 16-bit character set

See also

Basics of tags (Page 4197)

12.2.1.6 User-defined PLC data types (UDT)

Overview

You can connect with the HMI tags and DB instances of user-defined PLC data types (UDT).

The PLC data type and the corresponding DB instances are created and updated centrally in STEP 7. In WinCC, you can use the following sources as PLC tag (DB instances):

- Data block elements that use a UDT as data type
- Instance data blocks of a UDT

The data type is taken from STEP 7 and is not converted into an HMI data type. The access type is always "Symbolic access".

Elements of a PLC data type

You have access to the following elements in WinCC with a structured PLC data type:

- Elements that have been released for WinCC in STEP 7.
- Elements whose data types are supported in WinCC.

Note

Invalid elements of a PLC data type in WinCC

Invalid elements generate an error in WinCC.

If you disable the "Accessible from HMI" option for the corresponding elements of the associated PLC data type in STEP 7, these elements are excluded in WinCC.

Naming conventions

The following characters are invalid in the name of the PLC data type and generate an error in WinCC:

- Dot: "."
- Square brackets: "(" and ")"

Properties

The properties of the PLC data type and its elements are adopted in WinCC. Depending on the data type used, the properties are read-only or can be written to in WinCC.

If you change the connection of the PLC data type in WinCC, all elements of the PLC data type are deleted and the properties of the newly connected PLC tag are used.

In WinCC, you have access to STEP 7 comments on elements of the PLC data type.

You have limited access to properties in WinCC for the following elements of PLC data types:

- Elements of the data type "Struct"
- PLC data type
- Multidimensional arrays
- Array of complex data types except "DTL"

Mapping of the data type "DTL"

If a PLC data type contains elements of the data type "DTL", these elements are mapped in WinCC without lower-level elements. The data type "DTL" turns into "DateTime" in WinCC.

12.2.2 Working with Tags

12.2.2.1 Creating tags

Creating external tags

Introduction

You can access an address in the PLC via a PLC tag using an external tag. The following options are available for addressing:

- Symbolic addressing
- Absolute Addressing

You can find further details on symbolic addressing in section "Addressing external tags (Page 4202)". If possible, use symbolic addressing when configuring a tag. You create tags either in the standard tag table or in a tag table you defined yourself.

Requirement

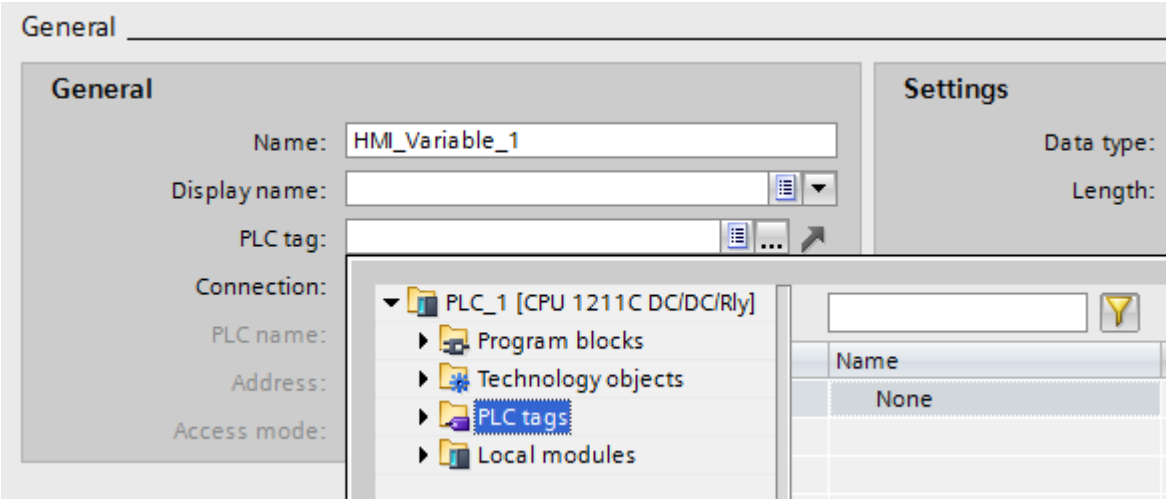
- You have opened the project.
- A connection to the PLC is configured.
- The Inspector window is open.

Procedure

To create an external tag, proceed as follows:

1. Open the "HMI tags" folder in the project tree and double-click the standard tag table. The tag table open.
   Alternatively, create a new tag table and then open it.
2. In the "Name" column, double-click "Add" in the tag table. A new tag is created.
3. Select the "Properties > Properties >General" category in the Inspector window and, if required, enter a unique tag name in the "Name" field. The tag name must be unique throughout the device.
4. If required, select the "Display name" field to enter a name to be displayed in Runtime. The name to be displayed is language-specific and can be translated for the required Runtime languages. The display name is available for Basic Panels, Panels and Runtime Advanced.
5. Select the connection to the required PLC in the "Connection" box. If the connection you require is not displayed, you must first create the connection to the PLC. You create the connection to a SIMATIC S7 PLC in the "Devices & Networks" editor. You create the connection to external PLCs in the "Connections" editor.
   If the project contains the PLC and supports integrated connections, you can also have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.

6. If you are working with an integrated connection, click the ... button in the "PLC tag" field and select an already created PLC tag in the object list. Click the ✓ button to confirm the selection.



7. If you are working with a non-integrated connection, enter the address from the PLC in the "Address" field. The "PLC tag" field remains empty.
8. Configure the other properties of the tag in the inspector window.

You can also configure all tag properties directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

You can also create tags alternatively directly at the application point, e.g. on an I/O field. To do this, click the Add new button in the object list. You then configure the new tag in the Inspector window.

Result

An external tag has been created and linked to a PLC tag or an address in the PLC.

Alternative procedure

You can also create external HMI tags by dragging and dropping data block elements or global PLC tags in an HMI tag table.

See also

Creating internal tags (Page 4209)

Creating multiple tags (Page 4210)

Basics of tags (Page 4197)

Addressing external tags (Page 4202)

Manage tags (Page 4211)

Tag Limits (Page 4218)

Indirect addressing of tags (Page 4232)

Creating internal tags

Introduction

You must at least set the name and data type for internal tags. Select the "Internal tag" item, rather than a connection to a PLC.

For documentation purposes, it is a good idea to enter a comment for every tag.

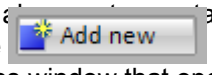
Requirement

You have opened the project.

Procedure

1. Open the "HMI tags" folder in the project tree and double-click the entry "Standard tag table". The tag table opens.
   Alternatively, create and then open a new tag table.
2. Double-click "Add" in the "Name" column of the tag table. A new tag is created.
3. If the Inspector window is not open, select the "Inspector window" option in the "View" menu.
4. Select the "Properties > Properties >General" category in the Inspector window and, if required, enter a unique tag name in the "Name" field. This tag name must be unique throughout the project.
5. If required, select the "Display name" field to enter a name to be displayed in Runtime. The name to be displayed is language-specific and can be translated for the required Runtime languages. The display name is available for Basic Panels, Panels and Runtime Advanced.
6. Select "Internal tag" as the connection in the "Connection" field.
7. Select the required data type in the "Data type" field.
8. In the "Length" field, you must specify the maximum number of characters to be stored in the tag according to the selected data type. The length is automatically defined by the data type for numerical tags.
9. As an option, you can enter a comment regarding the use of the tag. To do so, click "Properties > Properties > Comment" in the Inspector window and enter a text.

You can also configure all tag properties directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

You can a ags alternatively directly at the application point, e.g. on an I/O field. Click the 

button in the object list. You can then configure the new tag in the Properties window that opens.

Result

An internal tag is created. You can now use this in your project.

In additional steps you can configure the tag, for example, by setting the start value and limits.

See also

Creating external tags (Page 4207)

Creating multiple tags

Introduction

In a tag table, you create additional identical tags by automatically filling the rows of the table below a tag.

The tag names are incremented automatically when filling in automatically.

You can also use the auto fill function to fill table cells below a tag with a single tag property and thus modify the corresponding tags.

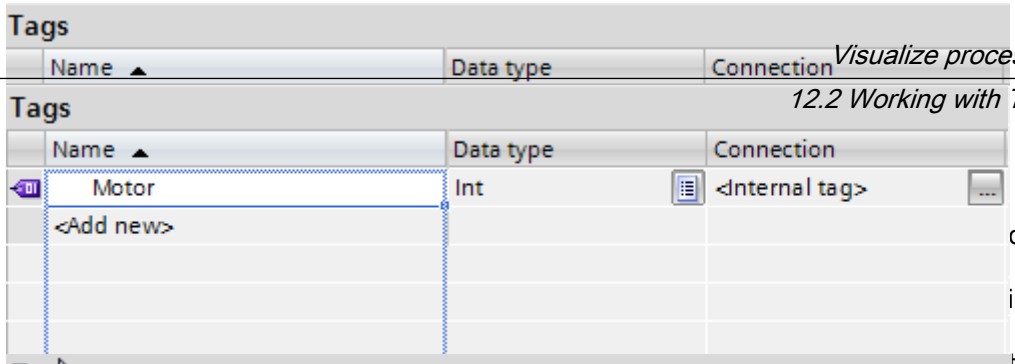
If you apply the automatic filling in to already filled cells of a tab table, you will be asked whether you want to overwrite the cells or insert new tags.

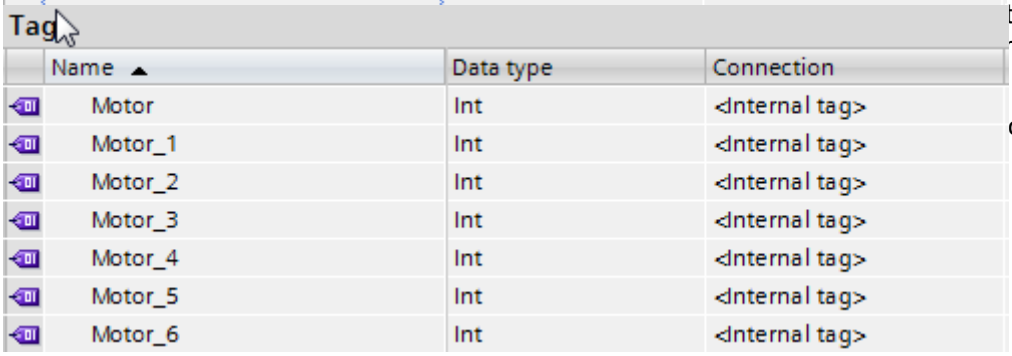
If you do not want to overwrite already configured tags, activate insert mode. Activate insert mode by keeping the <Ctrl> key pressed during insertion. Already existing entries in the tag table are moved down if insert mode is activated.

Requirement

- You have opened the project.
- A tag table is open.
- The tag which is to serve as a template for other tags is configured.

Procedure

1. 

d as
ins
ttom
ross.
2. o fill
3. 

Result

Depending on which cells were selected, the function may automatically fill individual properties or create new tags.

See also

Creating external tags (Page 4207)

12.2.2.2 Editing tags

Manage tags

Introduction

You can always rename, copy or delete tags.

When a tag is renamed, the new name must be unique for the whole device.

If you use the "Copy" command to copy a tag to the clipboard, the objects and references linked to the tag are copied as well.

If you use the "Insert" command to add a tag to another device, the tag will be added without the connected references. Only the object name of the reference will be inserted. If a reference of the same name and valid properties exists in the target system, the existing reference will then be connected to the copied tag.

If you copy a tag, some of the objects linked to the tag are copied as well. The following objects are copied:

- Logging tags
- Cycles
- Alarms

If you add the copied tag to another device, the tag is added together with the linked objects.

Requirement

- The tag which you wish to rename, copy or delete must exist.
- The tag table containing the tag is open.

Renaming tags

1. In the "Name" field, select the tag in the tag table.
2. Select "Rename" from the shortcut menu.
3. Type in a new name.
   The tag appears under its new name.

Copying tags

1. Mark one or more tags in the tags table.
2. Select "Copy" from the shortcut menu.
3. Click on the point at which you want to insert the tag. For example, click another tag table in the same device or the tag table in a second device.
4. Select "Paste" from the shortcut menu. The tag is inserted as described above.

Deleting a tag

1. Select one or more tags in the tag table.
2. Select the "Cross-reference" command from the "Tools" menu. In the "Cross-reference" editor, check to see where the tags are used. In this manner, you can see what impact the deletion of the tag will have on your project.
3. Select "Delete" in the pop-up menu of the tag.
   All marked tags will be deleted.

Export and import of tags

WinCC gives you the option to export and import tags. With Export and Import, you have the option to export tags from one project and import them into another project. There is also the option to create larger numbers of tags outside of WinCC, edit them and subsequently import into any WinCC project. For further details, refer to Auto-Hotspot.

See also

Edit tags (Page 4213)

Configuring multiple tags simultaneously (Page 4213)

Using multiple tags simultaneously in a screen (Page 4214)

Basics of tags (Page 4197)

Synchronizing tags (Page 4216)

Creating external tags (Page 4207)

Edit tags

Introduction

You can modify tags at any time to adapt them to changed requirements in the project.

Edit tags

You have the following options to change the configuration of a tag:

- You open the tag table in which the tag is contained.
- Open the tag table "Show all tags".
- You open the Inspector window of a tag with the "Edit" button in the object selection on the display and operating object.

In the tag tables, you can perform such tasks as comparing and adjusting the properties of multiple tags or sorting the tags by their properties.

You can change the properties either directly in the table or in the Inspector window.

If you change a tag property and this change causes a conflict with another property, it will be highlighted in color to draw your attention to this fact. If you link the tag with another controller, for example, which does not support the set data type, this property is highlighted in color.

See also

Manage tags (Page 4211)

Configuring multiple tags simultaneously

Introduction

In WinCC, you can assign the same properties to multiple tags in a single operation. This facilitates efficient programming.

Requirement

- You created the tags you want to configure.
- The tag table is open.
- The Inspector window is open.

Procedure

1. In the tag table, select all the tags that you want to configure at the same time.
   If the selected property is identical for all the tags, the setting for this property will appear in the Inspector window. The associated field will remain blank otherwise.
2. You can define the shared properties in the Inspector window or directly in the tag table.

if you change a property commonly on several tags, only this one property is changed. The other properties of the tag remain unchanged.

Result

All marked tags will be reconfigured.

To edit tag properties which differ from one tag to the other, simply clear the multiple selection.

See also

Manage tags (Page 4211)

Using multiple tags simultaneously in a screen

Introduction

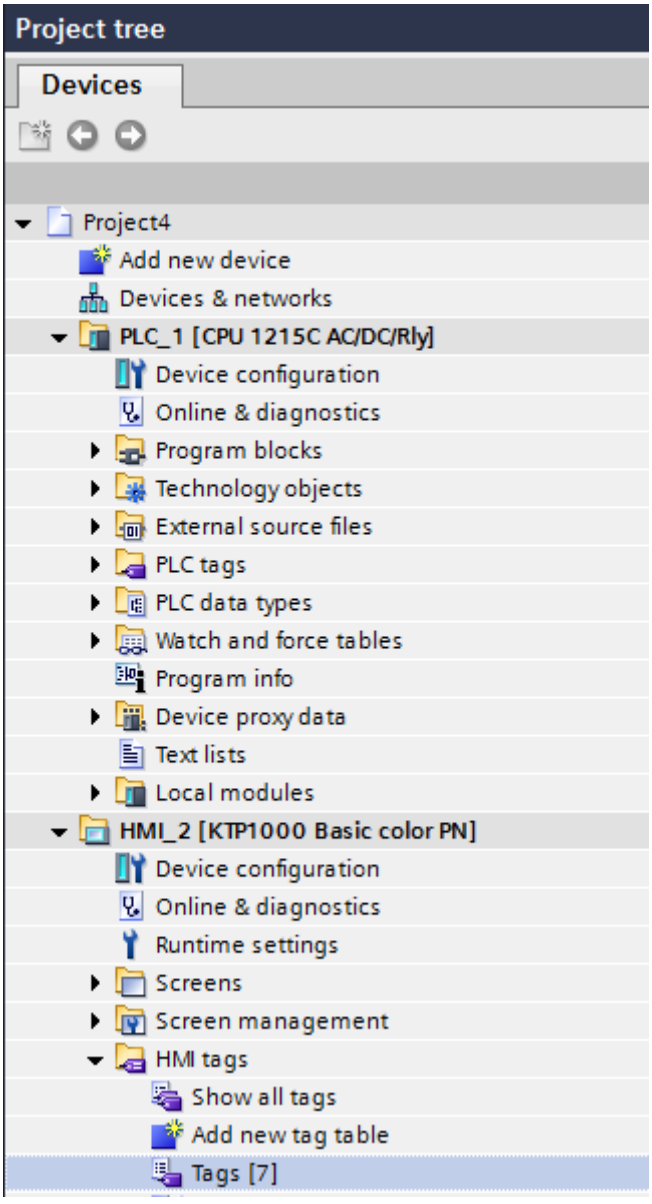
In WinCC, you can create multiple I/O fields that are linked with tags in one screen in a single operation. This facilitates efficient programming.

Requirement

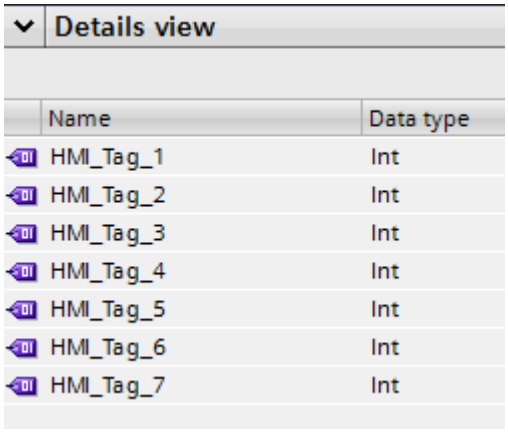
- Several tags are set up.
- A screen is open.

Procedure

1. In the project tree, select the required tag table under "HMI tags".



2. Select the detail view at the bottom of the project tree. The detail view shows the tags that exist in the selected tag group.



3. Mark the tags in the detail window.
4. Drag the tags to the screen. For each tag, this creates an I/O field that is connected to the tag.

Note

When you move a PLC tag from the detail window to the work area by drag&drop, a network and a connection are created additionally in the "Devices & Networks" editor.

See also

Manage tags (Page 4211)

Synchronizing tags

Introduction

To synchronize the PLC and HMI tags, WinCC offers the following options:

- Synchronizing tags with or without name matching between PLC and WinCC
  The following options are available for this:
- Link tags with addresses in the PLC
  This procedure is suitable, for example, if changes were made to the connection between the HMI device and the PLC and the tag connections were lost. The function can also be used if you have configured the control program and HMI project separately.

Requirement

- External HMI tags have been created.
- PLC tags have been created.
- An HMI connection to a PLC in the project has been established.

Procedure

To synchronize HMI tags with PLC tags, follow these steps:

1. In the project tree, select the directory that contains the tags in question.
2. Select "Connect HMI tags to matching PLC tags" from the shortcut menu.
   A dialog opens.

3. Select the option you want to use.
   If you want to synchronize the tags without name matching, disable "Replace WinCC tag name with PLC tag name".
   If you want to connect HMI tags new with absolute access, select "Data type and absolute address match".
4. Confirm with "Synchronize".
   The system searches for a suitable PLC tag according to the selected option.

Result

The external HMI tags are synchronized with the PLC tags.

If you have selected the option "Data type and absolute address match", the tag connection is established as soon as a suitable PLC tag is found.

If you have selected a different option, the WinCC tags are updated accordingly and the PLC tag names are applied in WinCC.

See also

Manage tags (Page 4211)

12.2.2.3 Configuring Tags

Tag Configuration Basics

Tag Limits

Introduction

You can restrict the value range with limits for numerical tags.

Principle

You can specify a value range defined by a high limit and a low limit for numerical tags.

If the operator enters a value for the tag that is outside the configured value range, the input is rejected. The value is not accepted. You configure a function list when configuring for Runtime Advanced and Panels. When the tag value leaves the value range, the function list is processed.

Note

If you want to output an analog alarm when a limit is violated, configure the respective tag in the "Analog alarms" tab. You can also configure the analog alarm in the "HMI alarms" editor. The values for output of an analog alarm depend on the configured tag limits.

Application example

Use the limits to warn the operator when the value of a tag enters a critical range, for example.

See also

Defining Limits for a Tag (Page 4219)

Start value of a tag (Page 4220)

Defining the start value of a tag (Page 4220)

Updating the Tag Value in Runtime (Page 4221)

Linear scaling of a tag (Page 4222)

Applying linear scaling to a tag (Page 4224)

Connecting a tag to another PLC (Page 4224)

Basics of tags (Page 4197)

Address multiplexing (Page 4225)

Configuring address multiplexing with absolute addressing (Page 4226)

Configuring address multiplexing with symbolic addressing (Page 4229)

Creating external tags (Page 4207)

Defining Limits for a Tag

Introduction

For numerical tags, you can specify a value range by defining a low and high limit.

For Runtime Advanced and Panels you can configure the system to process a function list whenever a tag value drops below or exceeds its configured value range.

Requirement

- The tag for which you want to set the limits is created.
- The Inspector window with the properties for this tag is open.

Procedure

To define limits for a tag, proceed as follows:

1. In the Inspector window select "Properties > Properties > Limits." If you want to define one of the limits as a constant value, select "Constant" using the button. Enter a number in the relevant field.
   If you want to define one of the limits as a tag value, select "HMI tag" using the button. Use the object list to define the tag for the limit.
2. To set additional limits for the tag, repeat step 1 with the appropriate settings.

Alternative procedure

You can also configure the high and low limit directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Configuring a function list for Runtime Advanced and Panels

When you configure Runtime Advanced and Panels, you can have a function list called up if a high or low limit is violated.

1. If you want to start a function list when the value drops below the value range, click "Properties > Events > Minimum violated" in the Inspector window. Create a function list in this dialog.
2. If you want to start a function list when the value exceeds the value range, click "Properties > Events > Maximum violated" in the Inspector window. Create a function list in this dialog.

Result

You have set a value range defined by a high and low limit for the selected tag. If the value range is exceeded or undershot, a function list may be carried out.

See also

Tag Limits (Page 4218)

Start value of a tag

Value of a tag at start of Runtime

You can configure a start value for numeric tags and tags for date/time values. The tag will be preset to this value when Runtime starts. In this way, you can ensure that the tag has a defined status when Runtime starts.

For external tags, the start value will be displayed on the HMI device until it is overwritten by the PLC or by input.

If no start value was configured, the tag contains the value "0" when starting Runtime.

In WinCC Runtime Professional you can enter a tag value in place of the start value on a tag with the "String" data type. The tag value is saved in the "Project texts" editor and is multilingual. After the text has been translated, it is displayed in Runtime as a language-dependent start value.

Application example

You can assign a default value to an I/O field. Enter the desired default value as start value for the tag that is linked to the I/O field.

See also

Tag Limits (Page 4218)

Defining the start value of a tag

Introduction

In WinCC you can configure a start value for a numeric tag and a tag for date/time values which this adopts at Runtime start.

Requirement

- You have created the tag for which you want to define a start value.
- The Inspector window with the tag properties is open.

Procedure

To configure a start value, proceed as follows:

1. In the Inspector window select "Properties > Properties > Values."
2. Enter the desired "Start value."

Alternative procedure

You can also configure the start value directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Result

The start value you selected for the tag is transferred to the project.

See also

Tag Limits (Page 4218)

Updating the Tag Value in Runtime

Introduction

Tags contain process values which change while Runtime is running. Value changes are handled differently at internal and external tags.

Principle

When Runtime starts, the value of a tag is equal to its start value. Tag values change in Runtime.

In Runtime, you have the following options for changing the value of a tag:

- A value change in an external tag in the PLC.
- By input, for example, in an I/O field.
- By running a system function, such as "SetValue."
- A value assignment in a script.

Updating the Value of External Tags

The value of an external tag is updated as follows:

- Cyclic in operation
  If you select the "Cyclic in operation" acquisition mode, the tag is updated in Runtime while it is displayed in a screen or is logged. The acquisition cycle determines the update cycle for tag value updates on the HMI device. You can either choose a default acquisition cycle or define a user-specific cycle.
- Cyclic continuous
  If you select the "Cyclic continuous" acquisition mode, the tag will be updated continuously in Runtime, even if it is not in the currently-open screen. This setting is activated for tags that are configured to trigger a function list when their value changes, for example.
  Only use the "Cyclic continuous" setting for tags that must truly be updated. Frequent read operations increase communication load.
- On demand
  If you select the "On demand" acquisition mode, the tag is not updated cyclically. It will only be updated on demand using the "UpdateTag" system function, for example, or by a script.

See also

Tag Limits (Page 4218)

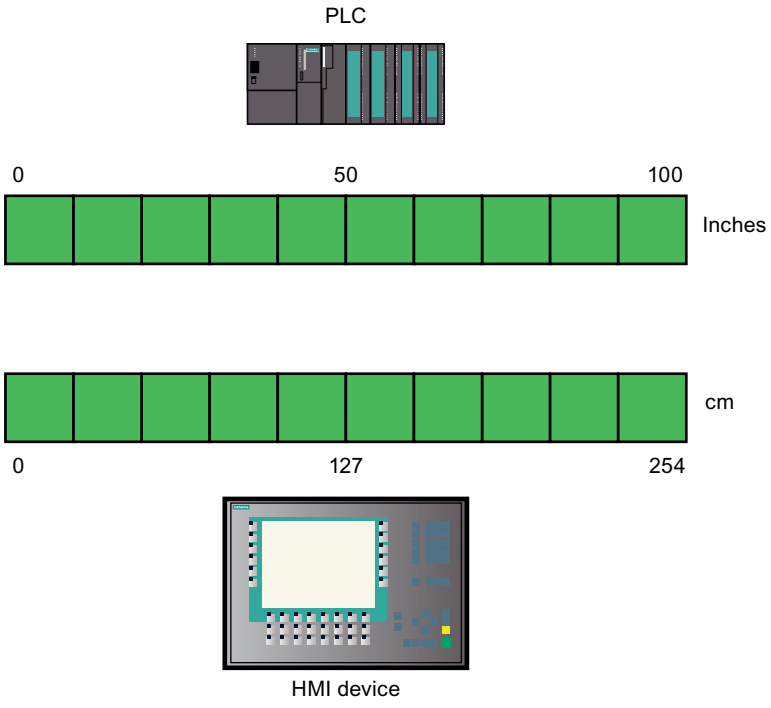
Linear scaling of a tag

Introduction

Numeric data types can be processed with linear scaling. The process values in the PLC for an external tag can be mapped onto a specific value range in the project.

Principle

To apply linear scaling to a tag, you must specify one value range on the HMI device and one on the PLC. The value ranges will be mapped to each other linearly.



As soon as data from the HMI device is written to an external tag, it will be automatically mapped to the value range of the PLC. As soon as data from the HMI device is read from the external tag, a corresponding transformation will be performed in the other direction.

Note

You can also use the system functions "LinearScaling" and "InvertLinearScaling" to automatically convert process values.

Application example

The user enters length dimensions in centimeters but the PLC is expecting inches. The entered values are automatically converted before they are forwarded to the controller. Using linear scaling, the value range [0 to 100] on the PLC can be mapped onto the value range [0 to 254] on the HMI device.

See also

Tag Limits (Page 4218)

Applying linear scaling to a tag

Introduction

To apply linear scaling to a tag, you must specify one value range on the HMI device and one on the PLC. The value ranges will be mapped to each other linearly.

Requirement

- The external tag to which linear scaling is to be applied must exist.
- The Inspector window with the properties for this tag is open.

Procedure

To apply linear scaling to a tag, follow these steps:

1. In the Inspector window select "Properties > Properties > Linear scaling."
2. Click on "Enable" to switch on linear scaling.
   Using this option, you can temporarily switch off linear scaling for testing purposes, for example. Settings which were made earlier for linear scaling remain unchanged.
3. In the "PLC" area, enter the start and end values of the value range to be applied to the process values on the PLC.
4. In the "HMI device" area, enter the end and start values of the value range to be applied to the process values on the HMI device.

Result

In Runtime the data will be automatically mapped from one value range to the other.

Note

You can also use the "LinearScaling" and "InvertLinearScaling" system functions to automatically convert process values.

See also

Tag Limits (Page 4218)

Connecting a tag to another PLC

Introduction

In WinCC, you can change the PLC connection of a tag at any time. This is needed when you change the configuration of your plant, for example.

Depending on the PLC selected, you may need to modify the configuration of the tag. The tag properties which must be changed will be highlighted in color.

Requirement

- The external tag, whose connection you wish to change, must already exist.
- The connection to the PLC must already exist.
- The Properties window for this tag is open.

Procedure

To change the PLC connection of a tag, proceed as follows:

1. In the Inspector window select "Properties > Properties > General."
2. Select the new connection in the "Connection" field.
   The tag properties that you must change will be highlighted in color in the tag table and in the Inspector window.
3. Change all highlighted properties of the tag to suit the requirements of the new PLC.

Result

The external tag is connected to the new PLC.

See also

Tag Limits (Page 4218)

Address multiplexing

Introduction

Using address multiplexing, you can use a single tag to access a multitude of memory locations within the PLC's address range. You read and write to the addresses without defining a tag for each individual address.

Multiplexing with absolute addressing

When using multiplexing with absolute addressing, you configure tags as placeholders for the address in the PLC to be addressed.

If you want to access, for example, an address of the format "%DBx.DBWy", the expression for multiplexing is as follows:

"%DB[HMITag1].DBW[HMITag2]"

In Runtime, you supply the tag "HMITag1" with the required value for the data block you want to address.

In Runtime, you supply the tag "HMITag2" with the required address from the data block.

Tags are supplied with values, for example, with the help of values from the PLC or via a script.

Multiplexing with absolute addresses is supported for the following PLCs and communication drivers.

- SIMATIC S7-300/400
- SIMATIC S7-1200
- SIMATIC S7-1500

Multiplexing with absolute addresses is not available for data blocks with optimized access.

Multiplexing with symbolic addressing

When multiplexing with symbolic addressing, you access an array element of an array tag in a data block of the connected PLC by means of a multiplex tag and an index tag. The multiplex tag contains the symbolic address of the data block which you want to access. The symbolic address also contains the index tag via which you access the index of the array tag.

If you want to access, for example, the array tag "Arraytag_1" in the data block "Datablock_1", the expression for symbolic addressing is as follows:

"Datablock_1.Arraytag_1["HMITag_1"]

You control the access to the index of the array elements with the HMI-Variable "HMITag_1". In Runtime, you supply the tag with the index of the array element that you want to access.

Multiplexing with symbolic addressing is only available if the following components support symbolic addressing:

- the HMI device
- PLC
- Communication driver

Symbolic addressing is supported by communication drivers:

- SIMATIC S7-1200
- SIMATIC S7-1500

See also

Tag Limits (Page 4218)

Configuring address multiplexing with absolute addressing

Introduction

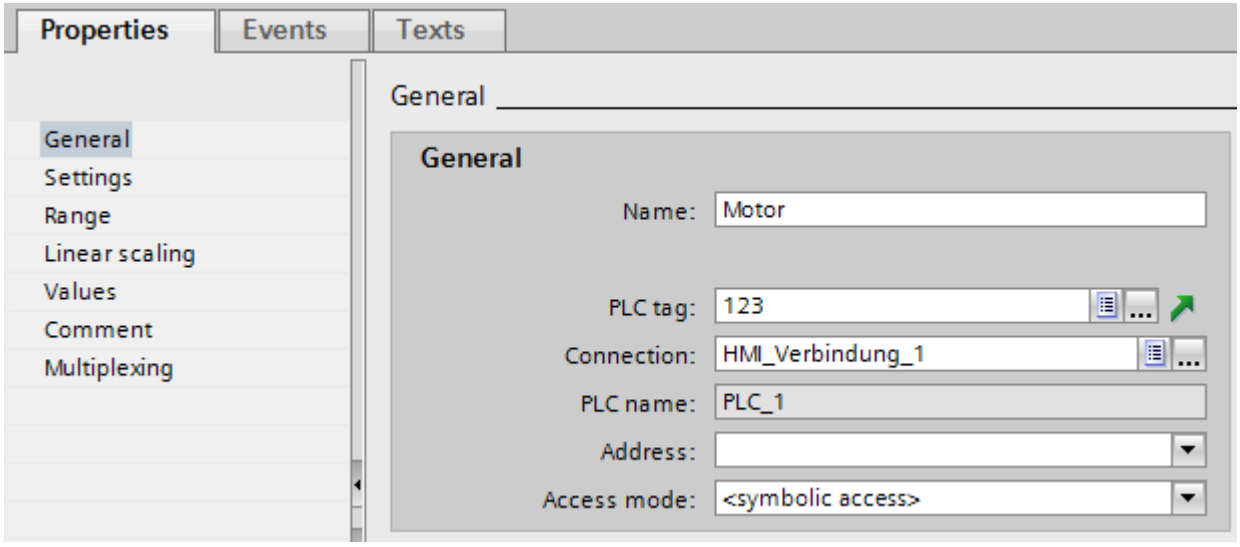
When using address multiplexing, you can efficiently access different addresses in the PLC with the help of a small number of tags. Instead of the absolute address in the PLC, you use tags in order to be able to change the address in Runtime.

Requirement

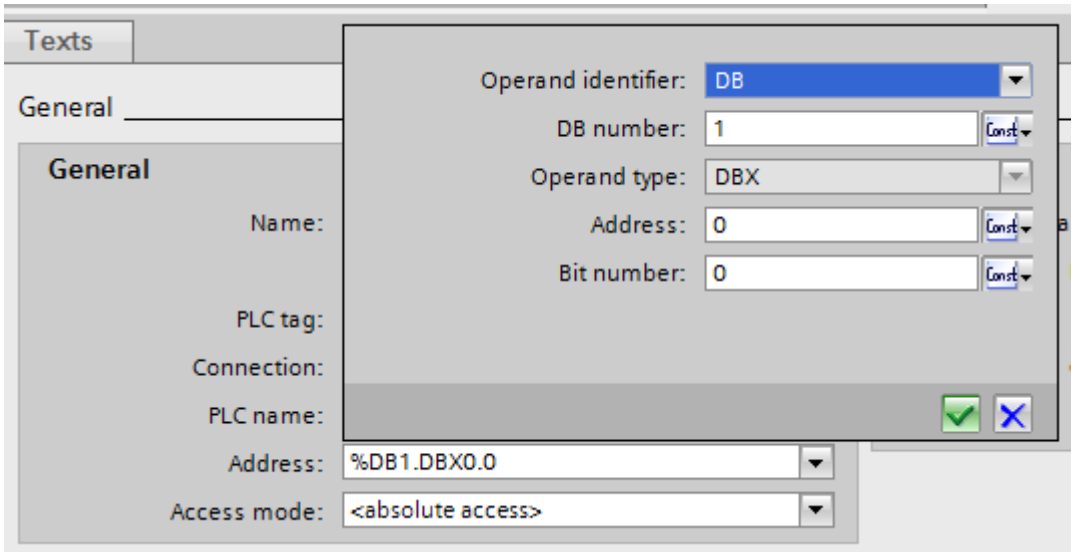
- The tag for address multiplexing is created and connected to the PLC.
- The Properties window for this tag is open.

Procedure

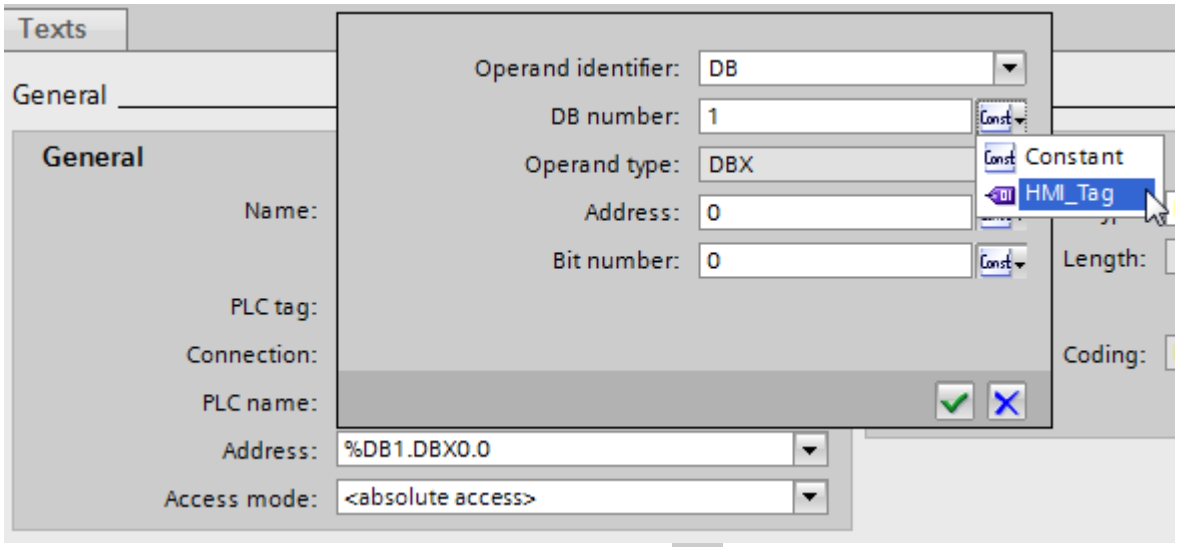
1. Select the tag for address multiplexing in the tag table, and select "Properties > Properties > General" in the Inspector window. The general properties of the tag are displayed.



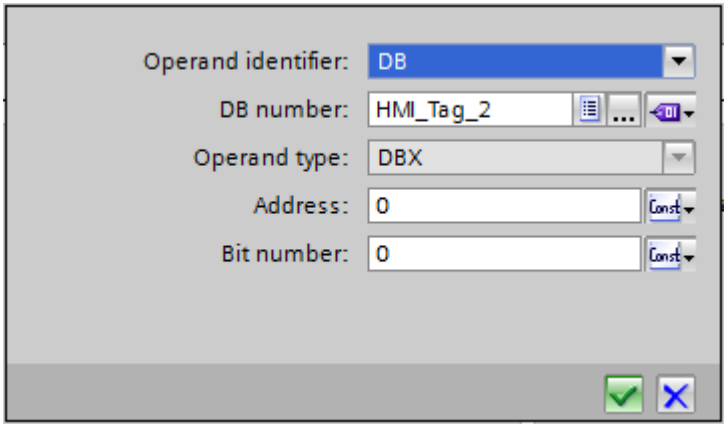
2. Select the "Int" data type for this example.
3. Select the access type "Absolute addressing".
4. Click the selection button in the "Address" field. The address dialog opens.



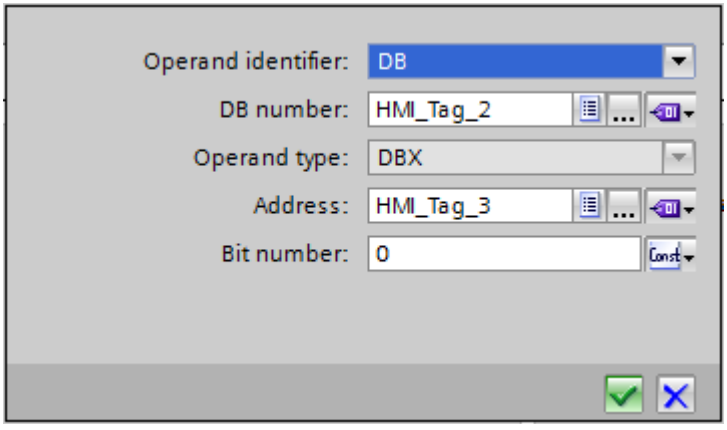
5. Click the selection button in the "DB number" field and select the entry "HMI tag".



6. In the "DB number" field, click the ... button and select a tag for the DB number in the object list. Or create a new tag with the help of the object list. Accept the tag by clicking the button.



7. Repeat steps 3 and 4 for the "Address" field and configure a further tag for calling the address area in the data block.



The selection options in the Address dialog depend on the selected data type of the multiplex tag. The Address dialog offers only address settings that can be configured with the selected data type.

Result

In runtime, the multiplex tag is used to access the memory location corresponding to the address currently found in the tag. You control access to the data block with the tag in the DB number field. You control access to the address in the selected data block with the tag in the "Address" field.

Note

The value in the memory location will only be read at the next update cycle for the addressed tag.

If, for example, you use a multiplex tag in a script, do not attempt to access contents of the memory location directly after changing it.

See also

Tag Limits (Page 4218)

Configuring address multiplexing with symbolic addressing

Introduction

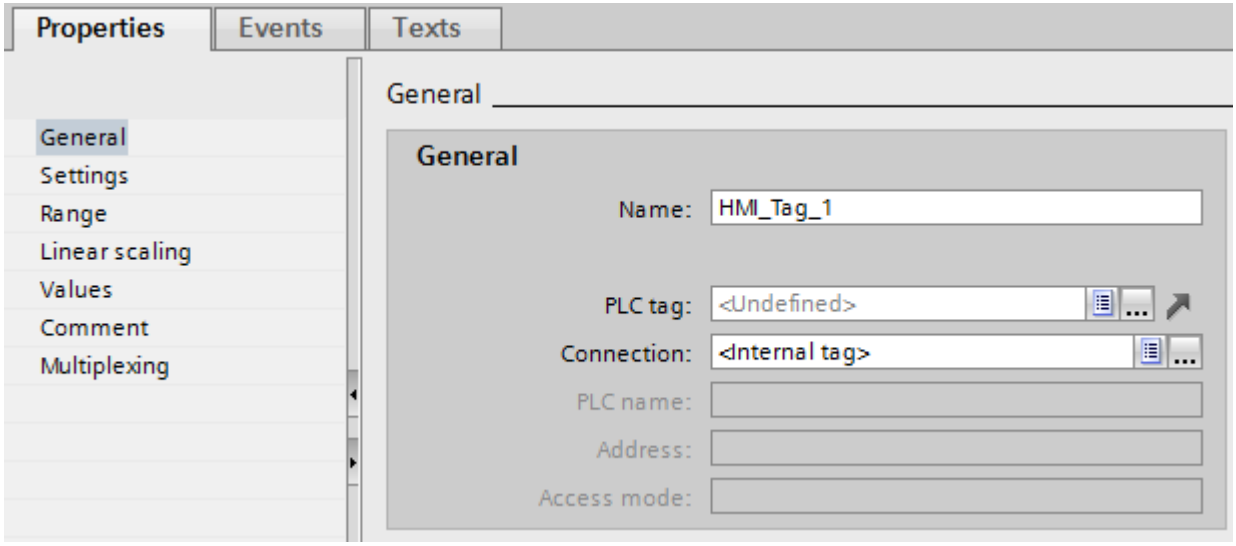
When using address multiplexing, you can efficiently access different addresses in the PLC with the help of a small number of tags. Instead of the symbolic address in the PLC, you use tags in order to be able to change the address in Runtime.

Requirement

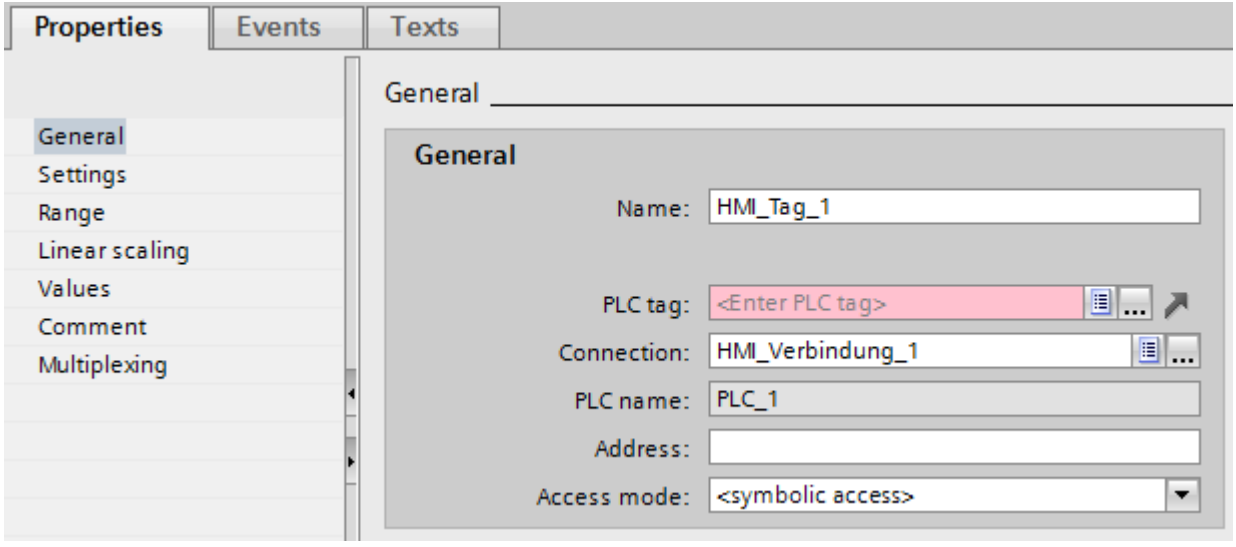
- The tag for address multiplexing is created.
- The Properties window for this tag is open.
- A data block with an array tag is created in the connected PLC.
- The data block was compiled.

Procedure

1. Select the tag for address multiplexing in the tag table, and select "Properties > Properties > General" in the Inspector window. The general properties of the tag are displayed.

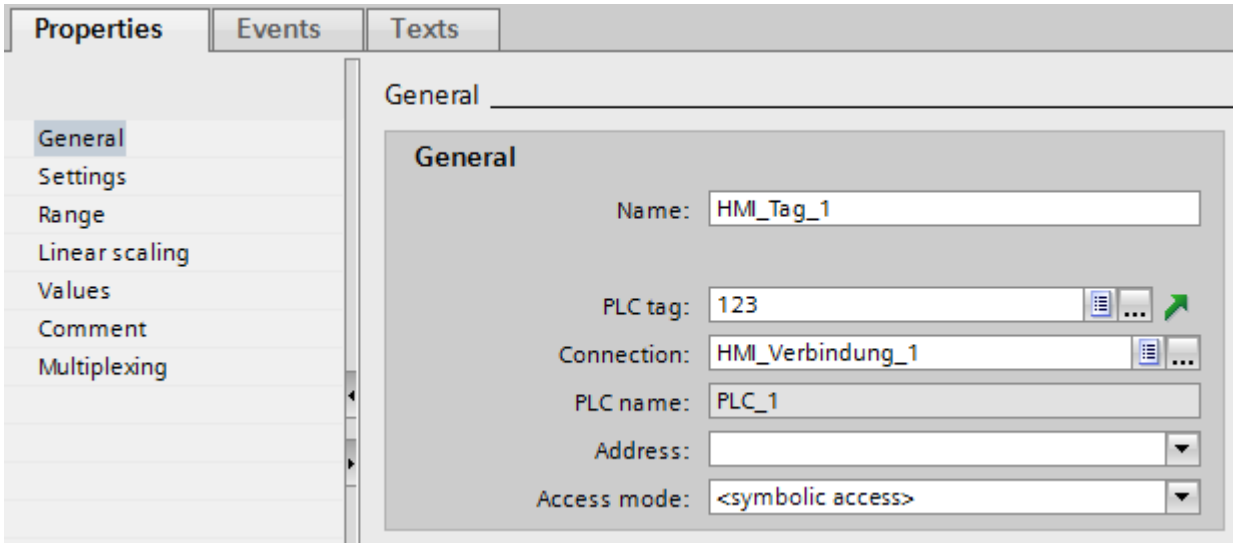


2. Select the connection to the PLC via the "Connection" field.

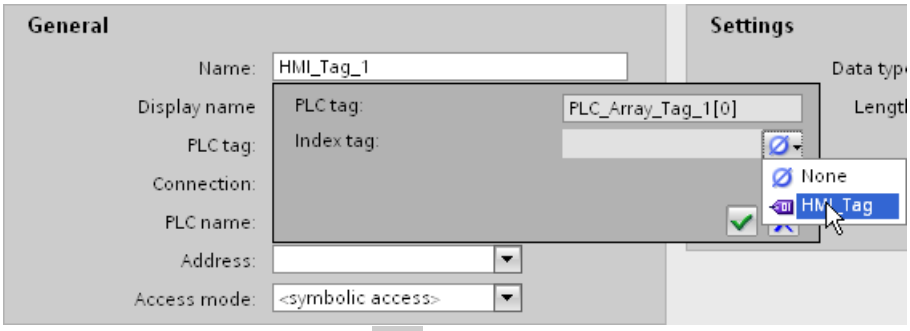


3. Select the access type "Symbolic access".

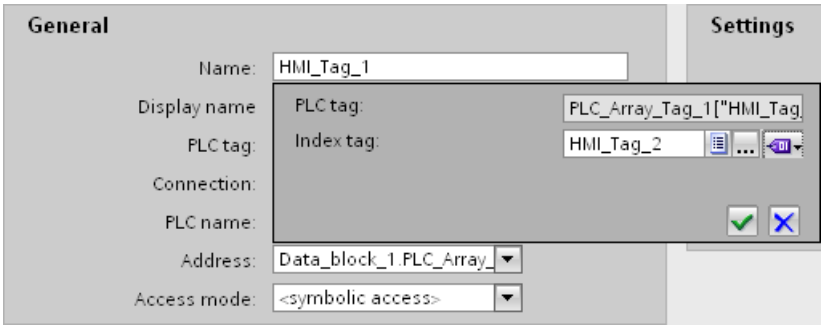
4. Navigate to the data block of the PLC via the "PLC tag" field and select an array element of the array tag.



5. Click the selection button in the "Address" field. The address dialog opens.
6. Click the selection button in the "Index tag" field and select the entry "HMI tag".



7. In the "Index tag" field, click the ... button and select a tag for the array index in the object list. Or create a new tag with the help of the object list. Accept the tag by clicking the button.



Result

In Runtime, the array element whose index value is contained in the index tag is accessed.

See also

Indirect addressing of tags (Page 4232)

Tag Limits (Page 4218)

Special configuration options

Indirect addressing of tags

Principle

In multiplexes, a type of indirect addressing, the tag used is first determined at runtime. A list of tags is defined for the multiplex tag. The relevant tag is selected from the list of tags in runtime. The selection of the tag depends on the value of the index tag.

In Runtime, the system first reads the value of the index tag. Then the tag which is specified in the corresponding place in the tag list is accessed.

Application example

Using indirect addressing, you could configure the following scenario:

The operator selects one of several machines from a selection list. Depending on the operator's selection, data from the selected machine will be displayed in an output field.

To configure such a scenario, configure the index tag for a symbolic I/O field. Configure the multiplex tag at an I/O field. Configure the tag list of the multiplex tag to reflect the structure of the selection list.

If the operator selects another machine, the value of the index tag will change. The selection field then displays the content of the tag that is pointed to in the tag list in the multiplex tag by the new index value.

See also

Addressing tags indirectly (Page 4233)

Using tags to trigger functions (Page 4234)

Defining the acquisition cycle for a tag (Page 4235)

Basics of tags (Page 4197)

Configuring address multiplexing with symbolic addressing (Page 4229)

Creating external tags (Page 4207)

Addressing tags indirectly

Introduction

With indirect addressing, the tag used is first determined at runtime. Instead of a single tag, a list of tags is defined. The list entries consist of an index value and the name of the tag to be used. Using an index tag, you can control which entry in the tag list will be accessed.

Requirement

- The tag which you wish to address indirectly must already exist.
- The index tag must exist.
- The tags which will be contained in the tag list must already exist.
- The Inspector window with the tag properties is open.

Procedure

To address tags indirectly, proceed as follows:

1. In the Inspector window select "Properties > Properties > Multiplexing".
2. Select the "Multiplexing" option to activate indirect addressing.
   Using this option, you can temporarily switch off indirect addressing for testing purposes, for example. Settings which were made earlier for indirect addressing remain unchanged.
3. Select an "Index tag" or define a new tag using the object list.
4. Click the first entry in the "Tags" column in the tag list.
5. Select a tag as a list entry or define a new tag using the object list.
   The entry in the "Index" column will be generated automatically.
6. Repeat step 5 for all tags that you wish to add to the tag list.
7. If necessary, you can use drag-and-drop to change the order of the entries in the list.

Result

In runtime, the system will dynamically access the tag in the tag list which has the same index value as the value currently in the index tag.

See also

Indirect addressing of tags (Page 4232)

Using tags to trigger functions

Introduction

You can use the values of variables as the triggering event for an action in runtime. To start an action in Runtime, configure a function list for a tag. Include one or more system functions or VB scripts in the function list. The function list is processed when the configured event occurs.

The following events are available for a tag:

- Change in value of the tag
  Function list processing is triggered by each change in the value of the variable.
  When the tag is defined as an array tag, the function list is processed whenever an array element changes.
- Violation of the tag's high limit
  The function list is processed when the high limit is violated.
- Violation of the tag's low limit
  The function list is processed when the low limit is violated.

Requirement

- The tag whose value you wish to use as an event already exists.
- The Inspector window with the properties for this tag is open.

Procedure

To configure a tag with a function list, proceed as follows:

1. Under "Properties > Events >" in the Inspector window, select the event for which you want to create a function list.
   The function list associated with the selected event is shown.
2. Click "<Add function>". The second table column contains a selection button.
3. Click the selection button and select a system function.
4. Define the parameter values.

Alternatively, select a script to run in the function list, rather than a system function. The "VB scripts" entry is only displayed if there is a script available in the project.

Result

If the configured event occurs in Runtime, the function list or the script is processed.

See also

Indirect addressing of tags (Page 4232)

Defining the acquisition cycle for a tag

Introduction

The value of an external tag can be changed in Runtime by the PLC to which the tag is linked. To ensure that the HMI device is informed of any changes in tag values by the PLC, the values must be updated on the HMI. The value is updated at regular intervals while the tag is displayed in the process screen or is logged. The interval for regular updates is set with the acquisition cycle. The update can also be made continuous.

Requirement

- You have created the tag for which you want to define an acquisition cycle.
- The Inspector window with the tag properties is open.

Procedure

To configure an acquisition cycle for a tag, follow these steps:

1. In the Inspector window select "Properties > Properties > General."
2. If you want to update the tag at regular intervals as long as it is being displayed on the screen or logged, select "Cyclic in operation" as the acquisition mode.
   Or:
   If you want to update the tag at regular intervals even though it is not being displayed on the screen or logged, select "Cyclic continuous" as the acquisition mode.
   The "Cyclic continuous" setting is selected for a tag, for example, that has a function list configured for a change of its value and that is not directly visible in a screen.
3. Select the required cycle time in the "Acquisition cycle" field or define a new acquisition cycle using the object list.

Alternatively, you can configure the acquisition cycle directly in the work area of the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Note

Only use the "Cyclic continuous" acquisition mode for tags that really have to be continuously updated. Frequent read operations generate a heavy communication load.

Result

The configured tag is updated in Runtime with the selected acquisition cycle.

See also

Indirect addressing of tags (Page 4232)

12.2.3 Working with arrays

12.2.3.1 Basics on arrays

Definition

Array data of a uniform data type is successively arranged and is addressed within the address space to allow access to these data by means of an index. The individual array elements are addressed by means of an integer index. Each array element is assigned the same properties which are configured at the array tag.

Default tag table

Name	Tag table	Data type	Connection
HMI_Tag_1	Default tag table	Array [0..4] of Int	<Internal tag>
[0]	Default tag table	Int	<Internal tag>
[1]	Default tag table	Int	<Internal tag>
[2]	Default tag table	Int	<Internal tag>
[3]	Default tag table	Int	<Internal tag>
[4]	Default tag table	Int	<Internal tag>

Advantages

You can configure numerous array elements with the same properties all at once with a single array tag. You can then use each array element as any other tag in your configuration.

Restrictions

The following restrictions apply to the use of arrays:

- An array may contain only one dimension.
- The lower index of an array must begin with "0".

Application examples

Array tags can be used in the following situations:

- To group process values in profile trends: You map process values to trends which are acquired at different points in time, for example.
- To access specific values which are grouped in trends: For example, display all recorded values of the profile trend by gradually increasing the index tag.
- To configure discrete alarms with successive bit number.
- To save the complete machine data records in a recipe.

License rule for runtime

An array tag is counted in WinCC Runtime as 1 PowerTag, regardless of the number of array elements.

Special features

Warning

Increased system load and performance losses

Read or write access to a single array element always includes read or write access to all array elements of the array tag. Transfer of the data of large arrays from and to the PLC usually takes longer compared to the transfer of a basic data type. This may cause communication overload and disruption as a result.

Example:

- An array tag which consists of 100 array elements of data type "Real" was configured.
- If an array element with a length of 4 bytes changes, 100 x 4 bytes are written to the PLC.

Use in scripts

To maintain performance, always use internal arrays to change arrays in scripts.

1. Copy the PLC array to the internal array at the start of the script.
2. Load on data transfer to the PLC is avoided while the script processes the internal array.

Caution

Data inconsistency at array tags

If the value of a single element must be changed in an array tag, the whole array is read, changed and rewritten as a complete array. Changes carried out in the meantime to other array elements in the PLC are overwritten during rewriting.

You should always prevent the HMI device and the PLC from concurrently writing values to the same array tag. Use synchronous transfer of recipe data records to synchronize an array tag with the PLC.

See also

Basics of tags (Page 4197)

Creating array tags (Page 4238)

Examples of arrays (Page 4239)

Basic Panel (Page 7009)

Basic Panel 2nd Generation (Page 7013)

Panel (Page 7016)

Mobile Panel (Page 7020)

Multi Panel (Page 7026)

Comfort Panel (Page 7029)

WinCC Runtime Advanced (Page 7034)

12.2.3.2 Creating array tags

Introduction

Create an array tag to configure a large number of tags of the same data type. The array elements are saved to a consecutive address space.

You can create an array tag as an internal tag or as an external tag.

If you want to create an array tag as an external tag, first configure an array tag in a data block of the connected PLC. You then connect the array tag to an HMI tag.

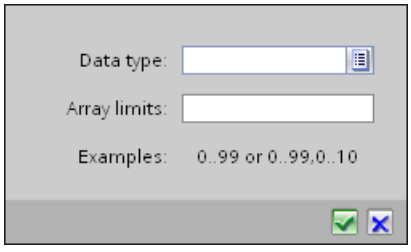
Requirement

- The HMI tag table is open.

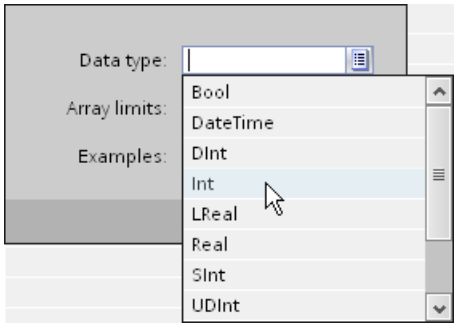
Procedure

To create an array tag, follow these steps:

1. Double click <Add> in the "Name" column of the HMI tag table.
   A new HMI tag is created.
2. Click in the Data type column and select the "Array" data type.
3. Click in the data type column. The dialog box for configuring the array is opened.



4. Select the desired data type for the array tag in the "Date type" field.



5. Define the number of array elements in the "Array limits" field. The lower limit must begin with "0".
6. Click . The settings for the array are saved.
7. Save the project.

Result

An array tag is created. The properties of the array elements are inherited from the parent array tag.

See also

Basics on arrays (Page 4236)

12.2.3.3 Examples of arrays

Introduction

Array tags combine a number of tags, e.g. 100 array elements. You use array tags as complete arrays in the following places:

- In the "Alarms" editor
- In the "Recipes" editor
- For address multiplexing
- In the trend view

You use individual array elements everywhere in the configuration like HMI tags.

Examples

You can configure an array tag with the corresponding number of array elements to handle multiple tags of the same data type.

- The individual array elements can be accessed indirectly by means of a multiplex index tag, for example.
- Use these index tags to operate and monitor the array elements.

See also

Basics on arrays (Page 4236)

12.2.4 Working with user data types

12.2.4.1 Basics on user data types

Introduction

With user data types you bundle a number of different tags that form one logical unit. You create a user data type as a type and use instances of this type in the project. User data types are project-associated data and are available for all HMI devices of the project.

User data types differ in their association to a family of devices. User data types are available for the following families of devices:

- WinCC Runtime Advanced and Panels
- WinCC Runtime Professional

WinCC also supports the connection of PLC data types (UDTs) and user data types to HMITags.

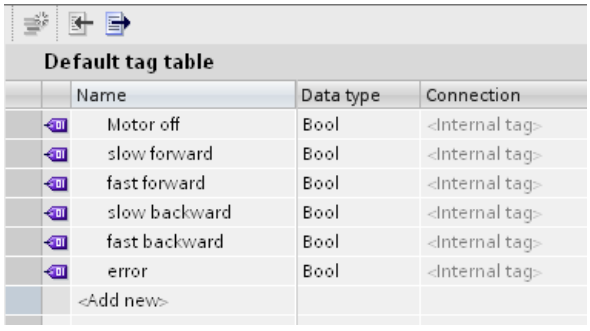
User data types also differ in their applicability with a specific communication driver. User data types are available for the following communication drivers:

- SIMATIC S7-300/400
- SIMATIC S7-1200
- SIMATIC S7-1500

Create user data types and user data type elements in the project library.

Principle

For example, the different conditions of a motor can be described using 6 unique Boolean tags.



Default tag table

	Name	Data type	Connection
	Motor off	Bool	<Internal tag>
	slow forward	Bool	<Internal tag>
	fast forward	Bool	<Internal tag>
	slow backward	Bool	<Internal tag>
	fast backward	Bool	<Internal tag>
	error	Bool	<Internal tag>
	<Add new>		

If the overall condition should be described with a single tag, this tag can be created based on a user data type. For each of the individual Boolean tags you create a user data type element in the user data type.

This user data type can then be assigned complete to a faceplate for the motor. The created and released version of user data type is displayed at the tag in the "Data type" selection field.

The configured properties of a user data type are used in the instances of the user data type. If required, you change individual properties directly at the point of use, e.g. at a tag. Changing a property at the tag does not affect the user data type created.

Notice

Availability of user data types

The connection of user data types to faceplates is only supported in WinCC Runtime Advanced and Panels.

License regulation in Runtime

If you use external tags of the "User data type" data type in a faceplate instance, each user data type element is counted as a tag in Runtime.

Example

You have created two screens in the "Screens" editor: Screen_1 and Screen_2.

3 instances of a faceplate type are inserted in Screen_1. 4 instances of a faceplate type are inserted in Screen_2.

Each faceplate instance is connected to an external tag of the "User data type" data type. The user data type includes 10 user data type elements.

Screen 1: 3 faceplate instances * 10 user data type elements corresponds to 30 external tags = 30 PowerTags.

Screen 2: 4 faceplate instances * 10 user data type elements corresponds to 40 external tags = 40 PowerTags.

In Runtime, 70 PowerTags are counted together for both screens. This also applies to the user data type elements that are not required.

See also

Basics of tags (Page 4197)

Creating tags with a user data type data type (Page 4245)

Managing versions of user data types (Page 4244)

Creating user data type elements (Page 4243)

Creating a user data type (Page 4241)

12.2.4.2 Creating a user data type

Introduction

You create a user data type in the project library.

Requirement

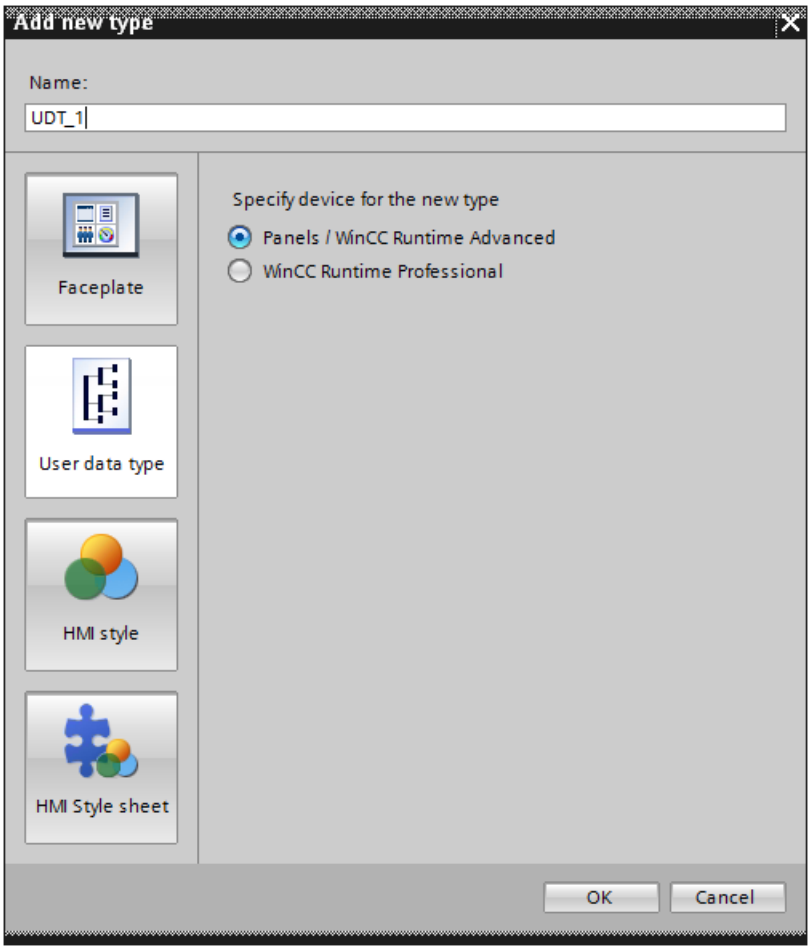
- A project is open.
- The "Libraries" task card is displayed or the library view is open.

Procedure

To create a user data type, follow these steps:

1. Click the "Libraries" task card.
2. Double-click the "Project library" item. The folder structure of the project library is open.

3. Click the "Create new type" button.
   A dialog opens.



4. Click the "HMI UDT" button in the dialog.
5. In the "Specify device for the new type" area select the HMI device on which the user data type is used.
6. Enter a descriptive name in the "Name" field.
7. Click "OK" to apply your settings. The user data type is created. The library view opens.

Result

A user data type with the configured properties is created. Version 0.0.1 of the user data type is created and receives the status "in progress".

Create the required user data type elements in the next step.

Before you use the version of user data types, for example at a tag, the version must be released.

See also

Basics on user data types (Page 4239)

Managing versions of user data types (Page 4244)

12.2.4.3 Creating user data type elements

Introduction

You define user data type elements in the library view. You add or delete elements in the "HMI user data types" table in the work area.

Requirement

- The library view is open.
- A user data type is created and opened in the editor.

Procedure

1. Select a communication driver for the user data type.

- If you select the <Internal communication> entry, you can only assign the user data type to the internal tags as the data type.
- If a connection to a PLC is selected as the communication driver, the user data type can only be assigned to tags with a connection to this PLC as the data type.
- The communication type set applies to all user data type elements of a user data type. In a user data type for the"WinCC Runtime Professional" family of devices, you can define for each user data type element whether the configured driver is used for control or internal communication.

1. Double-click "Add" in the "Name" column of the table. A new user data type element is added to the user data type.
2. Assign a name.
3. Select the required data type in the "Data Type" field.
4. Create as many user data type elements as you need.
5. You configure the user data type elements in the "Properties" group in the Inspector window.

Alternatively, you can configure the properties of the user data type elements directly in the table. To view hidden columns, activate the column titles using the shortcut menu.

Result

You have added elements to version 0.0.1 of the user data type. The version 0.0.1 has the status "in progress".

To use the version in the project, release the version in the next step.

See also

Basics on user data types (Page 4239)

12.2.4.4 Managing versions of user data types

Introduction

All user data types have at least one version. When a user data type is created, a version is created at the same time and this version has the status "in progress". You can edit the user data type in this status as required. When the editing is complete, you release the version of the user data type.

Requirement

- You have created a user data type.
- The user data type has the version 0.0.1. and the status "In progress".
- The "Libraries" task card or the library view is open.

Releasing a version

1. Select the version 0.0.1 of the user data type in the project library.
2. Select "Release version" in the shortcut menu.
   A dialog opens.
3. If necessary, change the properties of the version:
   - Enter a name for the type in the "Name" field.
   - In the "Version" field, define a main and an intermediate version number for the version to be released.
   - To clean up version management of the type, enable "Delete unused type versions from the library".

You have released version 0.0.1 of the user data type.

Editing a version

1. Select, for example, the released version 0.0.1 of a user data type in the project library.
2. Select "Edit type" in the shortcut menu.

The library view opens. The new version 0.0.2 of the user data type is created.

The version has the status "in progress".

Restoring the last version of a user data type

The last released version of the user data type is version 0.0.2.

You edit the user data type. A new version of the user data type, 0.0.3, is generated and receives the status "In progress".

1. Select the version of the user data type in the project library.
2. Select "Discard changes and restore version" in the shortcut menu.

Alternatively

1. If you have opened a version for editing, click "Discard changes and restore version" in the toolbar.
   The version is deleted.

All changes to the user data type since the last release operation are discarded. The user data type is released again and has version 0.0.2.

Deleting user data type

If you delete a user data type, all instances and master copies of this user data type are deleted as well. The same is true for HMI tags that use an HMI user data type.

To delete a user data type, follow these steps:

1. In the project library, under "Types", select the user data type you want to delete.
2. Select "Delete" from the shortcut menu.

See also

Basics on user data types (Page 4239)

Creating a user data type (Page 4241)

12.2.4.5 Creating tags with a user data type data type

Introduction

When a tag is created, you assign the version of user data type as a data type. In the "Tag" editor you can create internal tags or tags with a link to a PLC. A tag has access to all user data type versions that use the same communication driver as the tag itself.
In connection with a PLC, a user data type can only be used if the absolute addressing is selected.

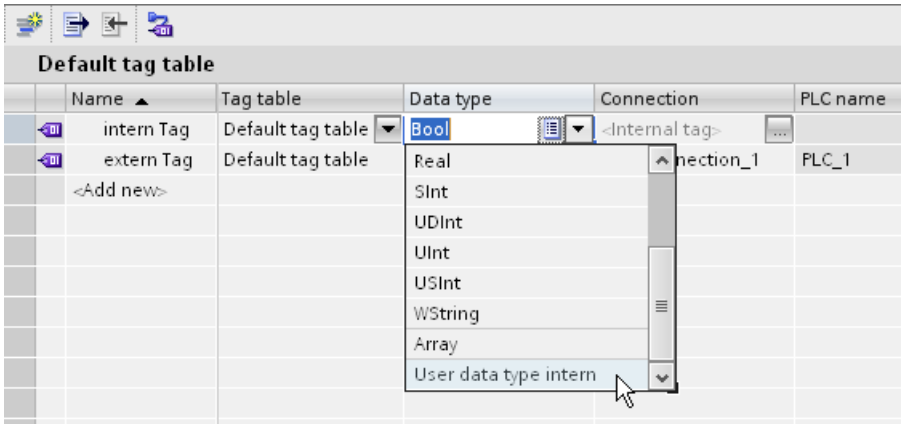
Requirement

- A user data type with a user data type element is created.
- The user data type is enabled.
- The tag table is open.
- The Inspector window with the tag properties is open.

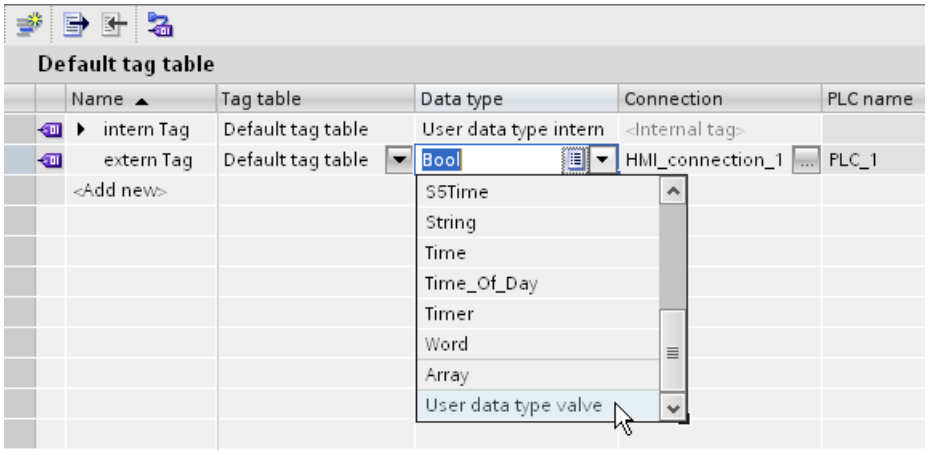
Procedure

To create a tag of the "User data type" data type, follow these steps:

1. In the "Name" column, double-click "Add" in the tag table. A new tag is created.
2. In the Inspector window select "Properties > Properties > General".
3. Enter a unique tag name in the "Name" field.
4. Select the connection to the required PLC in the "Connection" box.
5. Select the desired version of the user data type in the "Data type" field.
   The selected connection determines which user data types will be displayed.
   For internal tags, only user data type versions of the <Internal user data type> type are available.



   For tags with a connection to a PLC, only those user data types that have a link to a PLC can be accessed.



6. Enter the address of the PLC that you want to access with the external tags in the "Address" field of the "Settings" area. The specified address must always point to the start data bit, for example, <DB1.DBX0.0>.

Result

You have created a tag of the "User data type" data type. In additional steps you can configure the tag, for example, by setting the start value and limits.

With tags of the "User data type" data type, you can connect the dynamic properties of a faceplate instance, which then supplies these properties with values in Runtime.

If you wish to change the properties of a user data type tag, you must change the properties of the user data type element.

Properties such as "Start value", "Use substitute value" or " Linear scaling" can be changed in the employed instances of the user data type.

See also

Basics on user data types (Page 4239)

12.2.5 Working with cycles

12.2.5.1 Cycle basics

Introduction

Cycles are used to control actions that regularly occur in runtime. Common applications are the acquisition cycle, the logging cycle and the update cycle. You can also define your own cycles in addition to those already provided in WinCC.

Principle

In Runtime, actions that are performed regularly are controlled by cycles. Typical applications for cycles:

- Acquisition of external tags
  The acquisition cycle determines when the HMI device will read the process value of an external tag from the PLC. Set the acquisition cycle to suit the rate of change of the process values. The temperature of an oven, for example, changes much more slowly than the speed of an electrical drive.
  Do not set the acquisition cycle too low, since this will unnecessarily increase the communication load of the process.
- Logging process values
  The logging cycle determines when a process value is saved in the logging database. The logging cycle is always an integer multiple of the acquisition cycle.

The smallest possible value for the cycle depends on the HMI device that will be used in your project. For most HMIs, this value is 100 ms. The values of all other cycles are always an integer multiple of the smallest value.

If the cycles provided in WinCC do not meet the needs of your project, you can define your own cycles using the "Cycles" editor. User-defined cycles must always be an integer multiple of the smallest value.

Note

Device dependency

You cannot configure self-defined cycle times for Basic Panels.

Application example

You can use cycles for the following tasks:

- To regularly log a process.
- To draw attention to maintenance intervals.

See also

Basics of tags (Page 4197)

Defining cycles (Page 4248)

12.2.5.2 Defining cycles

Introduction

Use cycles to control actions that are run at regular intervals in Runtime. You can also define your own cycles in addition to those already provided in WinCC.

Requirement

You have opened the project.

Procedure

1. Double-click the "Cycles" entry in the project navigation.
   The "Cycles" editor opens.
2. In the "Name" column of the "Cycles" editor, double-click "Add".
   A new cycle time is created.
3. Enter a unique name in the "Name" field.
4. Select the desired cycle unit.

5. Select the desired value for the cycle time.
   The available selection of values varies depending on the cycle unit selected. The smallest possible value for the cycle depends on the HMI device that will be used in your project. For most HMIs, this value is 100 ms. The available values are always an integer multiple of this value.
6. As an option, you can enter a comment regarding the use of the cycle.

Result

The cycle you configured is created and beside the default cycles in WinCC for use during configuration.

See also

Cycle basics (Page 4247)

12.2.6 Logging tags

12.2.6.1 Basic principles for data logging

Introduction

Data logging is used to collect, process and log process data from an industrial system.

When you analyze the logged process data, you can extract important business and technical information regarding the operational state of the system.

Application of the data logging

You can use data logging to analyze error statuses and to document the process. By analyzing data logs, you can extract the information necessary to allow you to optimize maintenance cycles, increase product quality and ensure that quality standards are met.

See also

Data logging (Page 4250)

Logging Tags (Page 4251)

Basics of tags (Page 4197)

12.2.6.2 Data logging in Runtime Advanced and Panels

Basics

Data logging

Introduction

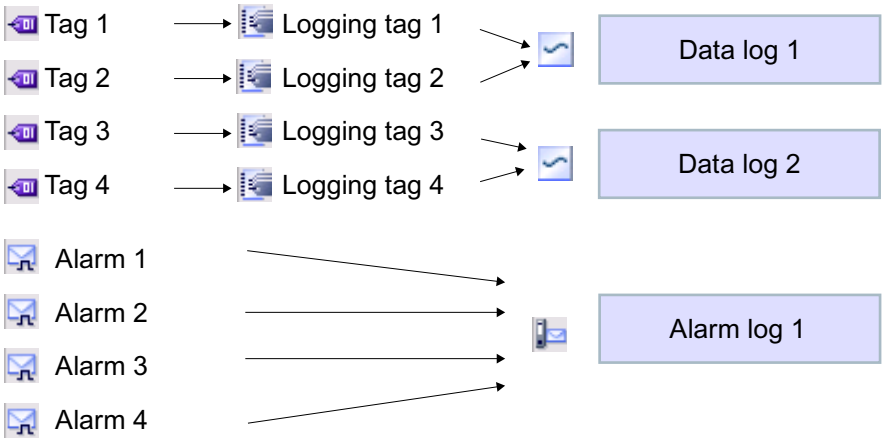
Data is information that is collected during the process and saved in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. You define tags for acquiring and editing the process values in WinCC.

External tags are used in WinCC to collect process values and to access a memory address in the connected automation system. Internal tags are not linked to any process and are only available at the associated HMI device.

Principle

The values of external and internal tags can be saved in data logs. Create a logging tag for every tag and specify the log in which it is saved.

Data logging is controlled via cycles and events. Logging cycles are used to ensure continuous acquisition and storage of the tag values. You can also trigger data logging in response to events, such as the change in value of a tag. Define these settings independently for each logging tag.



The tag values to be logged are compiled, processed and saved in the data log in Runtime. The HMI device you are using determines where the data log is saved. You can further process the saved data in other programs for analysis purposes, for example.

Logging methods

WinCC supports the following logging methods:

- Circular log
- Segmented circular log
- Circular log which sends a system alarm when it is full
- Circular log which executes system functions when it is full.

Outputting logged data

In runtime, you can output the logged tag values as trends in the process screens.

See also

Basic principles for data logging (Page 4249)

Storage locations of logs (Page 4420)

Working with Data Logs

Logging Tags

Introduction

In Runtime, you store tag values in logs. You can then analyze the logged data at a later date. You define the following conditions for logging a tag:

- The log tag, through which the values of the connected tag are logged.
- The data log in which the tag is stored.
- The cycle or the event with which the tag is stored.
- The value range within which the tag is stored.

Note

You use data logging mainly to log the values of external tags. You can also log the values of internal tags.

Principle

Several steps come together during data logging:

- Creating and configuring the data log
  When you create a data log, you define the following settings:
  - General settings, such as name, size and storage location
  - Runtime start characteristics
  - Behavior when the log is full
- Configuring the logging of tags
  For every log tag, specify a data log in which the values of the connected tags and other information are logged, such as the time of logging.
  When and how often the values of a log tag are logged are also defined. You have the following options:
  - "On demand":
    The tag values are logged by calling the "LogTag" system function.
  - "On change":
    The tag values are logged as soon as the HMI device detects a change in the value of the tag.
  - "Cyclic":
    The tag values are logged at regular intervals. You can supplement the default cycles in WinCC with your own cycles based on the default cycles. The smallest value that can be set is 1 s. All other values are integer multiples of this value.
    You can also restrict the logging of values within or outside a tolerance band. In this way, the logging of values within a relevant range of values is restricted.

  Note the following points if you want to log a tag on demand:
  - Do not log this type of tag in a segmented circular log in which tags are logged in a continuous cycle or in response to changes.

  Background:
  - If logging takes place on demand only rarely, the log segment will be filled by cyclically-logged values, for example, and the next log segment will be created. If you then attempt to access the tag that was logged on demand, it will not be possible to display the tag since it is the current log segment that is accessed in Runtime. To remedy this, you should create a separate data log for tags that are logged only rarely.
- Further processing logged tag values
  You can analyze the logged tag values directly in your project, such as in a trend view, or in another user program, such as Excel.

See also

Creating a data log (Page 4253)

Managing logging behavior when Runtime starts (Page 4254)

Controlling the Logging in relation to the Fill Level (Page 4255)

Logging process values (Page 4256)

Triggering a system function when log is full (Page 4258)

System functions for logs (Page 4259)

Basic principles for data logging (Page 4249)

Configuring logging tags (Page 4266)

Configuring a checksum for a log (Page 4260)

Evaluating the checksum of log data (Page 4261)

Creating a data log

Introduction

Data logs are used to store values from external and internal tags in runtime. When you create a log, you must give it a name and define its size and storage location. In addition you can enter comments for each log.

Requirement

- A project is open.
- The Inspector window is open.

Procedure

To create a data log, proceed as follows:

1. Double-click on the "Historical Data" entry in the project tree.
   The editor for data logs and alarm logs opens.
2. Open the "Data logs" tab and double-click "Add" in the "Name" column of the "Data logs" editor.
   A new data log is created.
3. In the Inspector window select "Properties > Properties > General."
4. Enter a unique name for the log in the "Name" field.
5. Define the number of data records to be logged in each log in the "Number of data records per log" field.
   The size of a log is calculated as follows: The number of items * the length of each tag value to be logged.
   In the Inspector window, the maximum size that the log can reach if the currently selected number of data records is observed is displayed beneath the "Number of data records" input field.
6. In the "Storage location" field you select where the log entries are saved.
7. Depending on the selected "Storage location", you either select the "Path" or the "Name of the data source".
8. If necessary, enter a descriptive text in the "Comment" category to document your configuration.

Alternatively you can configure log properties directly in the "Data log" editor. To view hidden columns, activate the column titles using the shortcut menu.

Result

The data log is created.

You can now configure tags so that their values are stored in this log.

To continue the log configuration, perform the following steps:

- Define the restart characteristics of the log when Runtime is started.
- Define how the log should behave when it is full.
- Configure a function list for the "Overflow" event.

See also

Logging Tags (Page 4251)

Managing logging behavior when Runtime starts

Introduction

When you configure a log, you define the restart characteristics of the log when Runtime is started. You define whether the logging should start when Runtime starts in the log properties. You can also define whether an existing log will be continued or overwritten.

You define the restart characteristics separately for each log.

Requirement

- You have created a log.
- The "Historical Data" editor is open.
- The Inspector window with the log properties is open.

Procedure

To configure the restart characteristics of a data log, proceed as follows:

1. Select the log for which you want to define the restart characteristics in the "Historical Data" editor.
2. In the Inspector window select "Properties > Properties > Restart behavior".
3. If you want logging to start when Runtime starts, enable the "Enable logging at runtime start" option in the "Logging" area.
   You can also start logging in Runtime using the "StartLogging" system function, for example.
4. Select the restart behavior of the log in the "Log handling at restart" area.
   - The logged values are deleted and logging is started again with the option "Reset log".
   - The option "Append data to existing log" is used to append the values to be logged to the existing log.

Alternatively you can configure the restart characteristics of a log directly in the "Historical Data" editor. To view hidden columns, activate the column titles using the shortcut menu.

Result

Logging will start in runtime according to your settings.

See also

Logging Tags (Page 4251)

Controlling the Logging in relation to the Fill Level

Introduction

The size of a log is determined by the number of entries. You use the logging method to determine how the log responds when it is full.

Logging methods

The following logging methods are available:

- Circular log
  When the configured log size has been reached, the oldest entries are deleted. When the configured log size has been reached, approximately 20% of the oldest entries are deleted. It is therefore not possible to display all the configured entries. During configuration, select an appropriate size for the circulation log. Alternatively, configure a segmented circular log.
- Segmented circular log
  In a segmented circular log, multiple log segments of the same size are filled in succession. When all logs are completely full, the oldest log is overwritten.
- Log that sends a system alarm when it is full
  When a defined level is reached, such as 90 %, a system alarm is triggered. When the log is 100% full, new tag values are not logged.
- Log with level-dependent triggering of an event.
  When the log is completely full, the "Overflow" event is triggered. Configure a function list for the event that will be carried out when the "Overflow" event occurs. When the configured size of the log is reached, new tag values are not logged.
  The following system functions are available for further processing of full logs: For further details, refer to System functions for logs (Page 4259).

Requirement

- You have created a log.
- The "Historical Data" editor is open.
- The Inspector window with the log properties is open.

Procedure

1. Select the log for which you want to define the logging method in the "Historical Data" editor.
2. Select "Properties > Properties > Logging method" in the Inspector window and select the required logging method.
3. If you have selected the "Segmented circular log" type, enter the number of log segments.
   If you selected a log with the "Display system alarm on" setting, specify the level as a percentage at which a system alarm is to be triggered.
   If you selected the "Trigger event" setting, configure the function list in the "Events" group.

Alternatively you can configure the logging method directly in the "Historical Data" editor table. To view hidden columns, activate the column titles using the shortcut menu.

The "Overflow" event is not available in the editor table. You must therefore configure the function list in the Inspector window.

Result

The selected log responds according to the settings in Runtime.

See also

Logging Tags (Page 4251)

System functions for logs (Page 4259)

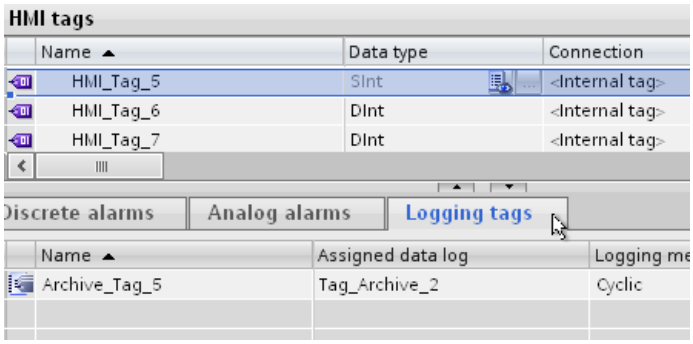
Logging process values

Introduction

You can save the process values of a tag in a data log in Runtime. You define the following conditions for logging a tag:

- The log tag, through which the values of the connected tag are logged.
- In which log the values are stored
- The conditions under which the values are stored
- If only process values for a certain range of values are stored

To log the tag values, assign a logging tag to an HMI tag. The logging tag is stored in the data log and logs the values of the connected HMI tag. You can configure logging tags directly in the "HMI tags" editor. The " HMI tags" editor contains the "Logging tags" editing table.



If the view of the "Logging tags" table is minimized, click the arrow button below the tag table.

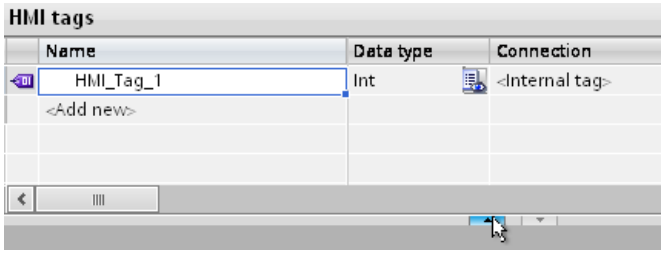


Figure 12-3 The "Logging tags" table is displayed.

Requirement

- The data log has been created.
- The tag for which you wish to configure the logging must already exist.
- The "Tags" editor is open.
- The "Logging tags" table is displayed.
- The Inspector window with the tag properties is open.

Procedure

To log process values in a tag, proceed as follows:

1. Select a tag in the tag table.
2. Double-click "Add" in the "Name" field in the "Logging tags" table.
   A new logging tag is created; it is given the same name as the associated HMI tag.
3. Select the data log in which the values of the tags are to be logged under "Properties > Properties > General" in the Inspector window.
4. Select "Properties > Properties > Logging type" in the Inspector window and select the log type for logging.
   - "Cyclic": The tag values are logged in accordance with the set logging cycle.
   - "On change": The tag values are logged, as soon as the operator device detects a change in value.
   - "On demand": The tag values are logged by calling the "LogTag" system function.

5. If you want to log tag values cyclically, select a cycle time in the "Logging cycle" area. Alternatively, you can define your own cycle using the object list. The smallest value that can be set is 1 s. All other values are integer multiples of this value.
6. If you only want to log tag values outside or inside a defined value range, select "Properties > Properties > Deadband for logging" in the Inspector window. Define the values for the high and low limits.
   If you want to configure a dynamic limit, select "HMI tag" using the selection button. In the second field, select the tag that contains the limit.
   If you want to configure a fixed limit, select "Constant." Enter the limit value in the second field.
   If you want to leave a limit undefined, select "None".
7. Under "Scope", specify whether tag values are to be logged only if they are within the defined limits or only if they are outside the defined limits.

Alternatively, you can configure the logging of a tag directly in the "Logging tags" editor table. To view hidden columns, activate the column titles using the shortcut menu.

You can also fully configure a logging tag in the "Historical Data" editor.

Result

The process values of the configured tag are logged in Runtime according to the selected settings.

Note

To have the tag values actually logged in runtime, you must ensure that the data log is started. Logging can be started automatically at start up or by means of a system function. To automatically start the log, use the setting in the log Properties window.

See also

Logging Tags (Page 4251)

Triggering a system function when log is full

Introduction

You can select a logging method that executes a function list as soon as the log is full.

Application example

You can use system functions, for example, to swap the data from a full log file out to another log before the full log is overwritten. You can then continue to process the transferred data in another program. Assign the "CopyLog" system function to the "Overflow" event accordingly.

Requirement

- A log with the "Trigger event" logging method is created.
- The "Data log" editor is open.
- The Inspector window with the data log properties is open.

Procedure

To configure a system function for the "Overflow" event, proceed as follows:

1. Select the required log in the table in the "Data Log" editor.
2. In the Inspector window select "Properties > Events > Overflow."
   The function list will open.
3. Double-click "Add function" and select the desired system function.
4. Configure the required parameters of the selected system function.

Result

During runtime, the function list will be executed as soon as the log is full.

See also

Logging Tags (Page 4251)

System functions for logs

System functions

The following system functions are available for logging:

Function name	How it works
ArchiveLogFile	This function moves or copies a log to another storage location for long-term archiving. Use the system function if you want to move the Audit Trail, for example, from a local storage medium to the server. With Audit Trails, always use the "Move log (hmiMove)" mode, otherwise you will be violating the FDA guideline by duplicating the data stored.
LogTag	Saves the value of the given tags in the given data log. Use the system function to log a process values at a specific time.
StartLogging	Starts the logging process in the specified log. You can interrupt logging in Runtime by calling the "StopLogging" system function.
StopLogging	Stops the logging process in the specified log. To resume logging in Runtime, select the "StartLogging" system function.
ClearLog	Deletes all entries in the specified log.
StartNextLog	Stops the logging process in the specified log. Logging is continued in the log segment that has been configured for the given log.

Function name	How it works
CloseAllLogs	Closes all logs. The connection between WinCC and the log files or log database is terminated. You can use this system function, for example, to enable hot-swapping of the storage medium on the HMI device without exiting the Runtime software.
OpenAllLogs	Opens all logs to resume logging. The connection between WinCC and the log files or log database is restored.
CopyLog	Copies the contents of the specified log to another log.

See also

Logging Tags (Page 4251)

Configuring a checksum for a log

Introduction

In a regulated project, you have the option of assigning a checksum to the log data of an data log or alarm log. This checksum can be used during plant operation to determine if the data of this log has subsequently changed.

Note

Device dependency

The "Checksum" option is only available for display and HMI devices which support "Configuration conforms to GMP".

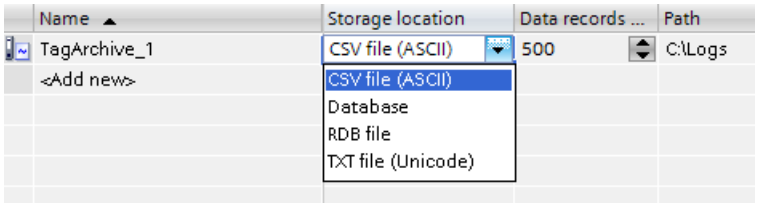
Requirement

- GMP compliant configuration is enabled.
- A data log or alarm log has been created.

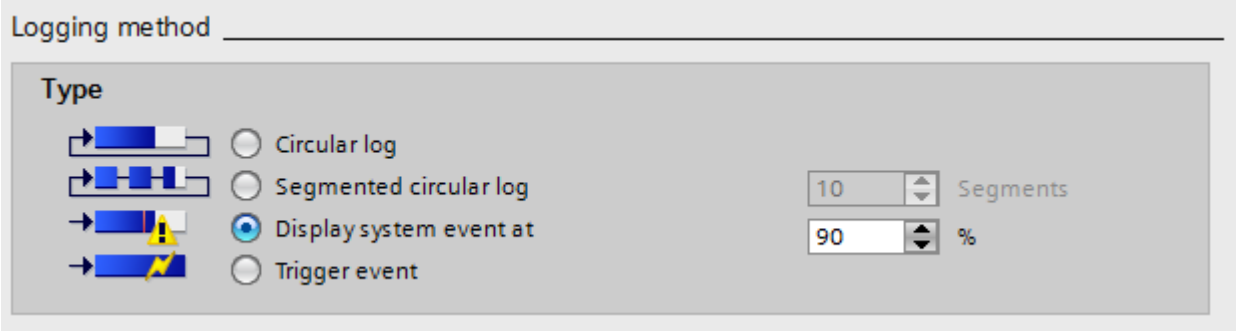
Procedure

Proceed as follows to configure a data log or alarm log for the use of a checksum:

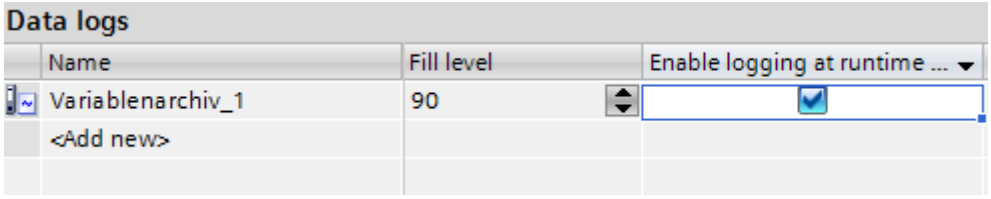
1. Open the data log or alarm log in the corresponding log editor.
2. In the "Storage location" box, select "File - CSV (ASCII)" or "File - TXT (Unicode)".



3. Under "Properties > Properties > Logging method" in the Inspector window , select the option "Display system event at" or "Trigger event".



4. In the editor table, activate the option "Enable logging at runtime start".
   Columns that are not displayed are activated with the shortcut menu of the column title.



5. Save the project.

Result

The log data of the log is assigned a checksum in runtime.

See also

Logging Tags (Page 4251)

Enabling GMP compliant configuration (Page 7046)

GMP-compliant configuration (Page 7043)

Evaluating the checksum of log data (Page 4261)

Evaluating the checksum of log data

Introduction

If you have configured a data log or alarm log with generation of a checksum, you can check if the log data has subsequently changed.

Note

Device dependency

The "Checksum" option is only available for display and HMI devices which support "Configuration conforms to GMP".

The DOS program "HmiCheckLogIntegrity" is available for checking the integrity of the log data.

The "HmiCheckLogIntegrity" program supports verification of the following files:

- Log files of alarm logs, data logs, and Audit in CSV format
- Log files of alarm logs, data logs, and Audit in TXT format
- Recipe data records in CSV format
- Recipe data records in TXT format

You can find the "HmiCheckLogIntegrity.exe" program in the installation directory of WinCC under the folder "WinCC Runtime Advanced", for example <C:\Program Files\Siemens \Automation WinCC Runtime Advanced>.

Note

Audit Trail and Log with Checksum

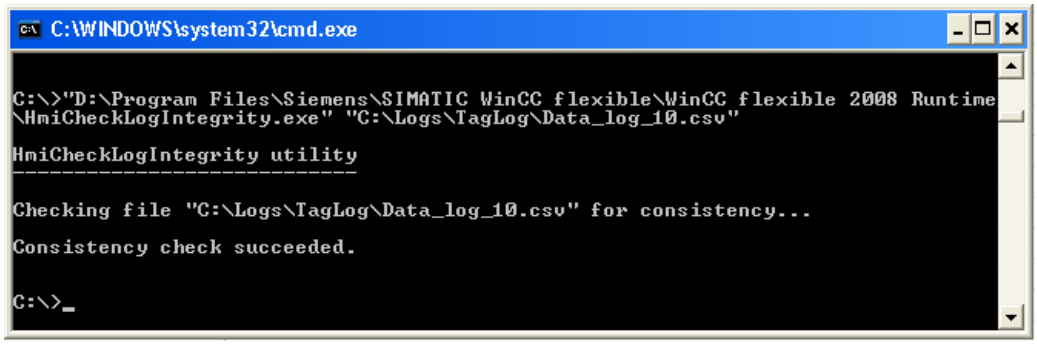
Before you update WinCC with a Service Pack or a new version, exit and save the Audit Trail or the logs using checksum. After WinCC is updated, the audit trail or logs will be continued with new files using checksum.

Make sure that the logs are started at a defined state with the new version.

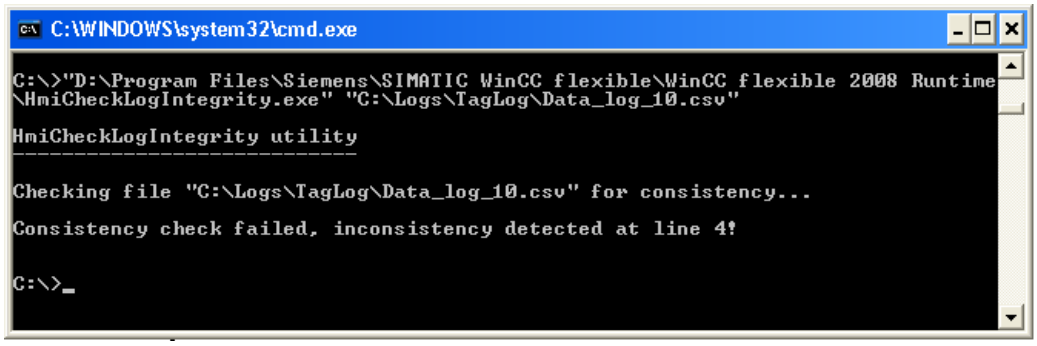
Procedure

1. Copy the file to be checked from the HMI device to your configuration computer.
2. Open a command line prompt with "Start > Programs > Accessories > Command Prompt".
3. Enter the path to "HmiCheckLogIntegrity.exe" followed by a space in the command line prompt. After the space, enter the storage location of the file to be checked within quotation marks.

4. Press <Enter>.
5. The check is performed.
   When the checked data are consistent, the "Consistency check succeeded" message appears.



   When the checked data are inconsistent, the "Consistency check failed" message appears. Information about the first inconsistent line in the file is also displayed.



Note

If there are spaces in the path to the "HmiCheckLogIntegrity.exe" program, you need to specify the path in quotation marks.

You can also check the integrity of the log data with the AuditViewer.

See also

Logging Tags (Page 4251)

Enabling GMP compliant configuration (Page 7046)

Configuring a checksum for a log (Page 4260)

GMP-compliant configuration (Page 7043)

Evaluating Audit Trails in AuditViewer (Page 7063)

Log response to language switching in runtime

Introduction

In the "Device Settings" editor, select the language to be used for writing to logs in runtime.

Requirements

- The languages used in your project are activated in the "Project languages" editor, for example, "German (Germany)" and "English (USA)" .

Procedure

1. Open the "Languages and fonts" editor.
2. Activate the runtime language, for example, "German (Germany)" and "English (USA)".
3. Set the "Language switch order":
   - German 0
   - English 1
     With "0", German is specified as the "Startup language".
4. Open the "Device settings" editor.
5. Select "Startup language" in the "Settings for Runtime" area.

Result

The project starts after it is transferred. German is specified as the "Startup language" with "Language switch order". The logs are now written in German. During runtime, the operator switches the runtime language to English. The logs will still to be written in German.

The operator closes runtime. Due to the previously performed language switch, the next time the system starts the "startup language" is English. Since English is the startup language, the logs are now written in English.

The logging language remains English even when the language is switched again in runtime until runtime is closed once again.

If you use another option instead of "Startup language" to select the language, the logs are always written in the same language. This is true regardless of the language selected by the operator in runtime.

Direct access to the ODBC log database (Advanced)

Overview

The storage location of a log can be a database or a file.

The database is addressed by means of its "Data source name" (DSN). Select the database to be used in WinCC as follows:

Select the database from the Windows Start menu under "Settings > Control panel > ODBC Data Sources".

In your configuration, specify the "Data source name" (DSN) instead of a directory name as storage path for log data. With the DSN, you are referencing the database and the storage location.

The entire functional scope of the database is available for additional processing and evaluation of log data.

Application

The data source sets up the connection to the database. Create the data source on the same PC on which the Runtime software is stored. Then enter the DSN configured on this PC when you create a log in WinCC.

The ODBC interface allows you to use other programs, such as MS Access or MS SQL Server, to access the database directly.

In addition, you can use the "StartProgram" system function to configure program calls, e.g. MS Access, on the HMI device. Runtime is not interrupted while you configure such calls.

Setting up the ODBC data source

Introduction

To store process values or alarms in the log database in Runtime, set up the ODBC data source.

Requirement

- A log database has been created.
  You create the log database using the SQL Enterprise Manager. You can find more detailed information on how to do this from Microsoft.

Procedure

To set up an ODBC data source, follow these steps:

1. In Control Panel, open "Administrative Tools" and select "Data Sources (ODBC)".
   The "ODBC Data Source Administrator" dialog is opened.
2. Click "User DSN > Add".
3. Select the "SQL-Server and click on "Finish".
4. In the dialog that follows, enter a name for the User DSN and the SQL-Server and click on "Next".
5. In the dialog that follows, define the logon procedure for the SQL database and click on "Next".
6. Activate "Change the default database to".

7. Select the database you have created and click on "Next".
8. In the dialog that follows, click "Finish".

Logging tags in the "Historical Data" editor

Configuring logging tags

Introduction

You can also create and edit logging tags in the "Historical Data" editor in WinCC. You also directly edit the properties of logging tags in the "Historical Data" editor.

Note

If you delete, move or copy in the "Historical Data" editor, the changes also take effect in the tag table.

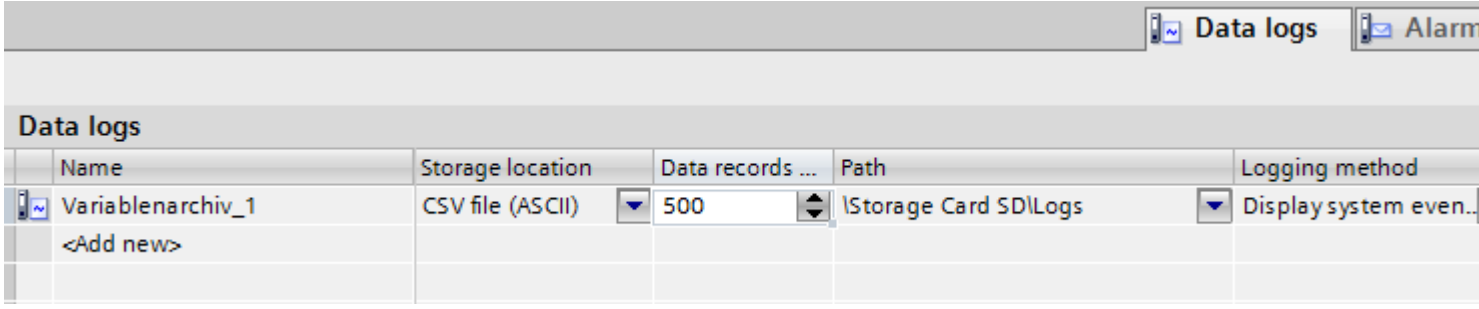
Requirement

- The "Data logs" tab is open in the "Historical Data" editor.
- A data log has been created.

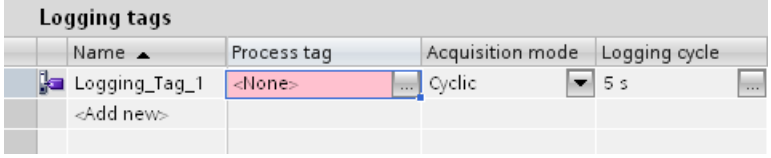
Procedure

Proceed as follows to configure a logging tag in the "Historical Data" editor:

1. Select an existing data log in the "Data logs" table of the editor.
   Alternatively, double-click on "Add..." in the "Name" column to create a new data log.

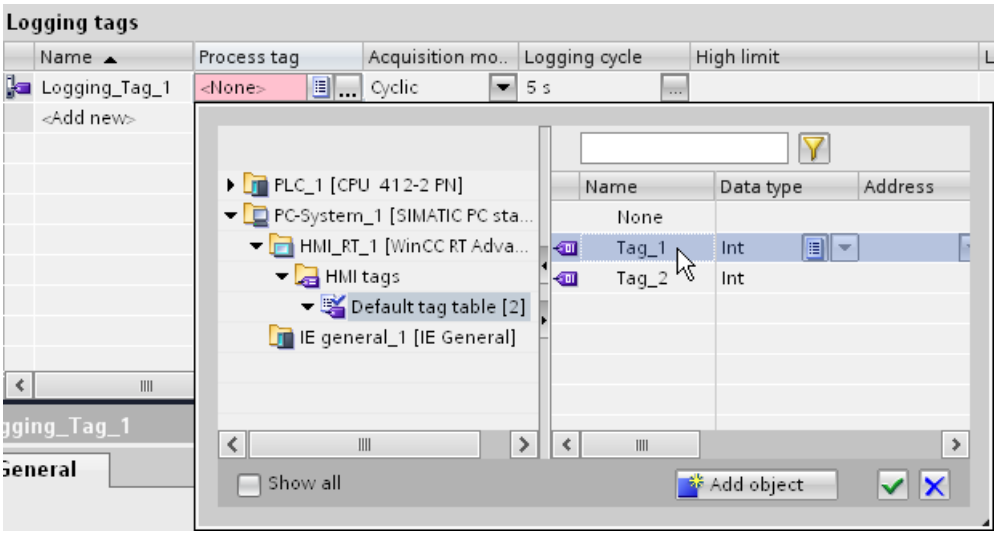


2. Double-click "Add ..." in the "Name" column of the editor "Logging tags" table.



3. Enter a unique name for the logging tag in the "Name" field.

4. In the "Process tag" field, click on the selection button and select the process tag for logging in the object list.



5. Select the desired trigger mode in the "Log type" field:
   - "Cyclic": The tag values are logged in accordance with the set logging cycle.
   - "On change": The tag values are logged, as soon as the operator device detects a change in value.
   - "On demand": The tag values are logged by calling the "LogTag" system function.
6. If you want to log tag values cyclically, select the desired cycle time in the "Logging cycle" area. Alternatively, you can define your own cycle using the object list. The smallest value that can be set is 1 s. All other values are integer multiples of this value.
7. Configure additional parameters for logging in the table of the editor or in the Inspector window.

Result

The configured logging tag is created in the "Historical Data" editor and is also displayed in the tag table.

See also

Logging Tags (Page 4251)

12.2.7 Displaying Tags

12.2.7.1 Displaying tags with Basic Panels

Outputting tag values in screens (Basic)

Introduction

In runtime you can output tag values in the screens of the operator device in the form of a trend. A trend is a graphic representation of the values that a tag takes during runtime. Use the "Trend display" graphic object to represent it. Process values for the trend display are loaded by the PLC from the ongoing process.

The values to be displayed are determined individually within a fixed, configurable cycle. Cyclically-triggered trends are suitable for representing continuous curves, such as the changes in the operating temperature of a motor.

Displayed values

You will need to configure a trend view in a screen so that tag values are displayed on the HMI device. When configuring the trend view, specify which tag values are to be displayed.

You can control the updating of the trend display by defining the cycle time.

See also

Configuring trend display for values from the PLC (Basic) (Page 4268)

Basics of tags (Page 4197)

Configuring trend display for values from the PLC (Basic)

Introduction

You use a trend view to graphically represent values that a tag assumes during the process.

Requirement

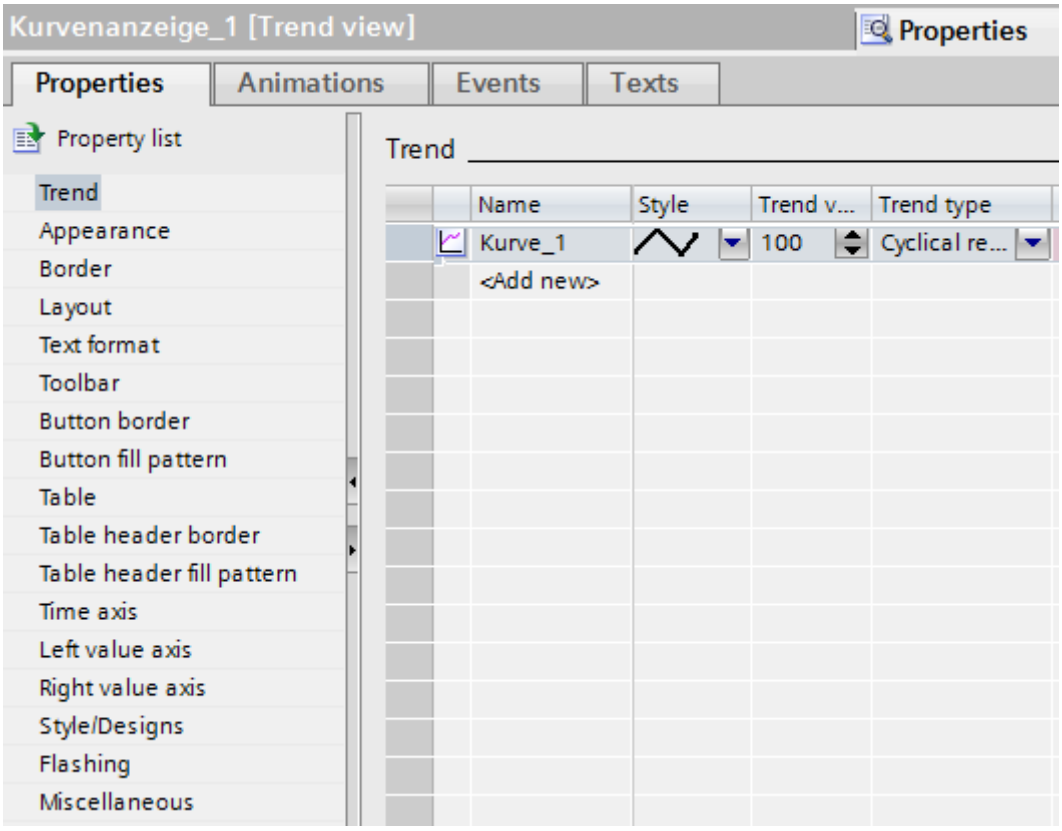
- A screen is open.
- The Inspector window with the trend view properties is open.

Procedure

To configure a trend view, follow these steps:

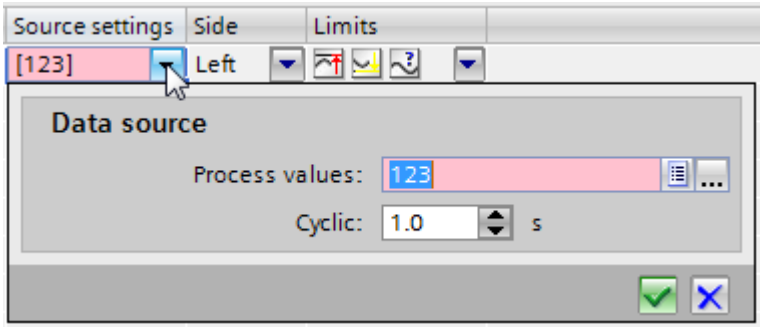
1. Add the "Trend view" object from the toolbox in the "Control" group to the screen.

2. Select the "Trend" category from the "Properties" group in the Inspector window and double-click "<Add>" in the "Name" column.



3. Assign a name to the trend in the "Name" column.
4. In the "Style" column, use the selection button to open the "Style" dialog and select the style of the line.
5. Select the number of trend values in the "Trend values" column.

6. In the "Settings" column, use the selection button to open the "Data source" dialog and select the tags to supply the trend with values.
   Specify the cycle for reading the tags from the PLC.



7. You can make other settings in the dialogs of the Inspector window. For example, you can select the "Display table" option in the "Table" category to display a value table beneath the trend view.

Note

If you hold down the <CTRL> key and double-click the trend view, the trend view is activated. You set the column width and the position of the columns in the table header of the values table in active mode. In order to activate the trend view the zoom factor has to be set to 100 %.

Result

In runtime, the values of the selected tags are displayed in the configured trend view.

See also

Outputting tag values in screens (Basic) (Page 4268)

12.2.7.2 Displaying tags with Runtime Advanced and Panels

Trends

Introduction

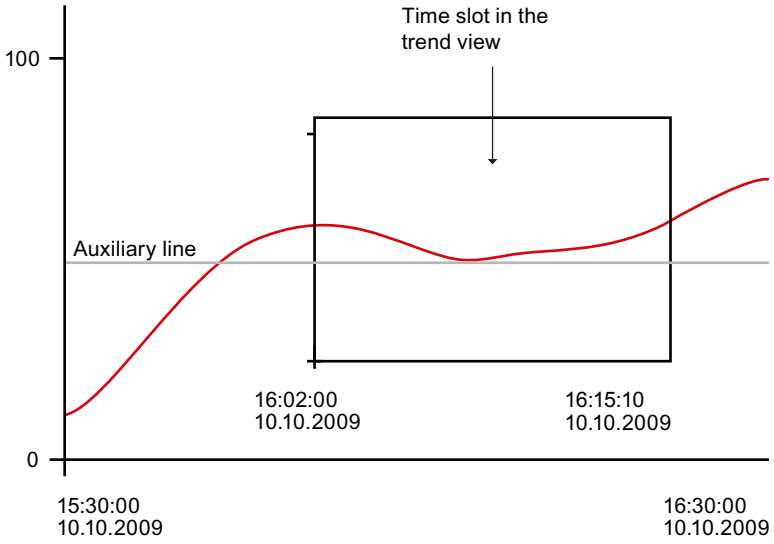
A trend is a graphic representation of the values that a tag takes during runtime. In order to display trends, configure a trend view in a screen of your project.

To configure the trend view, specify a trend type for the values to be displayed.

- Log: For displaying the logged values of a tag
- Cyclical real time: For time-triggered display of values
- Bit-triggered real time: For event-triggered display of values
- Bit-triggered buffer: For event-triggered display with buffered data acquisition

Displaying logged values

In Runtime, the trend view displays the values of a tag from a data log. The trend shows the logged values in a particular window in time. The operator can move the time window in Runtime to view the desired information from the log.



Pulse-triggered trends

The values to be displayed are determined individually with a definable time pattern. Cyclically-triggered trends are suitable for representing continuous curves, such as the changes in the operating temperature of a motor.

Bit-triggered trends

The values to be displayed are event-driven by setting a defined bit in "Trend transfer" tags. The bit is reset after reading has been completed. Bit-triggered trends are useful for displaying fast-changing values, such as the injection pressure for producing plastic parts.

Bit-triggered trends with buffered data acquisition

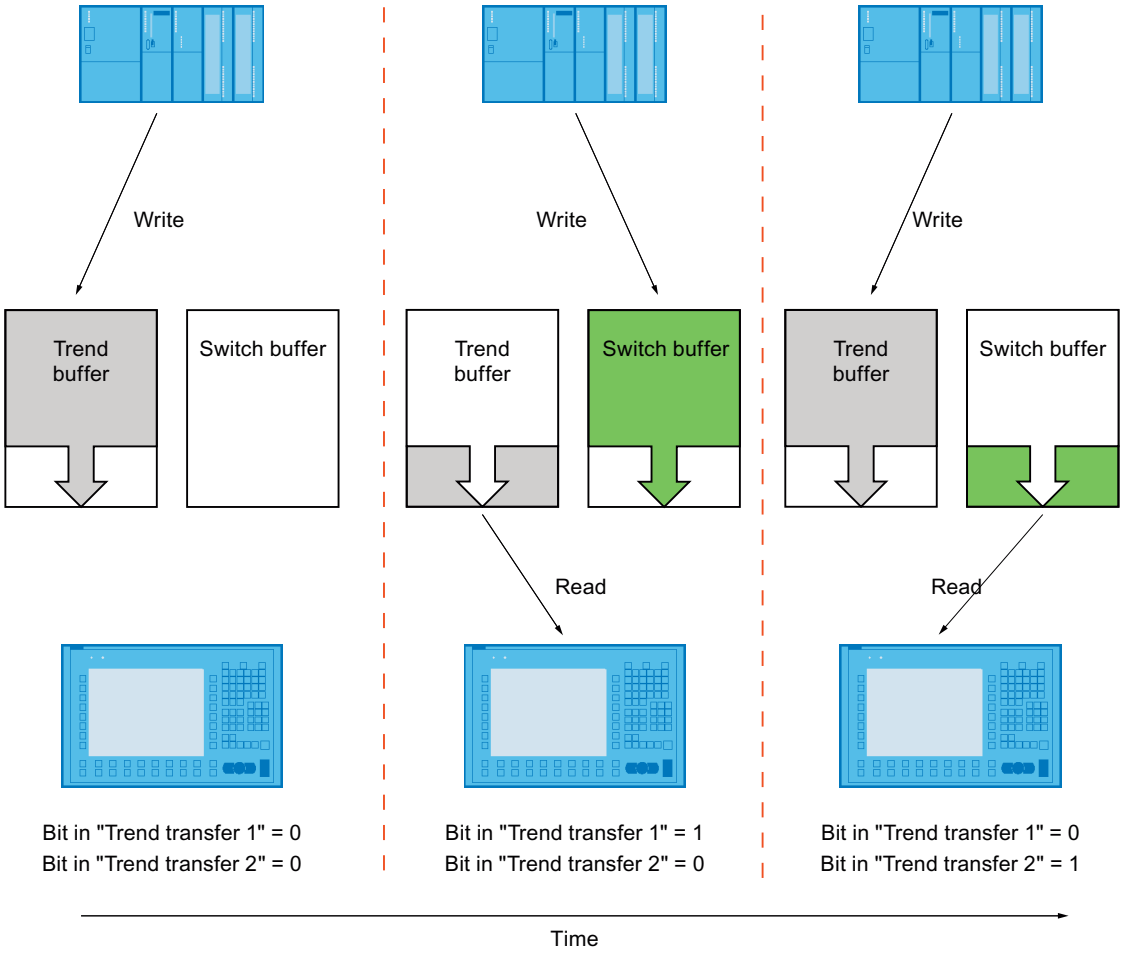
When you set buffered data acquisition, the values to be displayed are buffered in the PLC and read in bit-triggered as a block. These trends are suitable for displaying rapid changes when the course of the trend as a whole is interesting and not so much the individual values.

You configure a switch buffer in the PLC so it can continue to write the new values while the trend buffer is being read. The switch buffer ensures that the PLC does not overwrite values while the operator devices reads the values for the trend.

The switch between the trend buffer and the switch buffer functions as follows:

Whenever the bit assigned to the trend is set in the "Trend transfer 1" tag, all the values are read simultaneously from the trend buffer and are displayed as a trend on the HMI device. The bit in "Trend Transfer 1" is reset after reading has been completed.

While the operator device is reading the tag values from the trend buffer, the PLC writes the new tag values into the switch buffer. When the bit which is assigned to the trend is set in the "Trend transfer 2" tag , all the trend values are read from the switch buffer and displayed at the operator device. While the operating device is reading the switch buffer, the PLC writes again to the trend buffer.



Outputting tag values in screens

Introduction

In runtime you can output tag values in the screens of the operator device in the form of a trend. The process values are loaded by the PLC from the current process or from a log database.

Displayed values

You will need to configure a trend view in a screen so that tag values are displayed on the HMI device. When configuring the trend view, specify which tag values are to be displayed.

The following options are available:

- Current values from the PLC
  Writing of the trend is continued with individual values from the PLC. The following trigger types are available for display in real time:
  - Bit-triggered real time: The time of the update can be controlled by setting a bit. Bit-triggered trends are normally used to visualize rapidly changing values. One example might be the injection pressure in the production of plastic parts.
  - Cyclical real time: You control the update time with a cycle. Cyclically-triggered trends are suitable for representing continuous curves, such as the changes in the operating temperature of a motor.
  - Bit-triggered buffer: The trend contains values that were stored in a buffer in the period between two reads from the PLC. Use this format pattern if the course of the trend as a whole is more interesting than the individual values.
- Logged tag values
  In runtime, the trend view displays the values of a tag from a data log. The following trigger type is available to display logged tag values:
  - Data log: The trend shows the logged values in a particular window in time. To obtain the requested information from the archive, the operator uses the corresponding operator controls in the screen or in the trend view.

Configuring trend displays for values from the PLC

Introduction

You use a trend view to graphically represent values that a tag assumes during the process.

Requirement

- A screen is open.

Procedure

To configure a trend view, follow these steps:

1. Add the "Trend view" object from the toolbox in the "Controls" group to the screen.
2. Select "Properties . Properties > Trend" in the Inspector window and double-click "Add" in the "Name" field. A new trend is created.
3. If required, enter a unique name for the trend in the "Name" field.
4. In the "Style" column, use the selection button to open the "Style" dialog and select the style of the line.
5. Select the number of trend values in the "Trend values" column.

6. In the "Trend type" column, select the required trend type. The following entries are available:
   - "Bit-triggered real time"
   - "Cyclical real time"
   - "Bit-triggered buffer"
   - "Data log"
7. In the "Settings source" column, use the selection button to open the "Data source" dialog and select the tags to supply the trend with values.
   If you selected the "Cyclical real time" trend type, specify a cycle for reading the tags from the PLC.
   Only for bit-triggered trends:
   - Under "Bit", specify which bit of the "Trend request" tag and of the "Trend transfer" tags are to be assigned to the trend. Select a different value for each trend.
   - Enter a tag under "Trend request" which contains at least as many bits as trends are to be displayed in the trend view. The status of the bits of these tags specifies which trend is currently being displayed at the operator device.

   Only for the trend type "Bit-triggered real time":
   - Enter a tag under "Trend transfer" which contains at least as many bits as trends are to be displayed in the trend view. The status of the bits of these tags specifies for which trend values are to be read from the PLC.

   Only for the trend type "Bit-triggered buffer":
   - Enter a buffer tag for the switch buffer. The tags have to be of the same type and of the same length as the trend tag.
   - Enter a tag each under "Trend transfer 1" and "Trend transfer 2" which contains at least as many bits as trends are to be displayed in the trend view. The status of these tag bits specifies the trend for which values will be read from the PLC. "Trend transfer 1" controls reading from the trend buffer; "Trend transfer 2" controls reading from the switch buffer.
8. You can make other settings in the dialogs of the Inspector window. For example, you can select the "Display table" option in the "Table" category to display a value table beneath the trend view.

Note

If you hold down the <CTRL> key and double-click the trend view, the trend view is activated. You can set the column width and the position of the columns in active mode in the values table. In order to activate the trend view the zoom factor has to be set to 100 %.

Result

In runtime, the values of the selected tags are displayed in the configured trend view.

Configuring a trend view for a data log

Introduction

You can use a trend view to display logged tag values in Runtime. The trend view shows the logged tag values in a particular window in time. To obtain the requested information from the archive, the operator uses the corresponding operator controls in the screen or in the trend view.

Application example

At the end of a shift, for example, an operator wants information about the process carried out during the shift.

Requirement

- A data log has been created.
- A screen is open.
- The Inspector window with the trend view properties is open.

Procedure

To configure a trend view to display values from a data log, follow these steps:

1. Add the "Trend view" object from the toolbox in the "Controls" group to the screen.
2. Select "Properties . Properties > Trend" in the Inspector window and double-click "Add" in the "Name" field. A new trend is created.
3. If required, enter a unique name for the trend in the "Name" field.
4. In the "Style" column, use the selection button to open the "Style" dialog and select the style of the line.
5. In the "Trend type" column, select the trend type "Data log".
6. In the "Settings" column, use the selection button to open the "Data source" dialog; select the data log and the tags to supply the trend with values.
7. You can make other settings in the dialogs of the Inspector window. For example, you can select the "Display table" option in the "Table" category to display a value table beneath the trend view.

Note

If you hold down the <CTRL> key and double-click the trend view, the trend view is activated. You can set the column width and the position of the columns in active mode in the values table. In order to activate the trend view the zoom factor has to be set to 100 %.

Result

In runtime, the logged values of the selected tags are displayed in the configured trend view.

The structure of a *.CSV file with tags

Introduction

In the *.csv (comma-separated value) file format, the "Name" and "Value" table columns of the entry are separated by semicolons. Each table row ends with a line break.

Example of a *.CSV file

The example shows a file with logged tag values:

"VarName";"TimeString";"VarValue";"Validity";"Time_ms"

"Var_107";"01.04.98 11:02:52";66,00;1;35886460322,81

"Var_108";"01.04.98 11:02:55 AM";60.00;1;35886460358.73

"Var_109";"01.04.98 11:02:57 AM";59.00;1;35886460381.22

Structure of a log file in *.csv format

The following values are entered in the individual columns of a log file:

Parameters	Description
VarName	Name of the tag from WinCC
Time string	Time stamp as a STRING, e.g. readable data format
VarValue	Value of the tag
Validity	Validity: 1 = value is valid 0 = an error occurred (e.g. interrupted connection)
Time_ms	Specify a time stamp as a decimal value (see below for conversion). Only needed to display the tag values in a trend.

Conversion of the time stamp decimal value

If the value needs to be processed using a different program, proceed as follows:

1. Divide Time_ms by 1,000,000.
   Example: : 36343476928:1 000 000 = 36343,476928
2. The whole number portion (36344) is the date calculated from 31.12.1899.
   Example: 36343 results in 02.07.1999
   You can now convert the time stamp value to days in Excel by assigning a corresponding format from the "Date" group to the cells, which contain the time stamp.
   Result: 37986 results in 31.12.2003
3. The value after the comma (0,476928) indicates the time:
   - Multiply the value (0.476928) by 24 to obtain the hours (11.446272).
   - Multiply the remainder (0.446272) by 60 results in the minutes (26.77632).
   - Multiply the remainder (0.77632) by 60 results in the seconds (46.5792).

   Result 11:26:46.579
   This conversion is supported by Microsoft Excel, for example.

Accessing the ODBC log database directly

Introduction

The storage location of a log can be a database or a file.

The database is addressed by means of its "Data source name" (DSN). Select the database you want to use in WinCC from the Windows Start menu under "Settings > Control panel > ODBC Data Sources".

To store log data, specify the DSN, rather than a directory name during configuration. With the DSN, you are referencing the database and the storage location.

Application

The entire functional scope of the database is available for additional processing and evaluation of log data.

Principle

The data source establishes the connection to the database. You create the data source on the same computer that contains the Runtime software. Then enter the DSN configured here when you create a log in WinCC.

The ODBC interface allows you to use other programs, such as MS Access or MS SQL Server, to access the database directly.

With the "StartProgram" system function, you can also configure a program call (to MS Access, for example) on the HMI device. This does not interrupt Runtime.

12.3 Working with alarms

12.3.1 Basics

12.3.1.1 Alarm Logging in WinCC

Introduction

The alarm system allows you to display and record operating states and faults on the HMI device that are present or occur in a plant.

An alarm may have the following content:

No.	Time	Date	Alarm text	Status	Alarm class
5	12:50:24:590	24.02.2007	Boiler pressure above high limit.	Incoming Outgoing	Warning: Color Red

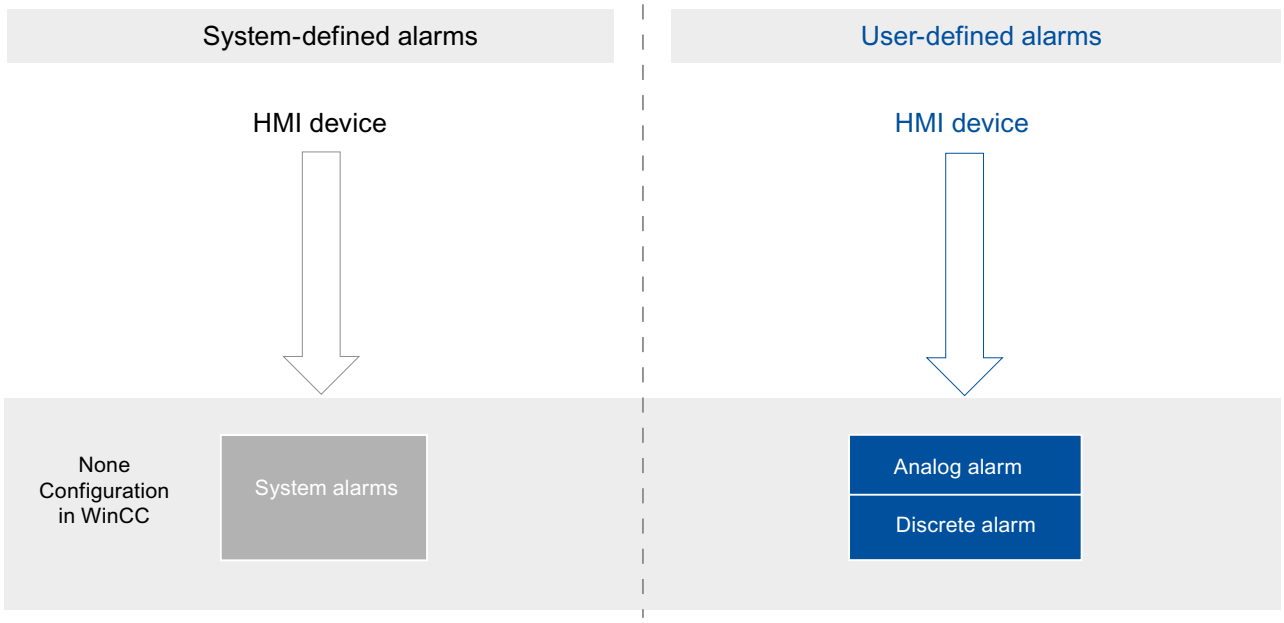
Alarm system in WinCC

The alarm system processes a variety of alarm types. The alarm procedures can be broken down into system-defined alarms and user-defined alarms:

- User-defined alarms are used to monitor the plant process.
- System-defined alarms monitor the HMI device.

The detected alarm events are displayed on the HMI device. Targeted access to the alarms combined with supplementary information about individual alarms ensures that faults are localized and cleared quickly. This reduces stoppages or even prevents them altogether.

The following figure shows the alarm system structure:



12.3.1.2 Alarm Logging in WinCC

Introduction

Alarm logging allows you to display and record operating states and faults on the HMI device that are present or occur in a plant.

An alarm may have the following content:

No.	Time of day	Date	Alarm text	Status	Alarm class
5	12:50:24 :590	24.02. 2007	Boiler pressure above high limit.	Incoming Outgoing	Warning: Color Red

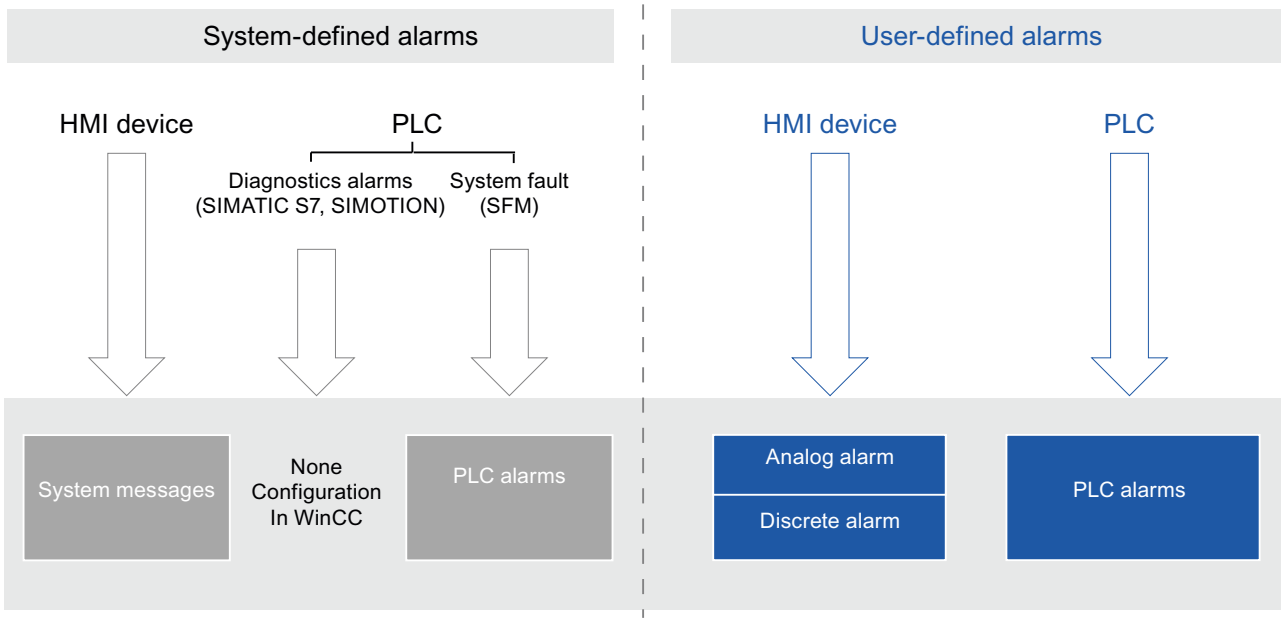
Alarm Logging in WinCC

Alarm logging processes various alarm procedures used by the PLC and the HMI device. The alarm procedures can be broken down into system-defined alarms and user-defined alarms:

- User-defined alarms serve to monitor the plant.
- System-defined alarms are used to monitor the HMI device or the PLC.

The detected alarm events are displayed on the HMI device. You can use the alarm logging system to log alarms from the ongoing process. Targeted access to the alarms combined with supplementary information about individual alarms ensures that faults are localized and cleared quickly. This reduces stoppages or even prevents them altogether.

The following figure shows the alarm logging structure:



12.3.1.3 Alarm Procedures

Overview of the alarm types

Introduction

The alarm types serve various purposes for monitoring the plant. The alarms from the individual alarm types are configured and triggered in different ways.

Select the relevant tab in the "HMI alarms" editor to configure alarms based on the individual alarm types.

Alarm types in WinCC

WinCC supports the following alarm types:

User-defined alarms

- **Analog alarms**
  - Analog alarms are used to monitor limit violations.
- **Discrete alarms**
  - Discrete alarms are used to monitor states.

System-defined alarms

- **System events**
  - System events belong to the HMI device and are imported into the project.
  - System events monitor the HMI device.

Overview of the alarm types

Introduction

The alarm types serve various purposes for monitoring the plant. The alarms from the individual alarm types are configured and triggered in different ways.

Select the relevant tab in the "HMI alarms" editor to configure alarms based on the individual alarm types.

Alarm types in WinCC

WinCC supports the following alarm types:

User-defined alarms

- **Analog alarms**
  - Analog alarms are used to monitor limit violations.
- **Discrete alarms**
  - Discrete alarms are used to monitor states.
- **Controller alarms**
  - You configure controller alarms in STEP 7.
  - You continue to process the controller alarms in WinCC.

  Note
  Device dependency
  The controller alarm is not supported on all HMI devices.

System-defined alarms

- **System-defined controller alarms**
  - System-defined controller alarms consist of diagnostic alarms (SIMATIC S7) and system errors (SFM)
  - System-defined controller alarms are used to monitor the PLC.

  Note
  Device dependency
  System-defined controller alarms are not supported on all HMI devices.
- **System events**
  - System events belong to the HMI device and are imported into the project.
  - System events monitor the HMI device.

User-defined alarms

Analog alarms

Description

Analog alarms signal limit violations during the process.

Example

The speed of the mixer in a fruit juice mixing plant must not be too high or too low. You can configure analog alarms to monitor the speed of the mixer. If the high or low limit for the speed of the mixer is violated, an alarm is output on the HMI device containing the following alarm text, for example: "Mixer speed is too low".

Discrete alarms

Description

Discrete alarms indicate a status in the current process.

Example

A fruit juice mixing plant consists of several tanks containing the ingredients. To ensure the correct mixing ratio of water, fruit concentrate, sugar, and flavoring, the valves in the intakes open and close at the right moment. This operation should be monitored.

You configure a suitable discrete alarm for all the valve states. If a valve on one of the four tanks opens or closes, an alarm is displayed, such as "Water valve closed".

The operator can thus monitor whether the plant is producing correctly.

PLC alarms

Example of an alarm

CPU operating mode switch to Stop

Description

The custom controller alarms are created by the control system engineer in STEP 7. The status values, such as time stamp and process values, are mapped in the controller alarm. If controller alarms are configured in STEP 7, accept them into the integrated WinCC operation as soon as a connection is established to the PLC.

In STEP 7, the controller alarm is assigned to an alarm class. You can import this alarm class with the controller alarm as a common alarm class.

Types of controller alarms

SIMATIC S7 supports different types of controller alarms. WinCC supports different types of controller alarms according to which Runtime is used.

Controller alarms for multiple HMI devices

If a PLC is connected to multiple HMI devices, the project engineer assigns display classes to the controller alarms in STEP7. The display classes determine the allocation to the HMI device. You can activate the display classes for your HMI device that are to be displayed on it. In this case, only the controller alarms from this display class will be displayed on the HMI device. Up to 17 display classes are possible.

See also

Editing controller alarms (Page 4310)

System-defined Alarms

System events

Examples for alarms

- "An online connection to the PLC is established."

Description

A system alarm indicates the status of the system, plus communication errors between the HMI device and system.

Under "Runtime settings > Alarms" you specify how long a system alarm is shown on the HMI device.

Support

The reference contains a list of the possible system events, along with the cause and possible remedies. If you contact online support because of a system alarm on the HMI device, you will need the alarm number and tags used in the system alarm.

System events

Examples of alarms

- "Overflow of the buffer for printing lines in text mode"
- "No printer is installed or a default printer has not been set up."

Description

A system event indicates the system status, including errors in the communication between the HMI device and system.

The system events depend on the HMI device.

You can load system events in the "HMI alarms" editor from a an *.xml file.

Support

The reference contains a list of the possible system events, along with the cause and possible remedies. When contacting Online Support because of a system event on the HMI device, you need the event number and tags used in the system event.

System-defined controller alarms

Example of an alarm

Valve open due to overpressure

Description

System-defined controller alarms are installed with STEP 7 and are only available if WinCC is operated in the STEP 7 environment.

WinCC processes system-defined controller alarms of the type "SIMATIC S7 diagnostic alarm".

S7 diagnostic alarms show states and events in the SIMATIC S7 controllers. You only have read access to S7 diagnostic alarms. They are assigned to the "Diagnostics" alarm class by the system. S7 diagnostic alarms are not acknowledged or reported. They are for signaling purposes only.

See also

Enabling system-defined controller alarms (Page 4311)

12.3.1.4 Alarm states

Introduction

An alarm assumes various alarm states in Runtime. The user analyzes and reports on the process execution with reference to the alarm states.

Note

Device dependency

Reporting and logging are not available for all HMI devices.

Description

Every alarm has an alarm status. The alarm states are made up of the following events:

- **Incoming**
  The condition for triggering an alarm is satisfied. The alarm is displayed, such as "Boiler pressure too high".
- **Outgoing**
  The condition for triggering an alarm is no longer satisfied. The alarm is no longer displayed as the boiler was vented.
- **Acknowledge**
  The operator acknowledges the alarm.

Alarms without acknowledgment

The following table shows the alarm states for alarms that do not have to be acknowledged:

Status	Description
Incoming	The condition for an alarm is satisfied.
Outgoing	The condition for an alarm is no longer satisfied.

Alarms with acknowledgment

The following table shows the alarm states for alarms that have to be acknowledged:

Status	Description
Incoming	The condition for an alarm is satisfied
Outgoing, not acknowledged	The condition for an alarm is no longer satisfied. The operator has not acknowledged the alarm.
Outgoing, and subsequently acknowledged	The condition for an alarm is no longer satisfied. The operator has acknowledged the alarm after this time.
Incoming, acknowledged	The condition for an alarm is satisfied. The operator has acknowledged the alarm.
Outgoing, but acknowledged first	The condition for an alarm is no longer satisfied. The operator acknowledged the alarm while the condition was still satisfied.

Each occurrence of these states can be displayed and logged on the HMI device and a protocol printed.

Note

The display text for the states of an alarm is language-specific and configuration-specific.

12.3.1.5 Alarm classes

Basics on alarm classes

Introduction

Many alarms occur in a plant. These are all of different importance. You can assign the alarms of your project to alarm classes to clearly show the user which of the alarms are most important.

Description

The alarm class defines how an alarm is displayed. The alarm class specifies if and how the user has to acknowledge alarms of this alarm class.

A new alarm class with mandatory acknowledgment is generated in WinCC.

Note

The choice of display modes for alarm classes depends on the options on your HMI device.

Examples of how to use alarm classes

- The alarm "Speed of fan 1 in upper tolerance range" has alarm class "Warnings". The alarm is displayed with a white background. The alarm does not have to be acknowledged.
- The alarm "Speed of fan 2 has exceeded upper warning range" is assigned to the "Errors" alarm class. The alarm is displayed with a red background and flashes at high frequency in runtime. The alarm is displayed until the user acknowledges it.

Using alarm classes

Use the following alarm classes to define the state machines and display of alarms for your project:

- Predefined alarm classes
  You cannot delete predefined alarm classes and edit them only to a limited extent. Predefined alarm classes have been created for each HMI device under "HMI alarms > Alarm classes".
- Custom alarm classes
  You can create new alarm classes under "HMI alarms > Alarm classes", configure how you want the alarms to be displayed, and define an acknowledgment model for alarms of this alarm class. The possible number of custom alarm classes depends on which runtime is used in your project.

See also

Predefined alarm classes (Page 4288)

Creating alarm classes (Page 4297)

Using project-wide alarm classes (Page 4298)

Basics on alarm classes

Introduction

Many alarms occur in a plant. These are all of different importance. You can assign the alarms of your project to alarm classes to clearly show the user which of the alarms are most important.

Description

The alarm class defines how an alarm is displayed. The alarm class specifies if and how the user has to acknowledge alarms of this alarm class.

A new alarm class with mandatory acknowledgment is generated in WinCC.

Note

The choice of display modes for alarm classes depends on the options on your HMI device.

Examples of how to use alarm classes

- The alarm "Speed of fan 1 in upper tolerance range" has alarm class "Warnings". The alarm is displayed with a white background. The alarm does not have to be acknowledged.
- The alarm "Speed of fan 2 has exceeded upper warning range" is assigned to the "Errors" alarm class. The alarm is displayed with a red background and flashes at high frequency in runtime. The alarm is displayed until the user acknowledges it.

Using alarm classes

Use the following alarm classes to define the state machines and display of alarms for your project:

- Predefined alarm classes
  You cannot delete predefined alarm classes and edit them only to a limited extent. Predefined alarm classes have been created for each HMI device under "HMI alarms > Alarm classes".
- Custom alarm classes
  You can create new alarm classes under "HMI alarms > Alarm classes", configure how you want the alarms to be displayed, and define an acknowledgment model for alarms of this alarm class. The possible number of custom alarm classes depends on which runtime is used in your project.
- Common alarm classes
  Common alarm classes are displayed under "Shared data > Alarm classes" in the project tree and can be used for the alarms of an HMI device. Common alarm classes originate in the alarm configuration of STEP 7. If needed, you can create additional common alarm classes in WinCC.

See also

Predefined alarm classes (Page 4288)

Creating alarm classes (Page 4297)

Using project-wide alarm classes (Page 4298)

Predefined alarm classes

Predefined alarm classes

The following alarm classes already created in WinCC for every HMI device:

Alarm classes for user-defined alarms

- "Warnings"
  The "Warnings" alarm class is designed to show irregular statuses and routines in the process. Users do not acknowledge alarms from this alarm class.
- "Errors"
  The "Errors" alarm class is intended to show critical or dangerous states or limit violations in the process. The user must acknowledge alarms from this alarm class.

Alarm classes for system-defined alarms

- "System"
  The "System" alarm class contains alarms that display states of the HMI device and the PLCs. Alarms of the "System" alarm class belong to the system alarms.
- "Diagnosis Events"
  The "Diagnosis Events" alarm class contains alarms that display states and events in SIMATIC S7 controllers. Users do not acknowledge alarms from this alarm class.

  Note

  Device dependency

  The "Diagnosis Events" alarm class is not available for all HMI devices.

- "Safety Warnings"
  The "Safety Warnings" alarm class contains alarms for fail-safe operation. Users do not acknowledge alarms from this alarm class. Alarms of the "Safety Warnings" alarm class belong to the system alarms.

  Note

  Device dependency

  The "Safety Warnings" alarm class is not available for all HMI devices.

See also

Basics on alarm classes (Page 4287)

Creating alarm classes (Page 4297)

Using project-wide alarm classes (Page 4298)

12.3.1.6 Acknowledgement

Acknowledging alarms

Introduction

To make sure that an alarm was registered by the plant operator, configure this alarm so that it is displayed until acknowledged by the operator. Alarms that display critical or hazardous states in the process have to be acknowledged.

Description

The acknowledgment of an alarm is an event that is logged and reported. Acknowledging an alarm changes the alarm status from "Incoming" to "Acknowledged". When the operator acknowledges an alarm, the operator confirms that he or she has processed the status that triggered the alarm.

Note

HMI device dependency

Reporting and logging are not available for all HMI devices.

Triggering acknowledgment of an alarm

In Runtime, you trigger alarm acknowledgments in various ways:

- Acknowledgement by the authorized user at the HMI device
- Automatic acknowledgment by the system without operator action, e.g. by means of
  - Tags
  - PLC
  - System functions in function lists or scripts

  Note

  HMI device dependency

  Scripts are not available for all HMI devices.

Acknowledging alarms that belong together

To make the alarm system clearer and easier to use in Runtime, you can configure an alarm group. You can acknowledge all alarms belonging to this alarm group in a single pass.

Acknowledgment by the PLC

Discrete alarms will be automatically acknowledged by the PLC, if necessary. The acknowledgment is triggered by a bit in the "Acknowledgment tag PLC". You define the bit and tag at the configuration stage.

Acknowledgment of an alarm on the HMI device

In Runtime, the user acknowledges an alarm in one of the following ways, depending on the configuration:

- Using the acknowledgment button <ACK> on the HMI device
- Using the button in the alarm view
- Using configured function keys or buttons in screens

Note

Acknowledgment button <ACK> on the HMI device

To ensure that critical alarms are processed only by authorized users, protect the "ACK" button on the HMI devices, including the operating controls and display objects of the alarms. Use the appropriate operator authorization for this.

Note

HMI device dependency

The acknowledgement key <ACK> is not available on all HMI devices.

Reaction to the acknowledgment of alarms of an alarm group

- Alarm acknowledgment by the operator
  You acknowledge the alarm of an alarm group, for example, using the <ACK> acknowledgment key or a function key. All alarms of the alarm group are acknowledged.
- Alarm acknowledgment by the controller
  The controller sets a tag bit to acknowledge the alarm of an alarm group.
  The respective alarm is acknowledged.

Acknowledgement Model

Overview

You define the state machines for an alarm class. Alarms that are assigned to this alarm class will be acknowledged on the basis of these state machines. The following state machines are used in WinCC:

- Alarm without acknowledgment
  This alarm comes and goes without having to be acknowledged. There is no visible response from the system.
- Alarm with simple acknowledgment
  This alarm must be acknowledged as soon as the event that triggers the alarm occurs. The alarm remains pending until it is acknowledged.

12.3.1.7 Alarm groups

Introduction

Many alarms from different areas and processes occur in a plant. You can compile associated alarms into alarm groups.

Alarm groups

You can use the alarm groups to monitor the parts of the plant and to acknowledge the associated alarms together as required.

Alarm groups can contain alarms from different alarm classes. You only assign alarms that require acknowledgment to alarm groups.

Using alarm groups

It is a good idea to compile alarm groups for alarms such as the following:

- Alarms that are caused by the same fault.
- Alarms of the same type
- Alarms from a machine unit, such as "Fault in drive XY"
- Alarms from an associated part of the process, such as "Fault in cooling water supply"

Display in Runtime

In Runtime, the "Alarm group" column displays the number of the alarm group to which the alarm belongs.

See also

Configuring Alarm Groups (Page 4300)

12.3.1.8 Alarm number

Assigning alarm numbers

The system assigns unique alarm numbers within an alarm type.

Note

When adapting alarm numbers, observe the uniqueness of the alarm number within an alarm type.

12.3.2 Working with Alarms

12.3.2.1 Alarm components and properties

Overview

You configure the components of alarms in WinCC. The following table shows the basic components of alarms:

Alarm class	Alarm number	Time of day	Date	Alarm status	Alarm text	Alarm group	Tooltip	Trigger tag	Limit
Warn-ing	1	11:09 :14	06.08.2 007	IO	Maximum speed reached	2	This alarm is ...	speed_1	27
Sys-tem	110001	11:25 :58	06.08.2 007	I	Switch to "Online" mode	0	This alarm is ...	PLC-Varia-ble_1	–

Alarm class

Alarm classes, such as "Warnings" or "Errors". The alarm class defines the following for an alarm:

- State machine
- Appearance in Runtime (e.g. color)
- Logging

Note

Device dependency

Logging is not available for all HMI devices.

Alarm number

An alarm is identified by a unique alarm number. The alarm number is assigned by the system. You can change the alarm number to a sequential alarm number, if necessary, to identify alarms associated in your project.

Time and date

Every alarm has a time stamp that shows the time and date at which the alarm was triggered.

Alarm status

An alarm has the events "Incoming," "Outgoing," "Acknowledge." For each event, a new alarm is output with the current status of the alarm.

Alarm text

The alarm text describes the cause of the alarm.

The alarm text can contain output fields for current values. The values you can insert depend on the Runtime in use. The value is retained at the time at which the alarm status changes.

Alarm group

The alarm group bundles individual alarms.

Tooltip

You can configure a separate tooltip for each alarm; the user can display this tooltip in Runtime.

Trigger tag

Each alarm is assigned a tag as trigger. The alarm is output when this trigger tag meets the defined condition, e.g. when its state changes or it exceeds a limit.

Limit

Analog alarms indicate limit violations. Depending on the configuration, WinCC outputs the analog alarm as soon as the trigger tag exceeds or undershoots the limit value.

See also

Output of a tag value in the alarm text (Page 4307)

Alarm states (Page 4285)

Basics on alarm classes (Page 4287)

Alarm groups (Page 4292)

12.3.2.2 Configuring Alarms

Overview of alarm configuration tasks

Steps to configure alarms

Configuring alarms in WinCC involves the following steps:

1. Edit and create alarm classes
   You use the alarm class to define how an alarm will be displayed in runtime and to define the state machines for it.
2. Creating tags in the "HMI tags" editor
   - Configure the tags for your project.
   - You create range values for the tags.
3. Creating tags in the "HMI alarms " editor
   - Create custom alarms and assign these the tag to be monitored, alarm classes, alarm groups, and other properties.
   - You can also assign system functions or scripts to the alarm events.
4. Output of configured alarms
   To output configured alarms, configure an alarm view or an alarm window in the "Screens" editor.

Additional configuration tasks

Additional tasks may be necessary for configuring alarms, depending on the requirements of your project:

1. Creating alarm groups
   You assign the alarms of your project to alarm groups according to their association, such as by the cause of the problem (power failure) or source of the error (Motor 1).
2. Configuring Loop-In-Alarm
   A Loop-In-Alarm is configured in order to change to a screen containing relevant information on an alarm received.

Overview of alarm configuration tasks

Steps to configure alarms

Configuring alarms in WinCC involves the following steps:

1. Edit and create alarm classes
   You use the alarm class to define how an alarm will be displayed in runtime and to define the state machines for it.
2. Creating tags in the "HMI tags" editor
   - Configure the tags for your project.
   - You create range values for the tags.
1. Creating tags in the "HMI alarms " editor
   - Create custom alarms and assign these the tag to be monitored, alarm classes, alarm groups, and other properties.
   - You can also assign system functions or scripts to the alarm events.
2. Output of configured alarms
   To output configured alarms, configure an alarm view or an alarm window in the "Screens" editor.

Additional configuration tasks

Additional tasks may be necessary for configuring alarms, depending on the requirements of your project:

- Activating and editing system events
  You can import system events when you initially open the "System events" tab in the "HMI alarms" editor. On completion of the import, you can edit the system events.
- Activating and editing controller alarms
  For integrated operation of a project in STEP 7, specify the controller alarms to be displayed on your HMI device in the alarm settings.
- Creating alarm groups
  Assign the alarms of your project to alarm groups based on their relation, e.g. by error cause (e.g. "power failure"), or by error source (e.g. "Motor 1").
- Configuring Loop-In-Alarm
  Configure a Loop-In-Alarm in order to change to a screen that contains information about an incoming alarm.

Configuring Alarm Classes

Creating alarm class

Introduction

Cre
cla
You

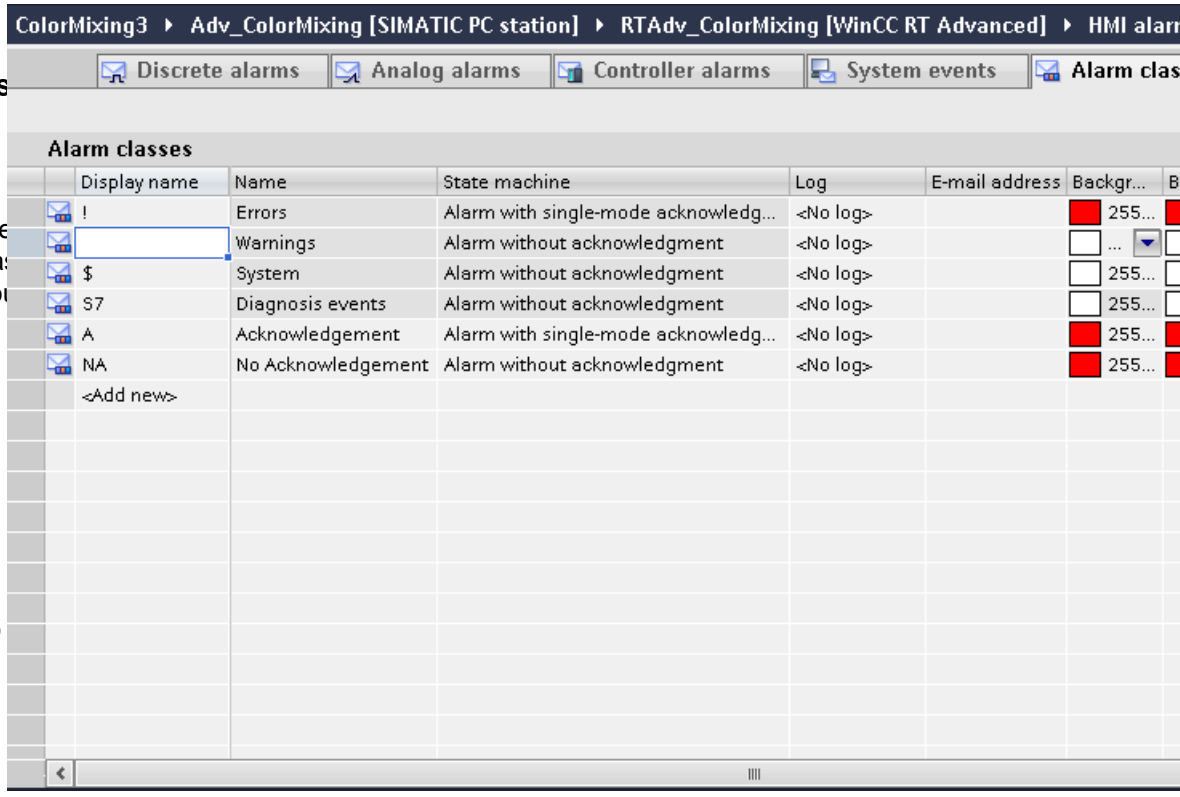
Requirement

-
-

Procedure

To

1.



1. Double-click "<Add>" in the table.
   A new alarm class is created. Each new alarm class is automatically assigned a static ID. The properties of the new alarm class are shown in the Inspector window.
2. Configure the alarm class under "Properties > Properties >General" in the Inspector window.
   - Enter a "Name" and the "Display name".
   - Depending on the HMI device, you can also activate logging, or automatic sending of e-mails.
3. Define the state machines for the alarm class under "Properties > Properties > Acknowledgment" in the Inspector window.
4. Change the default text under "Properties > Properties > Status texts" in the Inspector window.
   This text indicates the status of an alarm in Runtime.
5. Change the default colors under "Properties > Properties > Colors" in the Inspector window. Depending on the HMI device, also change the flashing characteristics.

These settings define how alarms from this alarm class are displayed in Runtime.

Note

To display the alarm classes in color in Runtime, the "Use alarm class colors" option must be activated. In the project navigation, enable "Runtime settings > Alarms > General > Use alarm class colors" accordingly. This option is selected in a new project in WinCC.

See also

Predefined alarm classes (Page 4288)

Basics on alarm classes (Page 4287)

Using project-wide alarm classes

Introduction

Project-wide alarm classes are displayed under "Shared data > Alarm classes" in the project navigation. Create additional common alarm classes in WinCC.

Controller alarms are assigned to common alarm classes. If you work with controller alarms, common alarm classes are automatically loaded in WinCC.

Requirements

- You have created a project.

Creating project-wide alarm class

To create a project-wide alarm class, proceed as follows:

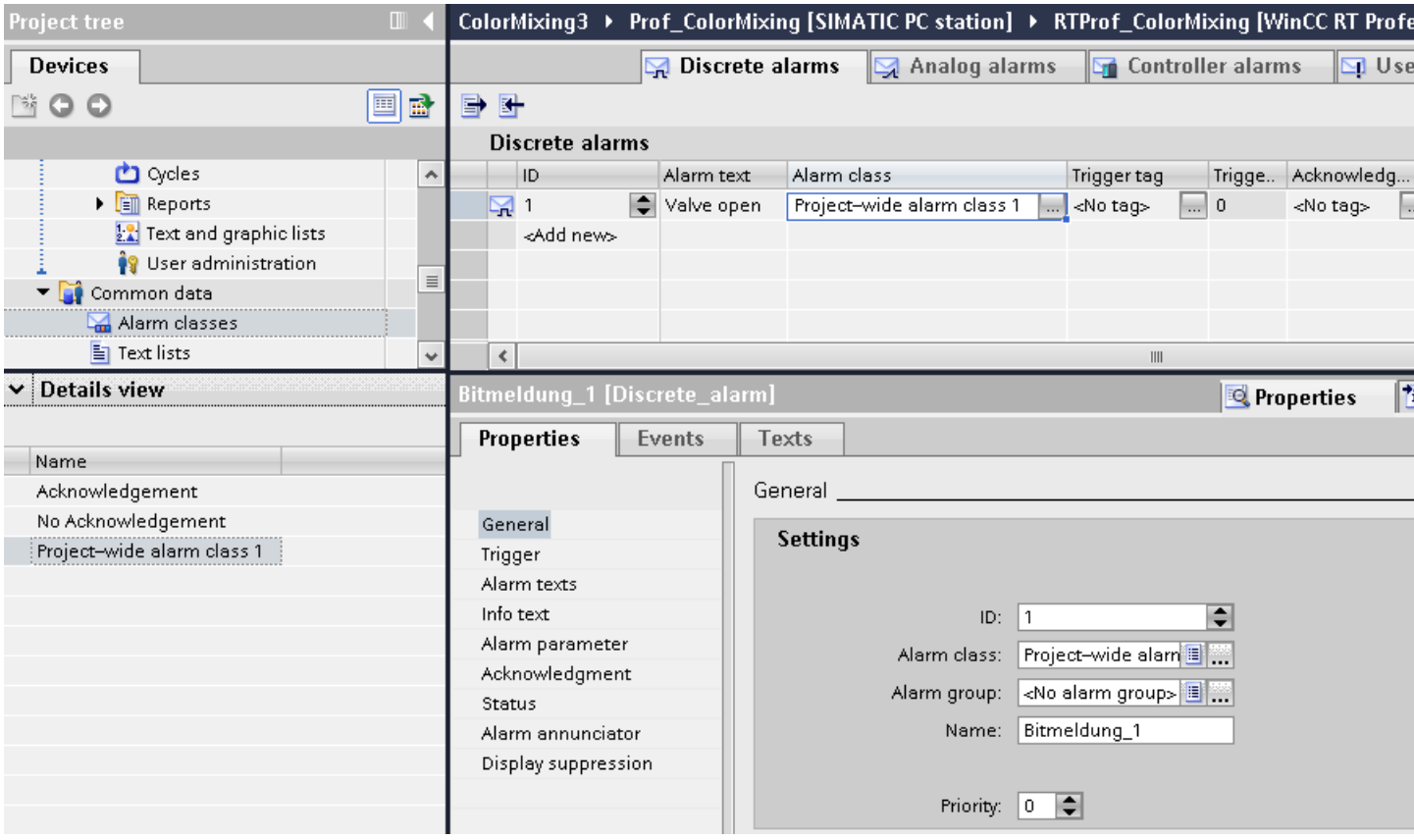
1. Double-click "Shared data > Alarm classes" in the project navigation.
   The "Project-wide message classes" editor opens in the work area.
2. To create a project-wide message class, double-click in the first empty line of the table editor.
3. Specify the name and display name of the alarm class.
4. Select "With acknowledgment" if the alarm class requires acknowledgment.

Assign alarms to a common alarm class

Proceed as follows to assign an analog or discrete alarm to a common alarm class:

1. In the "HMI alarms" editor, select the alarm that you want to assign to the common alarm class.
2. Click "General" in the Inspector window.
3. Click "Shared data > Alarm classes" in the project navigation.
4. Select the common alarm class in the detail view.

5. Drag-and-drop the alarm class to the "Alarm class" selection box or "Alarm class" column in the work area of the Inspector window of the alarm.



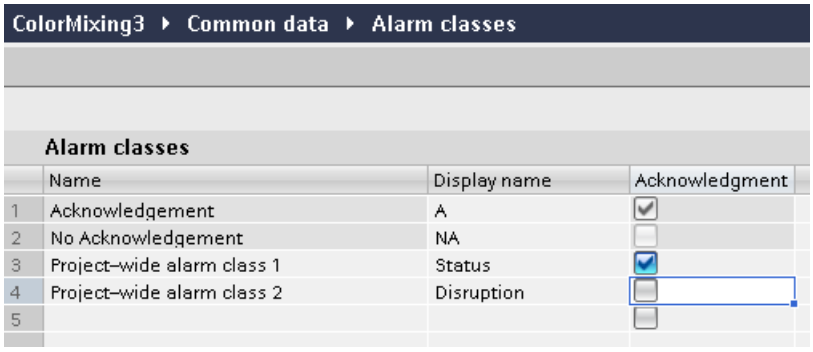
The system creates a new alarm class that is linked to the common alarm class. The new name of the common alarm class is displayed in the "Alarm class" area of the Inspector window of the alarm. The display name of the common alarm class is activated for the alarm class in your project.

6. Click the "Alarm Class" tab.
7. Check to see whether the display name of the common alarm class is displayed in WinCC.
8. You can change the object name of the alarm class.

Alternative procedure

To assign an alarm to a common alarm class, proceed as follows:

1. Click the "Alarm Classes" tab.
   The alarm classes are displayed.
2. Click "Shared data > Alarm classes" in the project navigation. The following table shows the common alarm classes:



3. Select the common alarm class in the detail view.
4. Drag-and-drop the alarm class to the "Alarm classes" tab of the "HMI alarms" editor
   The system creates a new alarm class that is linked to the common alarm class. The new name of the new alarm class in your project is displayed in the "Alarm class" tab.
5. Open the tab with the alarm procedure for the required alarm.
6. Under "General" in the Inspector window, select the alarm and the alarm class linked to the common alarm class.

Note

Deleting a common alarm class

If you delete a common alarm class, the linked alarm class is maintained. If necessary, link the alarm class with another common alarm class or use it as WinCC alarm class.

See also

Predefined alarm classes (Page 4288)

Basics on alarm classes (Page 4287)

Configuring Alarm Groups

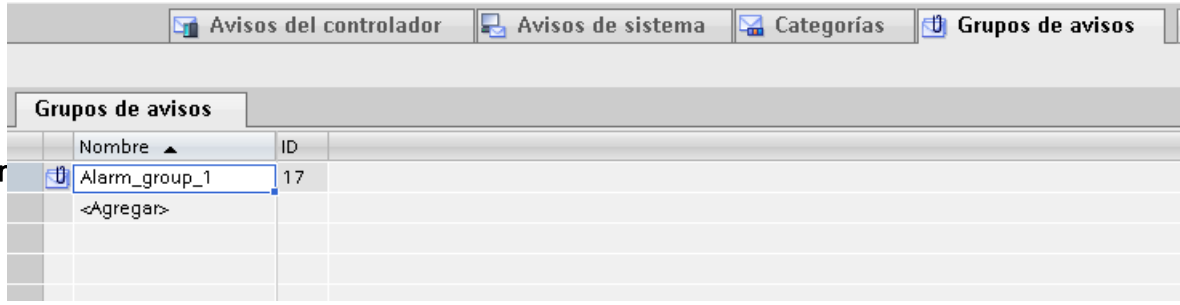
Introduction

Create alarm groups on the "Alarm Groups" tab in the "HMI alarms" editor. An alarm group is a compilation of single alarms. You assign alarms in an alarm group by association, such as cause of the problem or source of the error. If you acknowledge an alarm from this alarm group in Runtime, all other alarms in the alarm group are acknowledged automatically.

Requirement

- You have created a project.
-
-

Creating a new alarm

1. 

2. In the work area of the table, double-click "<Add>" in the first free row.
   A new alarm group is created.
3. You can overwrite the proposed "Name".

Result

An alarm group is created. For the group acknowledgment of alarms in Runtime, assign the associated alarms that require acknowledgment to an alarm group.

Configuring discrete alarms

Introduction

Discrete alarms triggered by the PLC indicate status changes in a plant. They indicate the opened or closed state of a valve, for example.

The following sections describes the configuration procedures in the "HMI alarms" editor. You can also configure discrete alarms in the "HMI tags" editor.

Requirements

- The "HMI alarms" editor is open.
- The Inspector window is open.
- You have created the required alarm classes and alarm groups.

Procedure

To configure a discrete alarm, proceed as follows:

1. Open the "Discrete alarms" tab.
2. To create a new discrete alarm, double-click in the work area on "<Add>".
   A new discrete alarm is created.

3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   - Enter an alarm text as event text.
     Use the functions of the shortcut menu to format the text on a character-by-character basis, or to insert output fields for HMI tags, or texts from the text lists.
   - You can renumber the alarm.
   - Select the alarm class and the alarm group, if necessary.
4. In the Inspector window, select the tag and the bit that triggers the alarm under "Properties > Properties > Trigger". Note the following information:
   - Use the data types "Int" or "UInt" to select an HMI tag.
   - When you select a PLC tag, use the data types "Int" or "Word". The input and output area of a PLC tag is unsuitable as trigger.
   - Use trigger tag bits only for alarms.
   - Do not use trigger tags for anything else.
   - If you want to acknowledge the alarm via the PLC, use this tag also as PLC acknowledgment tag.

Note

Note the method used to count bits in the utilized PLC when specifying the bit. For more information, refer to the "Communication" section in the PLC Online Help.

Note

If the object does not yet exist in the selection list, create it directly in the object list and change its properties later.

Status-dependent alarm texts

To display a different text independent of the alarm status, link a text list to the alarm text. You control the text list with a tag.

Additional settings for discrete alarms

Creating a tooltip

To configure a tooltip for the alarm, follow these steps:

- Enter your text under "Properties > Properties > Tooltip".

Configuring event-driven tasks

To configure event-driven tasks, such as a loop-in alarm, follow these steps:

1. Select the discrete alarm.
2. Select "Properties > Events" in the Inspector window and configure a new function list for the relevant event.

See also

Configuring trigger for alarm acknowledgment (Page 4334)

Output of a tag value in the alarm text (Page 4307)

Formatting alarm text (Page 4306)

Configuring loop-in alarm (Page 4311)

Configuring analog alarms

Introduction

Analog alarms indicate limit violations. For example, if the speed of a motor drops below a certain value, an analog alarm is triggered.

Requirements

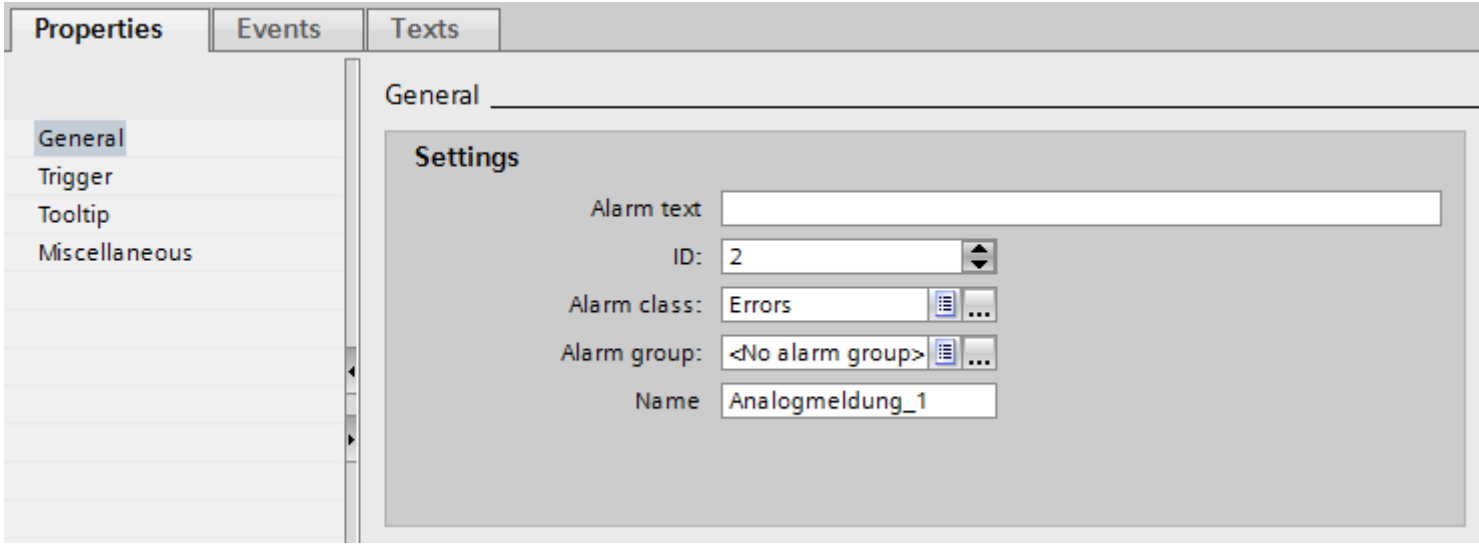
- The "HMI alarms" editor is open.
- The Inspector window is open.
- You have created the required alarm classes and alarm groups.

Procedure

To configure an analog alarm, proceed as follows:

1. Click the "Analog alarms" tab.
2. To create a new analog alarm, double-click in the table on "<Add>".
   A new analog alarm is created.

3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   - Enter an alarm text as event text.
     Format the text character-for-character using the shortcut menu.
     Using the shortcut menu, you can insert output fields for HMI tags, or text from text lists.
   - You can renumber the alarm.
   - Select the alarm class and the alarm group, if necessary.



4. Configure the tag that triggers the alarm under "Properties > Properties > Trigger > Settings".
   Do not use trigger tags for anything else.

Configure limit values for an analog alarm

1. In the Inspector window, click the Const button under "Properties > Properties > Trigger > Limit > Value".
   - To use a constant as limit value, select "Constant".
     Enter the required limit value.
   - To use a tag as limit value, select "HMI tag".
     The ... button appears. Use this button to select the tag you want to use.

   Note

   If the required tag does not exist in the selection list yet, create it in the object list and change its properties later.

2. Select the mode:
   - "Higher": The alarm is triggered when the limit is exceeded.
   - "Lower": The alarm is triggered when the limit is undershot.

Optional settings for analog alarms

Setting the delay time

To set the delay time, proceed as follows:

- Enter a time period in the Inspector window under "Properties > Properties> Trigger > Settings > Delay".
  The alarm is only triggered when the trigger condition is still present after the delay time has elapsed.

Setting the deadband

Note

If a process value fluctuates around the limit, the alarm associated with this fault may be triggered multiple times. To prevent this from happening, configure a deadband or delay time.

To enter the deadband, follow these steps:

1. Under "Properties > Properties> Trigger > Deadband > Mode", select the change in alarm status for which the deadband is to be taken into account.
2. Enter a constant value under "Value".
3. To define the deadband value as a percentage of the limit, set the "in %" check box.

Automatic reporting

To print the alarms continuously:

- Select the "Activate" option under "Properties > Properties > Miscellaneous > Report."

Note

Device dependency

Reporting is not available for all HMI devices.

Creating a tooltip

To configure a tooltip for the alarm, follow these steps:

- Select "Properties > Properties > Tooltip" in the Inspector window and enter your text.

Configuring event-driven tasks

To configure event-driven tasks, such as a loop-in alarm, follow these steps:

1. Select the analog alarm.
2. Select "Properties > Events" in the Inspector window and configure a new function list for the relevant event.

See also

Configuring trigger for alarm acknowledgment (Page 4334)

Output of a tag value in the alarm text (Page 4307)

Formatting alarm text (Page 4306)

Configuring loop-in alarm (Page 4311)

Formatting alarm text

Requirement

- The "HMI alarms" editor is open.
- An alarm has been created.

Procedure

To format an alarm text, proceed as follows:

1. Select the alarm to edit.
2. In the Inspector window, select the characters to format under "Properties > Properties > General > Alarm text".
3. Select the formatting from the shortcut menu, e.g. "Underscored" or "Uppercase".

Note

It is not possible to underline tags and text list entries.

Result

The selected characters are displayed in Runtime with the selected formatting.

Removing format settings

To remove all text formats, proceed as follows:

1. In the Inspector window, select the characters whose formatting you want to remove in the alarm text.
2. Select "Delete formatting characters" from the shortcut menu.

Result

The selected characters are displayed in unformatted notation in Runtime.

Output of a tag value in the alarm text

Introduction

In WinCC, you can insert output fields into the alarm text which display the content of tags.

Requirements

- The "HMI alarms" editor is open.
- The alarm is selected.

Output of a tag value in the alarm text

To insert an output field for a tag value in the alarm text, proceed as follows:

1. Place the cursor onto the required position in the event text.
2. Select "Insert tag output field" in the shortcut menu.
3. Open the object list under "Tag" and select a tag.
   You can also create the tag in the object list.
4. Under "Format", specify the length of the output field and the format for tag value output in the alarm text.
   Configure an output field of sufficient size. Otherwise, the tag content is not output to the full extent in the alarm.
5. Click to save your entries.

WinCC inserts a placeholder for the output field into the alarm text: "<tag: n, [tag name]>" whereby n = text string length.

Editing output field properties

To edit the properties of an output field, proceed as follows:

- Double-click on the output field in the alarm text and edit the settings.

Deleting an output field from the alarm text

To delete an output field from the alarm text, proceed as follows:

- Select the output field in the alarm text and then select the "Delete" command from the shortcut menu.

Note

The sequence of the tag output fields in the alarm text depends on the language. The sequence of the Runtime language is used for logging alarms to a CSV log file.

Changing the tag of an output field in one language causes the modified output field to appear at the end of the alarm text in all other languages. This changes the sequence of the output fields in the log.

Device dependency

Logging is not available for all HMI devices.

Output of texts from a text list in an alarm

Introduction

Text lists are used to create dynamic alarm texts. For example, you can configure the output of both states of a discrete alarm in an alarm text.

Insert a corresponding output field in the alarm text and specify the tag that returns the search key for the text list.

Requirements

- A text list is created.
- The "HMI alarms" editor is open
- The alarm is selected

Output of a text list value in an alarm text

To insert an output field for a value from a text list in the alarm text, proceed as follows:

1. Place the cursor at the required position in the alarm text.
2. Select "Insert text list output field" command from the shortcut menu.
3. Select the text list under "Text list".
4. Open the object list under "Tag" and select the tag that returns the search key for the text list.
   You can also create the tag in the object list.
5. Define the length of the output field under "Format".
   Configure an output field of sufficient size so that the complete text from the text list is displayed in the alarm.
6. Click on the icon to save your entries.
   WinCC inserts a placeholder for the output field into the alarm text:
   ""<textlist: n, [textlist name]>" whereby n = text string length.

Editing output field properties

To edit the properties of an output field, proceed as follows:

- Double-click on the output field in the alarm text and edit the settings.

Deleting text list from the alarm text

To delete an output field from the alarm text, proceed as follows:

- Select the output field in the alarm text and then select the "Delete" command from the shortcut menu.

Editing System Events

Editing system events

When you open the "System events" tab in the "HMI alarms" editor, WinCC prompts you to import or update the system events. Import the system events and specify the display period. You cannot delete or create new system events. You can only edit the alarm text with system events.

Requirements

- You have imported the system events.

Specifying the display period

To determine the display period, proceed as follows:

1. Double-click "Runtime settings" in the project navigation.
2. Enter a value under "Alarms > System events > Display period".
   This value specifies a time in seconds during which the system events are displayed on the HMI device. If you want to display system events permanently, enter "0".

Changing the alarm text

To change the alarm text of a system event, follow these steps:

1. Open the "HMI alarms" editor and then click the "System events" tab.
2. Select the system event that you want to edit.
1. Change the alarm text under "Properties > Properties > Properties > General" in the Inspector window.

Note

If you change an alarm text for a system event that contains a placeholder, leave the number of placeholders unchanged. A placeholder may be a "%1".

Configuring event-driven tasks

The "Incoming" event is available for certain system events, depending on the HMI device. To configure event-driven tasks, proceed as follows:

1. Select the system event.

2. Select "Properties > Properties > Incoming" in the inspector window and configure a function list.

Editing controller alarms

Introduction

Controller alarms are configured in STEP 7. Controller alarms are available in WinCC running in a STEP 7 environment.

In WinCC you only edit the properties of controller alarms that are specific to an HMI device. Different properties of a controller alarm can be edited, depending on the HMI device and PLC.

You can filter the displayed controller alarms for your project using the display classes that were configured on the PLC for controller alarms.

Requirements

- The connection was established to the PLC.
- Alarms were configured in STEP 7.

Editing controller alarms

Proceed as follows to edit controller alarms:

1. Double-click "HMI alarms" in the project navigation.
   The "HMI alarms" editor is opened.
2. Open the "Controller alarms" tab.
3. Select the alarm that you want to change in the work area.
4. Make your changes. Depending on the HMI device, it may not be possible to display or enable some of the controller alarm properties.

Editing SIMATIC S7 alarms

You can edit these controller alarms in the STEP 7 alarm configuration system. To do this, proceed as follows:

1. Select the alarm that you want to change in the work area.
2. Select "Open in alarm editor of the PLC" from the shortcut menu.
   The STEP 7 alarm configuration system is opened.
3. Make your changes.

Filtering the controller alarm using display classes

To filter controller alarms by display classes, proceed as follows:

1. Click "Runtime settings > Alarms" in the project navigation under your HMI device.
   One or several connections to a PLC are shown in "Contoller alarms".
2. Select the display classes whose controller alarms you want to display for the connection.

Result

Only those controller alarms whose display class you have enabled for your HMI device will be displayed on the "Controller alarms" tab of the "HMI alarms" editor.

See also

PLC alarms (Page 4283)

Enabling system-defined controller alarms

Enabling system-defined controller alarms

To activate system-defined controller alarms for your HMI device, proceed as follows:

1. Select "Runtime settings > Alarms" in the project navigation.
   The alarm settings open.
2. Under "System events" activate the system-defined controller alarms.
3. To display the system events with alarm text, activate "With text".
4. To display these alarms in Runtime, configure an alarm view that displays alarms from the "Diagnosis Events" alarm class.

Note

For HMI devices with small display units, specify to display only the alarm number for system-defined controller alarms.

See also

System-defined controller alarms (Page 4285)

Configuring loop-in alarm

Introduction

A Loop-In-Alarm is configured in order to change to a screen containing relevant information on an alarm received.

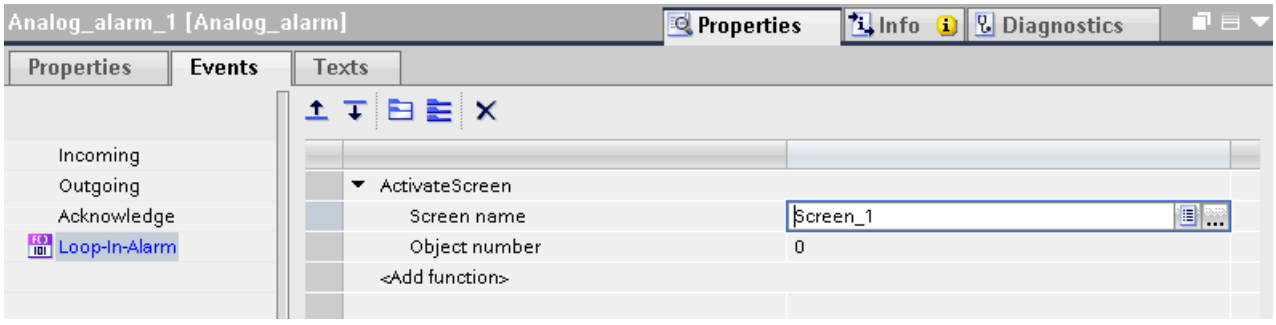
Requirement

- The screen called by the Loop-In-Alarm has been created.
- The "HMI alarms" editor is open.

Procedure

To configure a Loop-In-Alarm for an alarm, proceed as follows:

1. Click the tab that contains the alarm for which you want to configure the Loop-In-Alarm.
2. Select the alarm.
3. In the Inspector window, select "Properties > Events > Loop-In-Alarm".
4. Select the "ActivateScreen" system function.
5. Select the screen called by the Loop-In-Alarm as parameter.



Note

To configure the Loop-In-Alarm for an alarm view with an "alarm line" format, use a button with the following system functions:

- "EditAlarm" for HMI devices with keys
- "AlarmViewEditAlarm" for HMI devices without keys

The system functions trigger the "Loop-In-Alarm" event. The alarm line has no buttons.

Result

If you click on the "Loop-In-Alarm" button of the alarm view in Runtime, a screen is opened with information on the selected alarm.

See also

Configuring discrete alarms (Page 4301)

Configuring analog alarms (Page 4303)

Alarms in the "HMI tags" Editor

Configuring discrete alarms

Introduction

In WinCC, you can create and edit discrete and analog alarms, including the trigger tags, in the "HMI tags" editor.

Note

If you delete, move or copy objects in the "HMI tags" editor, the changes also take effect in the "HMI alarms" editor.

Requirements

The "HMI tags" editor is open.

Procedure

To configure a discrete alarm, proceed as follows:

1. To create a tag, click on "<Add>" in the table at the top of the work area.
   A new tag is created.
2. Configure an internal or external tag as required.
   - Use the data types "Int" or "UInt" to select an HMI tag.
   - When you select a PLC tag, use the data types "Int" or "Word". The input and output area of a PLC tag is unsuitable as trigger.
3. Select the tag at the top of the work area.
4. Click on "<Add>" in the table on the "Discrete alarms" tab at the bottom of the work area.
   A new discrete alarm is created for the tag. If you have selected the incorrect data type, the tag will be marked in the discrete alarm.
5. Configure the discrete alarm in the Inspector window:
   - Enter the alarm text under "Properties > Properties > General > Alarm text".
     You can also insert output fields into the alarm text.
   - Select an alarm class.
   - Select the trigger bit of the tag that triggers the discrete alarm under "Properties > Trigger".
6. You can create additional discrete alarms to monitor the tags.

   Note

   A tag is monitored using only one alarm type. You should therefore create either analog alarms **or** discrete alarms for a tag.

Result

The configured discrete alarms are created in the "HMI tags" editor and displayed in the "HMI alarms" and "HMI tags" editors.

Configuring analog alarms

Introduction

In WinCC, create the discrete and analog alarms, including the trigger tags, in the "HMI tags" editor. You can also edit the alarms as in the "HMI alarms" editor. You can create up to two range values for a tag. You monitor these limits with analog alarms.

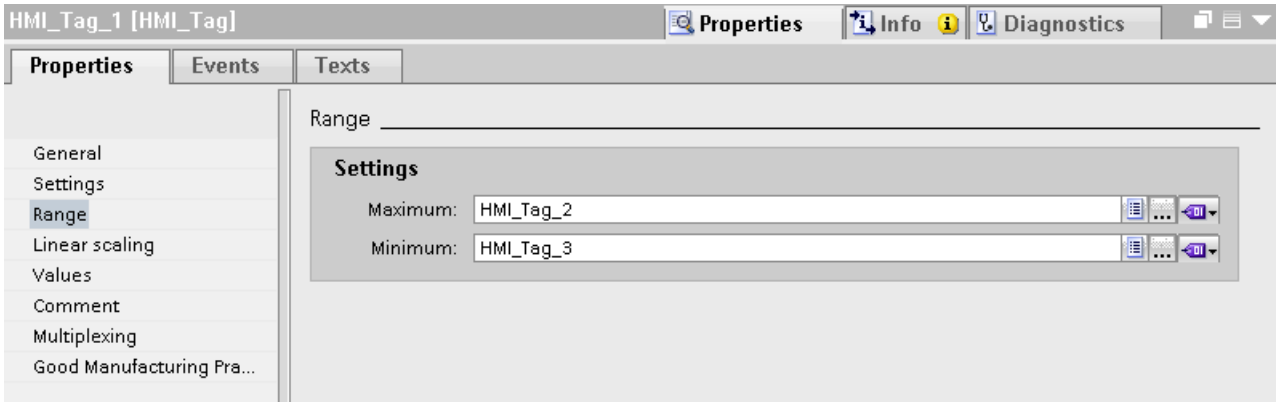
Requirements

The "HMI tags" editor is open.

Procedure

To configure an analog alarm in the "HMI tags" editor, proceed as follows:

1. To create a tag, click on "<Add>" in the table at the top of the work area.
   A new tag is created.
2. Configure an internal or external tag as required.
3. In the Inspector window, configure the range values of the tags under "Properties > Properties > Range":
   – Select whether to use a "Constant" or an "HMI tag" as limit value for your range values. The object list opens when you select "HMI tag". Select the tag you want to use.



1. Click the "Analog Alarms" tab at the bottom of the work area.
   Create an analog alarm for both range values.
2. Select an analog alarm and configure it in the Inspector window:
   – Enter the alarm text under "Properties > Properties > General > Alarm text".
   – You can also insert output fields into the alarm text.
   – You can change the default alarm class.
3. Configure the analog alarms as in the "HMI alarms" editor.
4. Complete the configuration of all analog alarms.

Note

A tag is monitored using only one alarm type. You should therefore create either analog alarms **or** discrete alarms for a tag.

Result

The configured analog alarms are created in the "HMI tags" editor and displayed in the "HMI alarms" and "HMI tags" editors.

12.3.2.3 Configuring the Outputting of Alarms

Overview of configuring alarm output

Steps to complete when configuring alarm output

You configure the alarm output in WinCC in the following steps:

1. Create alarm view
   Use the display and control objects in the "Screens" editor to display alarms in Runtime. You can also configure an alarm view for logged alarms.
2. Configure acknowledgment
   In the "Screens" editor, you can set the operator action that will trigger the acknowledgment.
3. Configure reporting
   You can create reports in the "Reports" editor to print alarms in Runtime. In the "Scheduler", "Screens", "HMI alarms" or "HMI tags" editors, you define when and how the printing of a report is triggered.

Note

Device dependency

Reporting and logging are not available for all HMI devices.

Additional configuration tasks

Additional tasks may be necessary for configuring alarm views, depending on the requirements of your project:

1. Setting up authorizations
   To make sure only authorized operators process the alarms, assign authorizations for the alarm view and the function keys of the HMI device.
2. Configuring the filtering of the alarm view
   You configure the filtering of the alarms in Runtime in the "Screens" editor. You can also configure alarm views that only display selected alarms.
3. Configure operator input alarms
   Configure the operator input alarms on the operator controls of the HMI device in the "Screens" editor. A preconfigured operator input alarm is output for an operator action. An operator input is, e.g. the acknowledgment of an alarm.

Displaying alarms

Options for displaying alarms on the HMI device

WinCC offers the following options for displaying alarms on the HMI device:

- Alarm view
  The alarm view is configured in a screen. More than one alarm can be displayed simultaneously, depending on the configured size. You can configure multiple alarm views with different contents.
- Alarm window
  The Alarm window is configured in the "Global screen" editor. The alarm window can display multiple alarms at the same time, depending on the configured size. An event can trigger closing and reopening of the alarm window. To hide it during configuration, create an alarm window on its own level.

Additional signals

- Alarm indicator
  The alarm indicator is a configurable, graphical icon. When an alarm comes in, the alarm indicator is displayed on the HMI device. You configure the alarm indicator in the "Global screen" editor.
  The alarm indicator has two states:
  - Flashing: At least one alarm that requires acknowledgment is pending.
  - Static: The alarms are acknowledged but at least one of them has not gone out yet. The alarm indicator also displays the number of pending alarms according to the HMI device.
- E-mail notification
  To inform someone other than the operator, such as an engineer, when an alarm with a specific alarm class arrives, assign the alarm class to an e-mail address.

  Note

  Device dependency

  E-mail notification is not available for all HMI devices.

- System functions
  You can configure a list of functions for the event associated with an alarm. These functions must be executed in Runtime when the event occurs.
  Use system functions for alarms in WinCC to control the alarm view or the alarm window other than via the toolbar.

Displaying the predefined alarm classes in Runtime

The following table shows the symbols used to display the predefined alarm classes in the alarm view:

Alarm class	Displayed icon
"Diagnosis Events"	S7
"Errors"	!
"System"	$
"Warnings"	<No symbol>

See also

Configuring an alarm view for active alarms (Page 4318)

Configuring an alarm window (Page 4325)

Configuring an alarm indicator (Page 4326)

Configuring alarm filtering (Page 4327)

Displaying Logged Alarms (Page 4330)

System functions for alarms (Page 4364)

Configuring an alarm view

Configuring an alarm view for active alarms

Introduction

Current alarms are displayed in Runtime in an alarm view or alarm window.

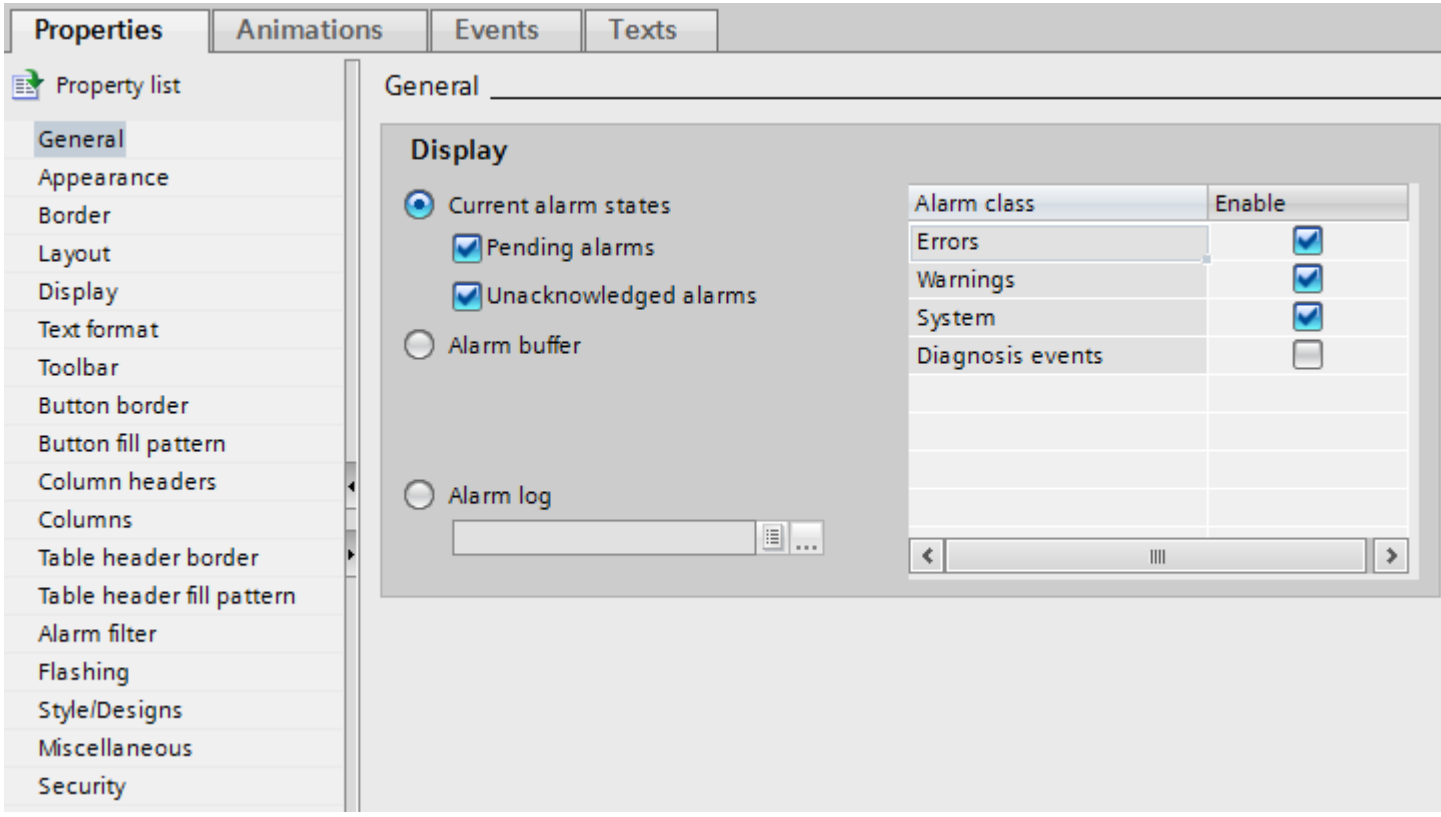
Requirement

- A screen is open in the "Screen" editor.
- The "Tools" task card is open.

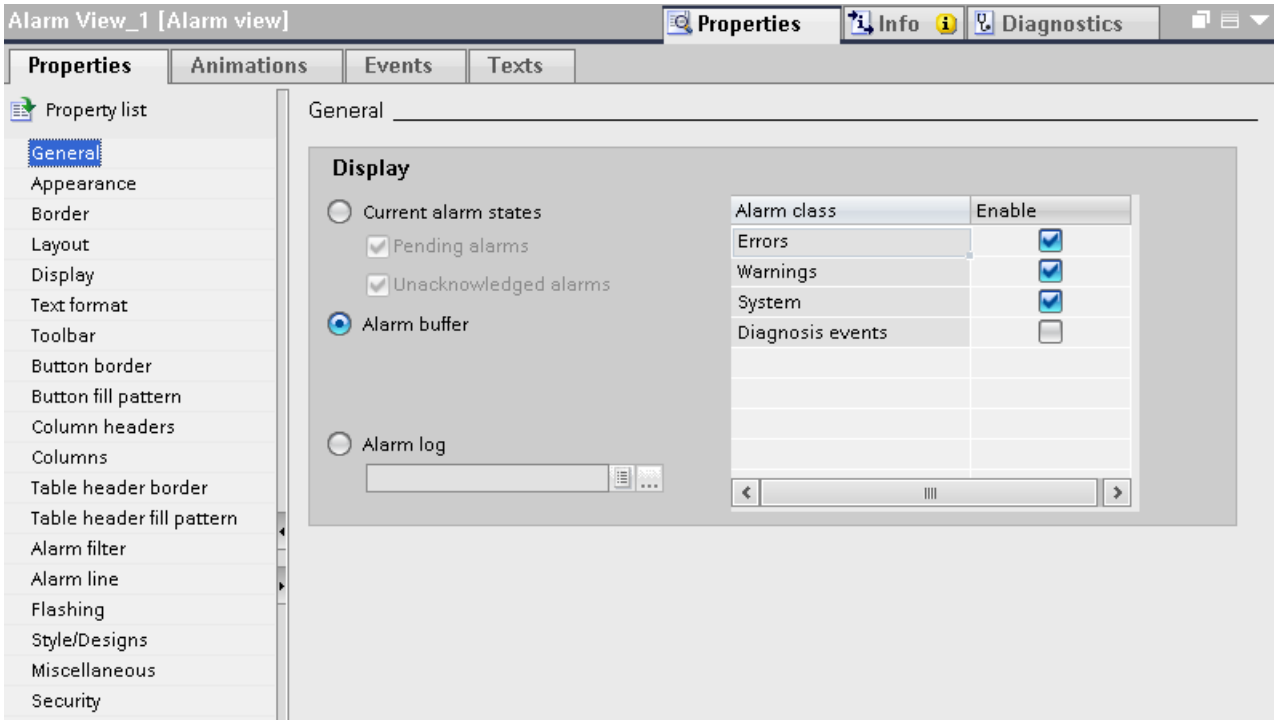
Configuring alarms for the alarm view

To specify the alarms that will be shown in the alarm view, proceed as follows:

1. Insert an "Alarm view" object from the "Tools" task card into the screen.
2. Select the alarm view.
   - In the Inspector window, select "Properties > Properties > General > View > Current alarm states".
     Set whether to display alarms with and/or without mandatory acknowledgment.



– To display all alarms in the alarm buffer, enable "Alarm buffer".

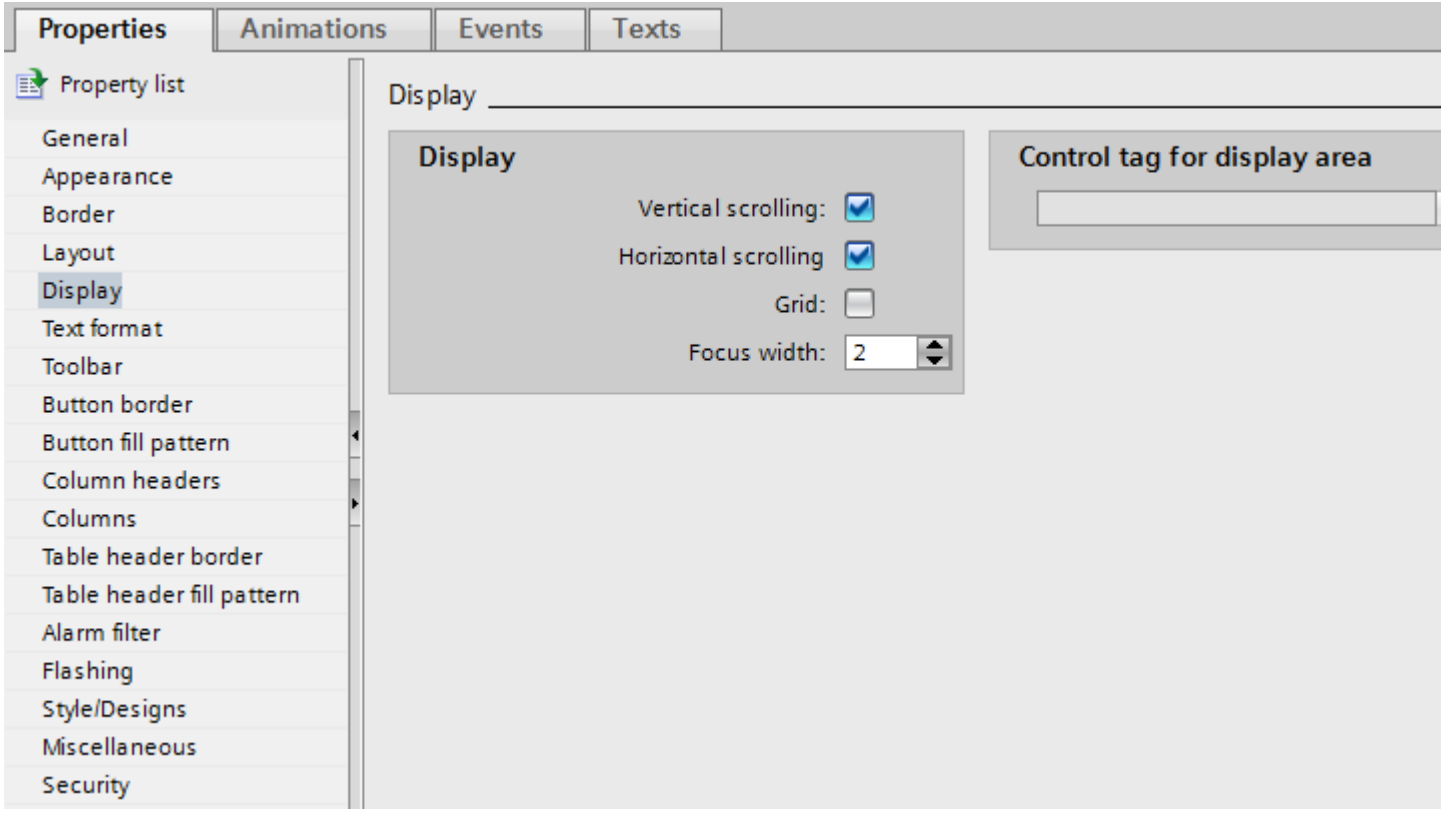


3. In the table, activate the alarm classes to be displayed in the alarm view.
4. Under ""Properties > Properties > View > Control tag for display area", define the tag that returns the date as of which the alarms will be displayed.

Note

Device dependency

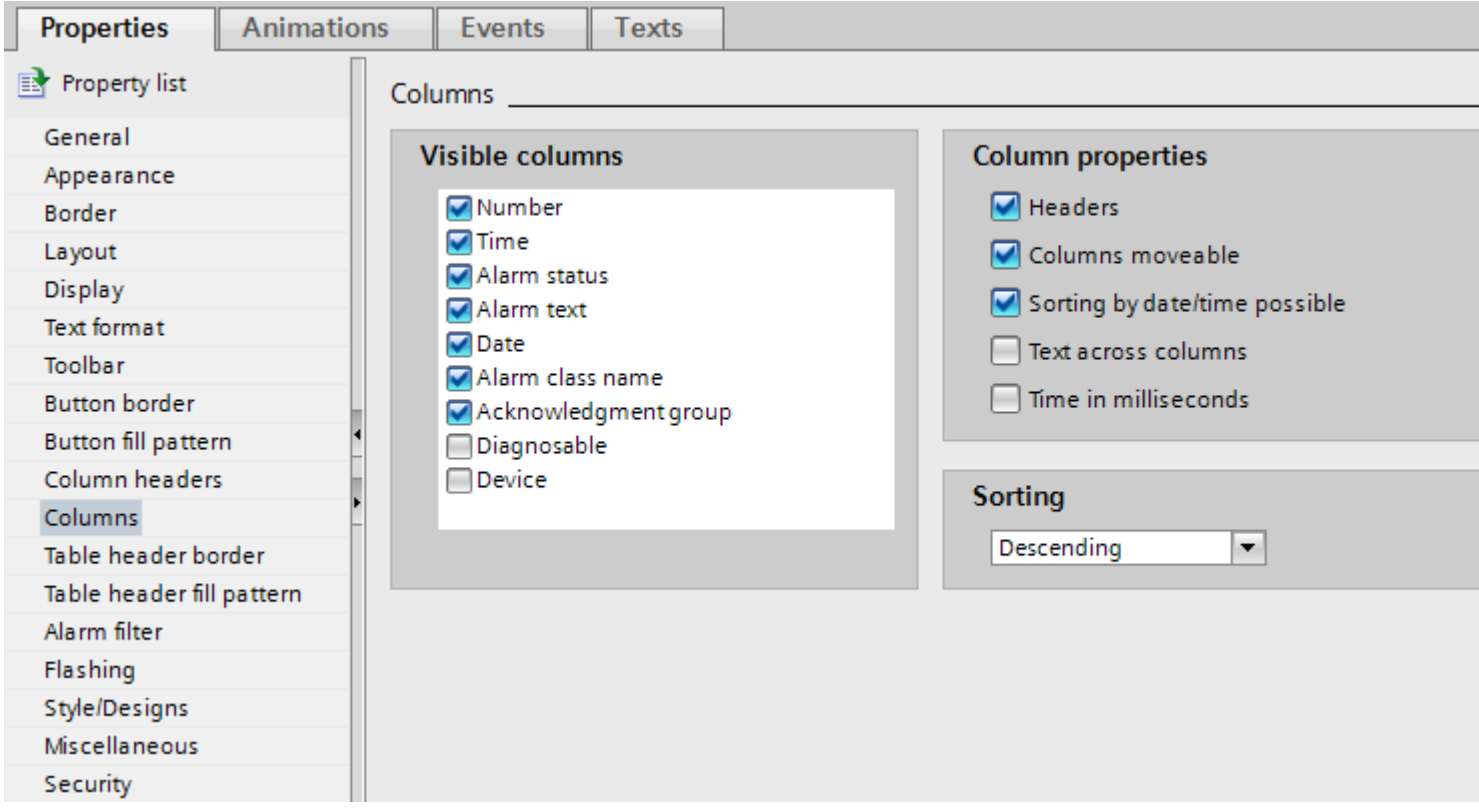
The "Control tag for display area" property is not available for all HMI devices.



Configuring the layout of the alarm view

To specify how the alarms are shown in the alarm view, proceed as follows.

1. Under "Properties > Properties > Layout > Settings > Lines per alarm" in the Inspection window, specify the number of lines to display for each alarm.
2. In "Properties > Properties > View", select the control elements that are available on the HMI device.
3. Configure the columns under "Properties > Properties > Columns":
   - Under "Visible columns" select the columns to be output in the alarm view.
   - Under "Properties Column", define the properties of the columns.
   - Under "Sort", select the sorting order of the alarms.



Note

Select the "Edit" command in the shortcut menu for the alarm view to activate the alarm view. You can set the column width and position in active mode. To enable the alarm view, set the zoom factor to 100 %.

The alarm view is deactivated as soon as you deselect the object in the screen.

Result

Alarms of various alarm classes are output in the alarm view during runtime.

See also

Creating alarm classes (Page 4297)

Configuring an alarm window (Page 4325)

Configuring an alarm view for the S7 diagnostic alarms

Introduction

To display S7 diagnostic alarms, configure an alarm view or alarm window for the predefined "Diagnosis Events alarm class. Activate the diagnostic alarms accordingly in the Runtime settings of your HMI device.

Note

HMI device dependency

The "Diagnosis Events" alarm class is not available for all HMI devices.

Requirements

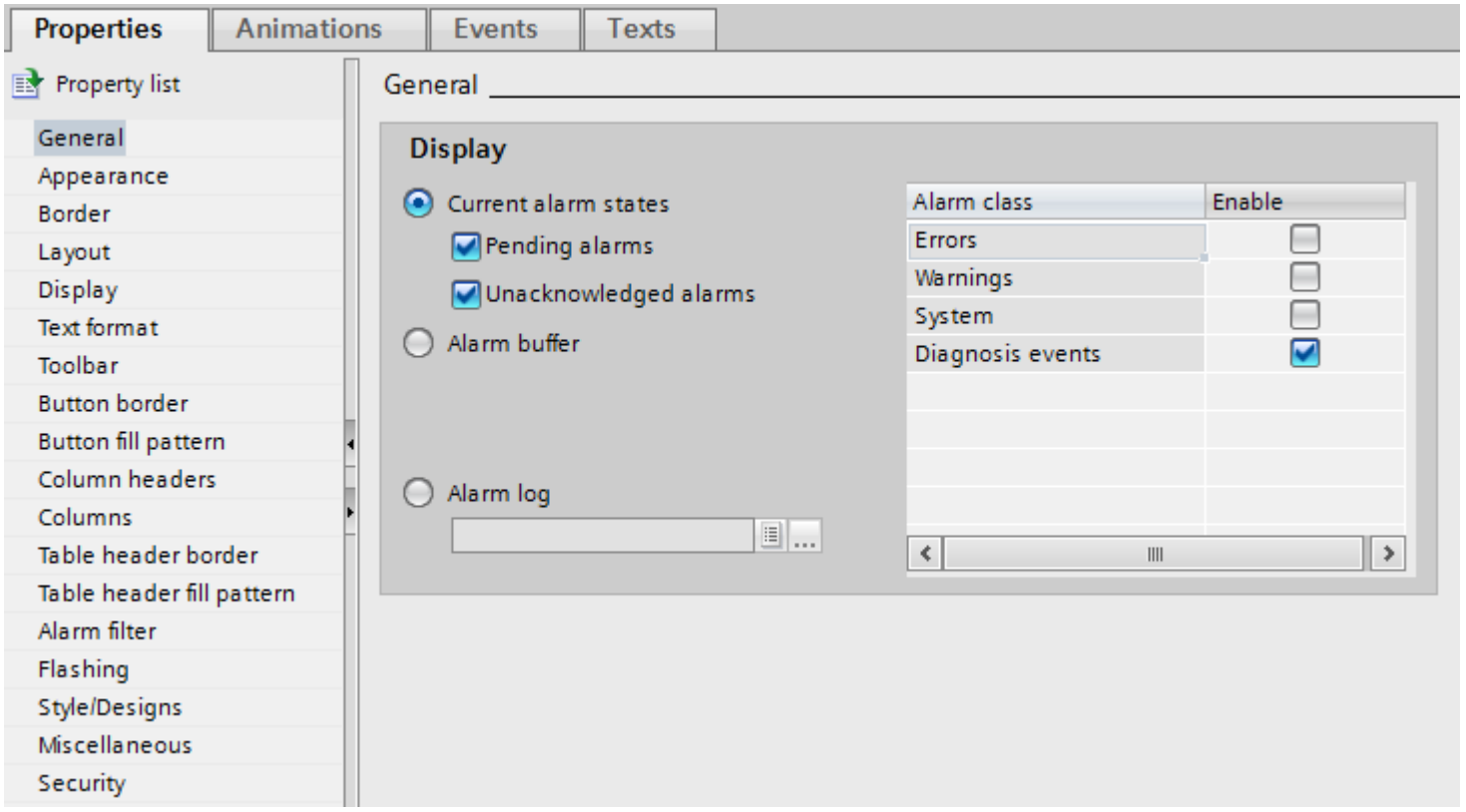
- An alarm view or alarm window is configured in the "Screens" editor
- A connection to the PLC is set up.

Configuring the display of S7 diagnostic alarms

To configure the display of S7 diagnostic alarms, follow these steps:

1. Select "Runtime settings > Alarms > System events > S7 diagnostics alarms" in the project navigation.
2. To display the S7 diagnostics alarms with text, activate "With text".
3. Open the screen with the alarm view and select the alarm view.

4. In the Inspector window, select "Properties > Properties > General > Display > Alarm class "Diagnosis Events".



5. Continue to configure the alarm view as for the display of current alarms.

Result

In Runtime the alarm view displays the S7 diagnostic alarms of the controller.

Configuring an alarm view for logged alarms

Introduction

In Runtime, logged alarms are displayed in an alarm view or alarm window.

Requirements

- An alarm view or alarm window is configured in the "Screens" editor
- An alarm log was created in the "Log" editor.
- The alarms are assigned the "loggable" attribute in the "HMI alarms" editor.

Configuring an alarm view for logged alarms

To configure an alarm view for logged alarms, proceed as follows:

1. Open the screen with the alarm view and select the alarm view.
2. In the Inspector window, select "Properties > Properties > General > Alarm log".
3. Click the "...." button and select the alarm log.
4. Continue to configure the alarm view as for the display of current alarms.

Result

In Runtime, the logged alarms are output in the alarm view.

Configuring an alarm window

Introduction

The alarm window displays current alarms. The alarm window is configured in the "Global screen" editor. The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active. Depending on the configuration, it is not closed until the alarm is acknowledged. The HMI device can still be used, even if alarms are pending and displayed. An alarm window is displayed and configured like an alarm view.

To hide an alarm window during configuration, create it on its own level.

Requirement

- The "Global Screen" editor is open.
- The "Tools" task card is displayed.
- The Inspector window is open.

Procedure

Proceed as follows to configure an alarm window:

1. Insert an "Alarm window" object from the "Tools" task card into the global screen.
2. Configure the alarm window like an alarm view.
3. Under "Properties > Properties > Mode > Window" in the Inspector window, select how the alarm window reacts and is operated in Runtime.
   - Activate "Modal" if the alarm window is to retain the focus in Runtime after a screen change.
     This option is important, as switching back and forth between the screen and different windows with <Ctrl+TAB> is not supported.

Result

During runtime, the alarms of the selected alarm class are displayed in the alarm window.

See also

Configuring an alarm view for active alarms (Page 4318)

Creating alarm classes (Page 4297)

Configuring an alarm indicator

Introduction

The alarm indicator uses a warning triangle to indicate that alarms are pending or require acknowledgment. If an alarm of the configured alarm class occurs, the alarm indicator is displayed.

The alarm indicator has two states:

- Flashing: At least one alarm that requires acknowledgment is pending.
- Static: At least one of the acknowledged alarms has not gone out yet.

During configuration, specify whether Runtime has to open an alarm window when you operate the alarm indicator.

Requirement

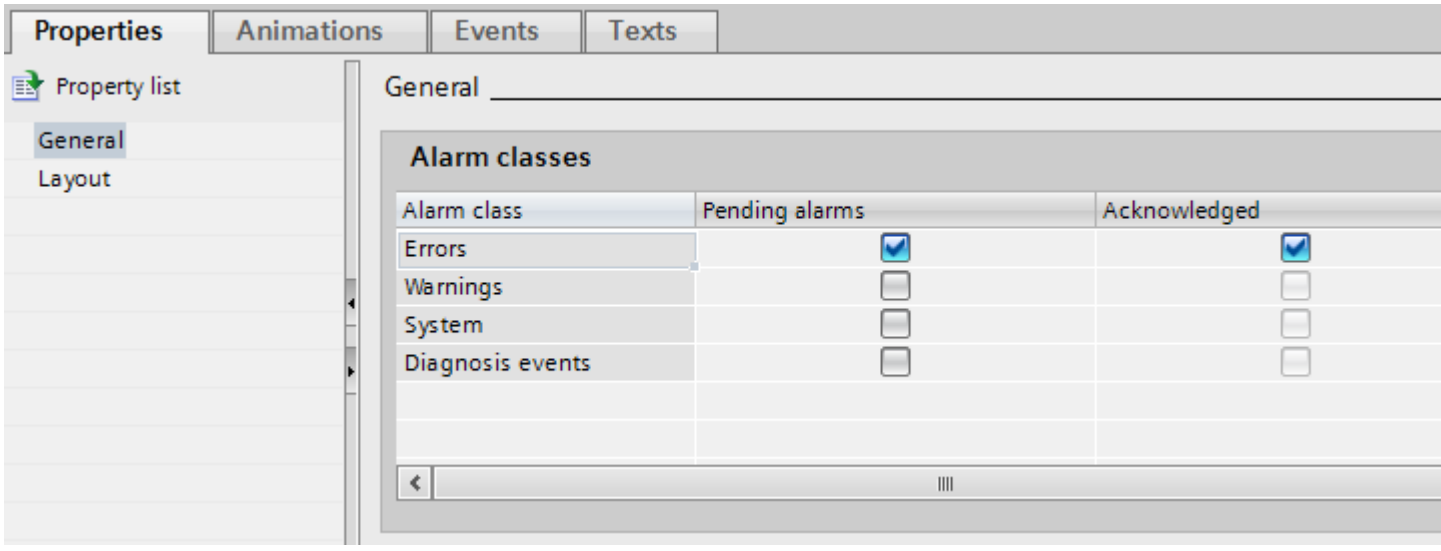
- The "Global Screen" editor is open.
- The "Tools" task card is open.
- The Inspector window is open.

Procedure

Proceed as follows to configure the alarm indicator:

1. Insert the "Alarm view" object from the "Tools" task card into the work area.
2. Select the alarm indicator.

3. Under "Properties > Properties > General" in the Inspector window, select the alarm classes to be displayed by the alarm view.
   Specify whether to display pending and/or acknowledged alarms in the alarm view.



4. Under "Properties > Event", assign the system function "ShowAlarmWindow" to an event of the alarm view.

Note

If you have configured a permanent window in the screen or template, do not position the alarm window and alarm view in the vicinity of the permanent window. Otherwise, the alarm window and the alarm view will not be displayed in Runtime. However, the permanent window is not visible in the "Global screen" editor.

Result

The alarm view is displayed if alarms from the selected alarm class are pending or need to be acknowledged in Runtime. The alarm window opens when the user operates the alarm view.

Configuring alarm filtering

Introduction

You can filter the display of alarms in the enhanced alarm view. The criterion is always a character string. There are two ways to filter the alarm view:

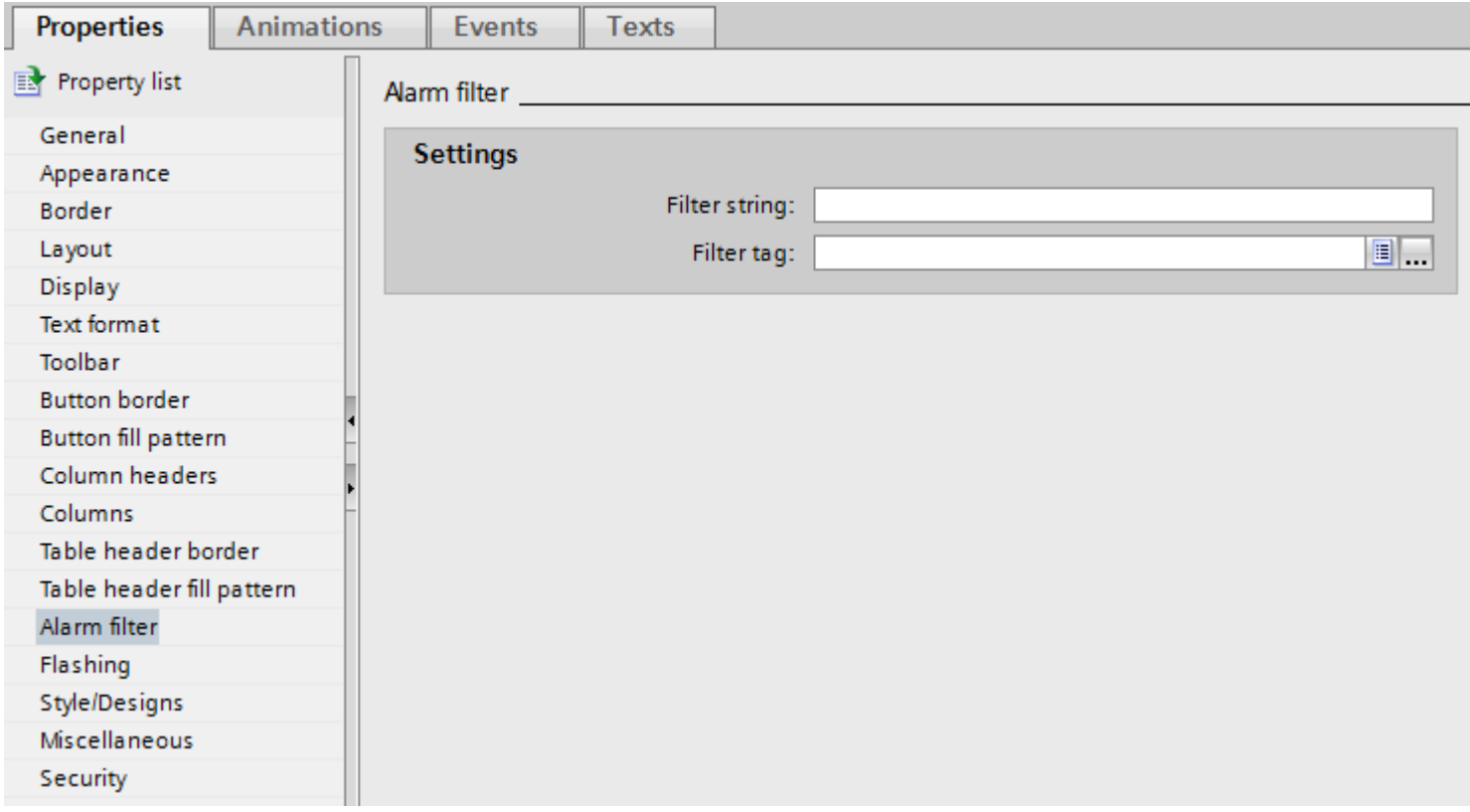
- A fixed criterion is configured in WinCC.
  In Runtime, all alarms that contain the complete character string in the alarm text are displayed.
- You filter the alarm view in Runtime.
  In Runtime, a filter tag sets the relevant character string in the alarm view by means of an I/O field, e.g. as described below.
  The filter affects the display in the alarm view. All alarms in the alarm buffer are retained.

Requirement

- The alarm view has been configured.
- The screen with the alarm view is open.
- The Inspector window is open.

Configuring filters for a fixed character string

1. Select the alarm view.
2. In the Inspector window, select "Properties > Properties > Alarm filter".
3. Enter a filter string in "Filter string".



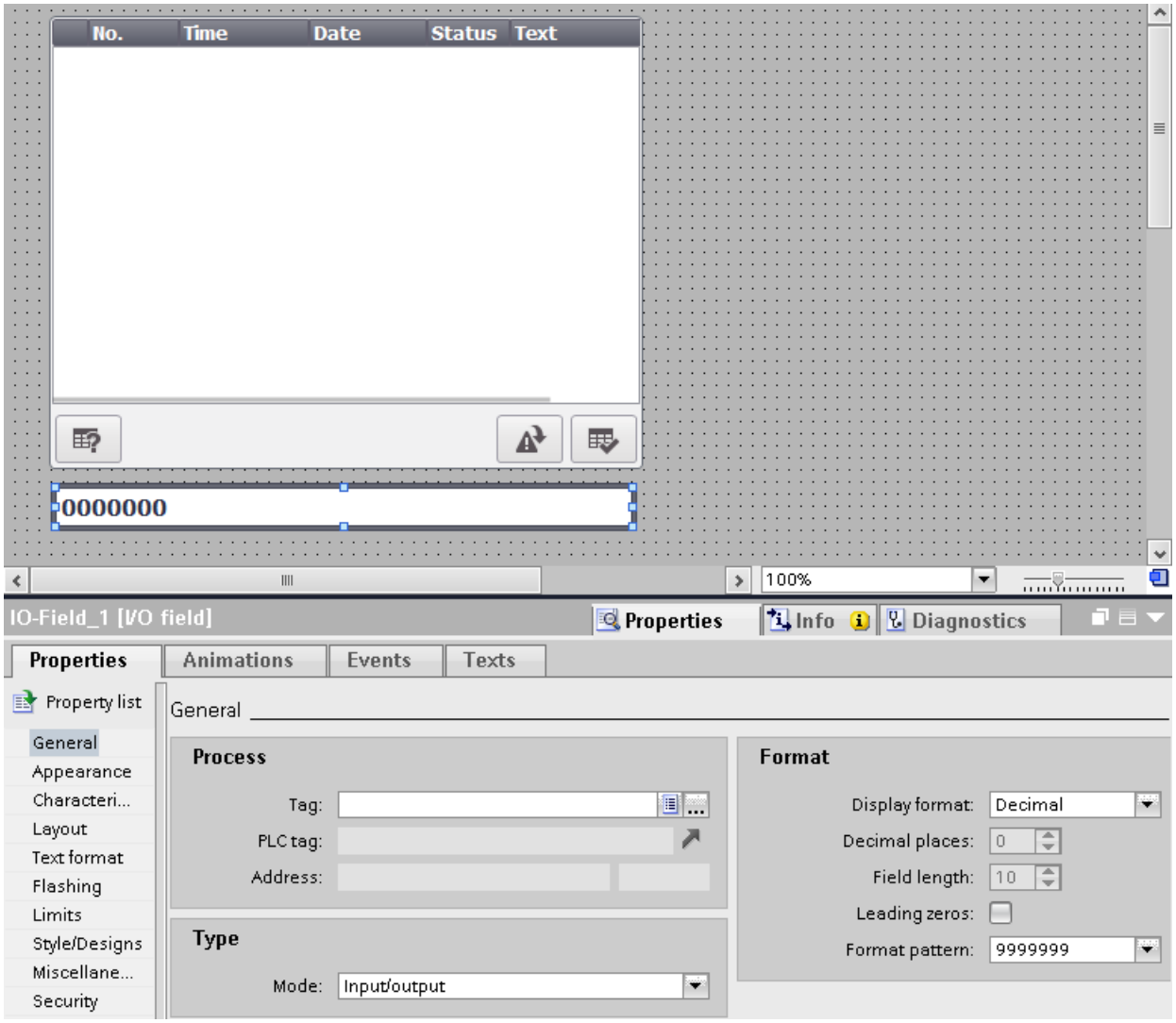
Result

Only those alarms are displayed in Runtime that contain the complete character string from the filter in the alarm text.

Configuring filters for a variable character string

1. Select the alarm view.
2. In the Inspector window, select "Properties > Properties > Filter".

3. Select a tag from "Filter tag". The tag must be of the "String" type.
4. In the screen, configure an I/O field for entering the filter string in Runtime.
   - In the Inspector window, select "Properties > Properties > General > Format > Display format > String".
   - To link the I/O field to the alarm view, select "Process" to set the filter tag you selected for the alarm view.



Result

If you enter a string in the I/O field in Runtime, the alarm view only displays alarms that contain the complete string in their alarm text.

If a permanently configured character string and a filter tag were configured, the alarms will be filtered in Runtime according to the content of the filter tag. If the filter tag is blank, the permanently configured character string is used as the filter.

Note

Filtering is not possible for the following alarm views:

- Simple alarm view
- Alarm line
- Alarm window
- Alarm view for logged alarms

Displaying Logged Alarms

Introduction

You display logged alarms in Runtime in an alarm view. The alarm view shows the content of an alarm log.

Application

For example, you want to view information about the process at the end of a shift.

Requirements

- An alarm log has been created.
- A screen is open.
- The Inspector window is open.

Procedure

Proceed as follows to configure an alarm view to output an alarm log:

1. Configure an alarm view in a screen.
2. Under "Properties > Properties > General > Display > Alarm log"" in the Inspector window, select the required alarm log.
   You can always create a new alarm log.

Result

During runtime, the alarms logged in the selected alarm log appear in the Alarm view.

Note

In runtime, the alarm texts logged are not displayed but rather the alarm texts in the current project. The logged alarm texts are only intended for external evaluation of the log file.

If the alarm log displayed was configured differently, the alarm texts displayed may not correspond with those logged.

Reporting Alarms

Overview

WinCC can report on all alarms that occur in the system. The following options are available:

- Output an alarm sequence report
  In Runtime, the standard printer of the HMI device will continuously print each alarm and any changes in its status.
- Output of an alarm report
  You configure an alarm report in the "Reports" editor and specify when it is output in Runtime:
  - For event-driven output, configure an object that is assigned the "PrintReport" system function. The object can be a button or tag.
  - For time-driven output, create a "Print job" in the scheduler. Assign an alarm report to the job.

    Note

    In the alarm report, specify whether to output current or logged alarms.

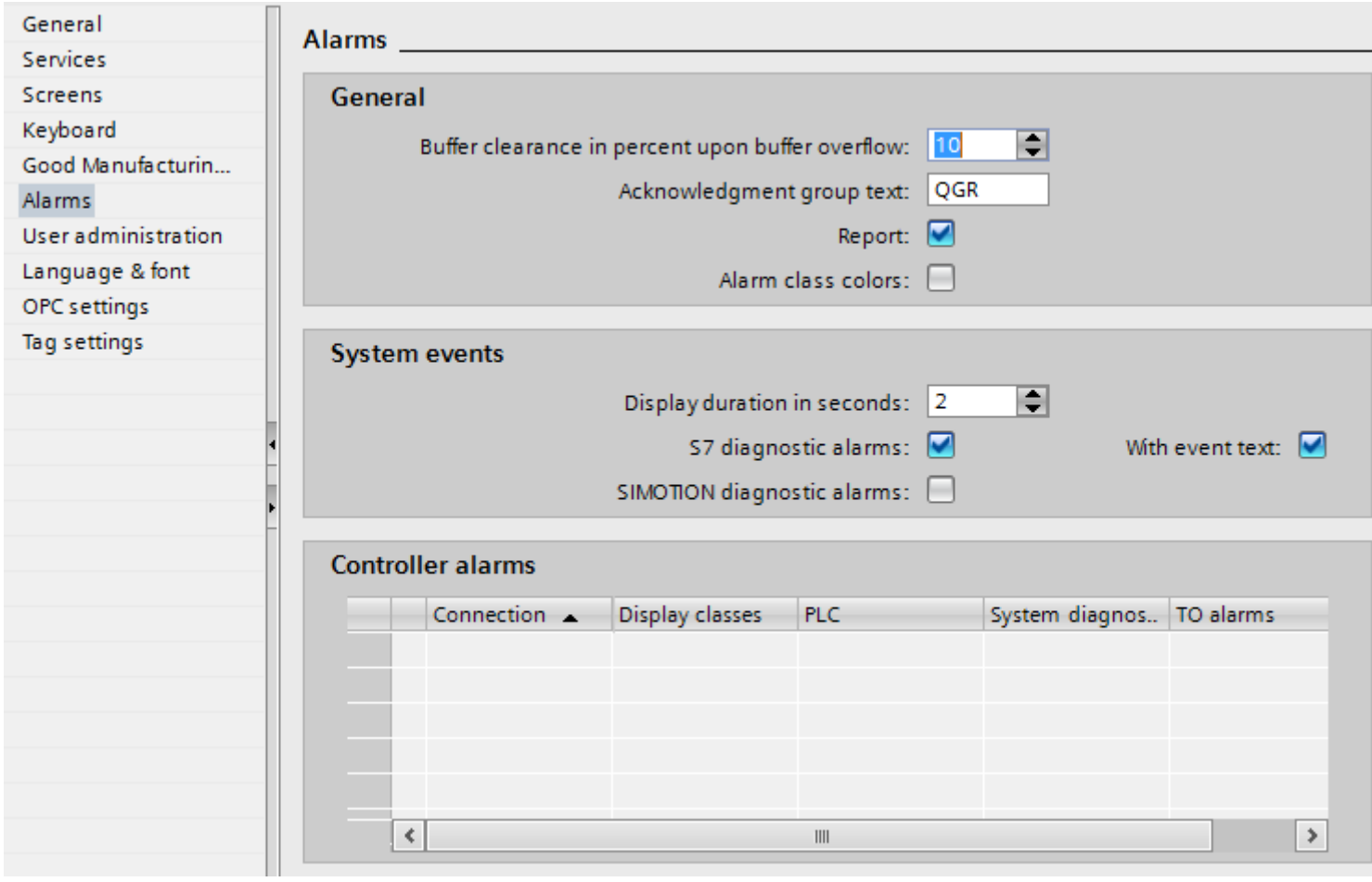
Requirements

- A printer was configured on the HMI device.

Activating continuous alarm reporting

To activate continuous reporting of alarms, follow these steps:

1. Double-click "Runtime settings" in the project navigation.
2. Select "Alarms > General > Report".



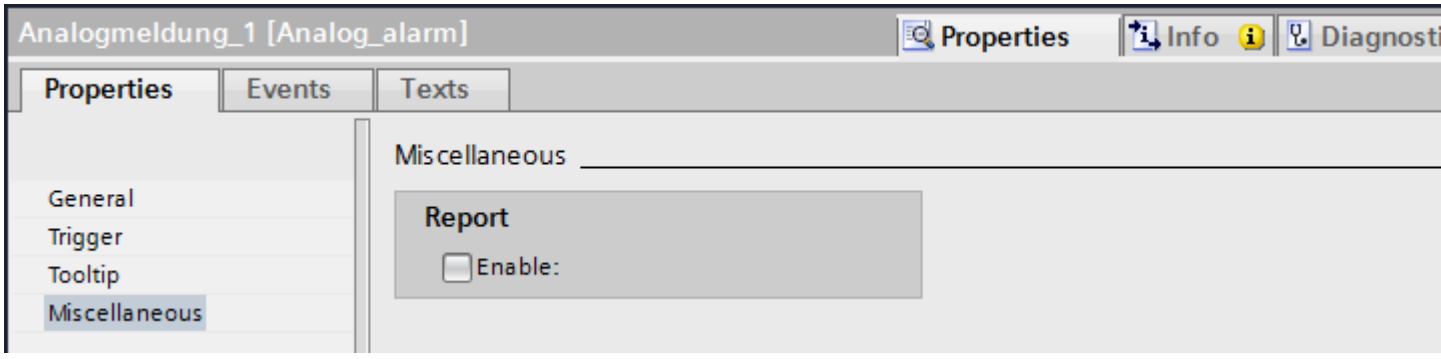
Result

The latest alarms are output at the printer of the HMI device.

Excluding an alarm from reporting

To exclude specific alarms from reporting, proceed as follows:

1. Open the "HMI alarms" editor.
2. Select the alarm to exclude from reporting on the tab of the selected alarm type.
3. In the Inspector window, disable "Properties > Properties > Miscellaneous > Report".



Result

WinCC does not output these alarms currently on the connected printer.

See also

Working with reports (Page 4496)

12.3.2.4 Acknowledging alarms

Configuring alarm acknowledgment by means of alarm class

Introduction

To configure an alarm with alarm acknowledgment, assign it to an alarm class with the "Alarm with single acknowledgment" state machine.

Requirement

- The "HMI alarms" editor is open.
- The required alarm class has been created.
- The required alarm has been created.

Selecting the state machines for an alarm class

The state machine for a predefined alarm class has already been set. You can only set the state machines for user-defined alarm classes. Proceed as follows:

1. In the "HMI alarms" editor, click the "Alarm class" tab and select the alarm class.
2. Select the required acknowledgment model under "Properties > Properties > Acknowledgment" in the Inspector window.

Assign alarms to an alarm class requiring acknowledgment

Proceed as follows to assign an alarm to an alarm class requiring acknowledgment.

1. In the "HMI alarms" editor, click the tab for the alarm type and select the alarm.
2. Under "Properties > Properties > General" in the Inspector window, select the alarm class of the alarm.

Result

The alarm will not disappear in Runtime until it is acknowledged by the operator.

Configuring trigger for alarm acknowledgment

Introduction

You always specify the acknowledgment requirement for an alarm using the alarm class. Then the operator acknowledges the alarm using the "ACK" function key of the HMI device or the "Acknowledgment" button of the alarm view.

The following options are also available to trigger acknowledgment:

- Configuring a button to acknowledge an alarm
- Acknowledgment of a Discrete Alarm by the PLC

Requirements

- The "HMI alarms" editor is open.
- The required alarm class has been created.
- The required alarm has been created.
- An alarm view and a button are created in the "Screens" editor.

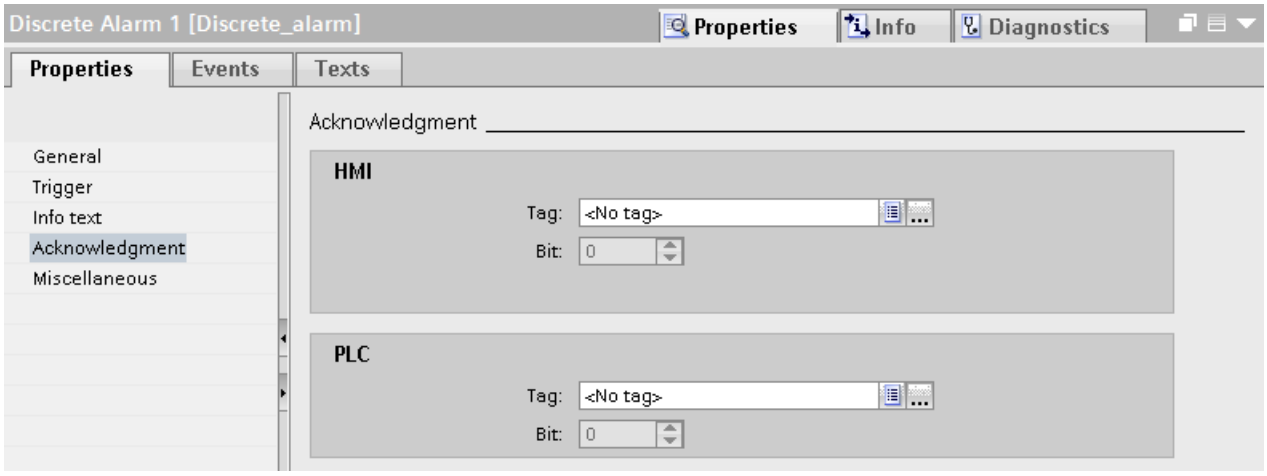
Configuring a button to acknowledge an alarm

To configure a button for acknowledging an alarm, proceed as follows:

1. Select the button in the "Screens" editor.
2. Under "Properties > Events" in the Inspector window, assign the "AlarmViewAcknowledgeAlarm" system function to the "Click" event.
3. Select the alarm view as parameter.

Acknowledgment of a Discrete Alarm by the PLC

1. In the "HMI alarms" editor, click the "Discrete alarm" tab and select the discrete alarm.
2. In the Inspector window, select the tag and the bit that acknowledges the PLC alarm under "Properties > Properties > Acknowledgment > PLC".



Acknowledging alarms in an alarm group

The various acknowledgment options have the following effect in alarm groups:

- Operator acknowledges the alarm
  You acknowledge the alarm of an alarm group, for example, using the <ACK> acknowledgment key or a function key. All alarms of the alarm group are acknowledged.
- The PLC acknowledges the alarm
  The alarm of an alarm group is acknowledged by setting a bit in the tag.
  Only the alarm to be acknowledged is acknowledged.

Sending alarm acknowledgments to the PLC

Requirement

- The "HMI alarms" editor is open.
- The required alarm has been created and assigned to an alarm class requiring acknowledgment.

Note

You cannot send the acknowledgment of analog alarms to the PLC.

Sending alarm acknowledgments to the PLC

To configure that acknowledgment of an alarm is sent to the PLC, follow these steps:

1. In the "HMI alarms" editor, click the "Discrete alarm" tab and select the discrete alarm.
2. In the Inspector window, select "Properties > Properties > Acknowledgment".
3. Under "HMI", select the tag and the bit set by the alarm acknowledgment function.

Note

The HMI device and PLC only have read access to the acknowledgment tag memory area.

Result

If the operator acknowledges the alarm in Runtime, the operating step is forwarded to the PLC.

12.3.2.5 Configuring alarm buffer overflow

Introduction

The size of the alarm buffer depends on the type of HMI device. In WinCC, specify the percentage of the alarm buffer to be deleted on alarm buffer overflow.

Requirements

- You have created a project.

Procedure

To configure the response to an alarm buffer overflow, follow these steps:

1. Double-click "Runtime settings" in the project navigation under your HMI device.
2. Under "Alarms > General > Buffer clearance in percent upon buffer overflow", enter a value between 1 and 100.
   This value specifies the percentage of the alarm buffer that is deleted on alarm buffer overflow.

Result

If data in the alarm buffer exceeds the resources of alarm buffer memory, the configured percentage of data will be deleted from the oldest alarms in the alarm buffer.

Note

You cannot check the alarm buffer overflow separately by alarm method. You can use the "ClearAlarmBuffer" system function to delete specific alarms of specific alarm classes from the alarm buffer.

See also

Alarm Logging Basics (Page 4337)

12.3.3 Logging Alarms

12.3.3.1 Alarm Logging Basics

Introduction

An alarm log is used to record project alarms.

Note

Alarm logging is not available for every HMI device.

Configuration steps

To log an alarm, follow these configuration steps:

1. Creating an alarm log
   You define the following properties for the alarm log:
   - Logging method
     Behavior of the log when reaching a specific fill level
   - Storage location and file format
   - Log size
   - Runtime start characteristics
2. Assigning an alarm log to an alarm class
   You can log the alarms from multiple alarm classes in an alarm log.
3. Assigning an alarm to a loggable alarm class
4. Configure display of logged alarms in an alarm view

Content of the alarm log

All states are logged for configured alarms. The following three entries, for example, are stored in the log for an alarm that requires acknowledgment:

- 04.08.2007 10:00:25:520, analog alarm, ID5, **K**, error, fill level exceeded by 10%
- 04.08.2007 10:01:20:442, analog alarm, ID5, **Q**, error, fill level exceeded by 10%
- 04.08.2007 10:01:30:112, analog alarm, ID5, **G**, error, fill level exceeded by 10%

In the example, the alarm states are identified by the following letters:

K = Incoming

Q = Acknowledged

G = Outgoing

All data belonging to an alarm is stored in the alarm log, including configuration data such as the alarm class, time stamp, and alarm text.

The potential number of logged alarms depends on the data medium used. You can further process the logged alarms in other programs for analysis purposes, for example.

Note

Alarm text and point of error are only logged if you have configured such a setting in the log properties.

Logged alarms that contain alarm text and the point of error exceed the estimated size of the configured alarms. Check to see whether the specified storage location still has sufficient space.

Note

The time stamp of a logged alarm is always specified in standard UTC format (Universal Time Coordinated).

Logging methods

The logging method determines how the alarm log responds when the configured size is reached. WinCC supports the following logging methods:

- Circular log
  When the configured log size has been reached, the oldest entries are deleted. When the configured log size has been reached, approximately 20% of the oldest entries are deleted. It is therefore not possible to display all the configured entries. During configuration, select an appropriate size for the circulation log. Alternatively, configure a segmented circular log.
- Segmented circular log
  In a segmented circular log, multiple single logs of the same size are filled in succession. When all logs are completely full, the oldest log is overwritten.
- Log with level-dependent system alarm
  When a defined level is reached, such as 90 %, a system alarm is triggered. When the configured size of the log has been reached, new alarm events are not logged.
- Log with level-dependent execution of system functions
  When the log is completely full, the "Overflow" event is triggered. You configure a function list for the event.
  When the configured size of the log has been reached, new alarm events are not logged.

Displaying logged data

On the HMI device, the logged data is displayed in an alarm view that is configured for this purpose.

See also

Checksums for logs in regulated projects (Page 4348)

Configuring alarm buffer overflow (Page 4336)

12.3.3.2 Creating an alarm log

Introduction

In Runtime, you save alarms to the alarm logs. Specify the alarm log in the alarm class. An alarm log contains the alarms of several alarm classes. Create the alarm log in the "Logs" editor. When creating an alarm log, define the following parameters:

- Name
- Size
- Storage location
- Runtime start characteristics
- Log type
- Advanced content
- Checksum

You can also enter comments for each log.

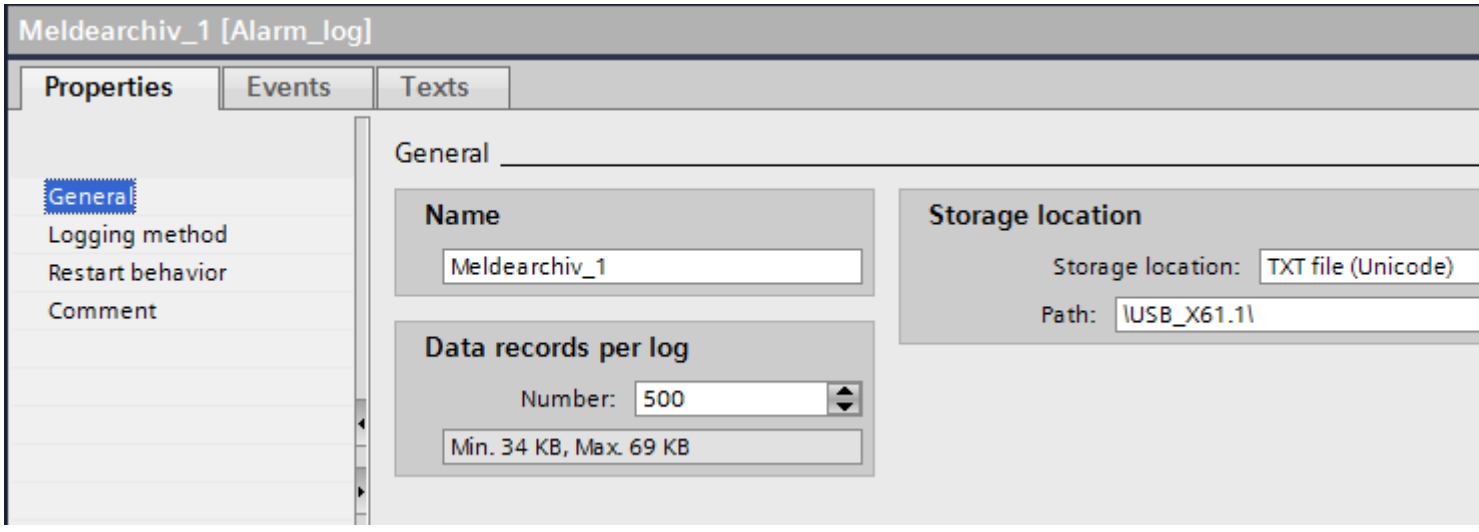
Requirement

- The "Alarm log" tab is open in the "Log" editor.
- The Inspector window is open.

Procedure

To create an alarm log, proceed as follows:

1. Double-click "<Add>" in the table.
   A new alarm log is created.
2. Under "Properties > Properties > General" in the Inspector window, enter a unique name for the alarm log.



3. Under "Number of data records per log", define the number of alarms to be saved to a log file.
   The approximate space required on the storage medium is displayed. Memory space requirements will increase accordingly if you log alarm texts with tag values.
4. In the "Storage location" field you select where the log entries are saved.
5. Depending on the selected "Storage location", you either select the "Path" or the "Name of the data source".

6. To determine if audit trail data were changed at a later time, activate "Properties > Properties > General > Checksum".

   Note
   Device dependency

   The "Checksum" option is only available for display and HMI devices which support "Configuration conforms to GMP".

7. If necessary, enter a descriptive text in the "Comment" category to document your configuration.

Result

The alarm log is created. You can assign one or several alarm classes to this alarm log.

See also

Managing logging behavior when Runtime starts (Page 4349)

Controlling the Logging in relation to the Fill Level (Page 4350)

12.3.3.3 Logging Alarms

Overview

To log alarms, follow these steps:

- Create an alarm log.
- Assign the created alarm log to an alarm class.
- Assign the alarm class with the created alarm log to an alarm.
- In the Runtime settings specify the language in which you want to write the logs.
- You evaluate logged alarms.
  You can analyze the logged alarms directly in your WinCC project, such as in an alarm view, or in another user program, such as Microsoft Excel.

  Note
  Tag fields in the alarm text

  The sequence of tag fields in the alarm text is language dependent. The sequence of the Runtime language is used for logging alarms to a "*.csv" log file.

  Changing the tag of an output field in one language causes the modified output field to appear at the end of the alarm text in all other languages. This means the sequence of the output fields in the log file can change.

Requirements

- You have created an alarm log.
- The "HMI alarms" editor is open.

Assigning an alarm log to an alarm class

To assign the alarm log to an alarm class, proceed as follows:

1. Open the "Alarm classes" tab in the "HMI alarms" editor.
2. Select the required alarm class.
3. Under "Properties > Properties > General > Log" in the Inspector window, select the alarm log.

Assign alarm to an alarm class

To assign an alarm to the alarm class, proceed as follows:

1. In the "HMI alarms" editor, open the "Analog alarms" tab, or the "Discrete alarms" tab.
2. Select the required alarm.
3. Under "Properties > Properties > General > Alarm class" in the Inspection window, select the alarm class for which the alarm log was configured.

Result

The alarm is saved to the configured alarm log.

12.3.3.4 Configuring an alarm view for logged alarms

Introduction

In Runtime, logged alarms are displayed in an alarm view or alarm window.

Requirements

- An alarm view or alarm window is configured in the "Screens" editor
- An alarm log was created in the "Log" editor.
- The alarms are assigned the "loggable" attribute in the "HMI alarms" editor.

Configuring an alarm view for logged alarms

To configure an alarm view for logged alarms, proceed as follows:

1. Open the screen with the alarm view and select the alarm view.
2. In the Inspector window, select "Properties > Properties > General > Alarm log".

3. Click the "...." button and select the alarm log.
4. Continue to configure the alarm view as for the display of current alarms.

Result

In Runtime, the logged alarms are output in the alarm view.

12.3.3.5 Direct access to the ODBC log database

Overview

The storage location of a log can be a database or a file.

The database is addressed by means of its "Data source name" (DSN). Select the database to be used in WinCC as follows:

Select the database from the Windows Start menu under "Settings > Control panel > ODBC Data Sources".

In your configuration, specify the "Data source name" (DSN) instead of a directory name as storage path for log data. With the DSN, you are referencing the database and the storage location.

The entire functional scope of the database is available for additional processing and evaluation of log data.

Application

The data source sets up the connection to the database. Create the data source on the same PC on which the Runtime software is stored. Then enter the DSN configured on this PC when you create a log in WinCC.

The ODBC interface allows you to use other programs, such as MS Access or MS SQL Server, to access the database directly.

In addition, you can use the "StartProgram" system function to configure program calls, e.g. MS Access, on the HMI device. Runtime is not interrupted while you configure such calls.

12.3.3.6 Setting up the ODBC data source

Introduction

To store process values or alarms in the log database in Runtime, set up the ODBC data source.

Requirement

- A log database has been created.
  You create the log database using the SQL Enterprise Manager. You can find more detailed information on how to do this from Microsoft.

Procedure

To set up an ODBC data source, follow these steps:

1. In Control Panel, open "Administrative Tools" and select "Data Sources (ODBC)". The "ODBC Data Source Administrator" dialog is opened.
2. Click "User DSN > Add".
3. Select the "SQL-Server and click on "Finish".
4. In the dialog that follows, enter a name for the User DSN and the SQL-Server and click on "Next".
5. In the dialog that follows, define the logon procedure for the SQL database and click on "Next".
6. Activate "Change the default database to".
7. Select the database you have created and click on "Next".
8. In the dialog that follows, click "Finish".

12.3.3.7 Configuring a checksum for a log

Introduction

In a regulated project, you have the option of assigning a checksum to the log data of an data log or alarm log. This checksum can be used during plant operation to determine if the data of this log has subsequently changed.

Note

Device dependency

The "Checksum" option is only available for display and HMI devices which support "Configuration conforms to GMP".

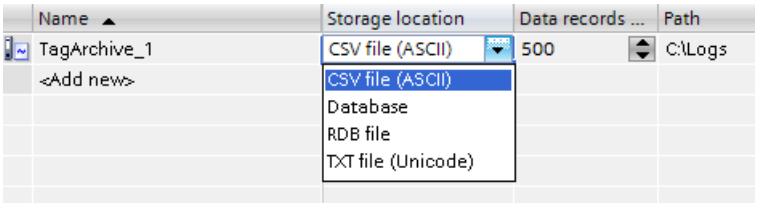
Requirement

- GMP compliant configuration is enabled.
- A data log or alarm log has been created.

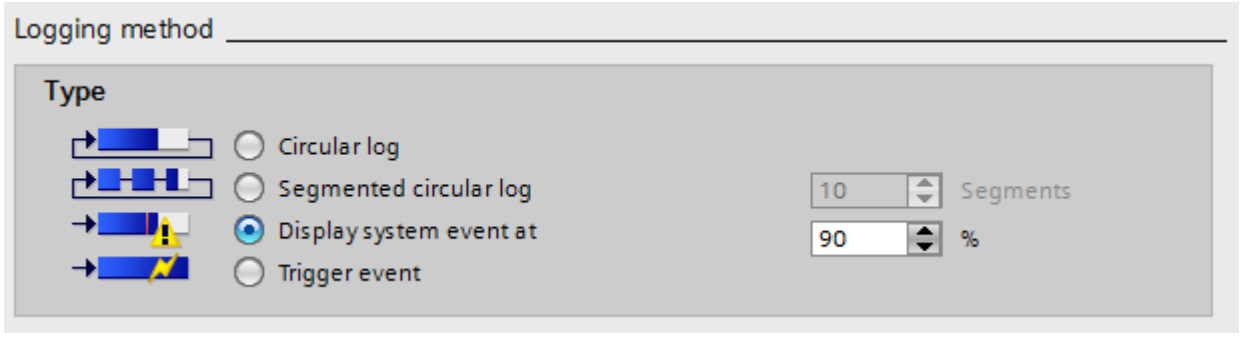
Procedure

Proceed as follows to configure a data log or alarm log for the use of a checksum:

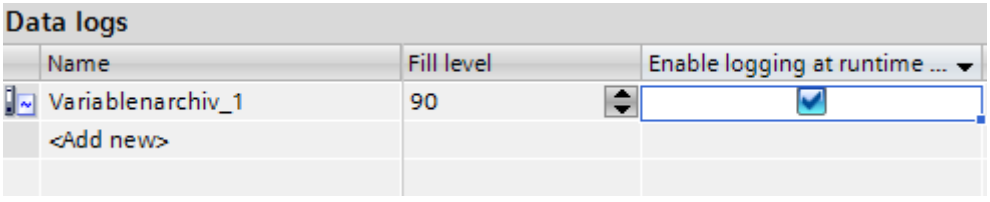
1. Open the data log or alarm log in the corresponding log editor.
2. In the "Storage location" box, select "File - CSV (ASCII)" or "File - TXT (Unicode)".



3. Under "Properties > Properties > Logging method" in the Inspector window , select the option "Display system event at" or "Trigger event".



4. In the editor table, activate the option "Enable logging at runtime start".
   Columns that are not displayed are activated with the shortcut menu of the column title.



5. Save the project.

Result

The log data of the log is assigned a checksum in runtime.

See also

GMP-compliant configuration (Page 7043)

Enabling GMP compliant configuration (Page 7046)

Checksums for logs in regulated projects (Page 4348)

12.3.3.8 Evaluating the checksum of log data

Introduction

If you have configured a data log or alarm log with generation of a checksum, you can check if the log data has subsequently changed.

Note

Device dependency

The "Checksum" option is only available for display and HMI devices which support "Configuration conforms to GMP".

The DOS program "HmiCheckLogIntegrity" is available for checking the integrity of the log data.

The "HmiCheckLogIntegrity" program supports verification of the following files:

- Log files of alarm logs, data logs, and Audit in CSV format
- Log files of alarm logs, data logs, and Audit in TXT format
- Recipe data records in CSV format
- Recipe data records in TXT format

You can find the "HmiCheckLogIntegrity.exe" program in the installation directory of WinCC under the folder "WinCC Runtime Advanced", for example <C:\Program Files\Siemens \Automation WinCC Runtime Advanced>.

Note

Audit Trail and Log with Checksum

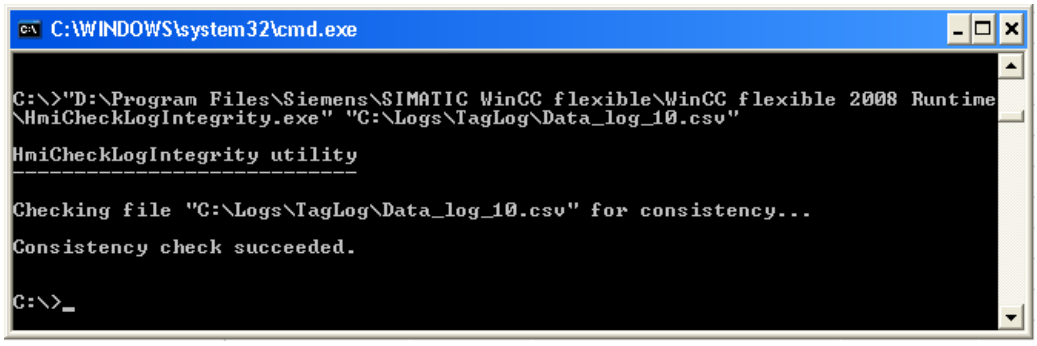
Before you update WinCC with a Service Pack or a new version, exit and save the Audit Trail or the logs using checksum. After WinCC is updated, the audit trail or logs will be continued with new files using checksum.

Make sure that the logs are started at a defined state with the new version.

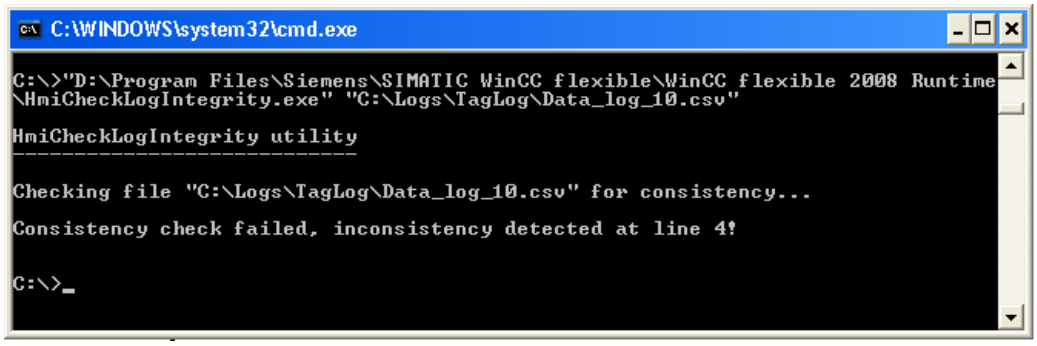
Procedure

1. Copy the file to be checked from the HMI device to your configuration computer.
2. Open a command line prompt with "Start > Programs > Accessories > Command Prompt".
3. Enter the path to "HmiCheckLogIntegrity.exe" followed by a space in the command line prompt. After the space, enter the storage location of the file to be checked within quotation marks.

4. Press <Enter>.
5. The check is performed.
   When the checked data are consistent, the "Consistency check succeeded" message appears.



   When the checked data are inconsistent, the "Consistency check failed" message appears. Information about the first inconsistent line in the file is also displayed.



Note

If there are spaces in the path to the "HmiCheckLogIntegrity.exe" program, you need to specify the path in quotation marks.

You can also check the integrity of the log data with the AuditViewer.

See also

Enabling GMP compliant configuration (Page 7046)

Evaluating Audit Trails in AuditViewer (Page 7063)

Checksums for logs in regulated projects (Page 4348)

12.3.3.9 Checksums for logs in regulated projects

Checksums in regulated projects

With WinCC you configure regulated projects as required by FDA guidelines, if necessary. Use the "Audit" option to create regulated projects.

In regulated projects you supply a checksum for the logged data of an alarm log. The operator will use this checksum during operation of the plant to see if the data of an alarm log have been changed.

For the checksum to be clear, it starts with the first line of an alarm log and integrates the previous lines when continuously generating the checksum. This means a checksum can only be generated for the following logs:

- Log that sends a system alarm when it is full
- Log with execution of system functions when log is full.

Generation of a checksum is also available for alarm logs that are saved as files of the "*.csv" or "*.txt" (Unicode) format.

If you want to continue an existing log without checksum as one with checksum, a backup copy of the existing log is created in the "*.keep" file format. A new log with checksums is created.

To maintain the validity of checksums, the following limitations apply when using the system function "CopyLog":

- You cannot copy a log without checksums into a log with checksums.
- You cannot copy a log with checksums into a log without checksums.

See also

Alarm Logging Basics (Page 4337)

Configuring a checksum for a log (Page 4344)

Evaluating the checksum of log data (Page 4346)

12.3.3.10 Log response to language switching in runtime

Introduction

In the Runtime settings of your HMI device, select the language to be used for writing to logs in runtime.

Requirement

- The languages used in your project are activated in the "Project languages" editor, for example, "German (Germany)" and "English (USA)" .

Procedure

To determine the startup language, follow these steps:

1. Select "Runtime settings > Language and fonts" in the project navigation.
2. Activate the runtime language, for example, "German (Germany)" and "English (USA)".
3. Set the "Language switch order". Use 0 to determine the startup language, for example:
   - German 0.
   - English 1
     With "0", German is specified as the "Startup language".
4. Select "Runtime settings > General" in the project navigation.
5. Select the "Logs > Logging language > Startup language".

Result

The project starts after it is transferred. German is specified as the "Startup language" with "Language switch order". The logs are now written in German. During runtime, the operator switches the runtime language to English. The logs will still to be written in German.

The operator closes runtime. Due to the previously performed language switch, the next time the system starts the "startup language" is English. Since English is the startup language, the logs are now written in English.

The logging language remains English even when the language is switched again in runtime until runtime is closed once again.

If you use another option instead of "Startup language" to select the language, the logs are always written in the same language. This is true regardless of the language selected by the operator in runtime.

12.3.3.11 Managing logging behavior when Runtime starts

Introduction

When you configure a log, you define the restart characteristics of the log when Runtime is started. You define whether the logging should start when Runtime starts in the log properties. You can also define whether an existing log will be continued or overwritten.

You define the restart characteristics separately for each log.

Requirement

- You have created a log.
- The "Historical Data" editor is open.
- The Inspector window with the log properties is open.

Procedure

To configure the restart characteristics of a data log, proceed as follows:

1. Select the log for which you want to define the restart characteristics in the "Historical Data" editor.
2. In the Inspector window select "Properties > Properties > Restart behavior".
3. If you want logging to start when Runtime starts, enable the "Enable logging at runtime start" option in the "Logging" area.
   You can also start logging in Runtime using the "StartLogging" system function, for example.
4. Select the restart behavior of the log in the "Log handling at restart" area.
   - The logged values are deleted and logging is started again with the option "Reset log".
   - The option "Append data to existing log" is used to append the values to be logged to the existing log.

Alternatively you can configure the restart characteristics of a log directly in the "Historical Data" editor. To view hidden columns, activate the column titles using the shortcut menu.

Result

Logging will start in runtime according to your settings.

12.3.3.12 Controlling the Logging in relation to the Fill Level

Introduction

The size of a log is determined by the number of entries. You use the logging method to determine how the log responds when it is full.

Logging methods

The following logging methods are available:

- Circular log
  When the configured log size has been reached, the oldest entries are deleted. When the configured log size has been reached, approximately 20% of the oldest entries are deleted. It is therefore not possible to display all the configured entries. During configuration, select an appropriate size for the circulation log. Alternatively, configure a segmented circular log.
- Segmented circular log
  In a segmented circular log, multiple log segments of the same size are filled in succession. When all logs are completely full, the oldest log is overwritten.

- Log that sends a system alarm when it is full
  When a defined level is reached, such as 90 %, a system alarm is triggered. When the log is 100% full, new tag values are not logged.
- Log with level-dependent triggering of an event.
  When the log is completely full, the "Overflow" event is triggered. Configure a function list for the event that will be carried out when the "Overflow" event occurs. When the configured size of the log is reached, new tag values are not logged.
  The following system functions are available for further processing of full logs: For further details, refer to Auto-Hotspot.

Requirement

- You have created a log.
- The "Historical Data" editor is open.
- The Inspector window with the log properties is open.

Procedure

1. Select the log for which you want to define the logging method in the "Historical Data" editor.
2. Select "Properties > Properties > Logging method" in the Inspector window and select the required logging method.
3. If you have selected the "Segmented circular log" type, enter the number of log segments.
   If you selected a log with the "Display system alarm on" setting, specify the level as a percentage at which a system alarm is to be triggered.
   If you selected the "Trigger event" setting, configure the function list in the "Events" group.

Alternatively you can configure the logging method directly in the "Historical Data" editor table. To view hidden columns, activate the column titles using the shortcut menu.

The "Overflow" event is not available in the editor table. You must therefore configure the function list in the Inspector window.

Result

The selected log responds according to the settings in Runtime.

12.3.4 Using Alarms in Runtime

12.3.4.1 Alarms in Runtime

Alarms

Alarms indicate events and states on the HMI device which have occurred in the system, in the process or on the HMI device itself. A status is reported when it is received.

An alarm could trigger one of the following alarm events:

- Incoming
- Outgoing
- Acknowledge
- Loop-in-alarm

The configuration engineer defines which alarms must be acknowledged by the user.

An alarm may contain the following information:

- Date
- Time
- Alarm text
- Location of fault
- Status
- Alarm class
- Alarm number
- Alarm group

Alarm classes

Alarms are assigned to various alarm classes.

- "Warnings"
  Alarms of this class usually indicate states of a plant such as "Motor switched on". Alarms in this class do not require acknowledgement.
- "Errors"
  Alarms in this class must always be acknowledged. Error alarms normally indicate critical errors within the plant such as "Motor temperature too high".
- "System"
  System alarms indicate states or events which occur on the HMI device.
  System alarms provide information on occurrences such as operator errors or communication faults.
- Custom alarm classes
  The properties of this alarm class must be defined in the configuration.

Alarm buffer

Alarm events are saved to an internal buffer. The size of this alarm buffer depends on the HMI device type.

Alarm view

The alarm view shows selected alarms or alarm events from the alarm buffer. Whether alarm events have to be acknowledged or not is specified in your configuration.

Alarm window

The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active.

You can configure the order in which the alarms are displayed. You can choose to display the alarms in ascending or descending order of their occurrence. The alarm window can also be set to indicate the exact location of the fault, including the date and time of the alarm event. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string will be shown.

Alarm indicator

The alarm indicator is a graphic symbol that is displayed on the screen when an alarm of the specified alarm class is activated.

The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated. The number indicates the number of queued alarms.

12.3.4.2 Alarms in Runtime

Alarms

Alarms indicate events and states on the HMI device which have occurred in the system, in the process or on the HMI device itself. A status is reported when it is received.

An alarm could trigger one of the following alarm events:

- Incoming
- Outgoing
- Acknowledge

The configuration engineer defines which alarms must be acknowledged by the user.

An alarm may contain the following information:

- Date
- Time
- Alarm text
- Location of fault
- Status
- Alarm class
- Alarm number
- Alarm group

Alarm classes

Alarms are assigned to various alarm classes.

- "Warnings"
  Alarms of this class usually indicate states of a plant such as "Motor switched on". Alarms in this alarm class do not require acknowledgment.
- "Errors"
  Alarms in this class must always be acknowledged. Error alarms normally indicate critical errors within the plant such as "Motor temperature too high".
- "System"
  System alarms indicate states or events which occur on the HMI device.
  System alarms provide information on occurrences such as operator errors or communication faults.
- "Diagnosis Events"
  SIMATIC diagnostic alarms show states and events in the SIMATIC S7 controller.
- "Safety Warnings"
  The "Safety Warnings" alarm class shows alarms for fail-safe operation. Users do not acknowledge alarms from this alarm class.
- STEP 7 alarm classes
  The alarm classes configured in STEP 7 are also available to the HMI device.
- Custom alarm classes
  The properties of this alarm class are defined in the configuration.

Alarm buffer

Alarm events are saved to an internal buffer. The size of this alarm buffer depends on the HMI device type.

Alarm report

When alarm logging is enabled in the project, alarms are output directly to the printer.

You can set the logging function separately for each alarm. Printing of such an alarm is initiated when the "incoming" and "outgoing" alarm events are generated.

The output of alarms of the "System" alarm class to a printer must be initiated by means of the corresponding alarm buffer. This outputs the content of the alarm buffer to the printer. To be able to initiate this print function, you need to configure an operating element in the project.

Alarm log

Alarm events are stored in an alarm log, provided this log file is configured. The capacity of the log file is limited by the storage medium and system limits.

Alarm view

The alarm view shows selected alarms or events from the alarm buffer or alarm log. Whether alarm events have to be acknowledged or not is specified in your configuration. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string in their alarm text are shown.

Alarm window

The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active. Depending on the configuration, it is not closed until the alarm is acknowledged.

You can configure the order in which the alarms are displayed. At the first position, the current, or the oldest alarm will be displayed. The alarm window can also be set to indicate the exact location of the fault, including the date and time of the alarm event. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string in their alarm text are shown.

Alarm indicator

The alarm indicator is a graphic symbol that is displayed on the screen when an alarm of the specified alarm class is activated.

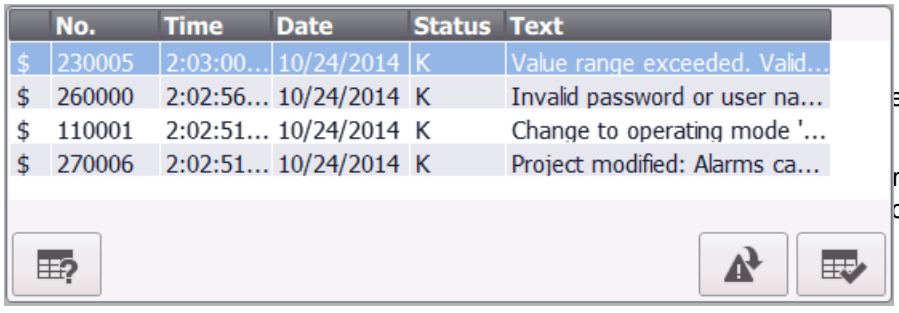
The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated. The number indicates the number of queued alarms.

12.3.4.3 Using the Alarm Window or Alarm View

Alarm window, alarm view in Runtime

Application



evice. The layout

ration the alarm

ctive. Depending

Layout

Depending on the configuration, in the alarm view different columns with information regarding an alarm or an alarm event are displayed. If a filter is configured, only alarms that contain a specific string in the alarm text will be displayed.

To differentiate between the different alarm classes, the first column in the alarm view contains an icon:

Symbol	Alarm class
!	"Errors"
Empty	"Warnings"
depends on the configuration	Custom alarm classes
$	"System"
S7	"Diagnosis Event"
!!	"Safety Warnings"

Operation

You use the alarm view as follows, depending on how it is configured:

- Change the order of the columns.
- Change the order in which the alarms are displayed.
- Acknowledging alarms
- Editing alarms

Operator controls

The buttons have the following functions:

Button	Function
	Displaying a tooltip for an alarm
	Loop-In-Alarm Changes to the screen that contains information about the error event
	Acknowledge alarm

Operation behavior

Linked alarm window for touch panels

When configuring the alarm window for keyboard units, enable the "Connected" (modal) property under "Properties > Mode". This ensures that the alarm window does not defocus during screen changes. Switching back and forth between the screen and different windows with <Ctrl+TAB> is not supported. If the linked alarm window has the focus, then the buttons

in the screen behind it cannot be operated. The functions configured on a function key are carried out.

Changing the order of the displayed alarms

When you click on the column, first of all alarms requiring acknowledgment are sorted according to date and time. Then those alarms that do not require acknowledgment are sorted according to date and time.

Using the Alarm Window, Alarm View

Introduction

As an alternative to using the mouse, you can operate the alarm view and the alarm window using the <TAB> key on your HMI device. This allows you to select the button and the most recently selected alarm in the alarm view. Depending on the configuration, you can also operate the alarm view via the function keys.

Operation using the mouse

1. Select the alarm to be edited.
2. Click the button whose function you want to run.

Operation using the keyboard

1. Press the <Tab> key until the list of displayed alarms is selected in the alarm view.
2. Click on the alarm to be edited. Use the <Up> and <Down> keys accordingly.
3. Press the <Tab> key until the button whose function you wish to use is selected.
4. Press <Enter>.

Change the order of the columns

1. Select the column header, such as the "Date" column header.
2. While holding down the mouse button, drag the column header to the column header "Time". The "Date" column is in front of the "Time" column.

Change the sorting

You can sort the list according to date or time.

1. Click on the corresponding column header.
   The list is sorted in descending or ascending order according to this criterion.
2. Click the same column header again to reverse the sort order.

Acknowledge alarm

1. Click on the alarm to be edited.
2. Click the button.

Loop-In-Alarm trigger

1. Click on the alarm to be edited.
2. Click the button.
   The screen containing information about the alarm is displayed.

Note

If you trigger a Loop-In-Alarm during an unacknowledged alarm, it is acknowledged automatically.

Displaying configured tooltip

1. Click on the alarm concerned.
2. Click the button.
   The tooltip configured for the alarm is displayed.

12.3.4.4 Using the Simple Alarm Window, Alarm View

Basic alarm view, alarm window in Runtime

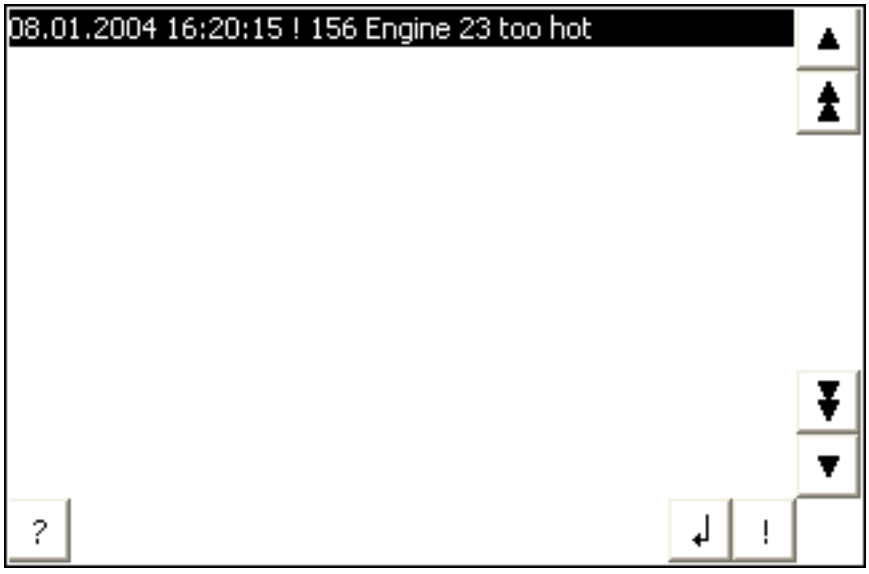
Application

The simple alarm view shows selected alarms or alarm events from the alarm buffer. The layout and operation of the simple alarm window correspond to that of the simple alarm view.

Note

The "Single alarm view" object is not assigned dynamic functions by means of scripting.

In the Engineering System, for example, dynamize the visibility of an object in the "Animations" tab of the Inspector window. In Runtime, the "Simple alarm view" does not support animations. If you configured an animation and, for example, run a consistency check on the project, an error alarm is displayed in the output window.



Layout

Depending on the configuration, in the alarm view different columns with information regarding an alarm or an alarm event are displayed.

To differentiate between the different alarm classes, the first column in the alarm view contains an icon:

Icon	Alarm class
!	"Errors"
empty	"Warnings"
depends on the configuration	Custom alarm classes
$	"System"

Operation

You use the alarm view as follows, depending on how it is configured:

- Acknowledging alarms
- Editing alarms

Control elements

The buttons have the following functions:

Button	Function
!	Acknowledge alarm
↲	Loop-In-Alarm Changes to the screen that contains information about the error event

Button	Function
?	Displaying a tooltip for an alarm
▶	Displays the full text of the selected alarm in a separate window, namely the alarm text window In the alarm text window, you can view alarm texts that exceed the space available in the Alarm view. Close the alarm text window with the ✕ button.
▲	Scrolls one alarm up.
⏫	Scrolls one page up in the alarm view.
⏬	Scrolls one page down in the alarm view.
▼	Scrolls one alarm down.

Format of the control elements

The display of the buttons for using the simple alarm view depends on the configured size. You should therefore check on the HMI device whether all the required buttons are available.

Basic alarm view, using the alarm window

Introduction

As an alternative to using the mouse, you can operate the simple alarm view using the <TAB> key on your HMI device. This allows you to select the button and the most recently selected alarm in the alarm view. Depending on the configuration, you can also operate the alarm view via the function keys.

Operation using the mouse

1. Select the alarm to be edited.
2. Click the button whose function you want to run.

Operation using the keyboard

1. Press the <Tab> key until the list of displayed alarms is selected in the alarm view.
2. Click on the alarm to be edited. Use the <Up> and <Down> keys accordingly.
3. Press the <Tab> key until the button whose function you wish to use is selected.
4. Press <Enter>.

Acknowledge alarm

1. Select the alarm to be acknowledged.
2. Click the ! button.

Loop-In-Alarm trigger

1. Select the alarm to be edited.
2. Click the ↲ button.
   The screen containing information about the alarm is displayed.

Calling the infotext

1. Select the alarm to be edited.
2. Click the ? button.
3. To close the window for displaying the operator note, press the × button or use the key combination <Alt+F4>.

12.3.4.5 Using the Alarm Indicator

Alarm indicator in Runtime

Application

The alarm indicator is displayed if alarms of the specified alarm class are pending or require acknowledgment.

Layout

The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated. The number indicates the number of queued alarms.

Operation

Depending on the configuration, when operating the alarm indicator an alarm window is opened. The icons from the symbol library can only be operated with a mouse or touch screen.

Note

Device dependency

Operation with mouse is not available for all HMI devices.

Operating the alarm indicator using the mouse

Operation using the mouse

1. Click on the alarm indicator with the mouse pointer. Depending on the configuration, the Alarm window is open.
2. Click to close the alarm window. The Alarm window can be opened again by clicking on the alarm indicator.

12.3.4.6 Acknowledging alarms

Introduction

You can acknowledge alarms in Runtime according to your project configuration settings. You can acknowledge alarms as follows:

- Using the display and control object buttons
- Using the "ACK" key on your HMI device
- Using individually-configured function keys or buttons

If an operator authorization is configured for an individual control, the alarms can only be acknowledged by authorized users.

To automatically acknowledge alarms in Runtime, use the system functions and scripts, plus the "Acknowledgment by the PLC" option.

Note

Device dependency

Scripts are not available for all HMI devices.

Acknowledgment variants

You acknowledge individual alarms or multiple alarms together in Runtime. They are distinguished as follows:

- Single acknowledgment
  Acknowledgment of an alarm using a button or a function key.
- Acknowledge alarm groups
  Acknowledgment of all the alarms of an alarm group using a button or a function key.

Requirement

- An alarm is displayed on the HMI device.

Procedure

To acknowledge an alarm, proceed as follows:

1. Select the alarm.
2. If you are using an alarm view or an alarm window, click the button.
3. If you are using a simple alarm view or a simple alarm window, click the button.
4. Press the "ACK" button on your HMI device to acknowledge an alarm in an alarm line.

Note

Device dependency

The extended alarm view or alarm window and the alarm line are not available for all HMI devices.

Result

The alarm status changes to "Acknowledged". If the condition for triggering an alarm no longer applies, the alarm status also changes to "Outgoing", and it is no longer displayed on the HMI device.

12.3.4.7 Filtering Alarms

Introduction

The alarm view in Runtime does not contain a control element for entering filter criteria. To provide this functionality, configure an I/O field using the "Screens" editor.

Requirements

- A filter has been configured for the alarm view.
- An I/O field for entering the criterion is displayed.
- Alarms are displayed in Runtime.

Enable filter

1. Enter the filter string in the I/O field.
2. Confirm your input.
   The alarm view displays only those alarms that are contained in the specified character string. Thereby this distinguishes between upper and lower case letters.

Disable filter

1. Delete the string from the I/O field that is used to filter the alarm view.
2. Confirm your input.
   All alarms are displayed in the alarm view once more.

12.3.5 Reference

12.3.5.1 System functions for alarms

System functions

System functions are predefined functions you can use to implement many tasks in runtime, even with no programming knowledge. You use system functions in a function list or in a script.

Note
Device dependency

Scripts are not available for all HMI devices.

The table shows all the system functions available for displaying and editing alarms.

System function	Effect
EditAlarm	Triggers the Loop-In-Alarm event for all selected alarms.
ClearAlarmBuffer	Deletes alarms from the alarm buffer on the HMI device.
ClearAlarmBufferProtoolLegacy	Function such as "ClearAlarmBuffer". This system function has been retained to ensure compatibility and uses the old ProTool numbering.
AlarmViewUpdate	Updates the alarm view.
AlarmViewEditAlarm	Triggers the event Loop-In-Alarm for all alarms selected in the specified alarm view.
AlarmViewAcknowledgeAlarm	Acknowledges the alarms that are selected in the specified alarm view.
AlarmViewShowOperatorNotes	Displays the configured tooltip for the alarm selected in the specified alarm view.
AcknowledgeAlarm	Acknowledges all selected alarms.
SetAlarmReportMode	Switches the automatic reporting of alarms on the printer on or off.
ShowAlarmWindow	Hides or shows the alarm window on the HMI device.
ShowSystemEvent	Displays the value of the delivered parameter as a system event on the HMI device.

Note

Device dependency

The following system functions are not available for all HMI devices:

- SetAlarmReportMode
- AlarmViewUpdate
- ShowSystemEvent

12.3.5.2 Alarm events

Alarm events and their display and control objects

In Runtime, the following alarm events are triggered by their display and control objects. You can configure a function list for every event.

Object	Configurable events
Alarms	Incoming Outgoing Acknowledge Loop-In-Alarm
Alarm view	Enabling Disable Selection changed
Alarm indicator	Click Click when flashing

Note

Device dependency

The following events in the alarm view are not available for all HMI devices:

- Enabling
- Disable
- Selection changed
- Click
- Click when flashing

12.3.5.3 System functions for logs

System functions

The following system functions are available for logging:

Function name	Function method
ArchiveLogFile	This function moves or copies a log to another storage location for long-term archiving. Use the system function, for example, to move the audit trail from a local storage medium to the server. When handling audit trails, always use the "Move log (hmiMove)" mode in order to avoid noncompliance with FDA guidelines as a result of redundant data storage.
LogTag	Saves the value of the given tags in the given data log. Use the system function to log a process values at a specific time.
StartLogging	Starts the logging process in the specified log. You can interrupt logging in Runtime by calling the "StopLogging" system function.
StopLogging	Stops the logging process in the specified log. To resume logging in Runtime, select the "StartLogging" system function.
ClearLog	Deletes all entries in the specified log.
StartNextLog	Stops the logging process in the specified log. Logging is continued in the sequential log that has been configured for the specified log file.
CloseAllLogs	Closes all logs. The connection between WinCC and the log files or log database is terminated. You can use this system function, for example, to enable hot-swapping of the storage medium on the HMI device without exiting the Runtime software.
OpenAllLogs	Opens all logs to resume logging. The connection between WinCC and the log files or log database is recovered.
CopyLog	Copies the contents of the specified log to another log.

12.3.5.4 Structure of *.csv files that contain alarms

Introduction

You can save an alarm log as a file in "*.csv" format. CSV stands for **C**omma **S**eparated **V**alue. In this format, the table columns that contain the names and the value of the entry are separated by semicolons. Each table row terminates with a line break.

Example of a log file in *.csv format

This example shows a file with logged alarms:

"Time_ms";"MsgProc";"StateAfter";"MsgClass";"MsgNumber";"Var1";...;"Var8";"TimeString";"MsgText";"PLC"37986550590,27;1;1;3;110001;"";...;"";"30.06.99 13:12:51";"Change to operating mode 'online'";37986550682,87;1;1;3;140010;"";...;"";"30.06.99 13:12:59";"Connection established: PLC_1, Station 2, Rack 0, Position 2";

Structure of a log file in *.csv format

The following values are entered in the log file columns:

Parameters	Description
Time_ms	Specify a time stamp as a decimal value (see below for conversion)
Msg_Proc	Alarm procedures: 0 = Unknown alarm procedure 1 = System event 2 = Alarm bit procedure (operating alarms) 3 = Alarm number procedure ALARM_S 4 = Diagnostic event 7 = Analog alarm procedure 100 = Alarm bit procedure (fault alarms)
State after	Alarm event: 0 = Incoming/Outgoing 1 = Incoming 2 = Incoming/Acknowledged/Outgoing 3 = Incoming/Acknowledged 4 = All alarms in the PLC were deleted with SFC106 or after PLC STOP/RUN an alarm is pending 6 = Incoming/Outgoing/Acknowledged
Msg_Class	Alarm class: 0 = No alarm class 1 = "Errors" 2 = "Warnings" 3 = "System" 4 = "Diagnostic Events" 64 ... = user-configured alarm classes
Msg Number	Alarm number
Var1 to Var8	Value of the trigger tag as a STRING
Time string	Time stamp, as STRING in legible data format
Msg text	Alarm in a readable STRING
PLC	Alarm localization (relevant PLC)

Conversion of the time stamp decimal value

To further process the time stamp value using a different program, proceed as follows:

1. Divide Time_ms by 1,000,000.
   Example: 37986476928 : 1,000,000 = 37986.476928
2. The whole number portion (37986) is the date calculated from 12/31/1899.
   In Excel you can now convert the time stamp into days. To do this, assign to the cell that contains the time stamp, a respective format form the "Date" group.
   Result: 37986 results in 12/31/2003
3. The value after the point (0.476928) indicates the time:
   - Multiply the value (0.476928) by 24 to obtain the hours (11.446272).
   - Multiply the remainder (0.446272) by 60 results in the minutes (26.77632).
   - Multiply the remainder (0.77632) by 60 results in the seconds (46.5792).

   Result 11:26:46.579
   This conversion is supported by Microsoft Excel, for example.

12.3.5.5 System events

Basics on system events

System events

System events on the HMI device provide information about internal states of the HMI device and PLC.

The following overview illustrates when a system event occurs and how to eliminate the cause of error.

Note

HMI device dependency

Some of the system events described in this section apply to the individual HMI devices based on their scope of functions.

Note

System events are output in an alarm view. System events are output in the language currently set on your HMI device.

System event parameters

System events may contain encrypted parameters. The parameters are of relevance when troubleshooting because they provide a reference to the source code of the Runtime software. These parameters are output after the "Error code:" text.

10000 - Printer alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below.

Table 12-2 10000 - Printer alarms

Number	Effect/cause	Remedy
10000	The print job could not be started or was canceled due to an unknown error. Faulty printer setup. Or: No authorization is available for accessing the network printer. Power supply failure during data transfer.	Check the printer settings, cable connections and the power supply. Set up the printer once again. Obtain a network printer authorization. If the error persists, contact the Hotline!
10001	No printer is installed or a default printer has not been set up.	Install a printer and/or select it as the default printer.
10002	Overflow of the graphics buffer for printing. Up to two graphics are buffered.	Allow sufficient intervals between successive print jobs.
10003	Graphics can now be buffered again.	--
10004	Overflow of the buffer for printing lines in text mode (e.g. alarms). Up to 1000 lines are buffered.	Allow sufficient intervals between successive print jobs.
10005	Text lines can now be buffered again.	--
10006	The Windows printing system reports an error. Refer to the output text and the error ID to determine the possible causes. Nothing is printed or the print is faulty.	Repeat the action if necessary.

20000 - Global script alarms

Meaning of the system events

All system events that can be displayed are listed below. The system events are divided into different ranges.

Table 12-3 20000 - Global script alarms

Number	Effect/causes	Remedy
20010	An error has occurred in the specified script line. Execution of the script was therefore aborted. Note the system event that may have occurred prior to this.	Select the specified script line in the configuration. Ensure that the tags used are of the allowed types. For system functions, verify the correct number of parameters and the parameter types.
20011	An error has occurred in a script that was called by the specified script. Execution of the script was therefore aborted in the called script. Note the system event that may have occurred prior to this.	In the configuration, select the script that has been called directly or indirectly by the specified script. Ensure that the tags used are of the allowed types. For system functions, verify the correct number of parameters and the parameter types.
20012	Inconsistent configuration data. The script could therefore not be generated.	Recompile the configuration data.
20013	Incorrect installation of the scripting component of WinCC Runtime. Therefore, no scripts can be executed.	Reinstall WinCC Runtime on your PC. Recompile your project using the "Compile > Software (compile all)" command from the context-sensitive menu and then download the project to the HMI device.
20014	The system function returns a value that is not written in any return tag.	Select the specified script in the configuration. Check whether the script name has been assigned a value.
20015	Too many successive scripts have been triggered in short intervals. When more than 20 scripts are queued for processing, any subsequent scripts are rejected. In this case, the script indicated in the alarm is not executed.	Check what is triggering the scripts. Extend the times, e.g. the acquisition cycle of the tag initiating the script.

30000 - Alarms errors when using system functions

Meaning of the system events

All system events that can be displayed are listed below. The system events are divided into different ranges.

Table 12-4 30000 - Alarms errors when using system functions

Number	Effect/causes	Remedy
30010	The tag could not accept the function result, e.g. when it has exceeded the value range.	Check the tag types of the system function parameters.
30011	A system function could not be executed because the function was assigned an invalid value or type in the parameter.	Check the parameter value and tag type of the invalid parameter. If a tag is used as a parameter, check its value.
30012	A system function could not be executed because the function was assigned an invalid value or type in the parameter.	Check the parameter value and tag type of the invalid parameter. If a tag is used as a parameter, check its value.

40000 - Linear scaling alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below. The system alarms are divided into different ranges:

Table 12-5 40000 - Linear scaling alarms

Number	Effect/causes	Remedy
40010	The system function could not be executed since the parameters could not be converted to a common tag type.	Check the parameter types in the configuration.
40011	The system function could not be executed since the parameters could not be converted to a common tag type.	Check the parameter types in the configuration.

50000 - Data server alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below. The system alarms are divided into different ranges:

Table 12-6 50000 - Data server alarms

Number	Effect/causes	Remedy
50000	The HMI device is receiving data faster than it is capable of processing. Therefore, no further data is accepted until all current data have been processed. Data exchange then resumes.	--
50001	Data exchange has been resumed.	--

60000 - Win32 function alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 12-7 60000 - Win32 function alarms

Number	Effect/causes	Remedy
60000	This alarm is generated by the "ShowSystemEvent" system function. The text to be displayed is transferred to the function as a parameter.	--
60010	The file could not be copied in the direction defined because one of the two files is currently open or the source/target path is not available. It is possible that the Windows user has no access rights to one of the two files.	Restart the system function or check the paths of the source/target files. Under Windows: Users executing WinCC Runtime must be granted access to the files.
60011	An attempt was made to copy a file to itself. It is possible that the Windows user has no access rights to one of the two files.	Check the path of the source/target file. In Windows with NTFS: Users executing WinCC Runtime must be granted access to the files.

70000 - Win32 function alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 12-8 70000 - Win32 function alarms

Number	Effect/causes	Remedy
70010	The application could not be started because it could not be found in the path specified or there is insufficient memory space.	Check whether the application exists in the specified path or close other applications.
70011	The system time could not be modified. The error message only appears in connection with area pointer "Date/time PLC". Possible causes: • An invalid time was transferred in the job mailbox. • The Windows user has no right to modify the system time. If the first parameter in the system event is displayed with the value 13, the second parameter indicates the byte containing the incorrect value.	Check the time that is to be set. Under Windows: Users running WinCC Runtime must be granted the right to set the system time of the operating system.
70012	Error when executing the system function "StopRuntime" with the "Runtime and operating system" option. Windows and WinCC Runtime are not closed. The error may have been generated because other programs cannot be closed.	Close all programs currently running. Now close Windows.
70013	The system time could not be modified because an invalid value was entered. Incorrect separators may have been used.	Check the time which is to be set.

Number	Effect/causes	Remedy
70014	The system time could not be modified. Possible causes: • An invalid time was transferred. • The Windows user has no right to modify the system time. Windows rejects the setting request.	Check the time that is to be set. Under Windows: Users running WinCC Runtime must be granted the right to set the system time of the operating system.
70015	The system time could not be read because Windows rejects the reading function.	--
70016	An attempt was made to select a screen by means of a system function or job. This is not possible because the configured screen number does not exist. Or: A screen could not be generated due to insufficient system memory. Or: The screen is blocked. Or: Screen call has not been executed correctly.	Check the screen number in the system function or job against the screen numbers configured. Assign the number to a screen if necessary. Check the details for the screen call and whether the screen is blocked for specific users.
70017	Date/time is not read from the area pointer because the address set in the PLC is either not available or has not been set up.	Change the address or set up the address in the PLC.
70018	Acknowledgment that the password list has been successfully imported.	--
70019	Acknowledgment that the password list has been successfully exported.	--
70020	Acknowledgment for activation of alarm reporting.	--
70021	Acknowledgment for deactivation of alarm reporting.	--
70022	Acknowledgment to starting the Import Password List action.	--
70023	Acknowledgment to starting the Export Password List action.	--
70024	The tag value range was exceeded during system function execution. No calculation of the system function.	Check and correct the calculation.
70025	The tag value range was exceeded during system function execution. No calculation of the system function.	Check and correct the calculation.
70026	No other screens are stored in the internal screen memory. No other screens can be selected.	--
70027	The backup of the RAM file system has been started.	--
70028	RAM file system backup is complete. The files from the RAM have been copied to the Flash memory. Following a restart, these saved files are copied back to the RAM file system.	--
70029	Backup of the RAM file system has failed. No backup copy of the RAM file system has been made.	Check the settings in the "Control Panel > OP" dialog and save the RAM file system using the "Save Files" button in the "Persistent Storage" tab.
70030	The parameters configured for the system function are faulty. The connection to the new PLC was not established.	Compare the parameters configured for the system function with the parameters configured for the PLCs and correct them as necessary.

Number	Effect/causes	Remedy
70031	The PLC configured in the system function is not an S7 PLC. The connection to the new PLC has not been established.	Compare the S7 PLC name parameter configured for the system function with the parameters configured for the PLC and correct them as necessary.
70032	The object configured with this number in the tab sequence is not available in the selected screen. The screen changes but the focus is set to the first object.	Check the number of the tab sequence and correct it if necessary.
70033	An e-mail could not be sent because a TCP/IP connection to the SMTP server no longer exists. This system event is only generated at the first failed attempt. All subsequent unsuccessful attempts to send an e-mail will no longer generate a system event. The event is only generated again if an e-mail has been successfully sent in the meantime. The central e-mail component in WinCC Runtime attempts to connect to the SMTP server at regular intervals (1 minute) in order to transmit the remaining e-mails.	Check the network connection to the SMTP server and re-establish it if necessary.
70034	Following a disruption, the TCP/IP connection to the SMTP server was successfully re-established. The queued e-mails are then sent.	--
70036	No SMTP server for sending e-mails is configured. An attempt to connect to an SMTP server therefore fails and it is not possible to send e-mails. WinCC Runtime generates the system event at the first attempt to send an e-mail.	Configure an SMTP server: In the WinCC Engineering System using "Device settings > Device settings" In the Windows CE operating system using "Control Panel > Internet Settings > E-mail > SMTP Server"
70037	An e-mail could not be sent for unknown reasons. The contents of the e-mail are discarded.	Check the e-mail parameters (recipient etc.).
70038	The SMTP server has rejected the sending or forwarding of an e-mail because the domain of the recipient is unknown to the server or because the SMTP server requires authentication. The contents of the e-mail are discarded.	Check the domain of the recipient address or disable the authentication on the SMTP server if possible. SMTP authentication is currently not used in WinCC Runtime.
70039	The syntax of the e-mail address is incorrect or contains invalid characters. The contents of the e-mail are discarded.	Check the e-mail address of the recipient.
70040	The syntax of the e-mail address is incorrect or contains illegal characters.	--
70041	The import of the user management has been aborted due to an error. Nothing has been imported.	Check your user administration or download it again to the panel.
70042	The range of values of the tag was exceeded while executing the system function. The system function was not calculated.	Check and correct the calculation.
70043	The range of values of the tag was exceeded while executing the system function. The system function was not calculated.	Check and correct the calculation.
70044	An error occurred while sending the e-mails. The e-mails were not sent.	Check the SMTP settings and the error message in the system event.

Number	Effect/causes	Remedy
70045	Cannot load a file required for encrypting the e-mail.	Update the operating system and Runtime.
70046	The server does not support encryption.	Select an SMTP server that supports encryption.
70047	The SSL versions of the HMI device and SMTP server may not be compatible.	Contact your network administrator or the operator of the SMTP server.

80000 - Log alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 12-9 80000 - Log alarms

Number	Effect/causes	Remedy
80001	The log specified is filled to the size defined (in percent) and must be stored elsewhere.	You swap out the file or table by executing a ‘move’ or ‘copy’ function.
80002	A line is missing in the specified log.	--
80003	The copying process for logging was not successful. In this case, it is advisable to check any subsequent system events, too.	--
80006	Since logging is not possible, this causes a permanent loss of the functionality.	In the case of databases, check whether the corresponding data source exists and start up the system again.
80009	A copying action has been completed successfully.	--
80010	An incorrect storage location was entered in WinCC and causes permanent loss of functionality.	Configure the storage location for the respective log again and restart the system when the full functionality is required.
80012	Log entries are stored in a buffer. If the values are read to the buffer faster than they can be physically written (using a hard disk, for example), overloading may occur and recording is then stopped.	Archive fewer values. Or: Increase the logging cycle.
80013	The overload status no longer exists. Archiving resumes the recording of all values.	--
80014	The same action was triggered twice in quick succession. Since the process is already in operation, the action is only carried out once.	--
80015	This system event is used to report DOS or database errors to the user.	--
80016	The logs are separated by system function "CloseAllLogs" and the incoming entries exceed the defined buffer size. All buffer entries are deleted.	Reconnect the logs.
80017	The number of incoming entries cause a buffer overflow. This can be caused, for example, by several copying actions being activated at the same time. All copy jobs in the buffer are deleted.	Stop the copy action.

Number	Effect/causes	Remedy
80019	The connection between WinCC and all log files was shut down, for example, after execution of the "CloseAllLogs" system function. Entries are written to the buffer and written to the logs when the connection is recovered. There is no connection to the storage location and the storage medium can be changed.	--
80020	The maximum number of simultaneously copy operations has been exceeded. Copying is not executed.	Wait until the current copying actions have been completed, then restart the last copy action.
80021	An attempt was made to delete a log which is still busy with a copy action. Deletion has not been executed.	Wait until the current copying actions have been completed, then restart the last action.
80022	An attempt was made to use system function "StartSequenceLog" to start a sequence log for a log which is not configured as sequence log. No sequence log file is created.	In the project, check • whether the "StartSequenceLog" system function was properly configured. • if the tag parameters are properly provided with data on the HMI device.
80023	An attempt was made to copy a log to itself. The log is not copied.	In the project, check • whether the "CopyLog" system function was properly configured. • if the tag parameters are properly provided with data on the HMI device.
80024	In your configuration, you specified that the "CopyLog" system function should not allow copying if the destination log already contains data ("Mode" parameter). The log is not copied.	Edit the "CopyLog" system function in your configuration, if necessary. Before you initiate the system function, delete the destination log file.
80025	You have canceled the copy operation. Data written up to this point are retained. The target log file (if configured) was not deleted. The cancellation is reported by an error entry RT_ERR at the end of the target log.	--
80026	This alarm is output after all logs are initialized. Values are written to the logs from then on. No entries are written to the logs before this time has expired, irrespective of the active state of WinCC Runtime.	--
80027	The internal Flash memory has been specified as the storage location for a log. This is not permissible. No values are written to this log and the log file is not created.	Configure "Storage Card" or network path as the storage location.
80028	The alarm serves as status report indicating that the logs are currently being initialized. No values are logged until the alarm 80026 is output.	--
80029	The number of logs specified in the alarm could not be initialized. Initialization of logs was finished. The faulty log files are not available for logging jobs.	Evaluate the additional system events related to this alarm. Check the configuration, the ODBC (Open Database Connectivity), and the specified drive.
80030	The structure of the existing log file does not match the expected structure. Logging is stopped for this log.	Delete the existing log data manually, in advance.
80031	The log in CSV format is corrupted. The log cannot be used.	Delete the faulty file.

Number	Effect/causes	Remedy
80032	Logs can be configured with events. These are triggered as soon as the log is full. Assuming WinCC Runtime is started and the log is already full, the event would never be triggered. The specified log is full and no longer accepts data.	Close WinCC Runtime, delete the log, and restart WinCC Runtime. Or: Configure a button which contains the same actions as the event and press it.
80033	"System Defined" is set in the data log file as the data source name. This causes an error. No data is written to the database logs, whereas the logging to the CSV logs works.	Reinstall SQL Sever 2005 Express.
80034	An error has occurred in the initialization of the logs. An attempt has been made to create the tables as a backup. This action was successful. A backup copy of the tables of the corrupted log file was generated and the cleared log restarted.	No action is necessary. However, it is recommended to save the backup files or delete them in order to make the space available again.
80035	An error has occurred in the initialization of the logs. An attempt has been made to create backups of the tables and this has failed. No logging or backup has been performed.	It is recommended to save the backups or to delete them in order to release memory.
80044	The export of a log was interrupted because Runtime was closed or due to a power failure. At the restart of Runtime, it was detected that the export must be resumed.	The export resumes automatically.
80045	The export of a log was interrupted due to an error in the connection to the server or at the server itself.	The export is repeated automatically. Check: • The connection to the server. • If the server is running. • If there is enough free space on the server.
80046	The destination file or the associated directory could not be created on the server.	Check whether there is enough space on the server and it you have permission to create the log file.
80047	The log could not be read while exporting it.	Check whether the storage medium is correctly inserted.
80049	The log could not be renamed while preparing to export it. The job can not be completed."	Check whether the storage medium is correctly inserted and if there is sufficient space on the medium.
80050	The log which shall be exported is not closed. The job can not be completed.	Make sure to call the "CloseAllLogs" system function before using the "ExportLog" system function. Change the configuration as required.
80051	The log to be copied contains an invalid checksum. The log was not copied.	Select a log with a valid checksum. The selected log may have been manipulated.
80052	The log cannot be read.	Check the log and the specified path.
80053	The closed log cannot be read.	Open the log.

90000 - FDA alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 12-10 90000 - FDA alarms

Number	Effect/causes	Remedy
90024	No operator actions can be logged due to lack of space on the storage medium for log. The operator action will therefore not be executed.	Provide more storage space by inserting a blank storage medium, or backup the log files to the server using the "ExportLog" function.
90025	No user actions can be logged because of error state of the archive. Therefore the user action will not be executed.	Check whether the storage medium is correctly inserted.
90026	No operator actions can be logged because the log is closed. The operator action will therefore not be executed.	Before further operator actions are carried out, the log must be reopened with the help of system function "OpenAllLogs". Change the configuration as required.
90028	The password you entered is incorrect.	Enter the correct password.
90029	Runtime was closed during ongoing operation (perhaps due to a power failure) or a storage medium in use is incompatible with Audit Trail. An Audit Trail is not suitable if it belongs to another project or has already been logged. It could also be that archiving was not stopped before automatic execution of the "CloseAllArchives" function was started (e.g. by the task scheduler).	Ensure that you are using the correct storage medium.
90030	Runtime was closed during ongoing operation (perhaps due to a power failure).	--
90031	Runtime was closed during ongoing operation (perhaps due to a power failure).	--
90032	Running out of space on the storage medium for log.	Provide more storage space by inserting a blank storage medium, or backup the log files to the server using the "ExportLog" function.
90033	No more space on the storage medium for log. As of now, no more operator actions requiring logging will be executed.	Provide more storage space by inserting a blank storage medium, or backup the log files to the server using the "ExportLog" function.
90039	You do not have the necessary authorization to perform this action.	Adapt or upgrade your authorizations.
90040	Audit Trail is switched off because of a forced user action.	Reactivate the "Audit Trail" with the help of system function "StartLog".
90041	A user action which has to be logged has been executed without a logged on user.	A user action requiring logging should only be possible with permission. Change the configuration by setting a required authorization for the input object.
90044	A user action which has to be confirmed was blocked, because there is another user action pending.	Repeat the user action if necessary.
90048	The Audit Trail cannot be printed while data relevant to the audit is being logged.	Stop logging by calling system function "StopLogging".
90049	Access to required file is not possible.	Check the network connection or the storage medium.
90056	The recipe was not imported because the file contains no checksum.	Select a file with a checksum. You can also disable checksum verification by calling system function "ImportDataRecords".
90057	The recipe was not imported because the file contains an invalid checksum. The selected file may have been manipulated.	Select a file with a valid checksum.

110000 - Offline function alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 12-11 110000 - Offline function alarms

Number	Effect/causes	Remedy
110000	The operating mode was changed. "Offline" mode is now set.	--
110001	The operating mode was changed. "Online" mode is now set.	--
110002	The operating mode was not changed.	Check the connection to the PLCs. Check whether the address range for the “Coordination" area pointer is present in the PLC.
110003	The operating mode of the specified PLC was changed by the "SetConnectionMode" system function. The "offline" operating mode is now set.	--
110004	The operating mode of the specified PLC was changed by the "SetConnectionMode" system function. The "online" operating mode is now set.	--
110005	An attempt was made to use the "SetConnectionMode" system function to set the specified PLC to "online" mode, although the entire system is in "offline" mode. This changeover is not allowed. The PLC remains in "offline" mode.	Switch the complete system to "online" mode and repeat execution of the system function.
110006	The content of the "project ID" area pointer does not match the project ID configured in WinCC. The WinCC Runtime is therefore terminated.	Check: • the project ID entered on the PLC. • the project ID entered in WinCC.

120000 - Trend alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below.

Table 12-12 120000 - Trend alarms

Number	Effect/causes	Remedy
120000	The trend is not displayed because you configured an incorrect axis to the trend or an incorrect trend.	Change the configuration.
120001	The trend is not displayed because you configured an incorrect axis to the trend or an incorrect trend.	Change the configuration.
120002	The trend is not displayed because the tag assigned attempts to access an invalid PLC address.	Check whether the data area for the tag exists in the PLC, the configured address is correct and the value range for the tag is correct.

130000 - System information alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 12-13 130000 - System information alarms

Number	Effect/causes	Remedy
130000	The action was not executed.	Close all other programs. Delete files no longer required from the hard disk.
130001	The action was not executed.	Delete files no longer required from the hard disk.
130002	The action was not executed.	Close all other programs. Delete files no longer required from the hard disk.
130003	No data medium found. The operation is canceled.	Check the following conditions, for example: • Access to the correct data carrier? • Is a data carrier inserted?
130004	The data carrier is write-protected. The operation is canceled.	Check whether the correct data medium is being accessed. Remove the write-protection if necessary.
130005	The file is read only. The operation is canceled.	Check whether the correct file is being accessed. Change the file attributes.
130006	No access to the file. The operation is canceled.	Check the following conditions, for example: • Is the correct file being accessed? • Does the file exist? • Is another action preventing concurrent access to the file?
130007	The network connection is interrupted. Records cannot be saved or read via the network connection.	Check the network connection and eliminate the cause of error.
130008	No storage card available. The specified records cannot be saved to or read from Storage Card.	Insert the storage card.
130009	The specified folder does not exist on the storage card. The files stored in this directory are not backed up after switching off the HMI device.	Insert the storage card.
130010	The maximum nesting depth can be exhausted when, for example, value changes in a script initiate the call of an infinite number of further scripts. The configured functionality is not provided.	Check the configuration.
130013	No storage card available. The specified records cannot be saved to or read from Storage Card.	Insert the storage card.

140000 - Connection alarms: Connection + device

Meaning of the system events

All system events that can be displayed are listed below.

Table 12-14 140000 - Connection alarms: Connection + device

Number	Effect/causes	Remedy
140000	An online connection to the PLC is established.	--
140001	The online connection to the PLC was shut down.	--
140003	No tag update or write operations are executed.	Check whether the connection is up and the PLC is switched on. In the Control Panel, check the set parameters using the "Set PG/PC interface" function. Restart the system.
140004	No tag update or write operations are executed due to an incorrect access point, or incorrect module configuration.	Check the connection and whether the PLC is switched on. Check the access point or the module configuration (MPI, PPI, PROFIBUS) in the Control Panel with "Set PG/PC interface". Restart the system.
140005	No tag update or write operations are executed due to an incorrect HMI device address (possibly too high).	Use a different address for the HMI device. Check the connection and whether the PLC is switched on. Check the parameters set in the "Set PG/PC interface" dialog of the Control Panel. Restart the system.
140006	No tag update or write operations are executed due to an incorrect baud rate setting.	Select a different baud rate in WinCC (according to module, profile, communication peer, etc.).
140007	An incorrect bus profile prevents tag updates or write operations (see %1). The following parameters could not be written to the registry database: 1: Tslot 2: Tqui 3: Tset 4: MinTsdr 5: MaxTsdr 6: Trdy 7: Tid1 8: Tid2 9: Gap Factor 10: Retry Limit	Check the custom bus profile. Check the connection and whether the PLC is switched on. Check the parameters set in the "Set PG/PC interface" dialog of the Control Panel. Restart the system.

Number	Effect/causes	Remedy
140008	An incorrect baud rate prevents tag updates or write operations. The following parameters could not be written to the registry database: 0: General error 1: Wrong version 2: Profile cannot be written to the registry database. 3: The subnet type cannot be written to the registry database. 4: The Target Rotation Time cannot be written to the registry database. 5: Incorrect Highest Station Address (HSA).	Check whether the connection is up and the PLC is switched on. In the Control Panel, check the set parameters using the "Set PG/PC interface" function. Restart the system.
140009	No tag updates or write operations because the S7 communication module was not found.	To reinstall the module, open the Control Panel and select "Set PG/PC interface".
140010	No S7 communication partner found because the PLC is shut down. DP/T: The option "PG/PC is the only master" is not set in the Control Panel under "Set PG/PC interface".	Switch on the PLC. DP/T: If only one master is connected to the network, disable "PG/PC is the only master" in "Set PG/PC interface". If several masters are connected to the network, enable these. Do not change any settings, for this will cause bus errors.
140011	No tag updates or write operations because communication is down.	Check the connection and whether the communication partner is switched on.
140012	Initialization error (e.g. if WinCC Runtime was closed in Task Manager). Or: Another application (e.g. STEP 7) with different bus parameters is active and the driver cannot be started with the new bus parameters (baud rate, for example).	Restart the HMI device. Or: Start WinCC Runtime and then start your other applications.
140013	The MPI cable is disconnected and, therefore, there is no power supply.	Check the connections.
140014	The configured bus address is in use by another application.	Change the HMI device address in the PLC configuration.
140015	Incorrect baud rate Or: Incorrect bus parameters (e.g. HSA) Or: OP address > HSA or: Incorrect interrupt vector (interrupt not registered by the driver)	Correct the parameters.
140016	The hardware does not support the configured interrupt.	Change the interrupt number.
140017	The set interrupt is in use by another driver.	Change the interrupt number.
140018	SIMOTION Scout disabled the consistency check. Only a corresponding note appears.	In SIMOTION Scout, reactivate the consistency check and once again download the project to the PLC.
140019	SIMOTION Scout is downloading a new project to the PLC. Connection to the PLC is canceled.	Wait until the end of the reconfiguration.
140020	Mismatch of the PLC and project (FWX file) versions. The connection to the PLC is canceled.	The following remedies are available: Download the current version to the PLC using SIMOTION Scout. Recompile the project using WinCC ES, close WinCC Runtime, and restart with the new configuration.

Number	Effect/causes	Remedy
140021	Setup of the connection to the PLC failed. Incorrect configuration of the "Access password" for the connection to the PLC.	Select the "Access password" area in the "Connections" editor to check the password entered for the connection to the PLC. Assign the correct password. The "Password" for the connection to the PLC is assigned in the "Security" area of the PLC properties.
140022	Setup of the connection to the PLC failed. Incorrect configuration of the access password for the connection to the PLC.	Select the "Access password" area in the "Connections" editor to check the password entered for the connection to the PLC. The "Password" for the connection to the PLC is assigned in the "Security" area of the PLC properties.
140023	Error during time synchronization: Unable to read system time of PLC %1.	Check whether the PLC is switched on. Select the "Access password" area in the "Connections" editor to check the password entered for the connection to the PLC.
140025	Setup of the connection to the PLC failed. The access password of the PLC is disabled in the PLC display.	Enable the access password in the PLC display.

160000 - Connection alarms: OPC: Connection

Meaning of the system events

All system events that can be displayed are listed below.

Table 12-15 160000 - Connection alarms: OPC: Connection

Number	Effect/causes	Remedy
160000	No more data is read or written. Possible causes: • The cable connection is interrupted. • The PLC does not respond or has failed, etc. • The wrong port is used for the connection. • System overload	Check whether the cable is plugged in and the PLC is OK, and whether the correct port is used. Restart the system if the system event persists.
160001	The connection is recovered because the cause of the interruption has been eliminated.	--
160010	No connection to the server because the server ID (CLS-ID) cannot be determined. Values cannot be read or written.	Check access rights.
160011	No connection to the server because the server ID (CLS-ID) cannot be determined. Values cannot be read or written.	Check the following conditions, for example: • Correct server name? • Correct station name? • Is the server registered?

Number	Effect/causes	Remedy
160012	No connection to the server because the server ID (CLS-ID) cannot be determined. Values cannot be read or written.	Check the following conditions, for example: • Correct server name? • Correct station name? • Is the server registered? Note for advanced users: Interpret the value from HRESULT.
160013	The specified server was started as InProc server. This has not been released and may lead to undefined behavior because the server is running in the same process area as WinCC Runtime.	Configure the server as OutProc Server or Local Server.
160014	Only one OPC server project can be started on a PC/MP. An alarm is output after an attempt was made to start a second project. The second project has no OPC server functionality and cannot be located as an OPC server by external sources.	Do not start a second project with OPC server functionality on the computer.

170000 - S7 dialog alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below.

Table 12-16 170000 - S7 dialog alarms

Number	Effect/causes	Remedy
170000	S7 diagnostics events are not indicated because it is not possible to log on to the S7 diagnostics functions at this device. The service is not supported.	--
170001	The S7 diagnostics buffer cannot be viewed because communication with the PLC is shut down.	Set the PLC to online mode.
170002	The S7 diagnostics buffer cannot be viewed because reading of the diagnostics buffer (SSL) was canceled with error.	--
170003	An S7 diagnostics event cannot be visualized. The system returns internal error %2.	--
170004	An S7 diagnostics event cannot be visualized. The system returns an internal error of error class %2, error number %3.	--
170007	It is not possible to read the S7 diagnostics buffer (SSL) because this operation was canceled with an internal error of class %2 and error code %3.	--

180000 - General alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 12-17 180000 - General alarms

Number	Effect/causes	Remedy
180000	A component/OCX received configuration data with a version ID which is not supported.	Install a newer component.
180001	System overload because too many actions are running in parallel. Certain actions can be executed, while others are discarded.	Several remedies are available: • Generate alarms at a slower rate (polling). • Initiate scripts and functions at greater intervals. If the alarm appears more frequently: Restart the HMI device.
180002	The screen keyboard could not be activated. Possible causes: "TouchInputPC.exe" was not registered due to faulty Setup.	Reinstall WinCC Runtime.

190000 - Tag alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 12-18 190000 - Tag alarms

Number	Effect/causes	Remedy
190000	It is possible that the tag is not updated.	--
190001	The tag is updated after the cause of the last error state has been eliminated (recovery of normal operation).	--
190002	The tag is not updated because communication with the PLC is down.	Select system function "SetOnline" to enable communication.
190004	The tag is not updated because the address configured for this tag does not exist.	Check the configuration.
190005	The tag is not updated because the configured PLC type does not exist for this tag.	Check the configuration.
190006	The tag is not updated because it is not possible to map the PLC type in the data type of the tag.	Check the configuration.
190007	The tag value is not modified because the connection to the PLC is interrupted or the tag is offline.	Set online mode or reconnect to the PLC.

Number	Effect/causes	Remedy
190008	The configured tag limits were violated due to one of the following events: • Value input • System function • Script	Observe the configured or current tag limits.
190009	An attempt was made to assign this tag a value that is outside the valid range of values for this data type. For example, input of the value 260 for a byte tag, or input of the value -3 for an unsigned word tag.	Observe the range of values for the data type of the tags.
190010	The rate at which values are written to the tag is too high (e.g. initiated in a script loop). Values will be lost because buffer capacity is limited to 100 operations.	The following remedies are available: • Extend the interval between multiple write actions. • Do not use an array tag longer than 6 words when configuring an acknowledgment on the HMI device using "HMI acknowledgment tag".
190011	Possible cause 1: The value entered could not be written to the configured PLC tag because the high or low limit was exceeded. The system discarded the entry and restored the original value. Possible cause 2: The connection to the PLC was interrupted.	Note that the value entered must be within the range of values of the control tag. Check the connection to the PLC.
190012	It is not possible to convert a value from a source format to a target format, for example: An attempt is being made to write a counter value that is outside the valid, PLC-specific range of values. A tag of the type Integer should be assigned a value of the type string.	Check the range of values, or the data type of the tag.
190013	You entered a string that exceeds the tag length. The string is truncated automatically to a valid length.	Always enter strings that do not exceed the valid tag length.

190100 - Area pointer alarms

190100 - Area pointer alarms

Number	Effect/causes	Remedy
190100	The area pointer is not updated because the address configured for this pointer does not exist. Type 1 Warnings 2 Errors 3 PLC acknowledgment 4 HMI device acknowledgment 5 LED image 6 Trend request 7 Trend transfer 1 8 Trend transfer 2 No.: Consecutive number displayed in WinCC ES.	Check the configuration.
190101	The area pointer is not updated because it is not possible to map the PLC type to the area pointer type. Parameter type and no.: see alarm 190100	--
190102	The area pointer is updated after the cause of the last error state has been eliminated (recovery of normal operation). Parameter type and no.: See alarm 190100.	--

200000 - PLC coordination alarms

200000 - PLC coordination alarms

Number	Effect/causes	Remedy
200000	Coordination is not executed because the address configured in the PLC does not exist/is not set.	Change the address or set up the address in the PLC.
200001	Coordination is canceled because the write access to the address configured in the PLC is not possible.	Change the address or set the address in the PLC at an area which allows write access.
200002	Coordination is not carried out at the moment because the address format of the area pointer does not match the internal storage format.	Internal error
200003	Coordination can be executed again because the last error state is eliminated (return to normal operation).	--
200004	The coordination may not be executed.	--
200005	No more data is read or written. Possible causes: • The cable is defective. • The PLC does not respond, is defective, etc. • System overload	Ensure that the cable is plugged in and the PLC is operational. Restart the system if the system alarm persists.

210000 - PLC job alarms

210000 - PLC job alarms

Number	Effect/causes	Remedy
210000	Jobs are not processed because the configured address does not exist/has not been set up in the PLC.	Change the address or set up the address in the PLC.
210001	Jobs are not processed because read/write access to the configured address is not possible in the PLC.	Change the address, or set up the address in a PLC area at which read/write access is possible.
210002	Jobs are not executed because the address format of the area pointer does not match the internal storage format.	Internal error
210003	The job buffer is processed again because the last error status has been eliminated (recovery of normal operation).	--
210004	The job buffer is possibly not going to be processed.	--
210005	A job mailbox with invalid number was initiated.	Check the PLC program.
210006	An error occurred while executing the job mailbox. As a result, the control job is not executed. Observe the next/previous system event.	Check the parameters of the control job. Recompile the configuration data.

220000 - WinCC communication driver alarms

220000 - WinCC communication driver alarms

Number	Effect/causes	Remedy
220001	The tag is not downloaded because write access to data type Bool/Bit is not supported by the sublevel communication driver/HMI device.	Change the configuration.
220002	The tag is not downloaded because write access to data type Byte is not supported by the sublevel communication driver/HMI device.	Change the configuration.
220003	The communication driver cannot be loaded as it is possibly not installed.	Install the driver by reinstalling WinCC Runtime.
220004	Communication is down and no update data is transferred because the cable is not connected or defective etc.	Check the connection.
220005	Communication is up.	--
220006	The connection between the specified PLC and the specified port is active.	--

Number	Effect/causes	Remedy
220007	The connection to the specified PLC is interrupted at the specified port.	Check the following: • Is the cable plugged in? • Is the PLC OK? • Is the right port being used? • Is your configuration OK (port parameters, protocol settings, PLC address)? Restart the system if the system event persists.
220008	The communication driver cannot access or open the specified port. This port might be in use by another application, or a port that does not exist on the target device is being used. No communication with the PLC.	Close all applications that access the port and restart the computer. Use another port that exists in the system.

230000 - Screen object alarms

230000 - Screen object alarms

Number	Effect/causes	Remedy
230000	The value entered could not be used. The system discards this entry and restores the previous value. Possible causes: • The range of values is exceeded. • You entered invalid characters • The valid maximum number of users has been exceeded.	Enter a practical value, or delete a user that is no longer required.
230002	The user currently logged on does not have the necessary authorization, so the system discards the entry and restores the previous value.	Log on as user with appropriate authorization.
230003	Change to the specified screen failed because this screen is not available/configured. The current screen remains selected.	Configure the screen and check the selection function.
230005	The range of values of the tag has been exceeded in the I/O field. The original tag value is retained.	Observe the range of values for the tag when entering a value.
230100	During navigation in the Web browser, the system returned a message which may be of interest to the user. The Web browser continues to run but may not (fully) show the new page.	Navigate to another page.
230200	The HTTP channel connection was interrupted due to an error. This error is explained in detail by another system event. Data is no longer exchanged.	Check the network connection. Check the server configuration.
230201	The HTTP channel connection is set up. Data is exchanged.	--

Number	Effect/causes	Remedy
230202	WININET.DLL has detected an error. Usually, this error occurs if it is not possible to connect to the server, or the server denies a connection because the client lacks proper authorization. An unknown server certificate may also be the cause if the connection is encrypted by means of SSL. The alarm text provides details. This text is always in the language of the Windows installation because it is returned by the Windows OS. Process values are no longer exchanged. The part of the alarm returned by the Windows OS might not be displayed, e.g. "An error has occurred". WININET.DLL returns the following error: Number: 12055 Text:HTTP: <no error text available>."	Depending on the cause: When an attempt to connect fails, or a timeout occurs: • Check the network connection and the network. • Check the server address. • Check whether the WebServer is actually running on the target station. Incorrect authorization: • The configured user name and/or password do not match the entries on the server. Set consistent data. If the server certificate is rejected: Certificate signed by an unknown CA ( ): • Ignore this point, or install a certificate that has been signed with one of the root certificates known to the client station. The date of the certificate is invalid: • Ignore this point, or install a certificate with valid date on the server. Invalid CN (Common Name or Computer Name): • Ignore this point, or install a certificate with a name that corresponds to the server address.
230203	Although a connection can be made to the server, the HTTP server refused to connect. Possible causes: • WinCC Runtime does not run on the server • The HTTP channel is not supported (503 Service unavailable). Other errors can only occur if the Webserver does not support the HTTP channel. The language of the alarm text depends on the Webserver. Data is not exchanged.	On 503 Service unavailable error: Check whether WinCC Runtime is running on the server and whether the HTTP channel is supported.
230301	Internal error. An English text explains the error in more detail. The error may be caused by insufficient memory. OCX does not work.	--
230302	The name of the remote server cannot be resolved. An attempt to connect has failed.	Check the configured server address. Check whether the DNS service is available on the network.
230303	The remote server is not running on the addressed computer. incorrect server address. An attempt to connect has failed.	Check the configured server address. Check whether the remote server is running on the target computer.
230304	The remote server on the addressed computer is incompatible with VNCOCX. An attempt to connect failed.	Use a compatible remote server.

Number	Effect/causes	Remedy
230305	Authentication has failed due to incorrect password. An attempt to connect failed.	Configure the correct password.
230306	Error in the connection to the remote server. This may occur as a result of network problems. An attempt to connect failed.	Check whether the network cable is plugged in, or whether there are network problems.
230307	The connection to the remote server was shut down. Possible causes: • The remote server was shut down • The user instructed the server to close all connections. The connection is cancelled.	--
230308	This alarm provides information on the connection status. An attempt is made to connect.	--

240000 - Authorization alarms

240000 - Authorization alarms

Number	Effect/causes	Remedy
240000	WinCC Runtime is running in demo mode. You have no authorization, or your authorization is corrupted.	Install the authorization.
240001	WinCC Runtime is running in demo mode. You configured too many tags for the installed version.	Load an adequate authorization / powerpack.
240002	WinCC Runtime is running with time-limited emergency authorization.	Restore the full authorization.
240004	Error when reading the emergency authorization. WinCC Runtime is running in demo mode.	Restart WinCC Runtime. Install or repair the authorization (see Commissioning Instructions for Software Protection).
240005	Automation License Manager has detected an internal system error. Possible causes: • Corrupted file • Faulty installation • No free memory space for Automation License Manager, or similar.	Restart the HMI device/PC. If this procedure does not solve the problem, remove and then reinstall Automation License Manager.

250000 - S7 Status/Force alarms

250000 - S7 Status/Force alarms

Number	Effect/causes	Remedy
250000	The tag in the specified line in "Status Force" is not updated because the address configured for this tag is not available.	Check the set address and then verify that the address is set up in the PLC.
250001	The tag in the specified line in "Status Force" is not updated because the PLC type configured for this tag does not exist.	Check the set address.
250002	The tag in the specified line in "Status Force" is not updated because it is not possible to map the PLC type in the tag type.	Check the set address.
250003	An attempt to connect to the PLC failed. The tags are not updated.	Check the connection to the PLC. Check that the PLC is switched on and is online.

260000 - Password system alarms

260000 - Password system alarms

Number	Effect/causes	Remedy
260000	An unknown user or an unknown password has been entered in the system. The current user is logged off from the system.	Log on to the system as a user with a valid password.
260001	The logged in user does not have sufficient authorization to execute the protected functions on the system.	Log on to the system as a user with sufficient authorization.
260002	This alarm output when the "TrackUserChange" system function is triggered.	--
260003	The user has logged off from the system.	--
260004	The user name entered into the user view already exists in the user management.	Select another user name because user names have to be unique in the user management.
260005	The entry is discarded.	Enter a shorter user name.
260006	The entry is discarded.	Use a shorter or longer password.
260007	The logoff time entered is outside the valid range from 0 to 60 minutes. The new value entered is discarded and the original value is retained.	Enter a logon timeout value between 0 and 60 minutes.
260008	An attempt was made in WinCC to read a PTProRun.pwl file created with ProTool V 6.0. Reading of the file was canceled due to incompatibility of the format.	--
260009	You have attempted to delete the user "Administrator" or "PLC User". These users are fixed components of the user management and cannot be deleted.	If you need to delete a user, because perhaps you have exceeded the maximum number permitted, delete another user.

Number	Effect/causes	Remedy
260012	The password entries in the "Change Password" dialog and in the confirmation field do not match. The password is not changed. User will be logged off.	You have to log on to the system again. Then enter the identical password twice to be able to change the password.
260013	The password entered in the "Change Password" dialog is invalid because it is already in use. The password is not changed. User will be logged off.	You have to log on to the system again. Then enter a new password that has not been used before.
260014	You have tried to log on with an incorrect password three times in a row. You will be locked out and assigned to group no. 0.	You can log on to the system with your correct password. Only an administrator can change the assignment to a group.
260024	The password you entered does not meet the necessary security guidelines.	Enter a password that contains at least one number.
260025	The password you entered does not meet the necessary security guidelines.	Enter a password that comprises at least three characters.
260028	Upon system start-up, an attempt to log on, or when trying to change the password of a SIMATIC log-on user, the system attempts to access the SIMATIC Logon Server. If attempting to log on, the new user is not logged in. If a different user was logged on before, then this user is logged off.	Check the connection to the SIMATIC Logon Server and its configuration; for example: 1. Port number 2. IP address 3. Server name 4. Functional transfer cable Or use a local user.
260030	The SIMATIC Logon user could not change his password on the SIMATIC Logon Server. The new password is possibly noncompliant with password rules set on the server, or the user is not authorized to change his password. The old password remains and the user is logged off.	Log in again and choose a different password. Check the password rules on the SIMATIC Logon Server.
260033	The action change password or log on user could not be carried out.	Check the connection to the SIMATIC Logon Server and its configuration; for example: 1. Port number 2. IP address 3. Server name 4. Functional transfer cable Or use a local user.
260034	The last logon operation has not yet ended. A user action or a logon dialog can therefore not be called. The logon dialog is not opened. The user action is not executed.	Wait until the logon operation is complete.
260035	The last attempt to change the password was not completed. A user action or a logon dialog can therefore not be called. The logon dialog is not opened. The user action is not executed.	Wait until the procedure is complete.

Number	Effect/causes	Remedy
260036	There are insufficient licenses on the SIMATIC Logon Sever. The logon is not authorized.	Check the licensing on the SIMATIC Logon Server.
260037	There is no license on the SIMATIC Logon Sever. A logon is not possible. It is not possible to log on via the SIMATIC Logon Server, only via a local user.	Check the licensing on the SIMATIC Logon Server.
260040	The system attempts to access the SIMATIC Logon Server upon system start-up or when trying to change the password. If attempting to log on, the new user is not logged in. If a different user was logged on before, then this user is logged off.	Check connection to the domain and its configuration in the Runtime security settings editor. Or use a local user.
260043	It was not possible to log the user on to the SIMATIC Logon Server. The user name or the password could be incorrect or the user does not have sufficient rights to log on. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Try again. If necessary, check the password data on the SIMATIC Logon Server.
260044	It was not possible to log the user on to the SIMATIC Logon Server as his account is blocked. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Check the user data on the SIMATIC Logon Server.
260045	The SIMATIC Logon user is not associated to any or several groups. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Check user data on the SIMATIC Logon Server and the configuration in your WinCC project. A user may only be assigned to one group.

270000 - System alarms

270000 - System Alarms

Number	Effect/causes	Remedy
270000	The alarm does not indicate the tag because it is accessing an invalid address in the PLC.	Check whether the data area for the tag exists on the PLC, whether the configured address is correct, and whether the value range for the tag is correct.
270001	There is a device-specific limit as to how many alarms may be queued for viewing (see the operating instructions). This limit has been exceeded. The view no longer contains all the alarms. However, all alarms are written to the alarm buffer.	--
270002	The view shows alarms of a log for which there is no data in the current project. Placeholders are output for the alarms.	Delete old log data, if necessary.

Number	Effect/causes	Remedy
270003	The service cannot be set up because too many devices want to access this service. A maximum of four devices can execute this action.	Reduce the number of HMI devices which want to use the service.
270004	Access to the persistent alarm buffer is not possible. Alarms cannot be restored or backed up.	If the problems persist at the next restart, contact Customer Support (delete Flash).
270005	Persistent alarm buffer corrupted: Alarms cannot be restored.	If the problems persist at the next restart, contact Customer Support (delete Flash).
270006	Project modified: Alarms cannot be restored from the persistent alarm buffer.	The project was compiled and downloaded again to the HMI device. The error should no longer occur at the next restart of the HMI device.
270007	A configuration problem is preventing you from the restoring the data (e.g. a DLL was deleted, or unknown directory).	Update the operating system and download your project again to the HMI device.

290000 - Recipe system alarms

290000 - Recipe system alarms

Number	Effect/causes	Remedy
290000	The recipe tag could not be read or written. It is assigned the start value. The alarm can be entered in the alarm buffer for up to four more faulty tags. After that, alarm 290003 is output.	Check the configuration to see whether the address has been set up in the PLC.
290001	An attempt was made to assign the recipe tag a value that is outside the valid range of values for this type. The alarm can be entered in the alarm buffer for up to four more faulty tags if necessary. After that, alarm 290004 is output.	Observe the range of values for the tag type.
290002	It is not possible to convert a value from a source format to a target format. The alarm can be entered in the alarm buffer for up to four more faulty recipe tags if necessary. After that, alarm 290005 is output.	Check the range of values or type of the tag.
290003	This alarm is output after alarm 290000 was triggered more than five times. In this case, no further separate alarms are generated.	Check the configuration to see whether the tag addresses have been set up in the PLC.
290004	This alarm is output after alarm 290001 was triggered more than five times. In this case, no further separate alarms are generated.	Observe the range of values for the tag type.
290005	This alarm is output after alarm 290002 was triggered more than five times. In this case, no further separate alarms are generated.	Check the range of values or type of the tag.

Number	Effect/causes	Remedy
290006	The limits configured for the tag have been exceeded by the values entered.	Observe the configured or current tag limits.
290007	There is a difference between the source and target structure in the recipe currently being processed. The target structure contains an additional recipe tag that is not available in the source structure. The recipe tag specified is assigned its start value.	Insert the specified recipe tag into the source structure.
290008	There is a difference between the source and target structure in the recipe currently being processed.The source structure contains an additional recipe tag that is not available in the target structure and cannot be assigned. The value is discarded.	Remove the specified recipe tag in the specified recipe from the project.
290010	The storage location configured for the recipe is invalid. Possible causes: Invalid characters, write protection, data carrier out of space or not available.	Check the configured storage location.
290011	A data record of the specified number does not exist.	Check the source for the number (constant or tag value).
290012	A recipe of the specified number does not exist.	Check the source for the number (constant or tag value).
290013	An attempt was made to save a data record under a data record number that already exists. The operation is not executed.	The following remedies are available: • Check the source for the number (constant or tag value). • First, delete the data record. • Modify the "Overwrite" function parameter.
290014	The specified import file was not found.	Check the following: • The file name • Ensure that the file is in the specified directory.
290020	Check back to verify that the download of data records from the HMI device to the PLC has started.	--
290021	Check back to verify that the download of records from the HMI device to the PLC was completed.	--
290022	Check back to indicate that the download of data records from the HMI device to the PLC was canceled due to an error.	Check the following conditions in the configuration: • Are the tag addresses configured in the PLC? • Does the recipe number exist? • Does the data record number exist? • Is the "Overwrite" function parameter set?
290023	Check back to verify that the download of data records from the PLC to the HMI device has started.	--

Number	Effect/causes	Remedy
290024	Check back to verify that the download of data records from the PLC to the HMI device was completed.	---
290025	Check back to indicate that the download of data records from the PLC to the HMI device was canceled due to an error.	Check the following conditions in the configuration: • Are the tag addresses configured in the PLC? • Does the recipe number exist? • Does the data record number exist? • Is the "Overwrite" function parameter set?
290026	An attempt was made to read/write a data record that is not free at present. This error can occur if recipes were configured for download with synchronization.	Set the mailbox status to zero.
290027	Unable to connect to the PLC at present. As a result, the data record cannot be read or written. Possible causes: No hardware connection to the PLC (no cable plugged in, cable is defect), or the PLC is switched off.	Check the connection to the PLC.
290030	This alarm is output after you selected screen which contains a recipe view in which a data record is already selected.	Reload the data record from the storage location, or retain the current values.
290031	While saving, it was detected that a data record with the specified number already exists.	Overwrite the data record, or cancel the action.
290032	During the export of data records, a file with the specified name was found.	Overwrite the file, or cancel the process.
290033	Confirmation prompt before deleting data records.	--
290040	A data record error with error code %1 that cannot be described in more detail occurred. The action is canceled. It is possible that the mailbox was not installed correctly on the PLC.	Check the storage location, the data record, the "Data record" area pointer, and the connection to the PLC. Restart the action after a short waiting time. If the error persists, contact Customer Support. Forward the relevant error code to Customer Support.
290041	A data record or file cannot be saved because the storage location is out of sufficient space.	Delete files no longer required.
290042	An attempt was made to execute several recipe actions simultaneously. The last action is not executed.	Retrigger the action after a short waiting time.
290043	Confirmation prompt before saving data records.	--
290044	The database for the recipe was corrupted and will be deleted.	--
290050	A check back indicates that the export of data records was started.	--
290051	A check back indicates successful completion of the export of data records.	--
290052	A check back indicates that the export of data records was canceled due to an error.	Ensure that the structure of the data records at the storage location and the current recipe structure on the HMI device are identical.
290053	A check back indicates that the import of records was started.	--

Number	Effect/causes	Remedy
290054	A check back indicates successful completion of the import of data records.	--
290055	A check back indicates that the import of data records was canceled due to an error.	Ensure that the structure of the data records at the storage location and the current recipe structure on the HMI device are identical.
290056	Error when reading/writing the value in the specified row/column. The action was canceled.	Check the specified row/column.
290057	The tags of the recipe specified were toggled from "offline" to "online" mode. Each change of a tag in this recipe is now immediately transferred to the PLC.	--
290058	The tags of the specified recipe were toggled from "online" to "offline" mode. Modifications to tags in this recipe are no longer immediately transferred to the PLC but must be transferred there explicitly by downloading a data record.	--
290059	A check back indicates that the specified record was successfully saved.	--
290060	A check back indicates that the specified data record memory was cleared.	--
290061	A check back indicates that deletion of the data record memory was canceled due to an error.	--
290062	The data record number exceeds the maximum of 65536. This data record cannot be created.	Select another number.
290063	This occurs when you execute system function "ExportDataRecords" while the "Overwrite" parameter is set to "No". An attempt was made to save a recipe under a file name that already exists. The export is canceled.	Check the parameters of the "ExportDataRecords" system function.
290064	A check back indicates that the deletion of data records was started.	--
290065	A check back indicates successful completion of the deletion of data records.	--
290066	Confirmation prompt before deleting data records.	--
290068	Confirmation prompt for deletion of all recipe data records.	--
290069	Confirmation prompt for deletion of all recipe data records.	--
290070	The data record specified was not found in the import file.	Check the source of the data record number, or the data record name (constant, or tag value).
290071	When the editing data record values, you entered a value that is below the low limit of the recipe tag. The entry is discarded.	Enter a value within the recipe tag limits.

Number	Effect/causes	Remedy
290072	When editing data record values, you entered a value that exceeds the high limit of the recipe tag. The entry is discarded.	Enter a value within the recipe tag limits.
290073	An action (e.g. saving a record) failed for an unknown reason. The error corresponds to status alarm IDS_OUT_CMD_EXE_ERR in the large recipe view.	--
290074	While saving, a data record with the specified number but with different name was found.	Overwrite the record, change the record number or cancel the action.
290075	A data record of this name already exists. The data record is not saved.	Select a different data record name.
290110	The default values could not be set due to an error.	--
290111	The recipes subsystem cannot be used. Recipe views have no content and recipe-specific functions will not be executed. Possible causes: • Error when loading the recipes. • The recipe structure was changed in the ES. The recipes were not included in the latest project download. This means that the new configuration data no longer matches the old recipes on the device.	Download the project to the device again, including the recipes (the corresponding check box in the download dialog must be check marked).

300000 - Alarm_S alarms

300000 - Alarm_S alarms

Number	Effect/causes	Remedy
300000	Faulty configuration of process monitoring (e.g. using PDiag or S7 Graph): More alarms are queued than specified in the specifications of the CPU. No further ALARM_S alarms can be managed by the PLC and reported to the HMI devices.	Change the PLC configuration.
300001	ALARM_S is not registered on this PLC.	Select a controller that supports the ALARM_S service.

310000 - Reporting alarms

310000 - Reporting alarms

Number	Effect/causes	Remedy
310000	An attempt is being made to print too many reports in parallel. Only one log file can be output to the printer at a given time; the print job is therefore rejected.	Wait until the previous active log was printed. Repeat the print job if necessary.
310001	An error occurred on triggering the printer. The report is either not printed or printed with errors.	Evaluate the additional system alarms related to this alarm. Repeat the print job if necessary.

330000 - Interaction alarms

330000 - Interaction alarms

Number	Effect/causes	Remedy
330022	Too many open dialogs on the HMI device.	Close all dialogs you do not require on the HMI device.

350000 - PROFIsafe alarms

350000 - PROFIsafe alarms

Number	Effect/causes	Remedy
350000	PROFIsafe packets were not received within the specified time. There is a communication problem with the F-CPU. RT is terminated.	Check the WLAN connection.
350001	PROFIsafe packets were not received within the specified time. There is a communication problem with the F-CPU. The PROFIsafe connection is set up again.	Check the WLAN connection.
350002	Internal error. Runtime is terminated.	Internal error
350003	Check back for indicating the connection setup to the F-CPU. The Emergency-Off buttons are active immediately.	--

Number	Effect/causes	Remedy
350004	PROFIsafe communication was set up and the connection was shut down. Runtime can be terminated. The Emergency-Off buttons are deactivated immediately.	--
350005	Incorrect address configured for the F-Device. A PROFIsafe connection cannot be set up.	Check and modify the address of the F-Device in WinCC ES.
350006	The project was started. At the start of the project, check the proper function of the ACK buttons.	Press the two ACK buttons successively in the "Acknowledge" and "Panic" positions.
350008	You configured an incorrect number of failsafe buttons. A PROFIsafe connection cannot be set up.	Modify the number of failsafe buttons in the project.
350009	The device is in Override mode. It may no longer be possible to detect the location because transponder detection fails.	Exit the override mode.
350010	Internal error: The device has no failsafe buttons.	Return the device to Siemens. Worldwide contact partners

12.3.6 Configuring system diagnostics

12.3.6.1 System diagnostics basics

Introduction

You use system diagnostics to detect problems and errors in any part of your plant. WinCC has two display and operating elements for quick error localization.

System diagnostics view

The alarm view shows the status of a PLC while the system diagnostics view gives you an overview of all devices available in your plant: You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

System diagnostics window

The system diagnostics window is an operating and display element that you can only use in the global screen.

The functions of the system diagnostics window are no different than those of the system diagnostics view. Because the system diagnostics window is configured in the global screen, you can, for example, also specify if the object is closable in Runtime.

Note

System diagnostics on Basic Panels

The "System--Diagnostics window" object is not available for Basic Panels.

See also

System diagnostics views (Page 4402)

12.3.6.2 System diagnostics views

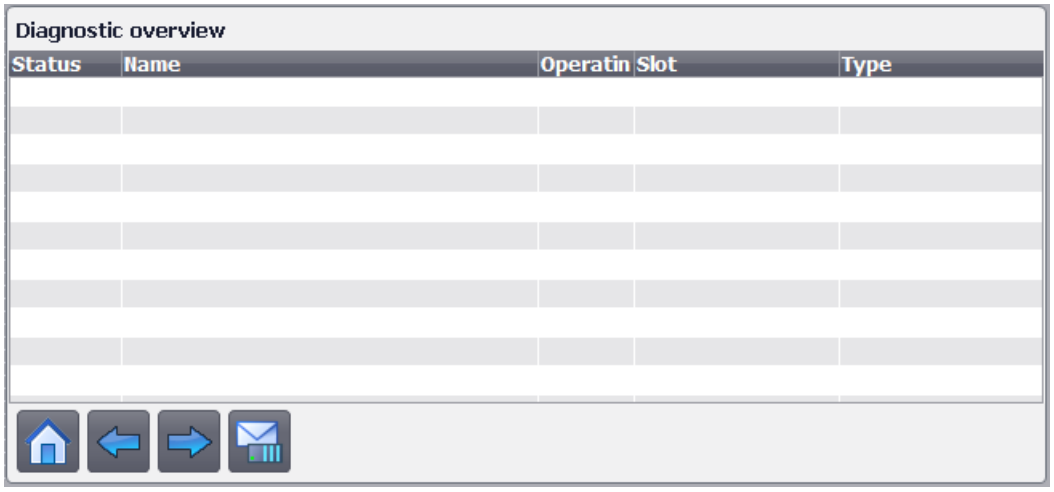
Introduction

There are four different views available in the system diagnostics view and the system diagnostics window.

- Device view
- Diagnostic buffer view
- Detail view
- Matrix view (for master systems, PROFIBUS, PROFINET only)

Device view

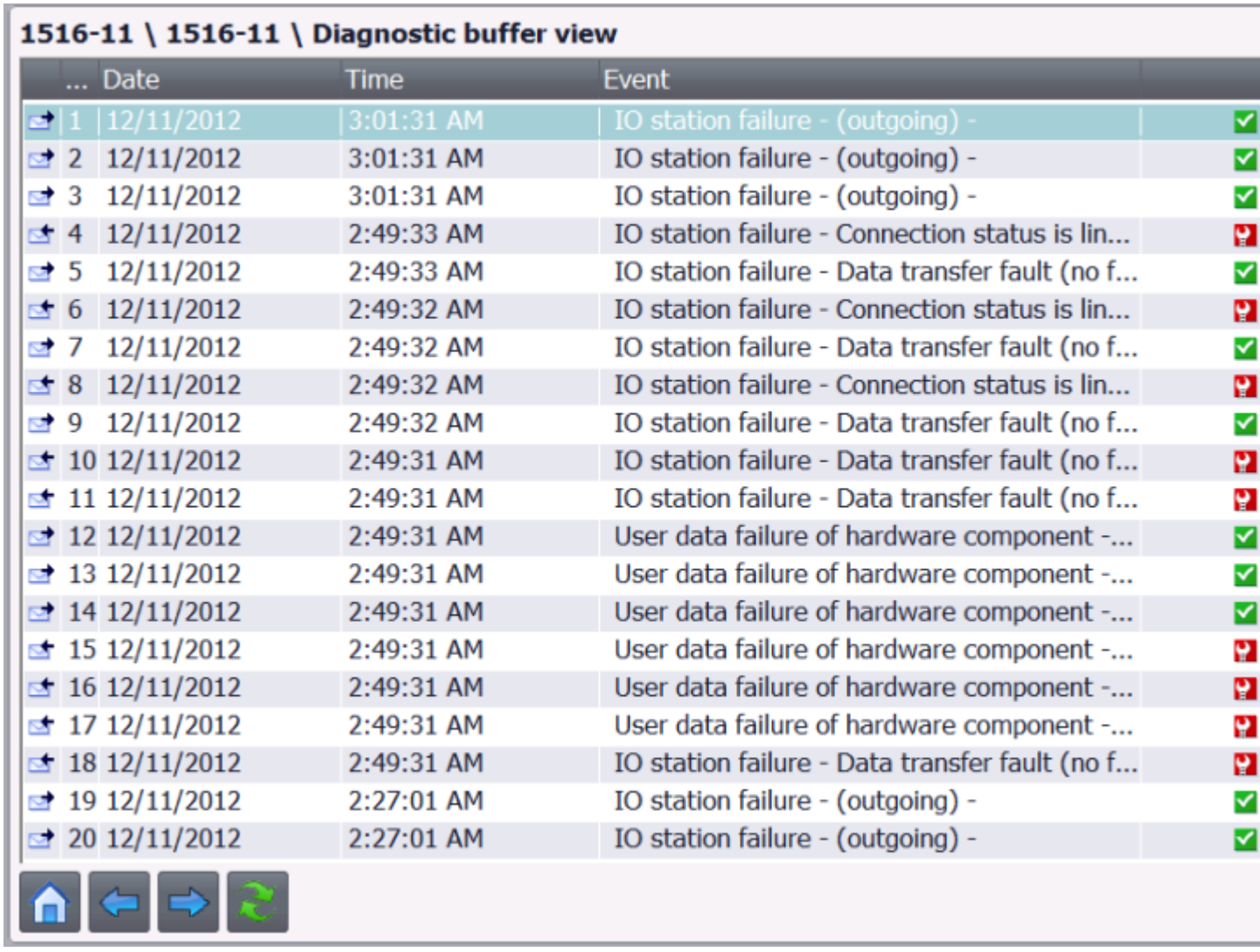
The device view shows all the available devices of a layer in a table. Double-clicking on a device opens either the child devices or the detail view. Symbols in the first column provide information about the current status of the device.



Diagnostic buffer view

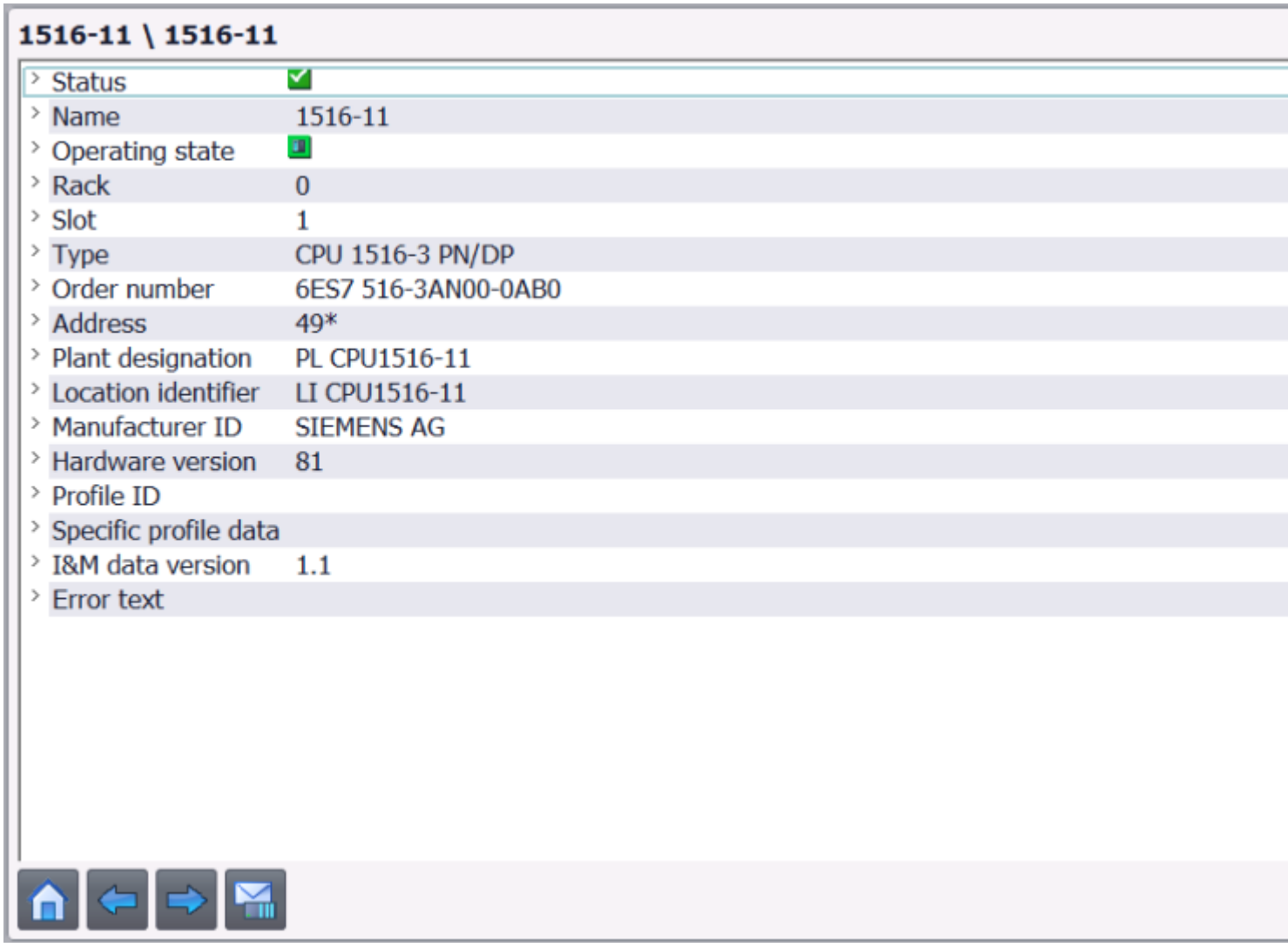
In the diagnostic buffer view, the current data from the diagnostic buffer of the device is displayed. The diagnostic buffer view can only be called up in the device view.

To update the diagnostic buffer view, you click .



Detail view

The detail view gives detailed information about the selected device and the pending errors. Check whether the data is correct in the detail view. You cannot sort error texts in the detail view.



Note

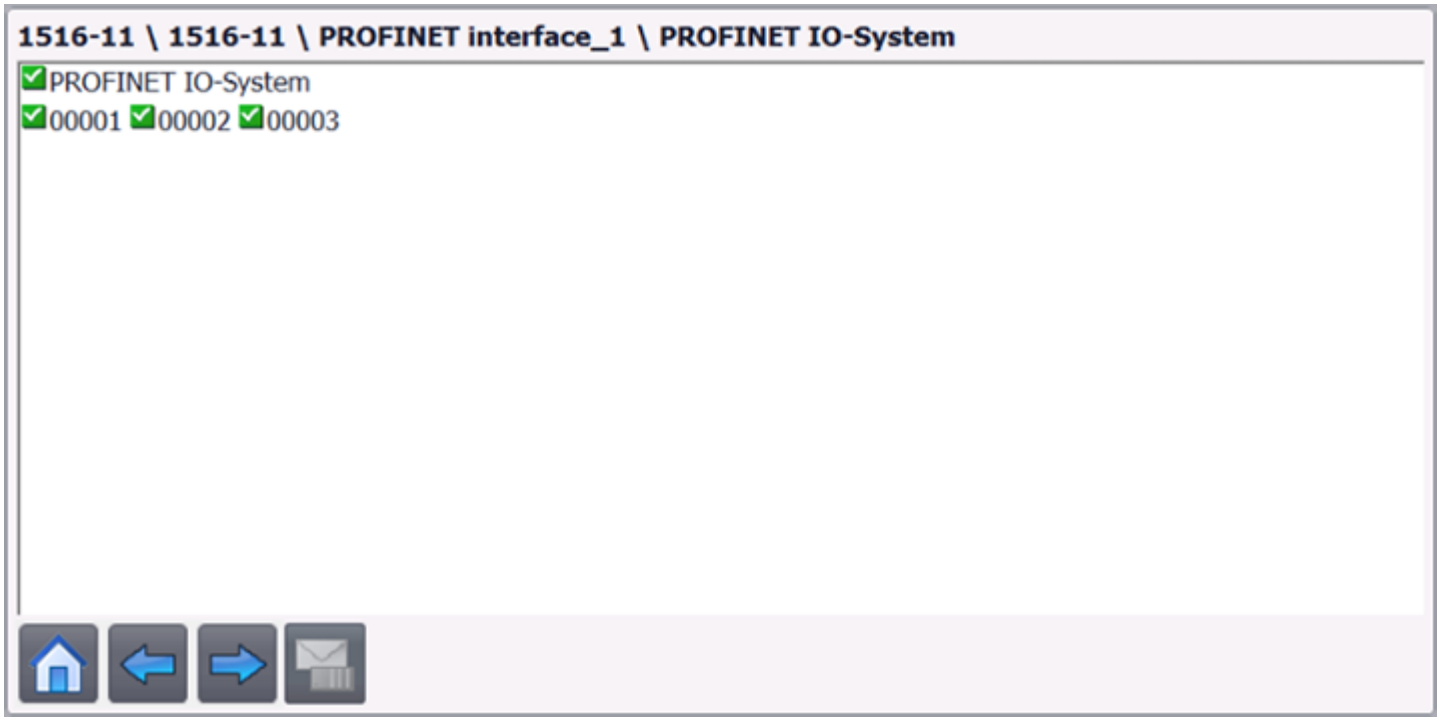
Contents of the detail view

The contents of the detail view are only available for integrated connections with S7 1200 and S7 1500 PLCs.

Matrix view

The matrix view is only available for master systems. In the matrix view you can see the status of the subdevices of the master system.

- In PROFIBUS, the numbers assigned by PROFIBUS are used as identification (DP station number).
- The IO devices are numbered consecutively beginning with 1 in PROFINET.



Navigation buttons

Button	Key	Function
	Enter key	Opens the child devices or the detail view if there are no child devices.
	Esc key	Opens the parent device or the device view if there is no parent device.
	Home key	Opens the device view.
	Configured function key, e.g. F1.	Opens the diagnostic buffer view. Only visible in the device view.
	Configured function key, e.g. F2.	Updates the diagnostic buffer view.

Note

Navigation in the matrix view of Comfort Panels with function keys

The Shift key is additionally used to select PNIO or PROFIBUS slaves in the matrix view.

Language switching in runtime

Switch over the language in runtime before you open the details view.

If you switch over the runtime language while the details view is opened, you change back to the diagnostic buffer view.

Open the details view again. The selected runtime language is now displayed correctly.

See also

System diagnostics basics (Page 4401)

12.3.6.3 Basic Panel basics

System diagnostics basics

Introduction

Using system diagnostics, you can display the messages from the diagnostic buffer of all integrated connections.

System diagnostics view

The system diagnostics display is an operating and display object that you use in a screen.

You navigate directly to the cause of the error and the associated connection. You have access to all integrated connections that you have configured in the "Devices & networks" editor.

System diagnostics views

Introduction

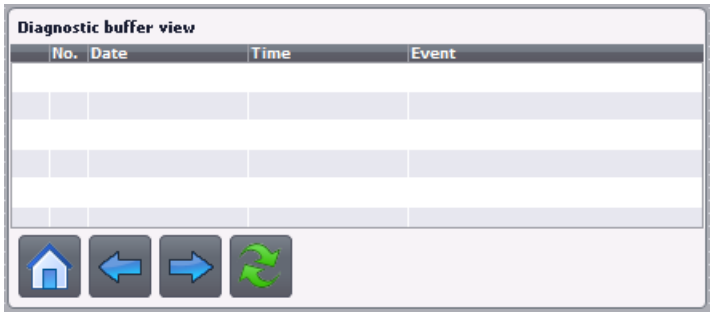
Three different views are available in the simple system diagnostic view.

- Device view
- Diagnostic buffer view
- Detail view

Device view

The device view is only displayed if more than one integrated connection has been configured.

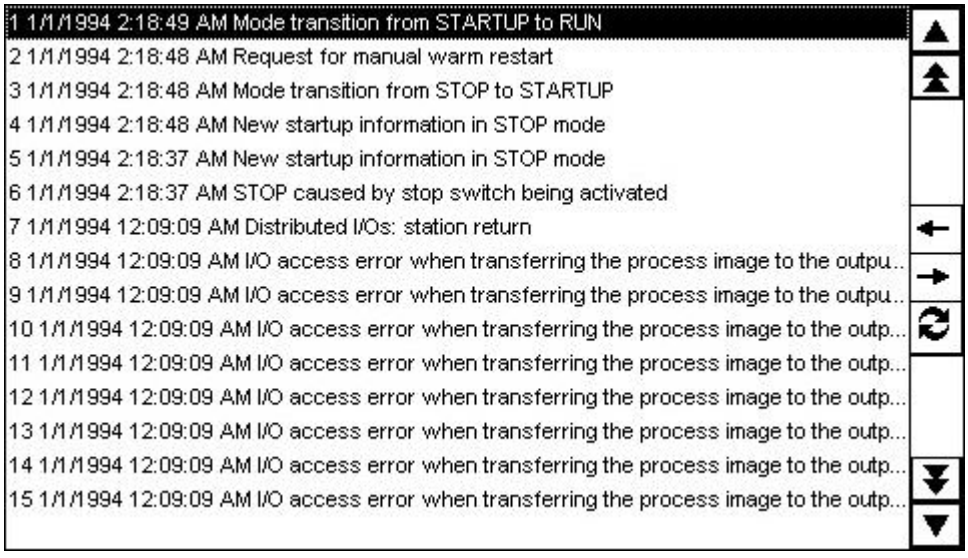
The device view shows all available connections in a table. Double-clicking a connection opens the diagnostic buffer view.



Diagnostic buffer view

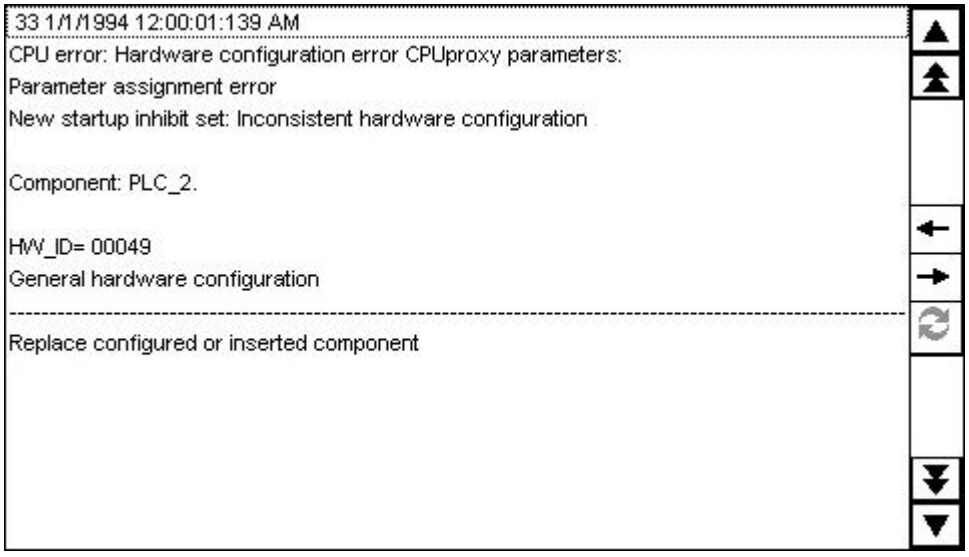
The diagnostic buffer view shows the current data from the diagnostic buffer.

To update the diagnostic buffer view, you click .



Detail view

The detail view gives detailed information about the selected connection and any pending errors. Check whether the data is correct in the detail view.



Note

Contents of the detail view

The contents of the detail view are only available for integrated connections with S7 1200 and S7 1500 PLCs.

Navigation buttons

Button	Key	Function
	Enter key	In the device view: Opens the diagnostic buffer view of the selected device. In the diagnostic buffer view: Opens the detail view.
	Esc key	In the diagnostic buffer view: Opens the device view. In the detail view: Opens the diagnostic buffer view.
	Configured function key, e.g. F1.	Updates the diagnostic buffer view.

12.3.6.4 Configuring system diagnostics objects

Enabling system diagnostics in the PLC

Introduction

During communication with the SIMATIC S7-1500 and SIMATIC S7-1200 PLCs, system diagnostics is automatically activated.

For other PLCs, the following settings must be implemented in the PLC to enable the PLC to transfer error messages to the system diagnostics view.

Requirements

- A PLC such as SIMATIC S7-300/400 or ET 200 has been created.
- The "Devices & Networks" editor is open in the "Device view".

Procedure

1. Select the PLC in the selection list.
2. Click on the PLC and select "Properties" in the shortcut menu. The properties of the PLC are displayed in the Inspector window.
3. Select "Properties > General > System diagnostics" in the Inspector window.
4. Enable "Activate system diagnostics for this CPU".
5. Select the PLC and then "Compile > Hardware configuration" in the shortcut menu.

Result

The settings will be effective following the next compilation. The PLC can now transfer error messages to the system diagnostics display.

Adding a system diagnostics indicator

Introduction

The system diagnostics indicator is a predefined graphic symbol of the library which alerts you to errors in your system.

If an error occurs, the appearance of the object changes. The library object can display two states:

- No error

- Error

Requirement

- The "Libraries" task card is opened.
- The global library "Buttons and Switches > Master copies > DiagnosticsButtons" is open.
- A screen is open.
- A system diagnostics window has been created in the global screen.

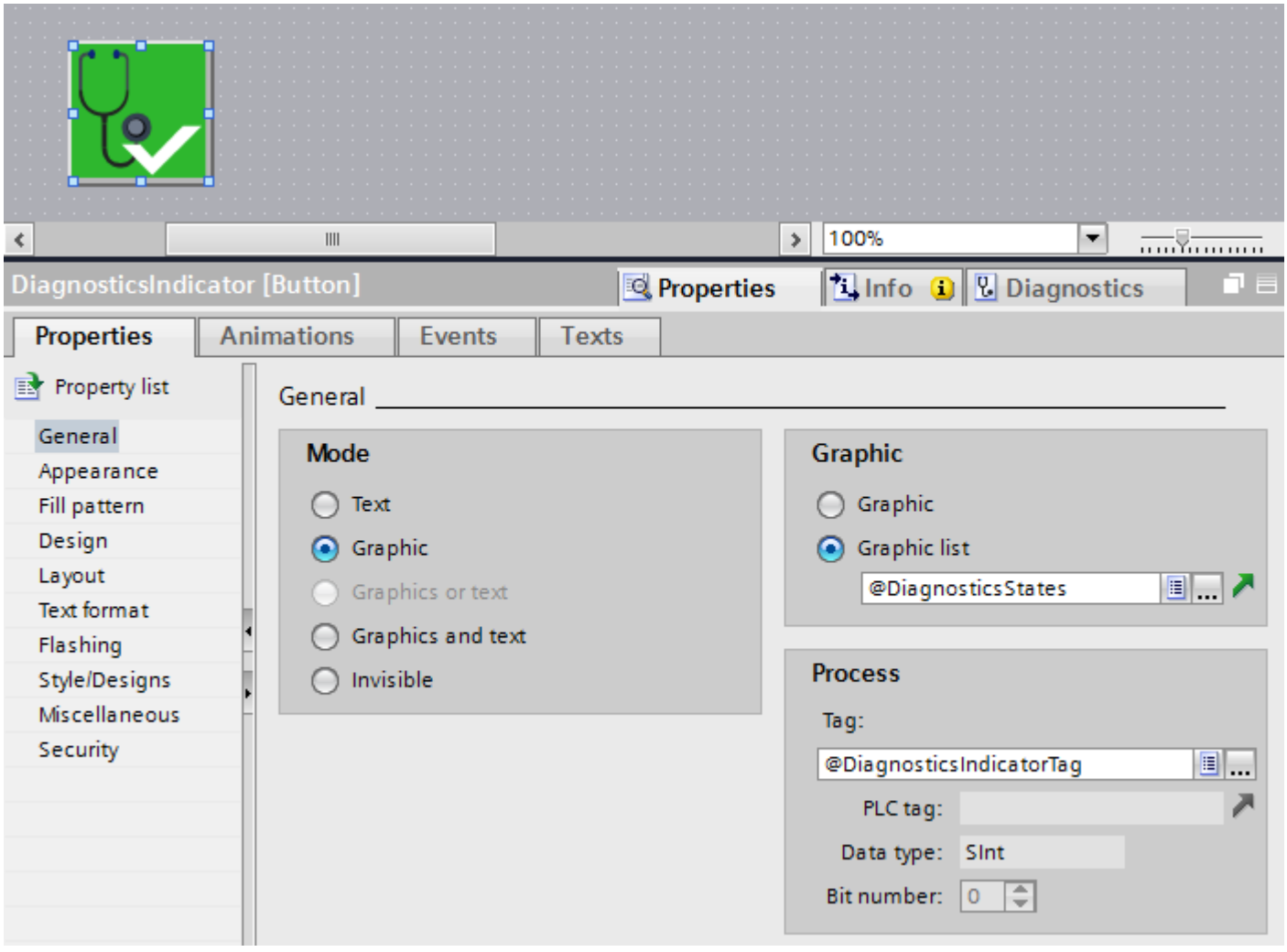
Procedure

1. Select the "DiagnosticsIndicator" object in the library.

   Note
   System diagnostics indicator for different device types

   When selecting the object ensure that you use the correct object for the device type used in the project.

2. Drag-and-drop the library object to the position in the work area where you want to insert the object.
   The library object is inserted.



3. Select the library object.
4. Click "Properties > Events" in the Inspector window.
   The system function "ShowSystemDiagnosticsWindow" is preset for the event "Click".

Result

The status diagnostics indicator has been added to the project and connected with the system diagnostics window.

The system diagnostics indicator changes its appearance if an error message is output in Runtime. The system diagnostics window opens when you click on the system diagnostics indicator. The system diagnostics window shows the detail view of the affected device.

Configuring access protection for the system diagnostics window

Configure access protection for the system diagnostics indicator to prevent unauthorized access to the system diagnostics window.

1. Select the "DiagnosticsIndicator" object in the screen.
2. Select an authorization in "Properties > Properties > Security in Runtime" in the Inspector window.

A logon dialog opens when you click on the system diagnostics indicator in Runtime. The system diagnostics window does not open unless you have the required authorization.

Configuring a system diagnostics window in the global screen

Introduction

The system diagnostics window gives you an overview of all devices available in your plant. The system diagnostics window operates like the system diagnostics view, but it is only available in the global screen.

Requirements

- At least one PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on each PLC.
- A Comfort Panel or WinCC RT Advanced has been created.
- The global screen is open.
- The Inspector window is open.

Procedure

1. Double-click the "System diagnostics window" object in the "Tools" task card. The object is added to the global screen.
2. Select "Properties > Properties > Columns > Devices/Detail view" in the Inspector window.
3. Enable the columns that you require in the device view for Runtime, for example: status, name, address.
4. Enable the properties which you require in the detail view for Runtime, for example: status, name, plant designation.

5. Activate the columns that you require in the diagnostic buffer view, for example, status, name.
6. Customize the column headers, if necessary.
7. Enable "Properties >Properties > Layout > Column settings > Columns moveable" to move the columns in Runtime.
8. Enable "Properties > Properties > Window > Closable" to allow the system diagnostics window to be closed in Runtime.

Result

The system diagnostics window has been added to the global screen. The system diagnostics window will respond to error messages in the plant and display the device affected.

Configure a system diagnostics indicator in a screen to open the system diagnostics window.

Configuring button as system diagnostics indicator

Introduction

Instead of using the object "DiagnosticsIndicator" from the library, you can configure a button, for example, to indicate errors in your plant.

Requirements

- At least one PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on each PLC.
- The "Tools" task card is open.
- A bit graphics list has been created with two different graphics for the statuses.
- A screen is open.
- You have created a system diagnostics view.

Procedure

1. Double-click the "Button" object in the "Tools" task card. A button will be added to the screen.
2. In the Inspector window, select "Properties > Properties > General > Mode > Graphic".
3. Select the bit graphics list under "Graphic > Graphics list".
4. Select the "@DiagnosticsIndicatorTag" tag under "Properties > Properties > General > Tag" in the Inspector window.

Configuring system function to the button

1. Click "Properties > Events" in the Inspector window.
2. Select the "Click" event.

3. Click on "Add function" in the table.
4. Select the "ActivateSystemDiagnosticsView" system function.
5. Select the system diagnostics view.

Result

You have now configured a button which will respond to error messages from the PLC. The button changes its appearance if an error message is output in Runtime.

The button has two states:

- Error
  The system diagnostics view opens when you click the button. The system diagnostics view shows the detail view of the device concerned.
- No error
  The system diagnostics view opens when you click the button. The system diagnostics view shows the device view.

Configuring the system diagnostic view

Introduction

You add a system diagnostics view to your project to get an overview of all devices available in your plant.

Requirements

- At least one PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on each PLC.
- A Comfort Panel or WinCC RT Advanced has been created.
- You have created a screen.
- The Inspector window is open.

Procedure

1. Double-click the "System diagnostics view" object in the "Tools" task card. The object is added to the screen.
2. Select "Properties > Properties > Columns > Devices/Detail view" in the Inspector window.
3. Activate the columns that you require in the device view for Runtime, e.g. Status, Name, Slot.
4. Activate the columns that you require in the detail view for Runtime, e.g. Status, Name, Plant designation.
5. Activate the columns that you require in the diagnostic buffer view, for example, Status, Name.

6. Enable "Properties >Properties > Layout > Column settings > Columns moveable" to move the columns in Runtime.
7. You can change the column headers under "Properties > Properties > Column headers", if necessary.

Result

The system diagnostics view has been added to the screen. Error for the entire plant are now displayed in the system diagnostics view in Runtime.

Configuring the system diagnostic view

Introduction

For an overview of all integrated connections, you insert a system diagnostics view in your project.

Requirements

- A PLC has been created.
- A Basic Panel has been created.
- An integrated connection has been created in the "Devices & Networks" editor.
- You have created a screen.
- The Inspector window is open.

Procedure

1. Double-click the "System diagnostics view" object in the "Tools" task card. The object is added to the screen.
2. In the Inspector window, select "Properties > Layout".
3. Enter a number under "Lines per entry", i.e. 5.

Result

The system diagnostics view has been added to the screen.

To obtain the latest alarms, update the diagnostic buffer.

Splitting the view in system diagnostics view

Introduction

The system diagnostics view also offers a split view of the display. You have the option to see the devices and the associated details at a glance.

Requirements

- At least one PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on each PLC.
- An HMI device has been created, for example, a Comfort Panel.
- You have created a screen.
- The Inspector window is open.

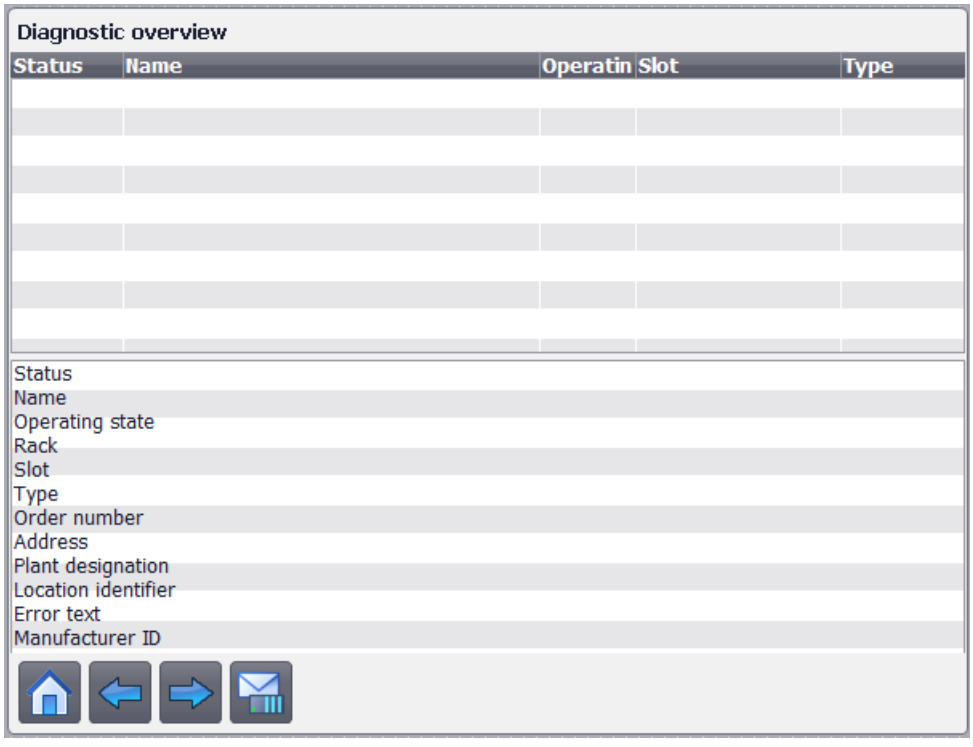
Procedure

1. Double-click the "System diagnostics view" object in the "Tools" task card. The object is added to the screen.
2. In the Inspector window, click "Properties > Properties > Layout".
3. Activate "Show split view".
4. Activate "Sizing > Auto-size"
5. You may also change the number of lines for the "Device view" and the "Detail view" if required.

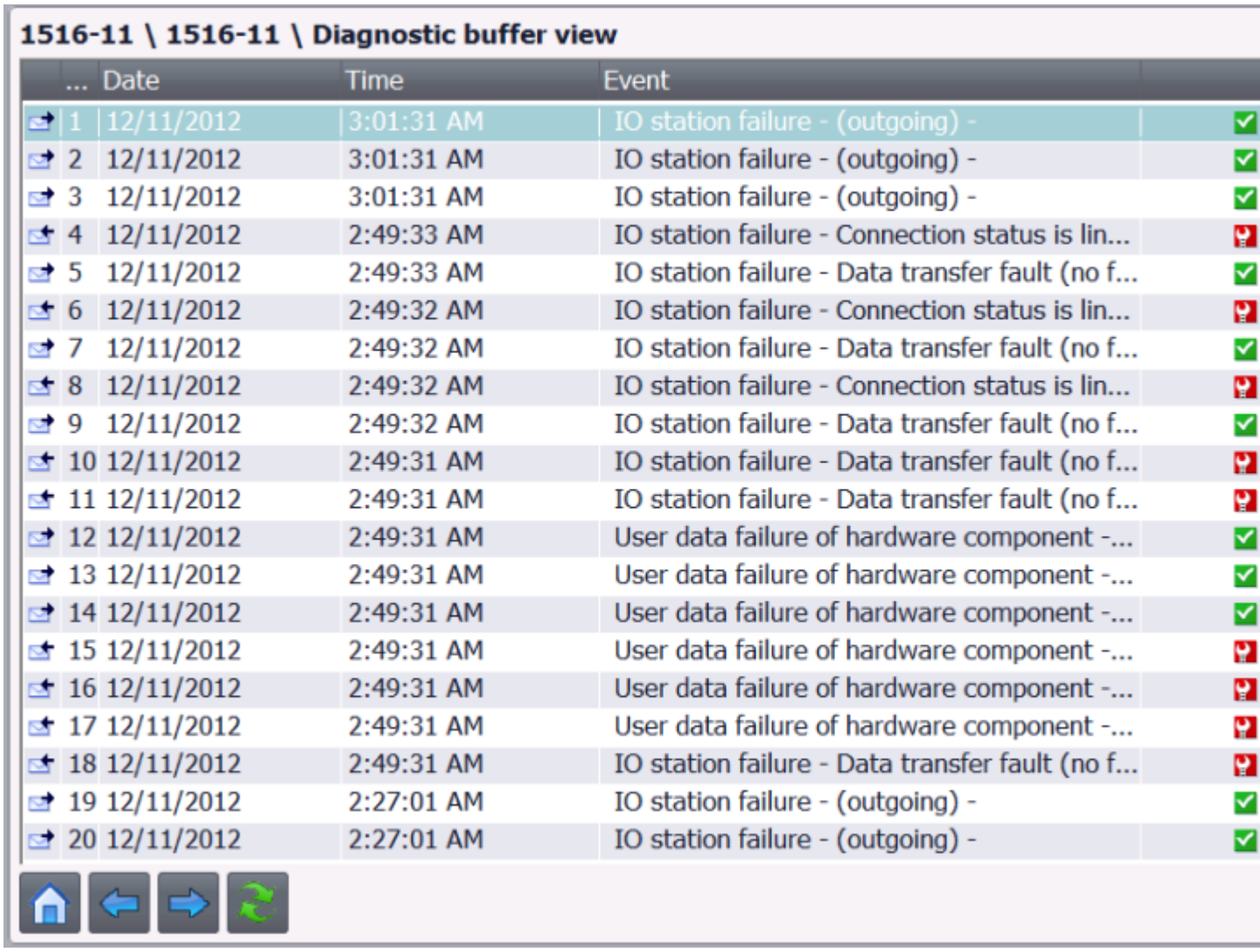
Result

The system diagnostics view has been added to the screen. The system diagnostics view is split into two areas.

The top area shows the device view. The bottom view shows the detail view.



If you open the diagnostic buffer view, the top area displays an overview of the diagnostic buffer and the bottom area displays the details.



Note

Contents of the detail view

The contents of the detail view are only available for integrated connections with S7 1200 and S7 1500 PLCs.

12.4 Working with logs

12.4.1 Working with logs for Runtime Advanced and Panels

12.4.1.1 Log Basics

Introduction

WinCC provides the following types of log for logging process data for HMI Runtime:

- Data logs
- Alarm logs

A data log is used to log process data from an industrial plant.

An alarm log is used to log alarms that occur in the monitored process.

Principle

The two types of log both have roughly the same structure and largely work in the same way. They are very clear and easy to configure. With both types of log you define the same properties for the log. The same logging methods are also available for both types of log.

The following logging methods are available:

- Circular log
  If a circular log is totally full, the oldest entries are overwritten.
- Segmented circular log
  In a segmented circular log, multiple single logs of the same size are filled in succession. When all log segments are filled, the oldest log segment is overwritten.
- Log with level-dependent system alarm
  A system alarm is triggered when a defined level is reached.
- Log with level-dependent triggering of an event
  The "Overflow" event is triggered when the log is full. The "Overflow" event triggers a system function.

12.4.1.2 Properties of Logs

Introduction

You define the properties of a data log in the "Data logs" editor.

You define the properties of an alarm log in the "Alarm log" editor.

The properties of the data log and alarm log are configured in the same way. You configure the properties either directly in the table of the respective editor or in the log properties of the Inspector window.

General properties

- Name
  You can assign any name to the log. The name must contain at least one letter or one number.
- Storage location
  The storage location specifies where the log is saved. The HMI device determines which storage locations are available.
- Size
  The size of a log depends on the type of log and the selected settings.
  - Size of a data log
    The size of a data log is calculated as follows:
    The number of items * the length of each tag value to be logged.
    In the "Properties" window, the maximum size that the log accepts for retention of the currently selected number of data records is displayed in the input field "Number of data records". The maximum log size is limited by the volume of the storage medium.
  - Size of an alarm log
    The size of an alarm log is calculated from the number of data records and the approximate size of an entry. The size of an entry depends on whether the alarm text and the associated tag values are to be logged as well.
- Restart characteristics
  Under Restart characteristics you can specify that the logging starts when Runtime starts. Check the "Enable logging at runtime start" check box.
  You can also control the behavior when Runtime starts. Enable "Reset log" if you want to overwrite existing logged data with the new data. Select the "Append data to existing log" option if you want to retain the existing logged data. This setting adds the data to be logged to an existing log.

See "Auto-Hotspot" and "Auto-Hotspot" for more details.

Note

You can use system functions to control the restarting of a log in Runtime.

Automatic log entries

In Runtime, the following log entries are created as standard:

Entry	File format	Log type	Meaning
RT_DIS	Any	Data log	Indicates that the connection to the log was interrupted at this point in time. (A bold line is shown in the trend view for this time period.)
RT_OFF	Any	Data log	Indicates that Runtime was shut down at this point in time. (No line is shown in the trend view for this time period.)

Entry	File format	Log type	Meaning
RT_ERR	Any	Data log Alarm log AuditTrail[1]	Indicates in the destination log that a copy operation was not successful or was interrupted. (The log copy was not fully created.)
$RT_COUNT	*.CSV *. TXT	Data log Alarm log AuditTrail[1]	This entry was created at the end of the log and serves to increase the system performance when Runtime starts.

[1] The "AuditTrail" logging method is not available for all HMI devices.

See also

Storage locations of logs (Page 4420)

12.4.1.3 Storage locations of logs

Storage location of a log

When you configure a log in WinCC, the available storage locations depend on the HMI device you are using.

HMI devices	Supported logs			Supported storage locations	Supported storage locations
	Alarms	Tags	Audit Trail		
Basic Panels[1]	No	No	No	-	-
Basic Panels 2nd Generation[2]	Yes	Yes	No	a TXT file (Unicode)	USB memory (at USB port)
Panels[3]	Yes	Yes	Yes	a CSV file (ASCII) RDB file a TXT file (Unicode)	Storage card (SD) Network drive
Comfort Panels[4]	Yes	Yes	Yes	a CSV file (ASCII) RDB file a TXT file (Unicode)	Storage card (SD) Storage card (USB) Network drive
Multi Panels[5]	Yes	Yes	Yes	a CSV file (ASCII) RDB file a TXT file (Unicode)	Storage card (MMC) Storage card (USB) Network drive
Mobile Panels[6]	Yes	Yes	Yes	a CSV file (ASCII) RDB file a TXT file (Unicode)	Storage card (MMC) Storage card (USB) Network drive

HMI devices	Supported logs			Supported storage locations	Supported storage locations
	Alarms	Tags	Audit Trail		
• Mobile Panels 2nd Generation[7]	Yes	Yes	Yes	a CSV file (ASCII) RDB file a TXT file (Unicode)	Storage card (SD) USB memory (at USB port) Network drive
PC systems with Runtime Advanced	Yes	Yes	Yes	a CSV file (ASCII) Database RDB file a TXT file (Unicode)	Local file system Network drive

1 KP 300, KP 400, KTP 1000, TP 1500

2 KTP 400, KTP 700, KTP 900, KTP 1200

3 TP 277 and OP 277 only

4 All HMI devices from the device list

5 MP 277 and MP 377 only

6 Mobile Panel 277 only

7 KTP 400 Mobile, KTP 700 Mobile, KTP 900 Mobile - including fail-safe versions

Note

Logging on network drives

Do not log alarms, tags and Audit Trail directly on a network drive. Power supply can be interrupted at any time. This means there is no guarantee for a reliable operation of logs and audit trails.

Save the logs on your local hard drive or a local storage card. Use the system function "ArchiveLogFile" to save the logs long-term on a network drive. This step ensures reliable operation.

Syntax examples for storage locations

Storage card storage location:

- <\Storage Card MMC\My_Archives\TagLogs>: Saves the log on the storage card MMC in the subdirectory "My_Archives\TagLogs".

Local file system storage location:

- <C:\My_File_Folder\My_Archives\Machine_1>: Saves the log on the local hard disk drive C: in the subdirectory "My_File_Folder\My_Archives\Machine_1"

Network drive storage location:

- <\\ArchiveServer\My_File_Folder\My_Archives\Machine_1>: Saves the log on the "ArchiveServer" in the subdirectory "My_File_Folder\My_Archives\Machine_1".

Naming conventions

Note

The log names must be unambiguous in a project. The name of a log must always be unique, regardless of whether different storage locations are selected for the log.

Note

The characters which can be used in the name of the data source depend on the storage location.

The characters \ / * ? : " < > | are not permitted for the following storage locations:

- File - RDB
- File - CSV (ASCII)
- File - TXT (Unicode)

If the "Database" storage location is used, the following characters may be used: a-z A-Z 0-9 _ @ # $

The characters _ @ # $ cannot be used as the first character of a name.

File - CSV (ASCII)

Data is saved to a CSV file in standard ASCII format.

If you want to read or evaluate logged data without using WinCC Runtime, use the "CSV file" storage location.

Note

Double quotation marks or multiple characters are not permitted as list separators for the "CSV file" storage location. You can find the settings for list separators under "Start > Settings > Control Panel > Regional and Language Options".

File - TXT (Unicode)

Data is stored in Unicode.

This file format supports all characters that can be used in WinCC and WinCC Runtime. For editing, you will need software that can save files in Unicode, such as Notepad.

Note

Use "File - TXT (Unicode)" as the storage location to log Asian languages.

File - RDB

Data is saved with quick access in a proprietary database.

If you require maximum read performance in Runtime, use the "RDB file" storage location.

Database

Data is saved to a database which is set up for ODBC access by the PC administrator.

Log with checksum (Audit Trail)

The following files are generated under special circumstances:

***.keep**

1. If a log is started without checksum and will be continued with a checksum.
2. If you update WinCC with a service pack or a new version and the Audit Trail or the log is continued with the checksum.

The content of the keep file will remain the same when compared with the original csv file or txt file.

***.bak**

If WinCC Runtime has determined a serious, irregular problem in the file.

See also

Data logging (Page 4250)

Properties of Logs (Page 4418)

12.5 Working with recipes

12.5.1 Basics

12.5.1.1 Definition and applications

Introduction

Recipes are collections of related data, for example, machine parameterizations or production data.

Note

Restrictions for recipes

Recipes are not available on the HMI device OP 73:

Examples:

- Machine parameter settings that are needed to convert production to a different product variant.
- Product components that result in different compositions for different end products.

A recipe has a fixed data structure. The structure of a recipe is defined during configuration. A recipe contains recipe data records. These differ in terms of their values, but not their structure.

Recipes are stored on the HMI device or on an external storage medium. Recipe data records are always transferred complete and in a single pass between the HMI device and the PLC. In addition, production data can be imported in runtime, in the form of a CSV file.

Note

Restrictions for Import / Export

It is not possible to export or import the recipes for the following HMI devices:

- Basic Panels
- OP 77A
- OP 177A
- TP 177A (Portrait)

Complete recipe data, but not single recipe data records, can be exported and imported using ProSave in CSV format and transferred to the HMI device. Thereby, Runtime is interrupted.

Using recipes

Recipes can be used in the following situations:

- Manual production
  You select the required recipe data and display it on the HMI device. You modify the recipe data as required and save it on the HMI device. You transfer the recipe data to the PLC.
- Automatic production
  The control program starts transfer of the recipe data between the PLC and HMI device. You can also start the transfer from the HMI device. Production is then implemented automatically. It is not essential to display or modify the data.
- Teach-in mode
  You optimize production data that was optimized manually on the system, e.g. axis positions or filling volumes. The values thus determined are transferred to the HMI device and saved in a recipe data record. You can then transfer the saved recipe data back to the PLC at a later date.

Entering and modifying the recipe data

You enter the data in the individual recipe data records and modify it as required. The following options are available:

- Data entry during configuration
  If the production data exists already, you enter the data in the "Recipes" editor during recipe configuration.
- Entering the data in Runtime
  If you have to frequently modify production data, you can do this directly in Runtime as follows:
  - Enter the data directly on the HMI device.
  - Set the parameters directly on the machine. You then transfer the data from the PLC to the HMI device and save it in the recipe.

12.5.1.2 Examples for using recipes

Recipes are used in the manufacturing industry and mechanical engineering, for example. The following recipes show typical applications which you can implement with the recipe function of WinCC:

- Machine parameter assignment
  One field of application for recipes is the assignment of machine parameters in the manufacturing industry: A machine cuts wooden boards to a certain size and drills holes. The guide rails and drill have to be moved to new positions according to the board size. The required position data are stored as data records in a recipe. You reassign the machine parameters using "Teach in" mode if, for example, a new board size is to be processed. You transfer the new position data directly from the PLC to the HMI device and save it as a new data record.
- Batch production
  Batch production in the food processing industry represents another field of application for recipes: A filling station in a fruit juice plant produces juice, nectar, and fruit drinks in a variety of flavors. The ingredients are always the same, differing only in their mixing ratios. Each flavor corresponds to a recipe. Each mixing ratio corresponds to a data record. All of the required data for a mixing ratio can be transferred to the machine control at the touch of a button.

12.5.1.3 Recipe structure

Introduction

The basic structure of a recipe is illustrated with reference to the filling station in a fruit juice plant.

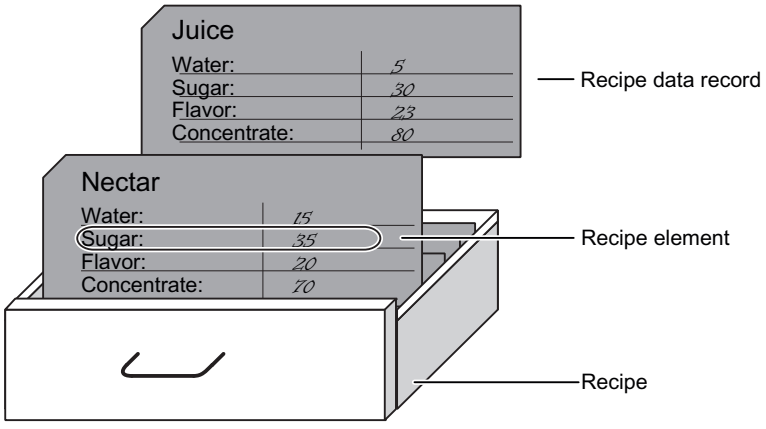
There may be several different recipes in an HMI device. A recipe can be compared to an index card box that contains several index cards. The index card box contains several variants for manufacturing a product family. All the data for each manufacturing variant is contained on a single index card.

Example:

In a soft drinks production plant, a recipe is needed for different flavors. Drink variants include fruit juice drink, juice and nectar.

Recipe

The recipe contains all the recipe data records for the different drink variants.



Recipe data records

Each index card represents a recipe data record needed to manufacture a product variant.

Recipe entries

Each index card in a drawer has the same structure. All the index cards contain fields for the different ingredients. Each field corresponds to a recipe entry. All the records of a recipe thus contain the same entries. The records differ, however, in the value of the individual entries.

Example:

All the drinks contain the following ingredients:

- Water
- Concentrate
- Sugar
- Flavoring

The records for juice drink, fruit juice or nectar differ, however, in the quantity of sugar used in production.

12.5.1.4 Display of recipes

Introduction

You can display recipes in the following ways:

- Recipe view
- Recipe screen

Input in the recipe view and recipe screen

You change the values of a recipe in the recipe screen or recipe view and thus modify the manufacturing process or a machine.

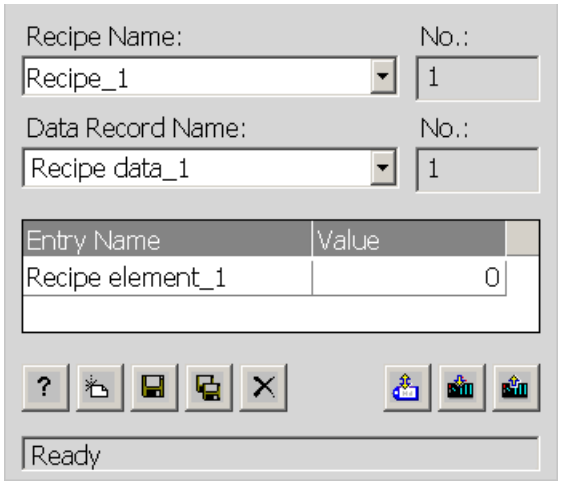
The recipe view and recipe screen can perform the same functions when using recipes. They differ in the following respects:

- Display options
- Operation
- Possibility of transferring data between the PLC and the HMI device

Recipe view

The recipe view is suitable for viewing simple recipes.

The recipe view is an off-the-shelf WinCC display and operator control object for managing recipe data records. The recipe view is always part of a screen. The recipe view shows recipe data records in tabular form. You adapt the appearance and the possible operations to suit your specific needs.

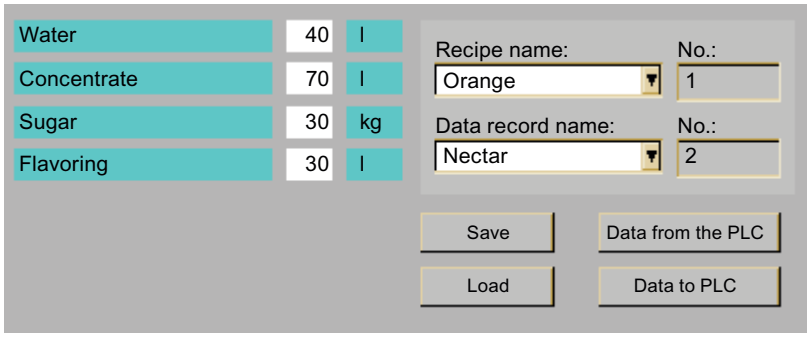


If you are editing recipes with a recipe view in your project, the values are saved in recipe data records. The values are not transferred between the HMI device and PLC until you use the relevant operator control object.

Recipe screen

The recipe screen is an individual screen that contains:

- Input fields for recipe variables
- Operator control objects for using the recipes, e.g. "SaveDataRecord"



The recipe screen is suitable in the following situations:

- Large recipes
- Allocation of recipe fields to the graphic display of the relevant plant area
- Breakdown of the recipe data into several screens

Note

The values of recipe variables are transferred between the PLC and recipe screen at the following times depending on the configuration:

- Immediately after modification
- When a relevant operator control object is used

Synchronizing recipe view and recipe screen

There may be differences in Runtime between the values displayed in the recipe view and the values saved in the associated tags when you edit recipes in both a recipe view and a recipe screen. To prevent this, you must synchronize the recipe data record values with the values of the recipe tags.

A complete recipe data record will always be saved or synchronized.

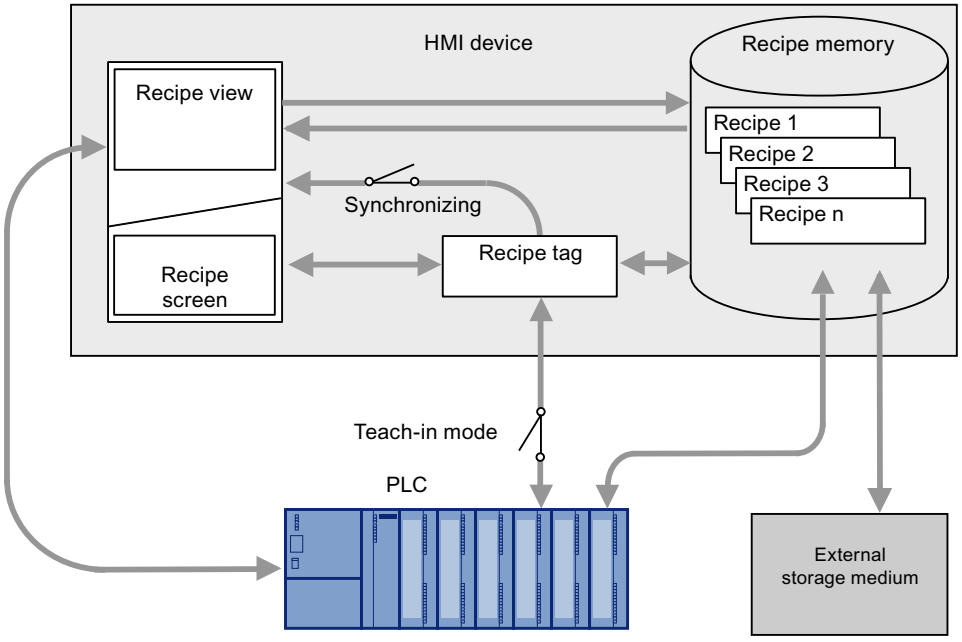
Note

Recipe tags can only be synchronized in the advanced recipe view. Synchronization only takes place if the "Synchronize recipe view and recipe tags" setting is activated for the recipe.

12.5.1.5 Transferring recipe data records

Flow of data in recipes

The following picture illustrates the flow of data in recipes:



Interaction between the components

There is interaction between the following components at runtime:

- Recipe view / recipe screen
  On the HMI device, recipes are displayed and edited in the recipe view or in a recipe screen:
  - The recipe data records from the internal memory of the HMI device are displayed and edited in the recipe view.
  - The values of the recipe tags are displayed and edited in the recipe screen.

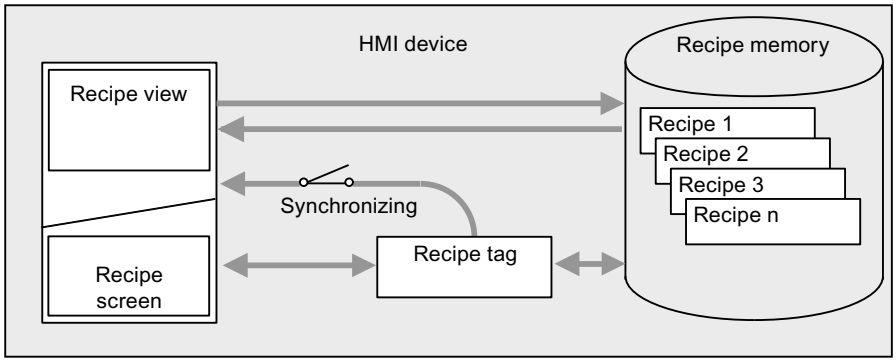
  You synchronize the values displayed in the recipe view with the values of recipe tags according to the configuration.
- HMI device recipe memory
  Recipes are saved in the form of recipe data records in the HMI device's recipe memory.
- Recipe tags
  The recipe tags contain recipe data. When you edit recipes in a recipe screen, the recipe values are stored in recipe tags. The point at which the values of the recipe tags are exchanged with the PLC depends on the configuration.

Note

You can synchronize the recipe tags with the recipe data records so that the same values are saved in both.

Loading and saving recipe data

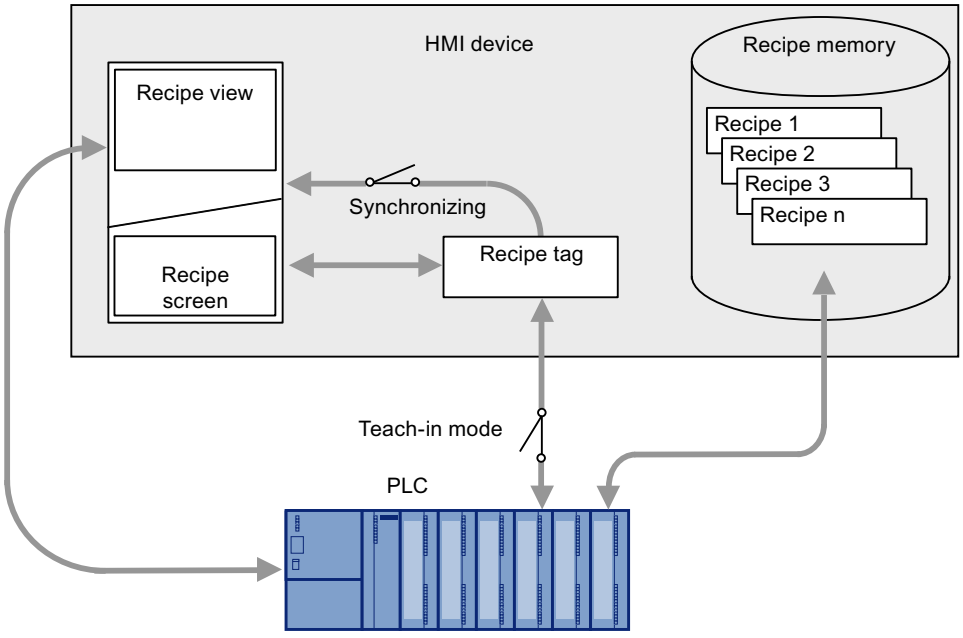
Complete recipe data records are loaded from or saved to the recipe memory on the HMI device in the recipe view.



The values of the recipe data record are loaded from the recipe memory to the recipe tags in the recipe screen. When they are saved, the values of the recipe tags are saved to a recipe data record in the recipe memory.

Transferring the recipe values between the HMI device and the PLC

Complete recipe data records are transferred between the recipe view and PLC.



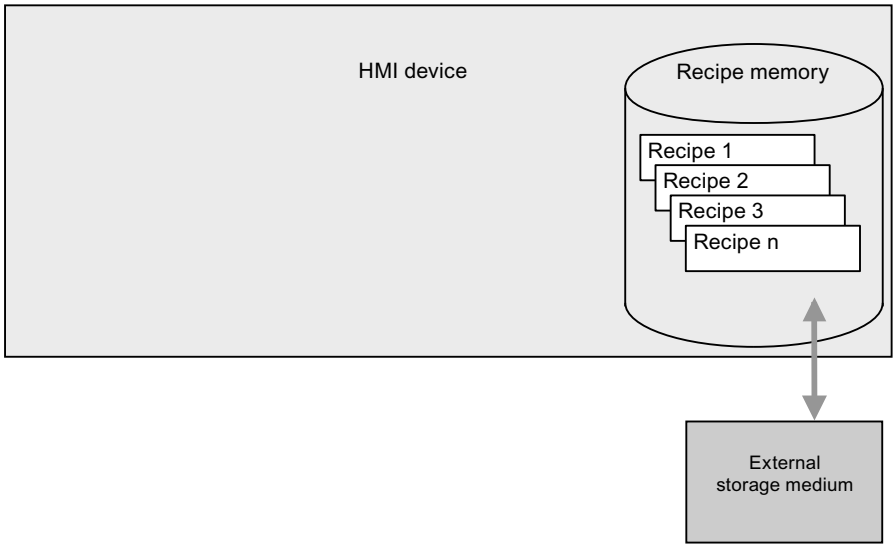
The following transfers are possible between the recipe screen and PLC, depending on the configuration:

- Transferring recipe data records between recipe view and recipe tags
- Immediate transfer of individual modified values between the recipe tags and PLC. The following settings are needed in the recipe in order to do this:
  - "Synchronizing recipe view and recipe tags" is activated.
  - "Manual transfer of individual modified values (teach-in mode)" is deactivated.

Recipe data records can be transferred directly between the HMI device and PLC. In these situations, the display on the HMI device is not essential.

Exporting and importing recipe data records

Recipe data records are exported from the HMI device recipe memory and are saved to a CSV file on the external storage medium. The records can be reimported from the storage medium to the recipe memory.



The following external storage media may be used, depending on the HMI device:

- No storage medium, for example, for basic panels
- Memory card
- USB stick
- Hard disk

12.5.1.6 Configuration of recipes

Introduction

Recipes are configured differently according to the intended use:

- If you are editing recipes with a recipe view in your project, the values are only saved in recipe data records.
- If you are editing recipes in a recipe screen in your project, the values are saved in recipe tags.

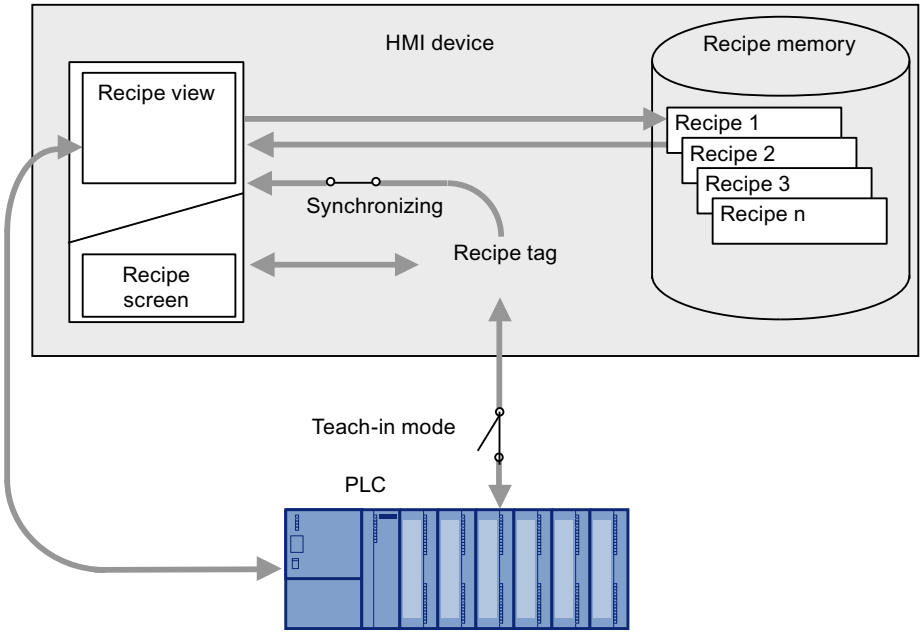
The following possible settings determine how the recipe data records, recipe tags and PLC all interact.

"Synchronizing recipe view and recipe tags" deactivated

The data in a data record is only displayed and can only be edited in the recipe view. If you use data outside the recipe view, they are not synchronized with the recipe view.

"Synchronizing recipe view and recipe tags" activated

There may be differences in Runtime between the values displayed in the recipe view and the values saved in the associated tags when you edit recipes in both a recipe view and a recipe screen. To prevent this, you must synchronize the recipe data record values with the values of the recipe tags.



Note

Recipe tags can only be synchronized in the advanced recipe view.

The values of the recipe view and the associated recipe tags are not synchronized automatically. The recipe tags and the recipe view are not synchronized until you use the operating element with the "RecipeViewSynchronizeDataRecordWithTags" function.

"Synchronize recipe view and recipe tags" activated and "Manual transfer of individual modified values (teach-in mode)" activated

With this setting, modified recipe values are not synchronized immediately between the recipe tags in the recipe screen of the HMI device,and PLC.

There must be an operating element with the "SetDataRecordToPLC" and "GetDataRecordFromPLC" functions present in order to synchronize the values.

If recipe values are changed in the controller, the modified values are displayed immediately in the recipe screen if you use the operating element with the "GetDataRecordFromPLC" function.

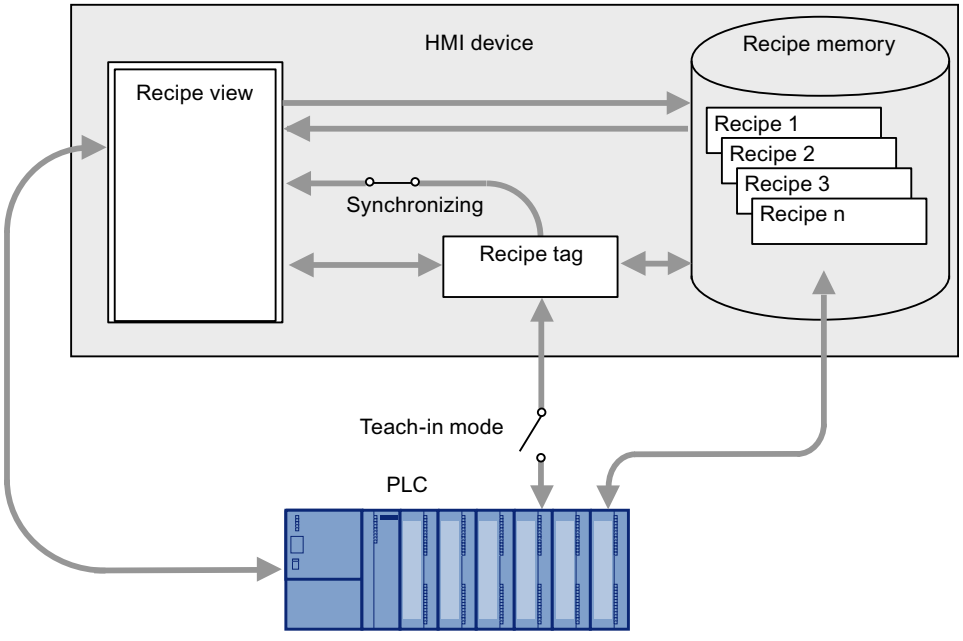
"Synchronize recipe view and recipe tags" activated and "Manual transfer of individual modified values (teach-in mode)" deactivated

With this setting, modified recipe values are synchronized immediately between the recipe tags on the HMI device and PLC.

When you change recipe values in the recipe screen, these changes are applied immediately by the PLC and immediately influence the process.

If recipe values are changed in the PLC, the changed values are displayed immediately in the recipe screen.

12.5.1.7 Special features of some devices



Basic panels, OP 77A, TP 177A and TP 177A (Portrait) respond differently to the other HMI devices in the following respects:

- Only the simple recipe view is supported.
- Recipe tags are always synchronized with the recipe view (see figure above): When the operator changes a value of the recipe element in recipe view, the value of the associated recipe tag will also change.
- Recipe tags are not automatically transferred between the PLC and HMI device when they are modified. Recipe tags are always transferred manually in teach-in mode.
  There must be an operator control object with the "SetDataRecordToPLC" and "GetDataRecordFromPLC" functions present in order to synchronize the values with the PLC.

Note

Restrictions for recipe tags

Generally, in Basic Panels and OP77A, TP177A (Portrait) recipe tags cannot be used, except in a recipe.

Note

Restrictions for Import / Export

It is not possible to export or import the recipes for the following HMI devices:

- Basic Panels
- OP 77A
- OP 177A
- TP 177A (Portrait)

Complete recipe data, but not single recipe data records, can be exported and imported using ProSave in CSV format and transferred to the HMI device. Thereby, Runtime is interrupted.

Notice

Data loss with several recipe views in the screen

Applies only to Basic Panels, OP73, OP77A, TP177A and TP177A (Portrait): If two or more recipe views show the same recipe in a screen, you have a conflict when accessing the data.

The result is data loss and unpredictable status of recipe data.

Make sure the operators do not select and edit the same recipe in different recipe views.

- Display only one recipe in a recipe view.
- Display a different recipe in each recipe view.

Interaction between the components

There is interaction between the following components at runtime:

- Recipe view
  Recipes are displayed and edited in the recipe view on the HMI device.
  The recipe data records from the internal memory of the HMI device are displayed and edited in the recipe view.
- HMI device recipe memory
  Recipes are saved in the form of recipe data records in the HMI device's recipe memory.
- Recipe tags
  The recipe tags contain the values of recipe elements.

12.5.1.8 Synchronization of recipe data records with the PLC

Overview

When recipe data records are transferred between the HMI device and PLC, both communication peers access common communication areas on the other peer.

Recipe data records are always transferred directly. The values of the tags are written directly to or read directly from the configured addresses without being placed on the clipboard.

Data transfer types

There are two ways to transfer recipe data records between the HMI device and PLC:

- Transfer without coordination
- Coordinated transfer via the "Data record" area pointer.

Note

Coordinated transfer

Transfer with coordinated transfer is used to prevent the uncontrolled overwriting of data in either direction in your control program.

Requirements for coordinated transfer

The following requirements apply to coordinated transfer:

- The "Data record" area pointer must be set up for the required connection in the "Communication > Connections" editor.
- In the properties of the recipe "Coordinated transfer of data records" is activated.
- The connection to the PLC is specified in the properties of the recipe with which the HMI device coordinates the transfer.

Coordinated transfer

In the case of coordinated transfer, both the PLC and the HMI device set the status bits in the shared data compartment.

Coordinated transfer of recipe data records can be a useful solution in the following cases:

- The PLC is the "active partner" for the transfer of recipe data records.
- The PLC evaluates information about the recipe number and name, as well as the recipe data record number and name.
- The transfer of recipe data records is started by the following PLC jobs:
  - "Set_data_record_in_PLC"
  - "Get_data_record_from_PLC"

12.5.1.9 "Recipes" editor

Introduction

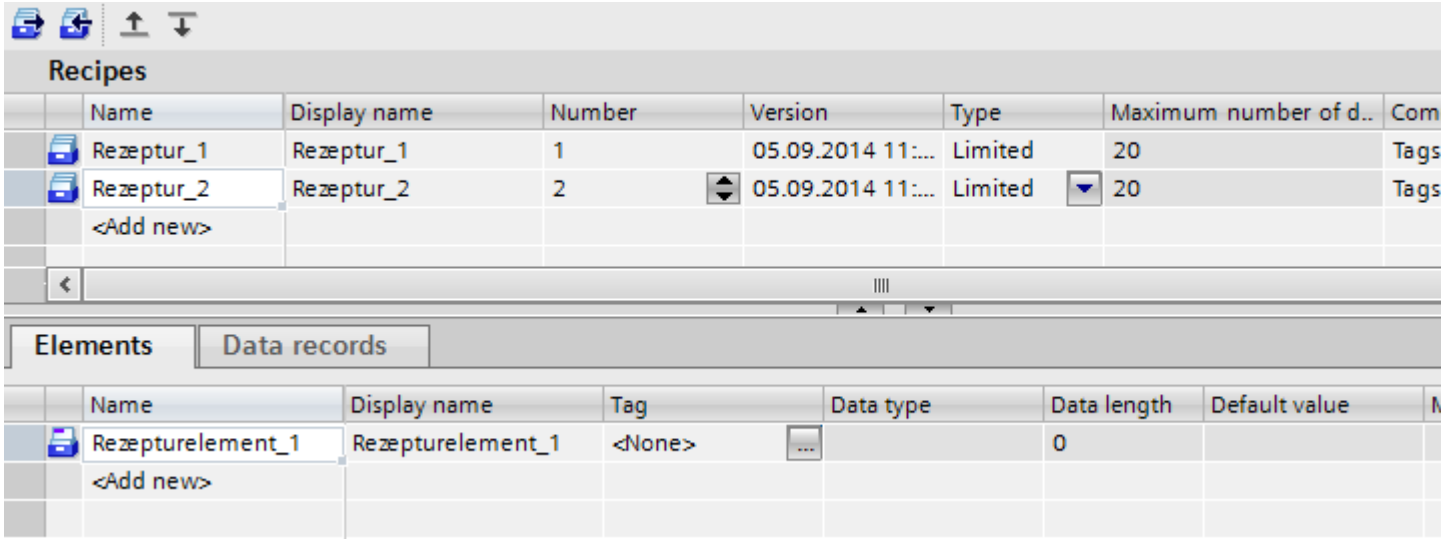
You can create, configure and edit recipes, recipe entries and recipe data records in the "Recipes" editor. The "Recipes" editor also allows you to enter values in recipe data records.

Structure of the "Recipes" editor

You create recipes in the top part of the table editor. You can also configure them there or in the Inspector window.

The bottom part of the table editor has the following tabs:

- Elements
  Define the recipe elements of the selected recipe using the table cells provided here. You can move recipe elements within the table with the shortcut menu commands, "Up" and "Down".
- Data records
  Define the values of the data records of the selected recipe using the table cells provided here.



You can then configure the selected recipe, the recipe element or the recipe data record in the Inspector window. You will find further notes on configuring the components of a recipe under "Configuring Recipes".

Recipe settings

The following settings are available for recipes:

Setting	Description
Name of the recipe	This is a unique identification for the recipe within the HMI device.
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign descriptive names or designations which the operator can associate directly with a recipe, e.g. "fruit juice drink".
Recipe number	This is a unique identification for the recipe within the HMI device.
Version	Information about the recipe. The date and time of the last change to the recipe is set by default.
Path	Defines the storage location for recipes. The recipes are stored as a file.
Size type [fixed]	The recipe data records are limited to a predetermined number by default.
Number of data records [fixed]	Maximum number of data records in a recipe in Runtime. The number is limited by the recipe memory of the HMI device.
Communication type [fixed]	The recipe data records are written directly to the addresses of the recipe tags and read from there.
Tooltip	Tooltip for the recipe which is shown to the operator in Runtime.

Note

Path

The storage location depends on the storage media available on the HMI device.

Basic Panels and OP77A, TP177A (Portrait)

These HMI devices have no external memory. Recipes are always saved in the internal Flash memory. The "Path" setting is therefore not available.

Recipe element settings

You can make the following settings on the "Elements" tab:

Setting	Description
Name of the recipe element	Identifies a recipe element uniquely within the recipe. Enter meaningful names or labels that you can allocate uniquely, such as axis labels on a machine or ingredients such as "Flavoring".
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign meaningful names or designations which the operator can associate directly, e.g. "fruit juice flavoring".
Recipe tag	An assigned tag in Runtime stores the current value of the recipe element in the recipe data record.
Data type	Data type of the recipe tag.
Data length [fixed]	Data length of the recipe tag, depending on the data type.
Text list	Text is assigned to a value or range of values in a text list. You can display this text in an output field, for example. The assigned recipe tag must have the data type of a number. The tag value must be within the range of values of the text list.
Default value	This is used as the default entry when you create a new recipe data record.
Minimum value [fixed]	The smallest representable value of a number-based recipe tag, depending on the type of data.
Maximum value [fixed]	The largest representable value of a number-based recipe tag, depending on the type of data.
Decimal places	Determines how many places a decimal number is rounded to, e.g. 3 decimal places and vice versa by what power of ten an integer value is multiplied, e.g. 1,000.
Tooltip	Tooltip about the recipe element which is shown to the operator in Runtime.

Recipe data record settings

You can make the following settings on the "Data records" tab:

Setting	Description
Name of the recipe data record	Identifies a recipe data record uniquely within the recipe.
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign meaningful names or product numbers which the operator can associate directly with a product, e.g. "yellow fruit juice E231".
Recipe data record number	Identifies a recipe data record uniquely within the recipe.
Recipe elements 1 to n	You can store various values for each recipe element even during configuration. Together with the values of the other recipe elements, a value always forms a recipe data record. You can store multiple recipe data records. If enabled in the transfer settings, the recipe data records are transferred to the HMI device when downloading the project and existing data records on the HMI device are overwritten.
Comment	Comment about the recipe data record

12.5.2 Displaying and Editing Recipes in Runtime

12.5.2.1 Recipe screen and recipe view

You can display and edit recipes on the HMI device with a recipe view or recipe screen.

Recipe view

The recipe view is an off-the-shelf WinCC display and operator control object.

The recipe view is available in the following views:

- As anvanced recipe view
- As simple recipe view

Note
Availability

In basic panels and OP77A, TP177A, only the simple recipe view is available.

Recipe screen

The recipe screen contains an individual screen form for entering the recipes. The screen form contains I/O fields and other display and operator control objects. The recipe functionality is implemented with system functions, e.g. for saving recipe data records.

Note

Availability

In basic panels and OP73, OP77A, TP177A (Portrait), the recipe screens are not available.

12.5.2.2 Simple recipe view

Recipe view

The simple recipe view is a ready-made display element and operator control that is used to manage recipe data records. The recipe view shows recipe data records in tabular form.

The displayed buttons and information in the columns are adjustable.

The values displayed or entered in the recipe view are saved in recipe data records. The displayed recipe data record can be written into the PLC by buttons or values can be read in from the PLC.

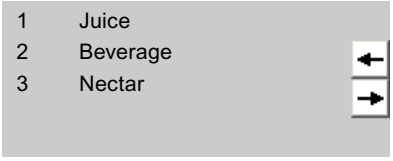
Layout of the display

The simple recipe view consists of three areas:

- Recipe list
- Data record list
- Element list

In the simple recipe view, each area is shown separately on the HMI device. Depending on the configuration, the simple recipe view starts with the recipe list.

The figure below shows an example of the data record list.



Display of values

Note

Processed recipe data record is changed in the background

Only applies for Basic Panels: if an operator has changed a recipe data record and a PLC job wants to read or write any recipe data record of this recipe, the PLC job is stopped and a system alarm is output. On the other hand, the changed value is displayed immediately if only the PLC job and no operator has changed recipe data.

Does not apply for Basic Panels: If an operator has changed a recipe data record and a PLC job has changed the values of the recipe data record concerned, the recipe view is not updated automatically. To update the recipe view, reselect the respective recipe data record.

12.5.2.3 Configuration options of the simple recipe view

The response of the simple recipe view can be defined in the recipe view inspector window.

Note

Validity

Some devices, for example basic panels, only support the simple recipe view.

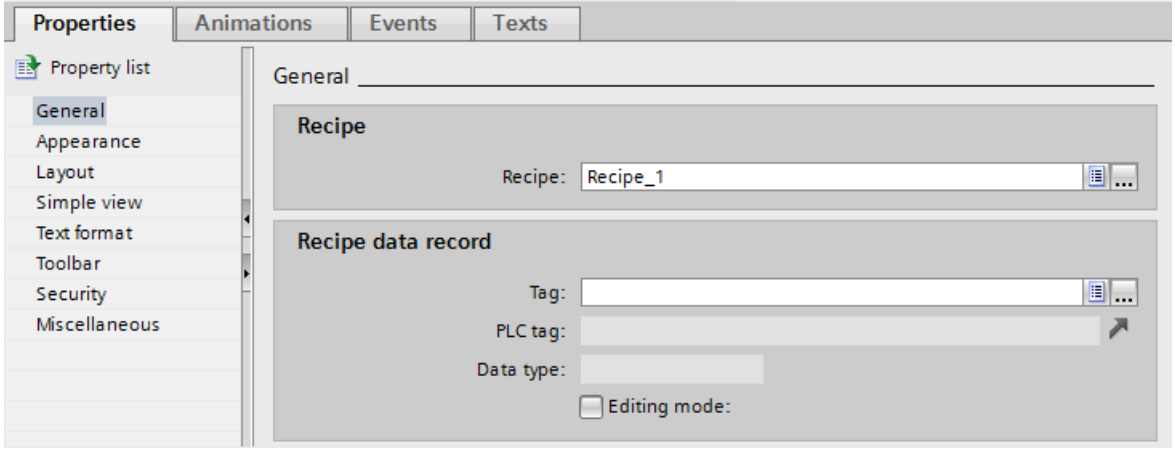
For all other devices, observe the following:

- In the Inspector window, under "Properties > Display > Mode", select "Simple view" as "View type".
- The "Properties > Simple view" area contains additional properties only applicable to the simple recipe view.
- All other properties are also applicable to the advanced recipe view.

Displaying recipe data record values only

To display the recipe data in a recipe view for checking only, proceed as follows:

1. Deselect "Edit mode" in the "General" group.



You cannot create, rename, edit or delete recipe data.

Writing a recipe data record number or name to a tag

A tag to each recipe data record can be configured in the advanced recipe view. Depending on the "String" or "Int" data type of the tag, the name or number of the recipe data record is stored in the tag. Conversely, the tag can also be used to select a recipe data record by entering the corresponding value. For example, the tag can be transferred as a parameter for a system function.

Proceed as follows:

1. Enter a tag of the "Int" type in the field "Tag", under "Properties > General > Recipe data record".

The number of the recipe data record is stored in a tag.

Configuring an event on the recipe view

Note

Events and buttons

The "Events" register is faded out when a minimum of one button is enabled.

To configure an event for the recipe view, proceed as follows:

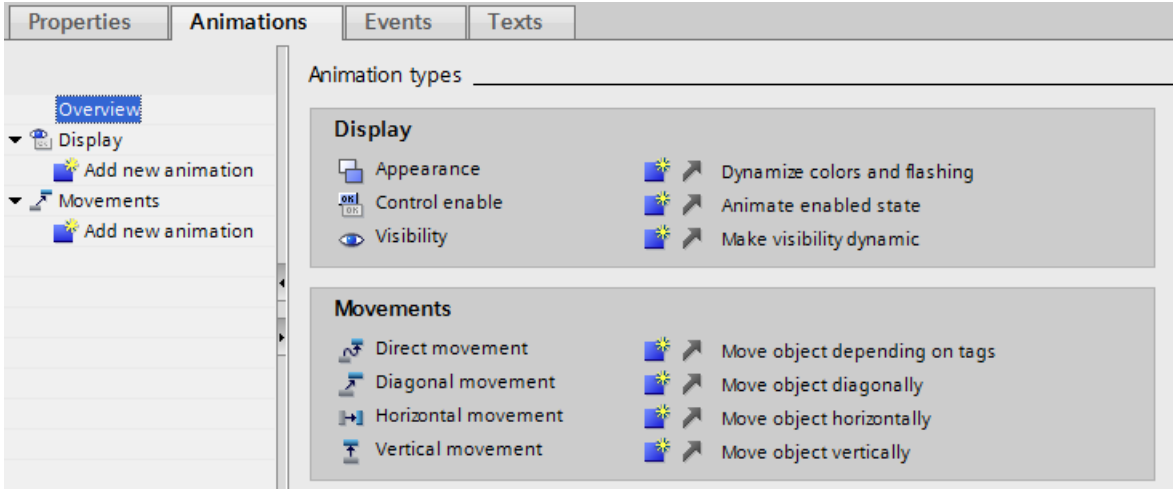
1. Select the recipe view that was added to the screen in the "Screens" editor.
   The properties of the recipe view are shown in the Inspector window.
2. Deactivate all buttons under "Properties > Toolbar" and "Properties > Simple view".
3. In the Inspector window, under "Properties > Events", click the event to be configured, e.g. "Enable".
4. Configure a function list for the event.

The function list is processed when the operator enables the recipe view.

Animation properties of the recipe view

To configure an animation of a recipe view, proceed as follows:

1. Select the recipe view that was added to the screen in the "Screens" editor. The properties of the recipe view are shown in the Inspector window.
2. Click under "Properties > Animations" in the Inspector window.



3. Link a tag to one or more of the following properties.
   - X-Position and y-Position
   - Design: Colors, flashing
   - Operability
   - Visibility
     In the "Animations> Overview" area, all animations are summarized in tabular form. Under "Animations> Tag links > Tag linkage", as well as visibility and position a tag can also be linked to height and width.

Note

Animations and buttons

If all buttons are not disabled, an error message appears whilst compiling the project for windows-CE HMI devices.

Constraints on the simple recipe view

The following functions are not possible in the simple recipe view:

- Synchronizing recipe view and recipe tags
- Writing a recipe number / name to a tag
- Display status bar
- Display data record number
- Display label
- Display table

12.5.2.4 Advanced recipe view

Recipe view

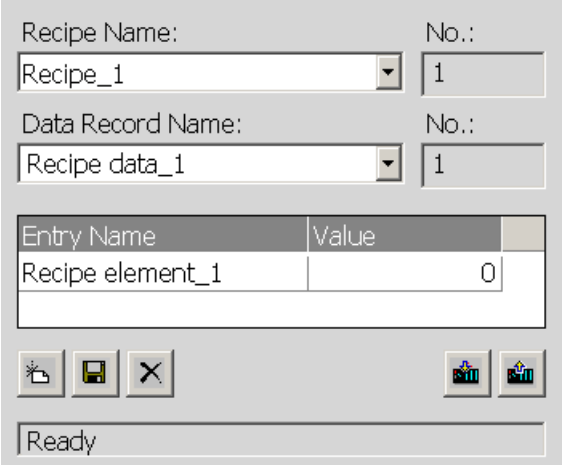
The advanced recipe view is an off-the-shelf display and operator control object that is used to manage recipe data records. The recipe view shows recipe data records in tabular form.

The buttons, headers and information displayed in the columns are variable.

The values displayed or entered in the recipe view are saved in recipe data records.

Layout of the display

The picture below contains an example of the advanced recipe view:



Display of values

Note

Changing the recipe data record in the background

Applies to the processing of a recipe data record:
If values of the corresponding recipe data record are changed by a control job, the recipe view is not updated automatically.

To update the recipe view, reselect the respective recipe data record.

12.5.2.5 Configuration options of the advanced recipe view

The response of the advanced recipe view in the recipe view inspector window can be determined.

Note

Validity

The following description also applies to the advanced recipe view. In the Inspector window, under "Properties > Display > Mode", select "Advanced view" as "View type".

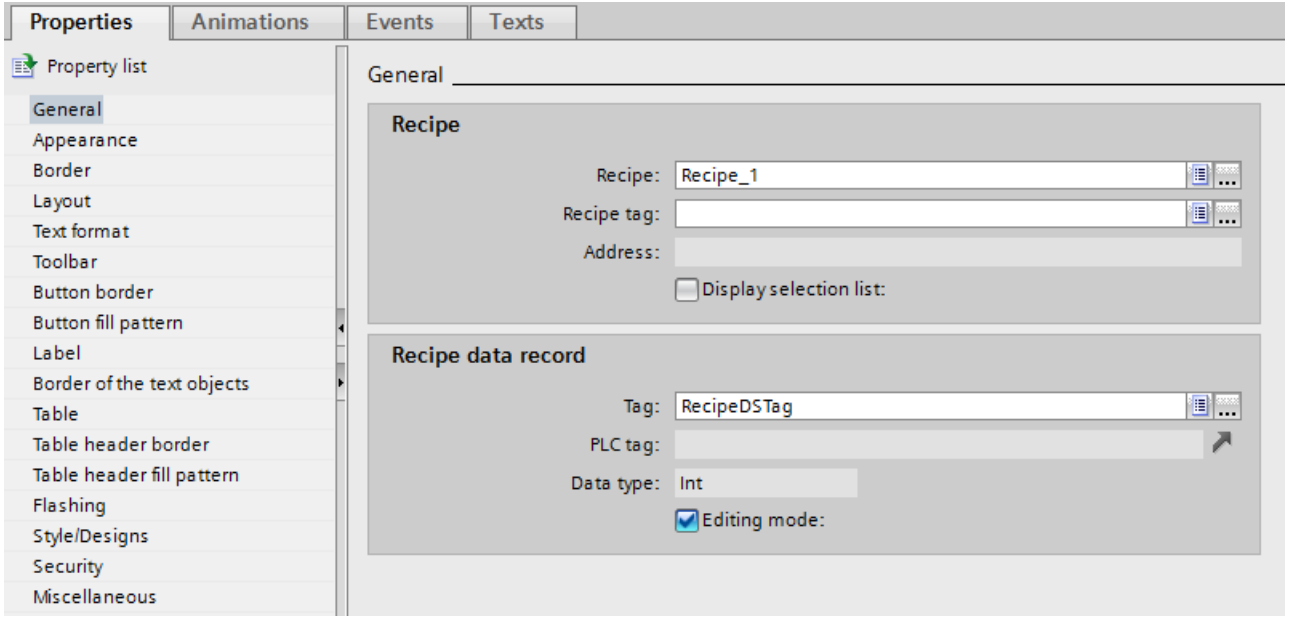
All properties described for the simple recipe view are also applicable in the advanced recipe view, except "Properties > Simple view". The "Simple view" area contains additional properties only applicable to the simple recipe view.

Basic Panels, OP77A and TP177A (Portrait) do not support any advanced recipe view.

Displaying a recipe

To allow access to the recipe data records of a specific recipe only in a screen, proceed as follows:

1. Enter the recipe or select an existing recipe in the "Recipe" field, under "Properties > General".
2. When a recipe is entered in the "Recipe" field, you will have to enable the "Selection list" if you want to display the recipe name in Runtime.



The selected recipe appears in the recipe view.

Writing a recipe number or name and recipe data record number or name to a tag

Both the recipe and the recipe data record can each be configured to a tag in the recipe view. Depending on the "String" or "Int" data type of the tag, the name or number of the recipe or recipe data record is stored in the tag. Conversely, the tag can be used to select the recipe or recipe data record by entering the corresponding value. For example, the tag can be transferred as a parameter for a system function.

Proceed as follows:

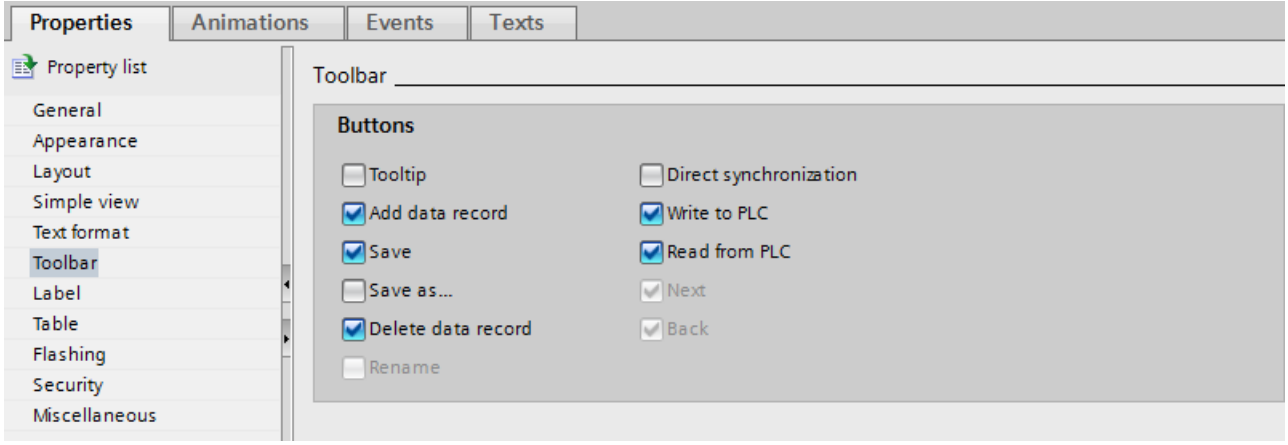
1. Enter a tag of the "String" type in the field "Recipe tag", under "Properties > General > Recipe".
2. Enter a tag of the "Int" type in the field "Tag", under "Properties > General > Recipe data record".

The name of the recipe and the number of the recipe data record are each stored in a tag.

Using the recipe view as a drop-down list

To use the recipe view as a selection field for recipes and recipe data records in a recipe screen, proceed as follows:

1. Select the tag for the recipe name under "General > Recipe > Recipe tag".
2. Select the tag for the recipe data record name under "General > Recipe data record >Tag".
3. Disable the "Edit mode". You cannot create, rename, edit or delete recipe data.
4. In order to select recipes, enable "Show selection list" and make sure that no recipe is selected under "Properties > General > Recipe".
5. Deactivate all buttons under "Properties > Toolbar".



Configuring an event on the recipe view

To configure an event on the recipe view, proceed as follows:

1. Select the recipe view that was added to the screen in the "Screens" editor.
   The properties of the recipe view are shown in the Inspector window.
2. In the Inspector window, click the event you want to configure under "Properties > Events".
3. Configure a function list for the selected event.

The function list is processed when the configured event occurs.

12.5.2.6 Advanced recipe view (as of V13)

Recipe view

The advanced recipe view is an off-the-shelf display and operator control object that is used to manage recipe data records. The recipe view shows recipe data records in tabular form.

The buttons, headers and information displayed in the columns are variable.

The values displayed or entered in the recipe view are saved in recipe data records.

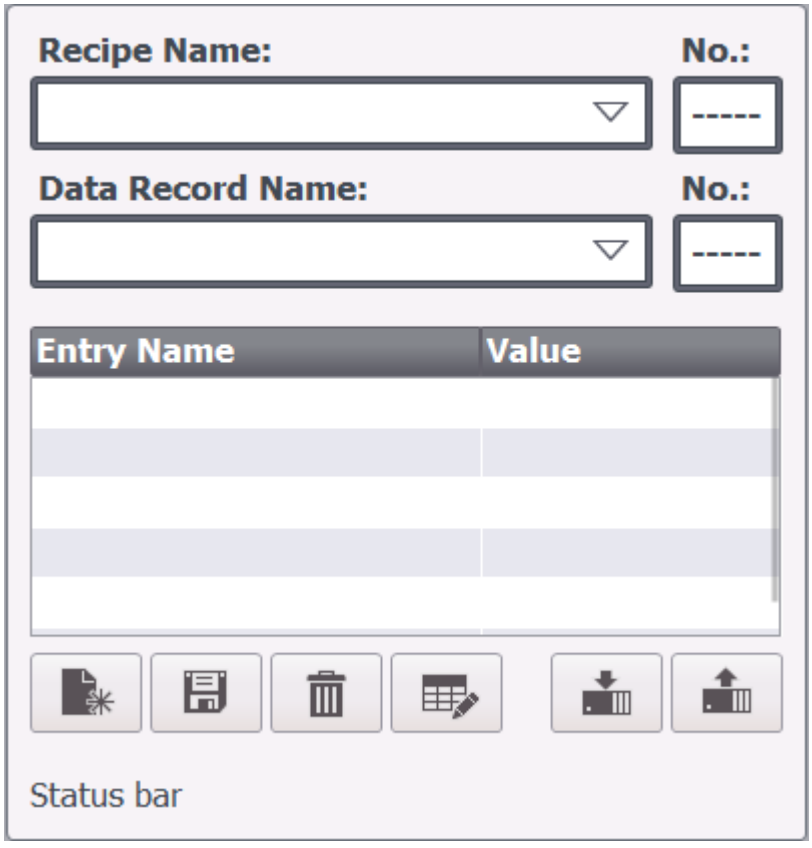
Note

Device dependency of the "Advanced recipe view" object

The "Advanced recipe view" object is available on the HMI devices Basic Panels 2nd Generation, Comfort Panels, Mobile Panels with the device version V13.

Layout of the display

The figure below shows an example of the advanced recipe view:



Show value

Note

Changing the recipe data record in the background

Applies to the processing of a recipe data record:
If values of the corresponding recipe data record are changed by a job mailbox, the recipe view is not updated automatically.

To update the recipe view, you must select the respective recipe data record again.

See also

Example of creating a recipe (Page 4485)

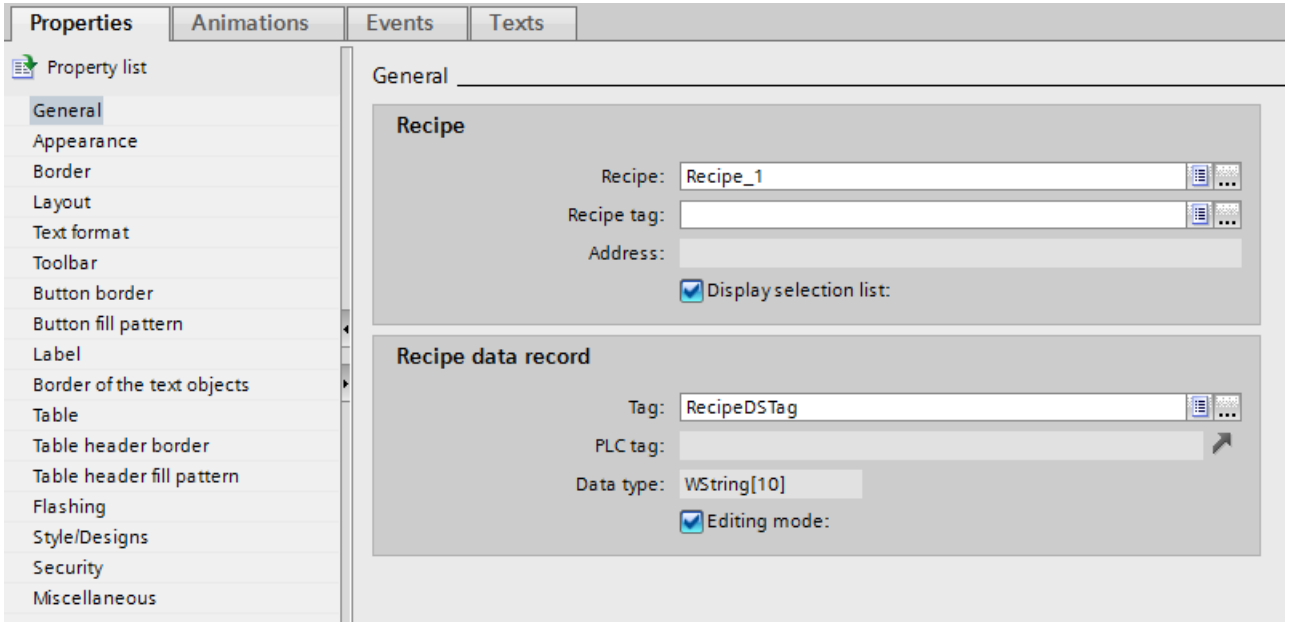
12.5.2.7 Configuration options of the advanced recipe view (V13 or higher)

You can specify the behavior of the advanced recipe view in the Inspector window of the recipe view.

Displaying a recipe

To only allow access to the recipe data records of a specific recipe in a screen, follow these steps:

1. Enter the required recipe or select an existing recipe in the "Recipe" field under "Properties > General".
2. When a recipe is entered in the "Recipe" field, you have to select "Display selection list" if you want to display the recipe name in Runtime.



The selected recipe appears in the recipe view.

Writing a recipe number or name and recipe data record number or name to a tag

Both the recipe and the recipe data record can each be configured to a tag in the recipe view. Depending on the "String" or "Int" data type of the tag, the name or number of the recipe or recipe data record is stored in the tag. Conversely, the tag can be used to select the recipe or recipe data record by entering the corresponding value. For example, the tag can be transferred as a parameter for a system function.

To do this, follow these steps:

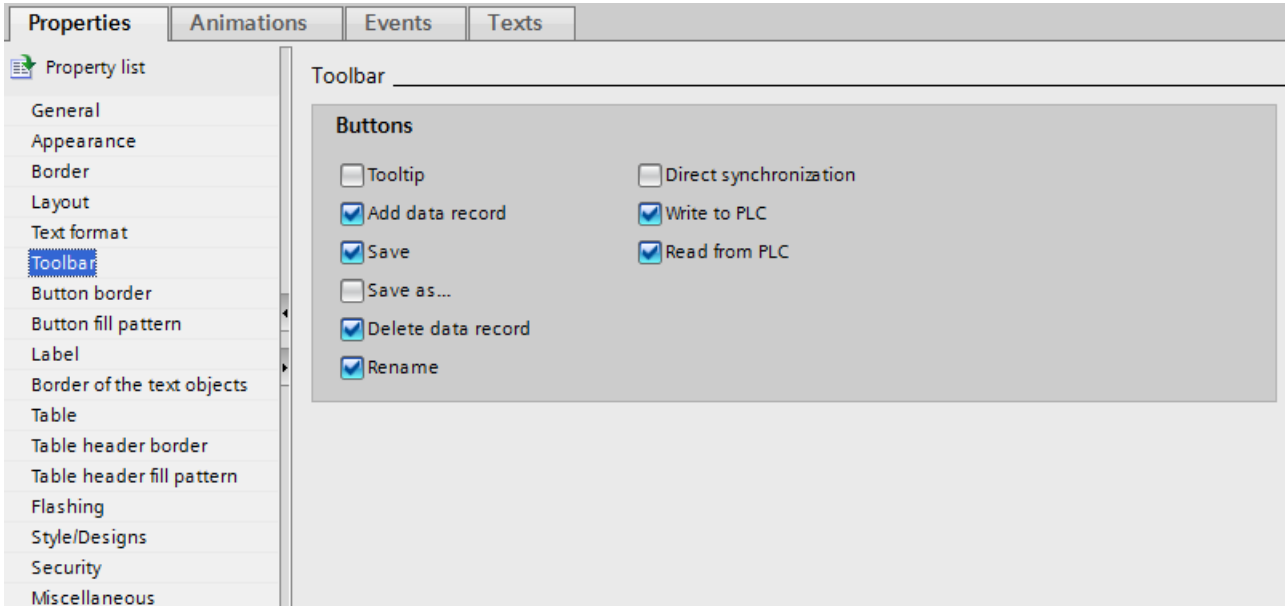
1. Enter a tag of the "String" type in the "Recipe tag" field under "Properties > General > Recipe".
2. Enter a tag of the "Int" type in the "Tag" field under "Properties > General > Recipe data record".

The name of the recipe and the number of the recipe data record are each stored in a tag.

Using the recipe view as selection field

To use the recipe view as a selection field for recipes and recipe data records in a recipe screen, follow these steps:

1. Select the tag for the recipe name under "General > Recipe > Recipe tag".
2. Select the tag for the recipe data record name under "General > Recipe data record >Tag".
3. Disable the "Editing mode". You cannot create, rename, edit or delete recipe data.
4. In order to select recipes, enable "Display selection list" and make sure that no recipe is selected under "Properties > General > Recipe".
5. Deactivate all buttons under "Properties > Toolbar".



Configuring an event on the recipe view

To configure an event on the recipe view, follow these steps:

1. Select the recipe view that was added to the screen in the "Screens" editor.
   The properties of the recipe view are displayed in the Inspector window.
2. In the Inspector window, click the event you want to configure under "Properties > Events".
3. Configure a function list for the selected event.

The function list is processed when the configured event occurs.

See also

Example of creating a recipe (Page 4485)

12.5.2.8 Response of the recipe view in Runtime

Screen change

If you change to another screen and have not yet saved changes to the recipe data in the recipe view, you will be prompted to save the recipe data. The recipe name and the name of the recipe data record are displayed to show which recipe data have not been saved yet.

Create, change, copy or delete recipe data records

If you attempt to create a new recipe data record and a recipe data record already exists, a system alarm will appear on screen.

Operating the recipe view with function keys

The Recipe view can be operated with function keys, e.g. if the HMI device does not have touch functionality. You can assign functions such as "SaveDataRecord" to the function keys on the HMI device.

Display after import of recipe data

Note

Availability

Import and export of recipe data is not available for Basic Panels and OP77A, TP177A (Portrait).

If you open the recipe view during the import of recipe data, only the recipe data that is already completely imported will be displayed. The recipe view is not automatically updated with a data import. In order to have a complete view of all the recipe data, do not open the recipe view until the system prompts you that the recipe data has been imported successfully. Alternatively, update the recipe view after successful completion of the import procedure.

Updating tag for recipes and recipe data records

Note

Availability

Tags for recipes and recipe data records are not available for Basic Panels and OP77A, TP177A (Portrait).

The current recipe data record or its number can be saved to a tag, depending on the configuration. The tag will be updated under the following conditions:

- The recipe data record has been loaded.
- The screen with the recipe view was not exited during loading.

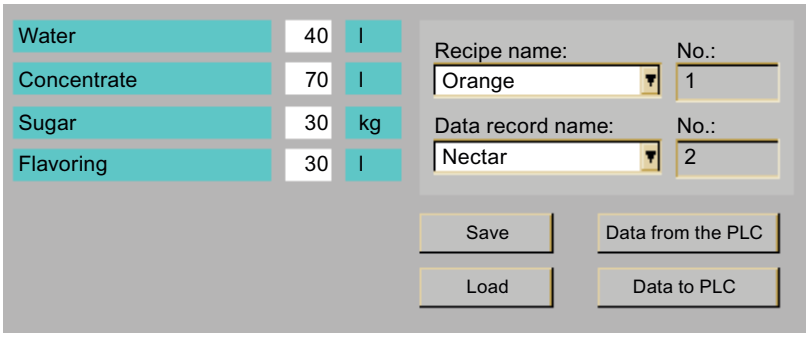
This operation may take some time.

12.5.2.9 Basics on the recipe screen

Introduction

The recipe screen contains an individual screen form for entering the recipes. The screen form contains I/O fields and other display and operator control objects. The recipe functionality is implemented with system functions, e.g. for saving recipe data records.

The picture below contains an example of a recipe screen:



Note

Availability of recipe screens

Basic Panels and OP73A, OP77A, TP177A (Portrait) do not support any recipe screens.

Principle

Configuring a recipe screen allows you to customize the display. You can spread large recipes over several screens according to topic and display them clearly using features such as graphical display and operator control objects.

- Spreading recipes over several screens according to topic
  - You can distribute recipe data records with multiple entries across several screens. For example, for each plant area you can configure a screen containing the associated screen forms for the recipe data records.

  Spreading recipes over several screens is useful for HMI devices with small displays as they avoid having to scroll through tables in Runtime.
- Visual machine simulation
  You can visually simulate your machine in a screen using graphical screen objects. This enables you to display parameter settings more clearly by positioning I/O fields immediately next to machine elements such as axes or guide rails. which produces a direct relationship between the values and the machine.

Synchronizing recipe values

To enter recipe data record values outside the advanced recipe view in the configured I/O fields, activate "Synchronize recipe view and recipe tags" under "Properties > Synchronization" in the recipe properties.

The "Synchronize recipe view and recipe tags" option is not available for the simple recipe view.

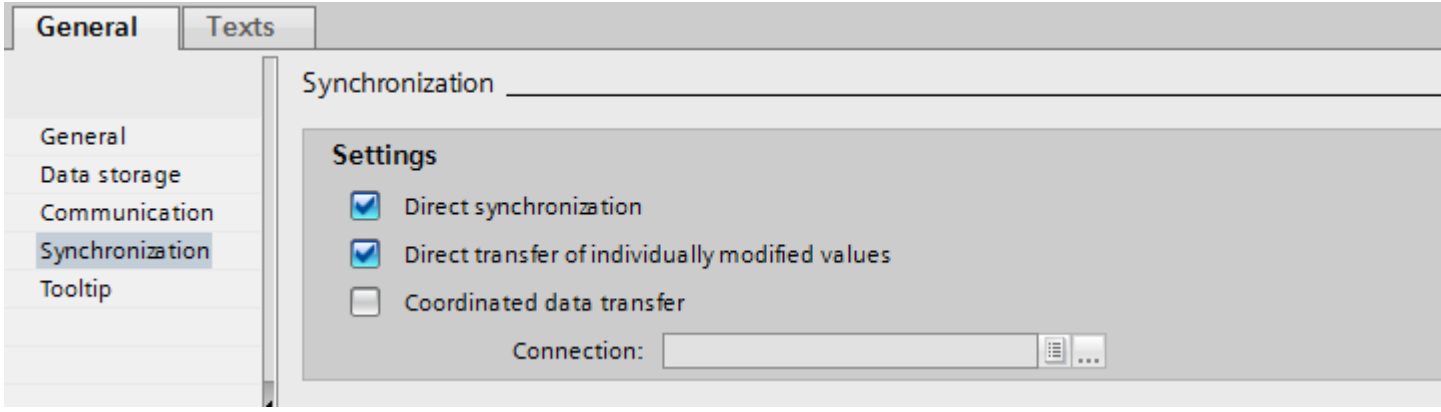


Figure 12-4 Recipes_Synchronization_Advanced

Transferring recipe values automatically

If the entered recipe values must be immediately transferred to the connected PLC in Runtime, deactivate "Manual transfer of individual modified values (teach-in mode)" under "Properties > Options".

Configure the "SetRecipeTags" system function if you want to enable and disable the immediate transfer of entered recipe values in Runtime.

System functions

The following system functions are available for operator control of a recipe screen:

- ImportDataRecords
- ExportDataRecords
- LoadDataRecord
- SaveDataRecord
- SetDataRecordTagsToPLC
- GetDataRecordTagsFromPLC

The following system functions are available for operator control of the recipe view when it is being used in the recipe screen:

- RecipeViewSaveDataRecord
- RecipeViewSaveAsDataRecord
- RecipeViewSynchronizeDataRecordWithTags

- RecipeViewDeleteDataRecord
- RecipeViewNewDataRecord
- RecipeViewGetDataRecordFromPLC
- RecipeViewRenameDataRecord (for simple recipe view only)
- RecipeViewShowOperatorNotes
- RecipeViewMenu (for simple recipe view only)
- RecipeViewOpen (for simple recipe view only)
- RecipeViewBack (for simple recipe view only)

The system functions for loading, saving and transferring recipe data records and recipes are located in the "Recipes" group.

12.5.3 Configuring Recipes

12.5.3.1 General configuration procedure

Carry out the following configuration steps when you create a new recipe:

Step	Description
1	Define the structure of the recipe.
2	Create tags according to the recipe structure. Assign process names to these tags.
3	Create the recipe.
4	Enter the required properties for the recipe: • Language-dependent view name of the recipe • "Coordinated transfer of data records" option Not for Basic Panels: • Recipe storage location • "Synchronize recipe view and recipe tags" option • "Manual transfer of individual modified values (teach-in mode)" option
5	Create the recipe elements and enter the required properties: • Language-dependent view name of the recipe elements • Tag binding of the recipe elements • Standard values and decimal places (power of ten) for the recipe elements
6	Create the recipe data records. Enter the language-specific display names for the recipe data records.
7	Configure a screen with recipe view or a recipe screen.

Note

Basic Panels and OP77A, TP177A (Portrait)

The selection of the storage location is not available for these devices. The recipes are always saved in the internal Flash memory.

Recipe tags cannot be used outside a recipe, e.g. not in I/O fields, not in alarms as trigger tags, not in systems functions as parameters, etc.

Note

Restrictions recipe view and recipe image

Only the simple recipe view is available in Basic Panels and OP77A, TP177A. Recipe images are not available in Basic Panels and OP73, OP77A, TP177A (Portrait).

12.5.3.2 Creating and Editing Recipes

Creating a new recipe

Introduction

To create a complete recipe, start by creating a new recipe, assign the corresponding recipe elements and then define the associated values in a recipe data record.

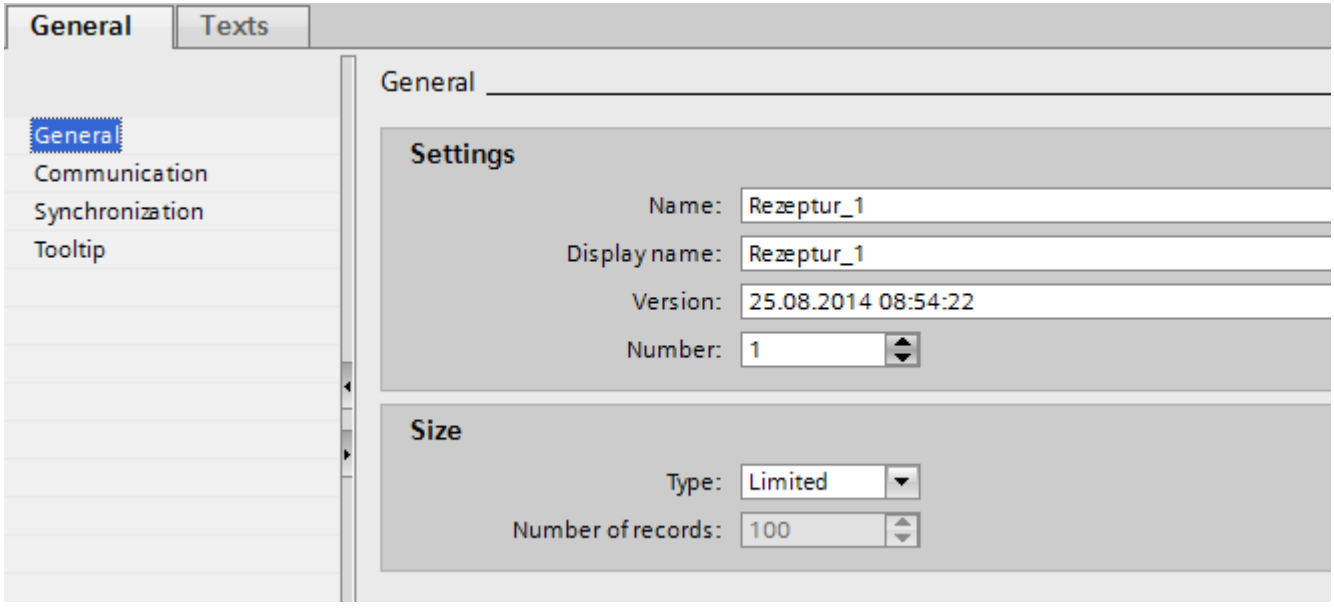
Requirement

- The tags for the recipe have been created.
- The "Recipes" editor is open.

Create recipe

Create a recipe as follows:

1. Click "Add" in the first free row of the table in the "Recipes" editor.
   The new recipe is created and displayed on a line.



2. Enter a descriptive name for the recipe under "Name" in the "General" area.
   This name identifies the recipe unambiguously within the project.
3. Select "Display name" to enter the language-specific name to be displayed in runtime.
4. Select a recipe number in "Number".
   The number identifies the recipe unambiguously within the HMI device.
   The recipe is automatically assigned a version that indicates the date and time of the last change. As an alternative, you can enter specific information relating to the recipe.
5. Specify the storage location for recipe data records in "Data medium". The options offered depend on the specific HMI device used.

   Note

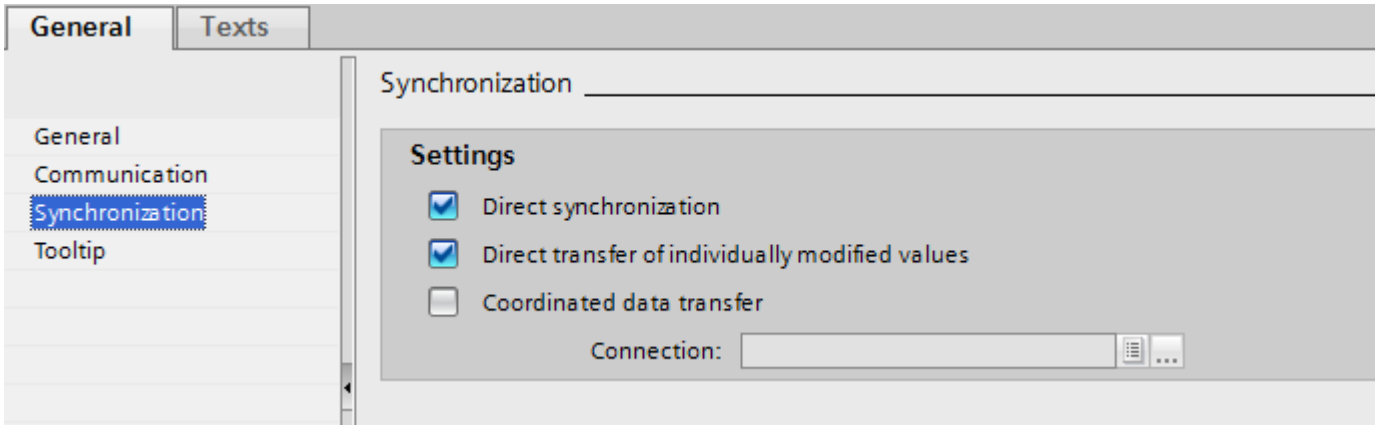
   Basic Panels and OP77A, TP177A (Portrait)

   The selection of the storage location is not available for these devices. The recipes are always saved in the internal Flash memory.

   Recipe tags cannot be used outside a recipe, e.g. not in I/O fields, not in alarms as trigger tags, not in systems functions as parameters, etc.

6. Enter a tooltip that is shown to the operator in runtime.

7. To compare recipe tags which are configured in I/O fields with the recipe view in Runtime, activate "Synchronize recipe view and recipe tags" in the Inspector window under "Properties > Synchronization".



Note

Basic Panels and OP77A, TP177A (Portrait)

Because the recipe tags cannot be additionally used in I/O fields in screens for Basic Panels, the "Synchronize recipe view and recipe tags" is not available; you will also not be able to use the "Manual transfer of individual modified values (teach-in mode)" option.

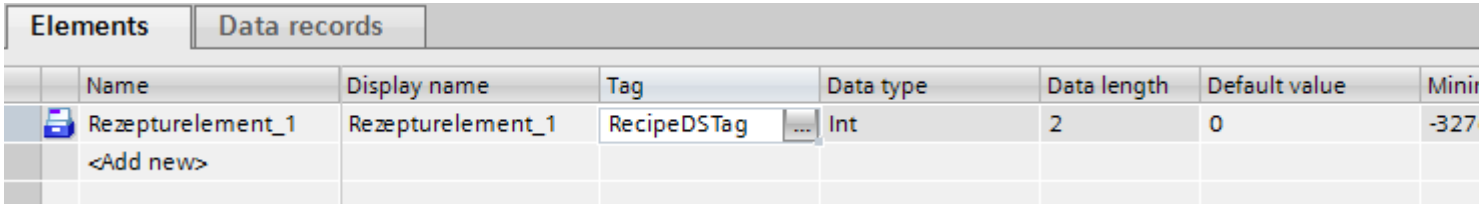
8. Deactivate "Manual transfer of individual modified values (teach-in mode)" to specify that the recipe tags are automatically transferred to the PLC when editing the I/O fields.
9. Activate "Coordinated transfer of data records" to monitor the transfer of recipe data in Runtime using area pointers.
10. Select the appropriate connection to the PLC for coordinated transfer under "Synchronize with".

Create recipe element

To create recipe elements, proceed as follows:

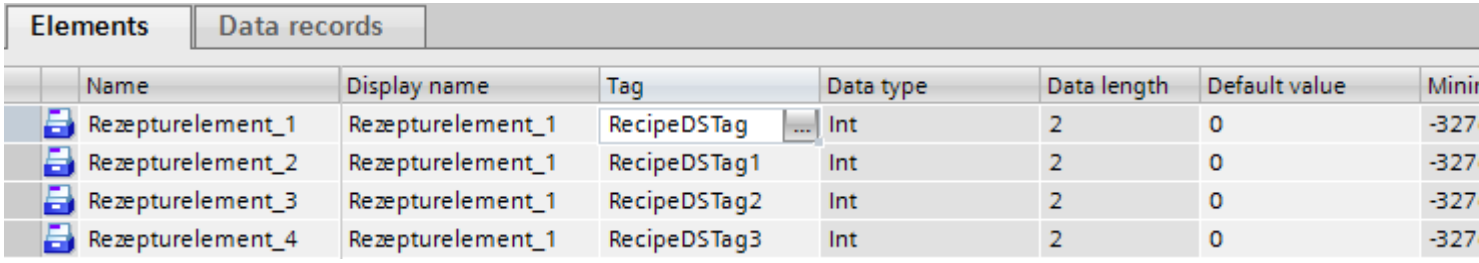
1. Click the "Elements" tab.
2. Click "Add" in the first free line of the table editor.
   A new recipe element is created.
3. Enter a descriptive name for the element under "Name".
   The name identifies the element uniquely within the recipe.
4. Enter a language-specific display name for the element under "Display name".
   The display name appears in the recipe view, for example, in runtime.

5. Select the tag you want to link to the recipe element under "Tag".
   The value of the recipe data element is saved in Runtime in this tag, which is stored in a recipe data record.



	Name	Display name	Tag	Data type	Data length	Default value	Mini
	Rezepturelement_1	Rezepturelement_1	RecipeDSTag	Int	2	0	-327
	<Add new>						

6. Enter a tooltip.
   The tooltip is shown to the operator in Runtime.
7. Under "Default value", enter the value that you want to use as the default entry when you create a new recipe data record.
8. To assign text to a value or range of values, select the relevant text list here. The assigned recipe tag must have the data type of a number. The tag value must be within the range of values of the text list.
   The text stored in the text list is displayed in an output field, for example, in Runtime.
9. Determine exactly how many places a decimal number is rounded to in the "Decimal places" column, e.g. 3 decimal places and vice versa by what power of ten an integer value is multiplied, e.g. 1,000.
   Examples for 3 decimal places: Entering "5" for a recipe element with the "Integer" data type gives the value "5000". Entering "5.6789" for a recipe element with the "Real" data type gives the value "5.679".
10. Create as many recipe entries as needed for the recipe. The maximum number of recipe entries possible depends on the HMI device being used.



	Name	Display name	Tag	Data type	Data length	Default value	Mini
	Rezepturelement_1	Rezepturelement_1	RecipeDSTag	Int	2	0	-327
	Rezepturelement_2	Rezepturelement_1	RecipeDSTag1	Int	2	0	-327
	Rezepturelement_3	Rezepturelement_1	RecipeDSTag2	Int	2	0	-327
	Rezepturelement_4	Rezepturelement_1	RecipeDSTag3	Int	2	0	-327

Create recipe data record with known recipe values

To create recipe elements, proceed as follows:

1. Click the "Data records" tab.
2. Click "Add" in the first free line of the table editor.
   A new recipe data record is created. The recipe data record has a separate column for every recipe element created in the recipe.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Recipe_data_record_1	Recipe_data_record_1	1	0	0	0	0	
<Add new>							

3. Enter a descriptive name under "Name".
   The name identifies the data record uniquely within the recipe.
4. Enter a language-specific name under "Display name".
   The display name appears in the recipe view, for example, in runtime.
5. Enter a recipe data record number under "Number".
   The recipe data record number identifies the recipe data record uniquely within the recipe.
6. If you already know the recipe values at the configuration stage, you can enter the relevant value for each recipe element.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
<Add new>							

7. Create as many data records as you need for the recipe.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
Nectar	Nectar	2	50	50	10	300	
Juice	Juice	3	5	95	3	100	
<Add new>							

Enter the values in runtime

The following options are available for entering values in the recipe data records at runtime:

- Transfer data directly from the PLC (Teach-in mode)
- Import of values from a CSV file
- Input values on the HMI device

Note
Basic Panels and OP77A, TP177A (Portrait)

The import of values is not available for these devices.

Result

The complete recipe is configured.

Recipe data records with date or time stamp

If you use date or time data, make sure that the system setting for time and date on the configuring computer match those on the target system. Example: You load a recipe data record on the target system at 13:55 in which 14 h is stored as the processing time. If it is already 14:05 on the target computer, the recipe will not be processed. If an operator processes the recipe, change information will not written back correctly into the database.

After loading to the target system, check the recipes with date or time stamps on the target system.

Editing a recipe

Purpose

You want to modify, extend or delete parts of a recipe.

Requirement

- You have created at least one recipe.
- The "Recipes" editor is open.

Changing recipe settings

To change the recipe settings, proceed as follows:

1. Select the recipe that you want to change in the "Recipes" editor.
   The Inspector window opens.
2. Change the recipe configuration in the Inspector window.

You change recipe elements and recipe data records in the same way.

Change recipe values

To change recipe values, proceed as follows:

1. Select the recipe whose values you want to change.
2. Click the "Data records" tab.
3. Enter the new values in the value columns.

Adding a recipe element

To add more recipe elements to a recipe, proceed as follows:

1. Select the recipe to which you want to add more elements in the "Recipes" editor.
2. Click the "Elements" tab.
3. Click "Add" in the first free line.
   The recipe element is created.
4. Configure the recipe element.

You add recipe data records in the same way.

Managing recipes

Requirement

- You have created a recipe with recipe elements and recipe data record.
- The "Recipes" editor is open.

Renaming recipes

We distinguish between internal names and display names for recipes, recipe entries and recipe data records.

To rename recipe elements, proceed as follows:

1. Select the recipe that you want to rename.
   The Inspector window opens.
2. Select the "Rename" command from the shortcut menu.
3. Enter the new name.
   You rename recipe elements and recipe data records on the relevant tab in the same way.

Note

The view names in the "Recipes" editor can also be renamed under "Languages & Resources > Project Texts". This possibility is useful when you have already configured in several languages for example.

Copying and pasting recipes

To copy and paste recipes, proceed as follows:

1. Select the recipe that you want to copy.
2. Select the "Copy" command from the shortcut menu.
3. Select the "Paste" command from the shortcut menu in the first free table row.

The copied recipe is pasted into the table. The recipe elements and recipe data records are also copied in the appropriate tab with the recipe.

You also copy the recipe elements and recipe data records on the appropriate tab in the same way.

If a recipe data record of the same name already exists, the name of the copied recipe data record is extended by one digit. This ensures that the name is unique. Recipe data records can only be copied or pasted within the same recipe.

Deleting a recipe

To delete a recipe, proceed as follows:

1. Select the recipe that you want to delete.
2. Select the "Delete" command from the shortcut menu.
   The recipe is deleted.

You delete recipe elements and recipe data records on the relevant tab in the same way.

Note

When a recipe is deleted, the recipe data records contained in the recipe are also deleted.

Note

When you delete a recipe element, the associated values in the recipe data records are also deleted. The assigned tags are retained.

12.5.3.3 Configuring the display of recipes

Configuring the simple recipe view

Requirement

- The recipe has been created.
- The "Screens" editor is open.
- The screen has been created and opened.

Notice

Data loss with several recipe views in the screen

Applies only to Basic Panels, OP73, OP77A, TP177A and TP177A (Portrait): If two or more recipe views show the same recipe in a screen, you have a conflict when accessing the data.

The result is data loss and unpredictable status of recipe data.

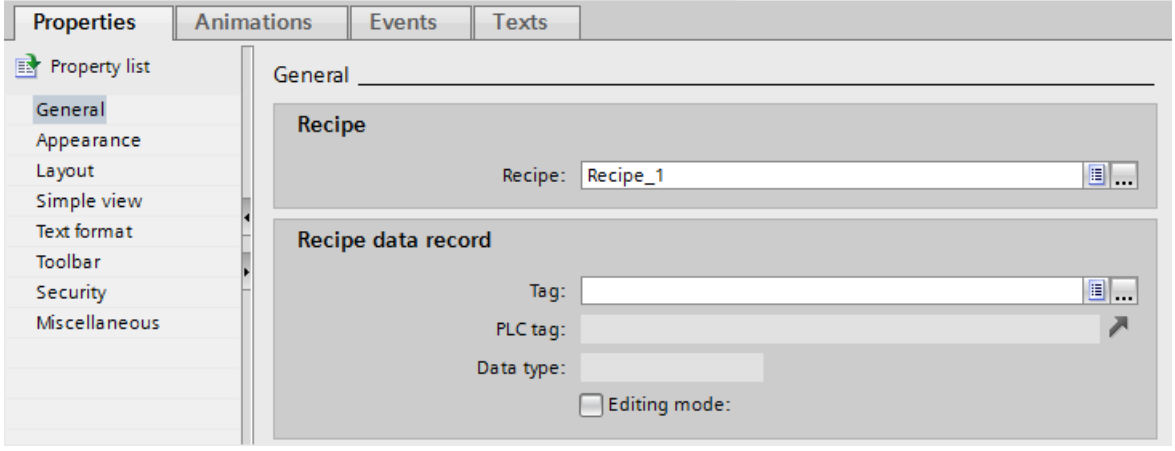
Make sure the operators do not select and edit the same recipe in different recipe views.

- Display only one recipe in a recipe view.
- Display a different recipe in each recipe view.

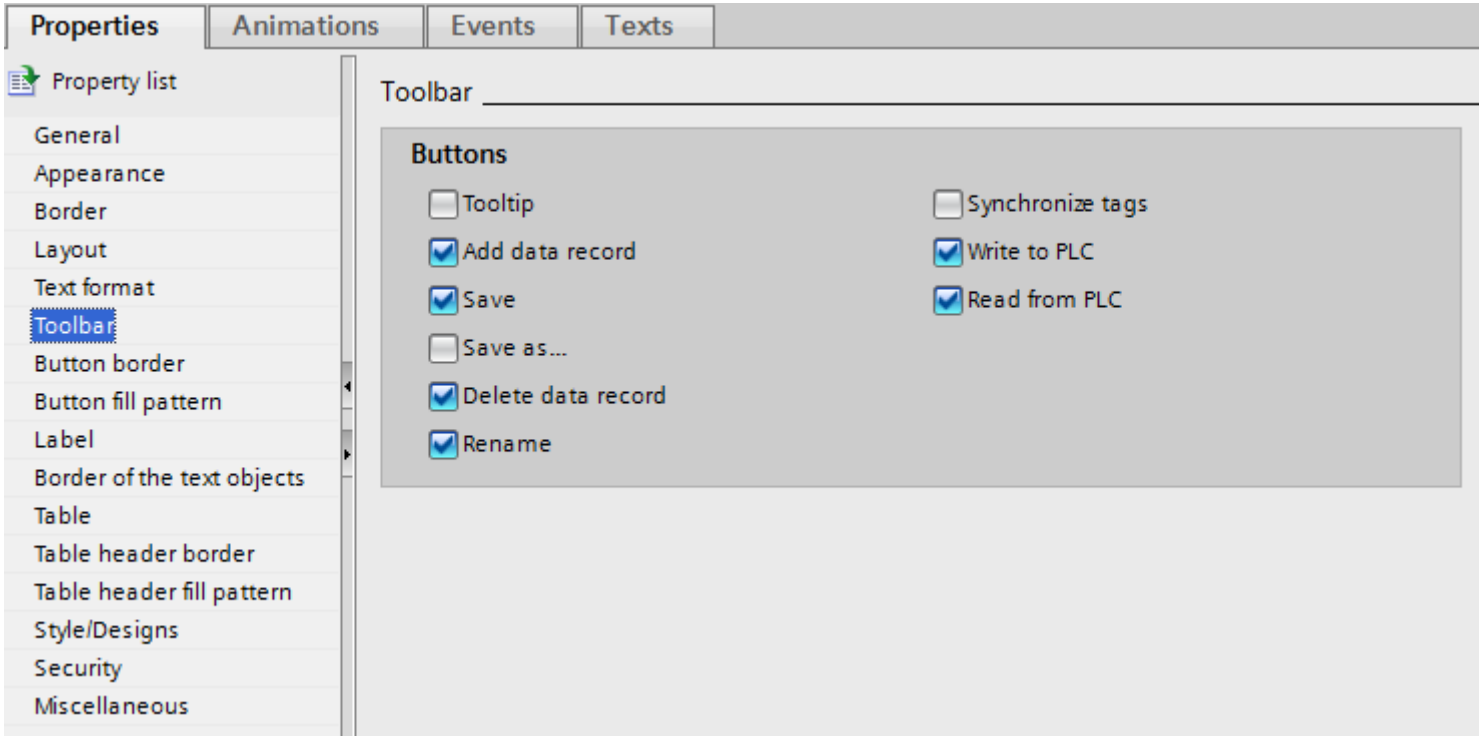
Procedure

To configure a simple recipe view, proceed as follows:

1. Paste the recipe view into the screen. You will find the recipe view under "Controls" in the "Tools" task card.
2. Only in devices which also support the extended recipe view: Activate "Simple view" under "Properties > Display > Mode".
3. If you want to display only the recipe data records of a specific recipe in the recipe view, select the specific recipe under "Properties > General > Recipe".



4. If you only want to display the recipe data in the recipe view, deactivate "Processing mode" in the "Recipe data record" area.
5. You can define additional options for the recipe view under "Properties > Appearance" and "Properties > Layout".
6. Under "Properties" > Toolbar" specify which menu commands are available in the recipe view in Runtime.



Result

The simple recipe view is configured. You can use the recipe view to display and edit recipe data during runtime.

Deactivation of the editing mode in "Properties > Properties > General" has no impact on the toolbar icons. The buttons activated under "Properties > Toolbar" can still be used even if editing mode is disabled.

Configuring the Advanced Recipe View

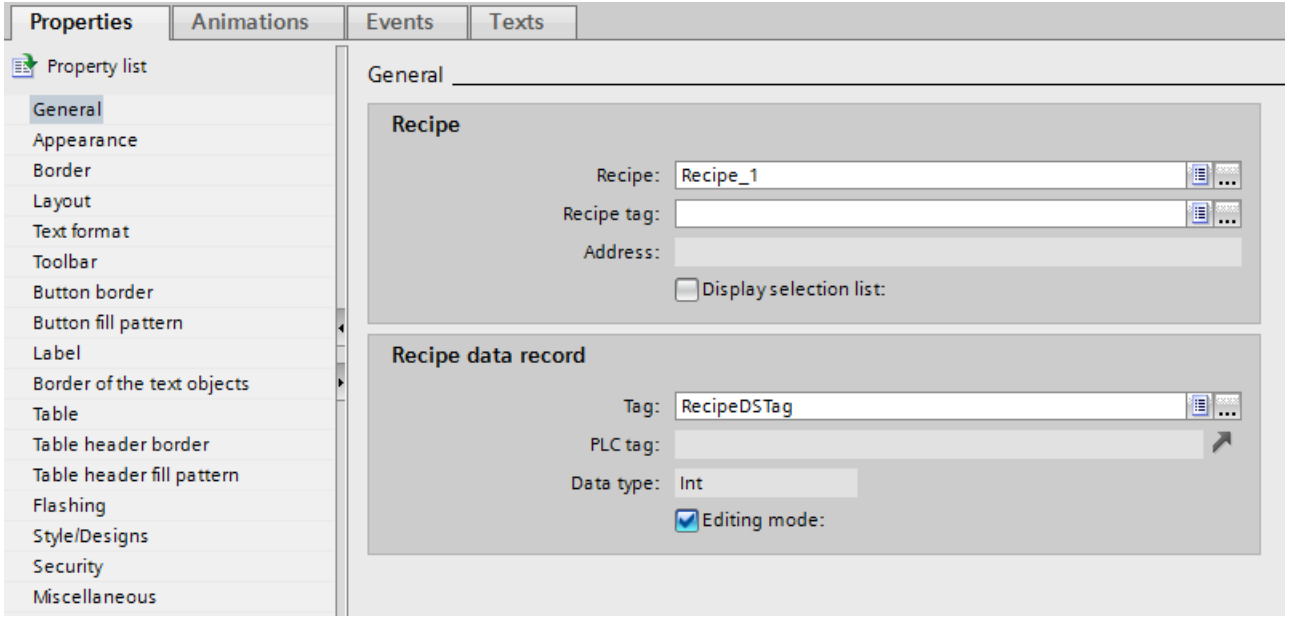
Requirement

- The recipe has been created.
- The "Screens" editor is open.
- The screen has been created and opened.

Procedure

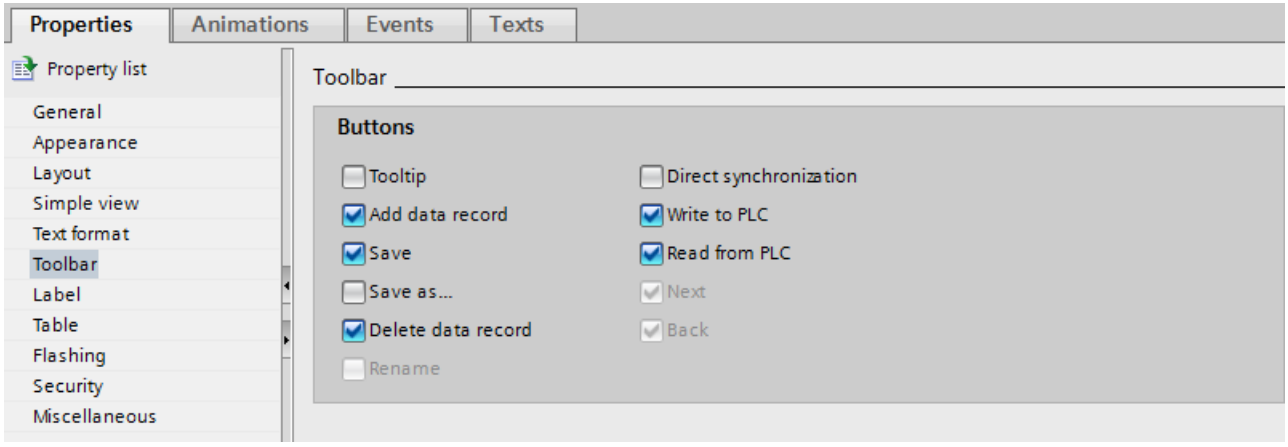
To configure an advanced recipe view, proceed as follows:

1. Paste the recipe view into the screen. The recipe view is found in the task card "Tools" > "Controls".
2. Select "Properties > Display > Mode > Advanced view" in the Inspector window.



3. Select the required settings from the "General" group in the Inspector window.
   - If you want to display only the recipe data records for a particular recipe in the recipe view, select the recipe under "Recipe" in the "Recipes" area.
   - If you want to save the recipe name or the recipe number in a tag, select the tag under "Recipe tag" in the "Recipe" area.
   - If you want to save the recipe data record name or number in a tag, select the tag under "Tag" in the "Recipe data record" area.
   - If you only want to display the recipe data in the recipe view, deselect "Edit mode".
   - If you only want to use the recipe view to select recipes, disable "Display table" under "Properties > Table".
4. You can define additional options for the recipe view under "Properties > Appearance" and "Properties > Layout".

5. If you want to change the label in the recipe view, enter a suitable text under "Properties" > "Label".
6. Under "Properties" > Toolbar" specify which buttons are available in the recipe view in Runtime.



Note

If you select the "Edit" command in the context menu of the recipe view, the recipe view becomes active. To activate the recipe view, the zoom factor must be set to 100%.

You can set the column width and the position of the "Entry name" and "Value" columns in active mode.

Result

The recipe view is configured. You can use the recipe view to display and edit recipe data during runtime.

Deactivation of the editing mode in "Properties > Properties > General" has no impact on the toolbar icons. The buttons you activated in "Properties > Toolbar" can still be used even if editing mode is disabled.

Configuring a recipe screen

Introduction

The recipe screen is a screen in which you configure a customized screen form in the "Screens" editor. You create the screen form from input/output fields and other display and operator control objects. System functions are used to configure the recipe functionality, such as saving recipe data records.

Note

Availability

Recipe screens cannot be created for basic panels and OP73, OP77A and TP177A (Portrait).

Application

You can spread recipe data records containing lots of entries across several screens. For example, for each plant you can configure a screen containing the associated screen forms for the recipe data records.

You can visually simulate your machine in a screen using graphical screen objects. This enables you to display parameter settings more clearly by positioning I/O fields immediately next to machine elements, such as axes or guide rails. You can use this to produce a direct reference between the values and the machine.

Requirement

- You have created the recipe.
- The "Screens" editor is open.

Procedure

To configure a recipe screen, proceed as follows:

1. Configure the screen and create the I/O fields for the input mask of the recipe.
   You can create multiple screens to suit the size and complexity of the recipe.
2. Configure the I/O fields with the tags you have linked to the recipe element.
3. Configure I/O fields for selecting recipes and recipe data records.

Alternatively:

1. Configure a recipe view as a selection list for recipe data records and recipes.
2. Hide the buttons that are not required in the recipe view.
3. Configure the system functions for editing recipe data records on the configured control elements.
   Control elements are configured buttons in the screen or function keys on the HMI device.
   You will find the system functions for editing recipe data records under "Recipes" and "Keyboard operation for screen objects".

Result

The recipe screen is created.

12.5.3.4 Importing recipes into the configuration and exporting them

Introduction

You can export recipes as a CSV file and import them again into the configuration.

Application case

Recipe data is exported for long-term storage and backup on a computer.

In order to unify and distribute recipe data, export the recipe data to an HMI device. Change the CSV file in Microsoft Excel and import the CSV file to all HMI devices that require the same recipe data.

You want to exchange recipe data between different projects. You change the recipe data in Runtime. To transfer the modified recipe data to WinCC ES, export the recipe data records in Runtime and copy the CSV file to the configuring computer. There you import the CSV file containing the recipe data records into the recipe. And in the opposite direction, you transfer the modified recipe data records to the HMI device in WinCC ES in Runtime.

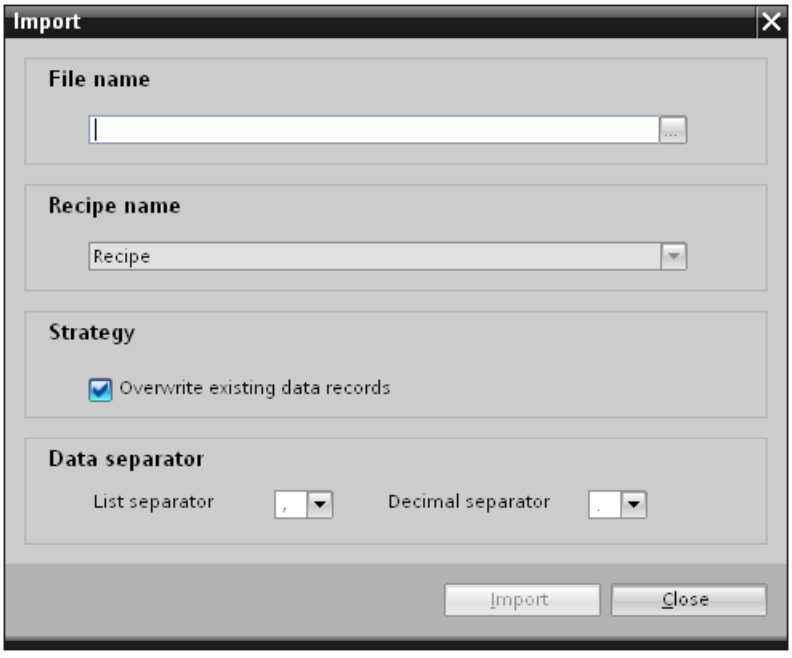
Requirement

- You have created the recipe.
- The structures of the exported CSV file and the recipe in WinCC ES match; name, number and data type of the recipe elements are identical.
- The "Recipes" editor is open.

Importing CSV files

To import recipe data records into a recipe, proceed as follows:

1. Select the row which contains the desired recipe in the "Recipes" tab.
2. Select the "Import recipe data" command from the shortcut menu.
   The "Import" dialog box opens.



3. To select the desired CSV file, go to "File name".
4. Under "Strategy", specify whether a recipe data record with the same recipe number in WinCC ES should be overwritten.

5. Specify the list separator and decimal separator under "Data separation". The list separator separates individual recipe elements in the CSV file. The decimal separator separates integers and decimal places.

   Note

   Use the same list separator for import as in the CSV file for export.

6. Click "Import" to start the operation.

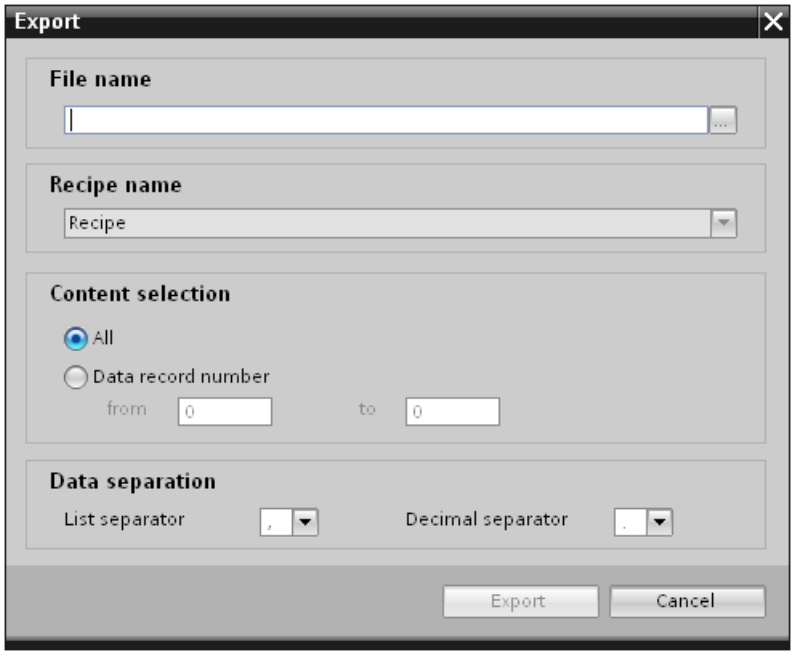
Result

The recipe will be amended by recipe data records from the CSV file that have not yet been entered. If the option is selected, existing recipe data records will be overwritten.

An error message is shown in the "Info" of the Inspector window if the structure of the recipe does not match the structure of the CSV file.

Exporting CSV files

1. Select the row which contains the recipe to export in the "Recipes" tab.
2. Select the "Export recipe data" command from the shortcut menu.
   The "Export" dialog box opens.



3. Under "File name", specify the path of the CSV file.
4. Select all recipe data records under "Selection content", or restrict the selection to specific record numbers of the recipe data.

5. Specify the list separator and decimal separator under "Data separation". The list separator separates individual recipe elements in the CSV file. The decimal separator separates integers and decimal places.
6. Click "Export" to start the operation.

Result

The configuration is exported as a CSV file.

12.5.4 Using Recipes in Runtime

12.5.4.1 Simple recipe view

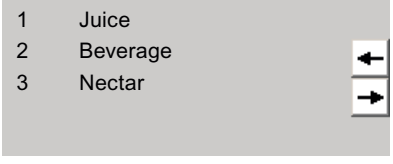
Description of the simple recipe view

Layout

The simple recipe view consists of the following display areas:

- Recipe list
- Data record list
- Element list

This application is illustrated below:



In the simple recipe view, each area is shown separately on the HMI device. You can use the shortcut menu to operate each of these display areas.

The simple recipe view always begins with the recipe list.

Operation

You have the following options for using the simple recipe view, according to the configuration:

- Create, change, copy or delete recipe data records
- Read recipe data records from the PLC or transfer to the PLC

Using the display area and shortcut menu

Toggle between the display areas and the shortcut menus to operate the simple recipe views.

The table below shows the operation of the display area.

Button	Key	Function
	<Enter>	The next lowest display area is opened, i.e. the data record list or the element list.
←	<Esc>	The previous display area opens.
	<INS>	Creates a new data record for the selected recipe if the list of recipes or recipe data records is displayed. Then changes to the list of recipe element. Requirement: "Properties >General > Processing mode" is activated. The button can be simulated with the "Key SimulateSystem-Key" function even on devices without keys.
	<DEL>	Deletes the selected recipe data record in the list of recipe data records. Requirement: "Properties >General > Processing mode" is activated.
	<Up>/<Down>	Selects the previous/next entry.
	<Pg Up>/<Pg Down>	Moves the display up or down one page.
	<Home>/<End>	Selects the first/last entry. The first/last entry is selected.

The table below shows the operation of the shortcut menu:

Button	Key	Function
→	<Right>	The shortcut menu of the display area opens.
←	<Esc>	The menu is closed. The display area opens.
	Input of the number of the menu command	The menu command is executed.

Shortcut menus of the simple recipe view

You can click the → button in each display area to call up a selection of commands. The command selection lists those commands that are available in the current display area. A number is assigned to each command. The command is executed when you enter this number. Alternatively select the command and press the <Return> key.

Shortcut menus in the recipe list

Menu command	Function
New	A new recipe data record is created for the selected recipe. If a start value is configured, it is displayed in the input field.
Display tooltip	The tooltip configured for the recipe is displayed.
Open	The record list of the selected recipe opens.

Shortcut menus of the recipe data record list

Menu command	Function
New	Creates a new recipe data record. If a start value is configured, it is displayed in the input field.
Deleting	The displayed record is deleted.
Save as	The selected data record is saved under a different name. A dialog box opens where you can enter the name.
Rename	Renames the selected data record. A dialog box opens where you can enter the name.
Open	The element list of the selected data record opens.
Previous	The recipe list opens.

Shortcut menus of the recipe element list

Menu command	Function
Save	The selected data record with the recipe element is saved.
To PLC	The displayed values of the selected data record are transferred from the HMI device to the PLC.
From PLC	The recipe values from the PLC are displayed in the recipe view of the HMI device.
Save as	The data record is saved under a new name. A dialog box opens where you can enter the name.
Display tooltip	The tooltip configured for the recipe element is displayed.
Rename	The selected recipe element is renamed. A dialog box opens where you can enter the name.
Previous	The data record list opens.

Shortcut menus in the data record list

Note

HMI device dependency

The following menu commands are configured in Basic Panels and in OP 77A, TP 177A, TP 177A (Portrait) and TB 177B.

Menu command	Function
To PLC	The displayed values of the selected data record are transferred from the HMI device to the PLC.
From PLC	The recipe values from the PLC are displayed in the recipe view of the HMI device.

Using the simple recipe view

Controlling the simple recipe view with mouse or touchpad

To control the simple recipe view with mouse or touchpad, proceed as follows:

1. Select the desired recipe from the recipe view.
2. Click the → button.
   The shortcut menu is opened.
3. Select the desired menu command.
   The menu command is executed.

Controlling the simple recipe view with the keyboard

To control the simple recipe view with the keyboard, proceed as follows:

1. Press the <Tab> key until the simple recipe view is selected.
2. Select the desired recipe with the cursor keys.
3. Press <Right>.
   The shortcut menu is opened.
4. Press the <Down> key until the desired menu command is selected.
5. Press <Enter> to confirm the command.

Key shortcuts for the simple recipe view

The following key shortcuts are activated for the simple recipe view in Runtime if "Activate keyboard operation" is enabled in the ES.

Key shortcut	Effect	Menu command
<Insert>	Generates a new recipe data record	New
<Del>	Deletes the recipe data record displayed.	Delete

Managing recipe data records

Recipe data record administration

You have the following options for managing the simple recipe view, according to the configuration:

- Creating new recipe data records
- Copy recipe data records
- Edit recipe data records
- Delete recipe data records

Creating new recipe data records

To create a new recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to create a new recipe data record.
2. Select the "New" command from the shortcut menu for the recipe list.
   A new data record with the next available number will be created.
   The element list of the new recipe data record opens.
3. Enter values for the elements of the recipe data record.
   The configuration data may already contain default values for the recipe data record.
4. Select the "Save" command from the shortcut menu for the element list.
   The dialog "Save as" opens.
5. Enter the name and number of the recipe data record.
6. Click the "OK" button.

Result

The new recipe data records will be saved to the selected recipe. If the recipe data records already exists, a system event will be output to the screen.

Copying a recipe data record

To copy a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to copy an existing recipe data record.
2. On the HMI device, select the recipe data record of which you want to save a copy.
3. Select the "Save As" command from the shortcut menu for the data record list.
   The dialog "Save as" opens. The recipe data record is automatically given the next free recipe data record number.
4. Under name, enter the name of the record.
5. Click the "OK" button.

Result

The recipe data record is stored under the new name.

Modify recipe data record

To change a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to edit an existing recipe data record.
2. Select the recipe data record that you want to edit on the HMI device.
3. Select the recipe data record.
   The element list of the recipe data record is displayed.
4. Replace the old values with new ones.
5. Select the "Save" command from the shortcut menu for the element list.

Result

The modified values are applied to the recipe data record.

Deleting a recipe data record

To delete a recipe data record, proceed as follows:

1. Select the recipe on the HMI device from which you want to delete an existing recipe data record.
2. Select the recipe data record that you want to delete on the HMI device.
3. Select the "Delete" command from the shortcut menu for the data record list.
4. Confirm this security prompt to delete the data record.

Result

The recipe data record is deleted.

Read recipe data record from PLC

Introduction

In Runtime, you can change values directly in the plant that are also stored in recipes in the HMI device. This applies if a valve was opened further directly in the plant than was specified in the recipe. The values of the recipe data records saved in the HMI device possibly no longer match the values in the PLC.

You can read the values of the recipe tags from the PLC and write them to a recipe data record.

The read values are written to the recipe data record that is currently displayed on the HMI device.

Procedure

To read a recipe data record from the PLC, proceed as follows:

1. Open the recipe on the HMI device.
   The data record list opens.
2. Select the element list of the recipe data record to which you want to apply the values from the PLC.
3. Select the "From PLC" command from the shortcut menu for the element list.
   The values are read from the PLC and displayed in the current recipe data record.
4. If you want to save the values, select the "Save" or "Save As" command.

Result

The values are read from the PLC, visualized on the HMI device and saved to the recipe data record.

Note

Basic Panels

With Basic Panels, the "From PLC" menu command can also be configured for the data record list. In this case, you can also select the "From PLC" menu command in the data record list.

Transferring a recipe data record to the PLC

Introduction

For the values of a data record that was changed in the recipe view to take effect, you must transfer the values to the PLC.

The values displayed in the recipe view are always transferred to the PLC.

Procedure

To transfer a recipe data record to the PLC, proceed as follows:

1. Open the recipe you want to use.
   The data record list opens.
2. Select the element list of the recipe data record whose values you want to transfer to the PLC.
3. Select the "To PLC" command from the shortcut menu for the element list.

Result

The values of the recipe data record are transferred to the PLC.

Note

Basic Panels

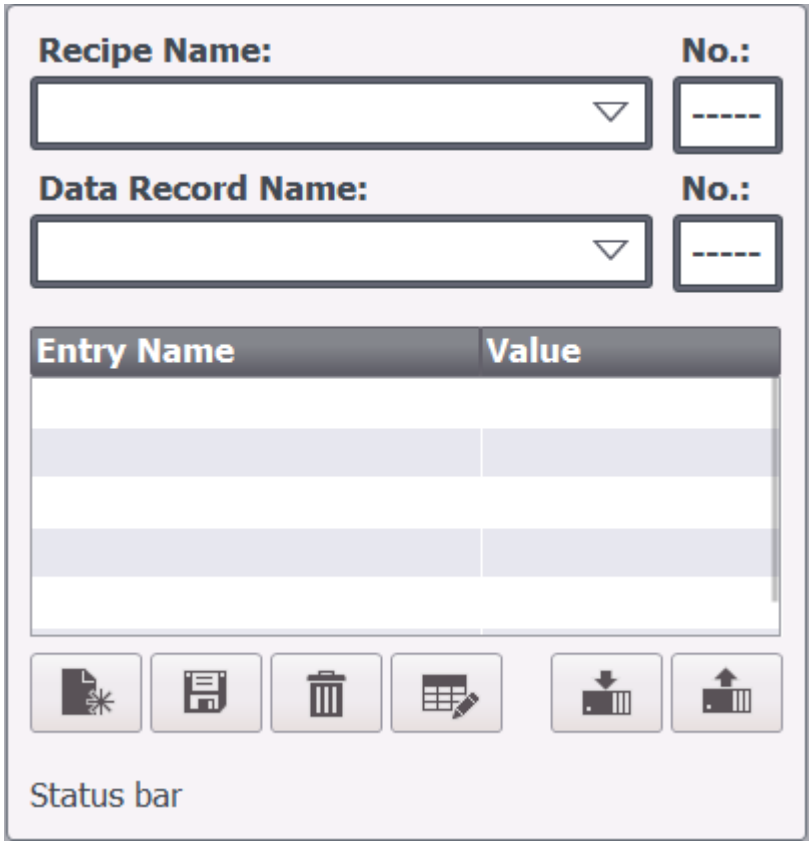
With Basic Panels, the "From PLC" menu command can also be configured for the data record list. In this case, you can also select the "From PLC" menu command in the data record list.

12.5.4.2 Advanced recipe view

Description of the advanced recipe view

Application

The recipe view is used to display, edit and manage data records.



Operation

Depending on the configuration you can:

- Create, change, copy or delete recipe data records
- Synchronizing recipe data records with the associated recipe tags
- Read recipe data records from or transfer to the PLC

Operating elements

The following operating elements can be configured in the recipe view:

Operating element	Key combination	Function
		The configured tooltip is displayed.
	<Ctrl+Space Bar>	Creates a new recipe data record. If a start value is configured, it is shown in the input field.
	<Ctrl+Enter>	Saves the displayed values of the recipe data record. The storage location is predefined by the project.
	<Ctrl+*>	The recipe data record is saved under a different name regardless of the recipe view. A dialog box opens where you can enter the name.
	<Ctrl+Del>	The displayed recipe data record is deleted.
	<Ctrl+=>	The system always updates the current value of the recipe view with the up-to-date recipe tag value. When the value shown in the recipe view is more recent than the current recipe tag value, the system writes this value to the recipe tag. "Synchronize recipe view and recipe tags" must be activated in the recipe properties before this function can be used.
	<Ctrl+Down>	The values of the set recipe data record displayed in the recipe view are transferred to the PLC.
	<Ctrl+Up>	The recipe values from the PLC are displayed in the recipe view.

Using the advanced recipe view

Introduction

You can control the recipe view using both mouse or touchpad or with the keyboard.

Controlling the recipe view with mouse or touchpad

To control the recipe view with mouse or touchpad, proceed as follows:

1. Select the recipe you want to use.
   The records for the recipe are displayed.
2. Click on the data record you wish to edit.
3. Click the button whose function you want to run.

Controlling the recipe view with the keyboard

To control the recipe view with the keyboard, proceed as follows:

1. Press the <Tab> key until the cursor reaches the field for selecting the recipe.
2. Press <Enter>.
   The drop-down list box for the recipes opens.
3. Select a recipe. You navigate between the next or previous entry in the list by using the cursor keys <Left>, <Right>, <Up> and <Down>.
4. Select a data record.
5. Press the <Tab> key until the operator control object you wish to use is selected.

Alternatively you can control the recipe view using certain key combinations.

Key shortcuts for the advanced recipe view

The following key shortcuts are activated for the advanced recipe view in Runtime if "Activate keyboard operation" is enabled in the ES.

Keys shortcut	Effect	Menu command	Button
<Ctrl+Space>	Generates a new recipe data record. Any configured start value is displayed in the input field.	Add data record	
<Ctrl+Del>	Deletes the recipe data record displayed.	Delete data record"	
<Ctrl+Enter>	Saves the modified record with its current name.	Save	
<Ctrl+*>	Saves the modified record with a new name.	Save as	
<Ctrl+=>	Compares the values of the selected data record with the values on the PLC. Any value in the recipe view which is more recent compared to the current recipe tag value is written to the recipe tag. This function is only available if enabled in the ES.	Synchronizing recipe view and recipe tags	
<Ctrl+Down>	Sends the current value to the PLC.	Write to PLC	
<Ctrl+Up>	Reads the current value from the PLC.	Read from PLC	

Managing recipe data records

Administration of recipe data records

You have the following options for managing recipe data records, according to the configuration:

- Create new recipe data records
- Copy recipe data records

- Edit recipe data records
- Delete recipe data records

Creating new recipe data records

To create a new recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to create a new recipe data record.
2. Click the button or press the <Ctrl + Spacebar> keys.
   A new recipe data record with the next available number is created.
   If you change the new data record number to an existing data record number, the existing data record is overwritten.
3. Enter values for the elements of the data record.
   The elements of the recipe data record can be assigned default values depending on the configuration.
4. Click the button or press the <Ctrl + *> keys.
   The dialog "Save as" opens.
5. Enter a name for the data record.
6. Click on "OK" to confirm your input.
   The data record is saved under the new name.
   If the recipe data record already exists, a dialog is opened. In this dialog, specify whether the existing data record is to be overwritten.

Result

The new recipe data records will be saved to the selected recipe. If the recipe data records already exists, a system alarm will be output to the screen.

Copying a recipe data record

To copy a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to copy an existing recipe data record.
2. Select the recipe data record that you want to copy on the HMI device.
3. Click the button in the recipe view or press the <Ctrl + *> keys.
   The dialog "Save as" opens.
4. Enter a name for the data record.
5. Click on "OK" to confirm your input.

Result

The recipe data record is stored under the new name.

Modify recipe data record

To change a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to edit an existing recipe data record.
2. Select the recipe data record that you want to edit on the HMI device.
3. Replace the old values with new ones.
4. Click the button in the recipe view or press the <Ctrl + Enter> keys.

Result

The modified values are applied to the recipe data record.

Delete recipe data record

To delete a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to delete an existing recipe data record.
2. Select the recipe data record that you want to delete on the HMI device.
3. Click the button in the recipe view or press the <Ctrl + Del> keys.

Result

The recipe data record is deleted.

Synchronizing recipe data record

Introduction

Differences between the following values may occur during runtime:

- The values displayed in the recipe view
- The actual values of the recipe tags

Depending on the configuration, the values displayed in the recipe view are synchronized with the recipe tags. Synchronization encompasses all tags of a recipe data record.

Note

Changed tag name

The tag and the value of the recipe data record cannot be associated if you have renamed the tag you want to synchronize. The tags in question are not synchronized.

Note

Recipe tags can only be synchronized in the advanced recipe view.

Requirement

- A recipe data record is displayed in the recipe view.
- The values of recipe tags were modified by teaching, for example.

Procedure

To synchronize a recipe data record, proceed as follows:

1. Click the button in the recipe view or press the <Ctrl + => keys.

Result

The system always updates the current value of the recipe view with the up-to-date recipe tag value.

When the value shown in the recipe view is more recent than the current recipe tag value, the system writes this value to the recipe tag.

Read recipe data record from PLC

Introduction

In Runtime, you can change values directly in the plant which are also stored in recipes in the HMI device. This applies if a valve was opened further directly in the plant than was specified in the recipe. The values of the recipe data records saved in the HMI device possibly no longer match the values in the PLC.

You can read the values of the recipe tags from the PLC and write them to a recipe data record.

The read values are written to the recipe data record that is currently displayed on the HMI device.

Procedure

To read a recipe data record from the PLC, proceed as follows:

1. Select the recipe on the HMI device.
2. On the HMI device, select the recipe data record of which you want to fetch the values from the PLC.
3. Click the button in the recipe view or press the <Ctrl + Up> keys.

Result

The values are read from the PLC and displayed on the HMI device.

Transferring recipe data records to the PLC

Introduction

For the values of a data record that was changed in the recipe view to take effect, you must transfer the values to the PLC.

The values displayed in the recipe view are always transferred to the PLC.

Procedure

To transfer a recipe data record to the PLC, proceed as follows:

1. Select the recipe on the HMI device.
2. On the HMI device, select the recipe data record of which you want to transfer the values to PLC.
3. Click the button in the recipe view, or press the <CTRL+DOWN> shortcut key.

Result

The values of the recipe data record are transferred to the PLC.

12.5.4.3 Exporting and importing recipe data records

Introduction

Depending on configuration and the HMI device, either export recipe data records to a CSV file, e.g. for editing in MS Excel, or import these from a CSV file. The extent to which you can influence these processes is determined by the project configuration.

Note

Restrictions for Import / Export

It is not possible to export or import the recipes for the following HMI devices:

- Basic Panels
- OP 77A
- OP 177A
- TP 177A (Portrait)

Complete recipe data, but not single recipe data records, can be exported and imported using ProSave in CSV format and transferred to the HMI device. Thereby, Runtime is interrupted.

A list separator is used to separate the data records during importing and exporting.

Note

The list separator used as the default depends on the setting for formats and numbers in the operating system. Select "Start > Settings > Control Panel > Regional and Language Options". If you want to import or export recipe data records, do not use this list separator in the display name of the recipe data records.

The following fields can be configured on the user interface, for example, in order to use the export/import function:

- Selection field for the recipe
- Selection field for the recipe data record
- Operating element with the "ExportDataRecords" functionality
- Operating element with the "ImportDataRecords" functionality

Export recipe data record

To export a recipe data record, proceed as follows:

1. Select the relevant recipe and recipe data record from the selection boxes.
2. Click the control element with the "ExportDataRecords" functionality.

Result

The recipe data record are exported to a CSV file.

Note

New data records created in runtime can be exported to an external file.

Importing recipe data records

To import a recipe data record, proceed as follows:

1. Select the relevant recipe and recipe data record from the selection boxes.
2. Click the control element with the "ImportDataRecords" functionality.

Result

The recipe data record are imported.

12.5.4.4 Reactions to modifications of the recipe structure

Introduction

Different recipe structures can occur in the following situations:

- In the event of changes during commissioning
- When work is carried out on the machine by the machine manufacturer (retrofit)
- When CSV files are imported, the structure of the CSV file can differ from the recipe structure.

Nevertheless, you can still use any recipe data records already created.

Notice

When a tag is renamed, the assignment is lost.

Effects

Handle any structural differences as follows:

- If the old recipe data record or the CSV file contains additional values, these values will be discarded.
- If the old recipe data record or the CSV file contains values of the wrong data type, the configured default value will be used in the recipe data record.
  Example: The recipe data record contains values that show the tank contents and were input as floating point numbers. However, the corresponding recipe tag expects an integer value. In this case, the system discards the transferred value and the configured default value is used.
- If the old recipe data record or the CSV file contains too few values, the configured default value will also be used in the recipe data record.

12.5.5 Examples

12.5.5.1 Example of creating a recipe

Task

In this example, you create three recipes for a fruit juice mixing machine. The fruit juice mixing machine produces drinks with "orange", "apple" and "tropical" flavors. You create a recipe for each flavor.

Each recipe contains a recipe data record for the following mixing ratios:

- Beverage
- Nectar
- Juice

Settings

The settings relate to an HMI device which is connected to a SIMATIC S7-300 or SIMATIC S7-400.

In this example, you will need the following tags, recipes, recipe entries and recipe data records:

Tags:

Name	PLC connection	Address	Type
Liter water	Yes	DB 120, DBW 0	Integer
Liter concentrate	Yes	DB 120, DBW 4	Integer
Kilo sugar	Yes	DB 120, DBW 8	Integer
Gram flavoring	Yes	DB 120, DBW 12	Integer

Recipes:

- Orange
- Apple
- Tropical

Recipe entries:

Recipe element	Associated tag
Liter water	Liter water
Liter concentrate	Liter concentrate
Kilo sugar	Kilo sugar
Gram flavoring	Gram flavoring

Recipe data records for drink, nectar and juice:

Data record name	Liter water	Liter concentrate	Kilo sugar	Gram flavoring
Beverage	30	70	45	600
Nectar	50	50	10	300
Juice	5	95	3	100

Procedure

To create a recipe, proceed as follows:

1. Create the following tags with the settings specified above: "LiterWater", "LiterConcentrate", "KiloSugar" and "GramFlavoring".
2. Create the "Orange", "Apple" and "Tropical" recipes with the settings indicated above. Create the recipe entries in each recipe.

Elements | Data records

Name	Display name	Tag	Data type	Data length	Default value
Rezepturelement_1	Rezepturelement_1	RecipeDSTag	Int	2	0
Rezepturelement_2	Rezepturelement_1	RecipeDSTag1	Int	2	0
Rezepturelement_3	Rezepturelement_1	RecipeDSTag2	Int	2	0
Rezepturelement_4	Rezepturelement_1	RecipeDSTag3	Int	2	0

3. Not for Basic Panels: Configure each recipe so that you can synchronize the recipe data records between the recipe screen and recipe view. The values of the recipe tags should not be transferred automatically to the PLC.
   You will have to make the following settings in the Properties dialog for the recipe concerned: Under "Properties > Options":
   - Activate the "Synchronize recipe view and recipe tags" option.
   - Activate the "Manual transfer of individual modified values (teach-in mode)" option.
4. Create the data records indicated above in each recipe. Enter the values indicated above in each of the data records.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
Nectar	Nectar	2	50	50	10	300	
Juice	Juice	3	5	95	3	100	
<Add new>							

Result

The "Orange", "Apple" and "Tropical" recipes have been created.

See also

Advanced recipe view (as of V13) (Page 4447)

Configuration options of the advanced recipe view (V13 or higher) (Page 4448)

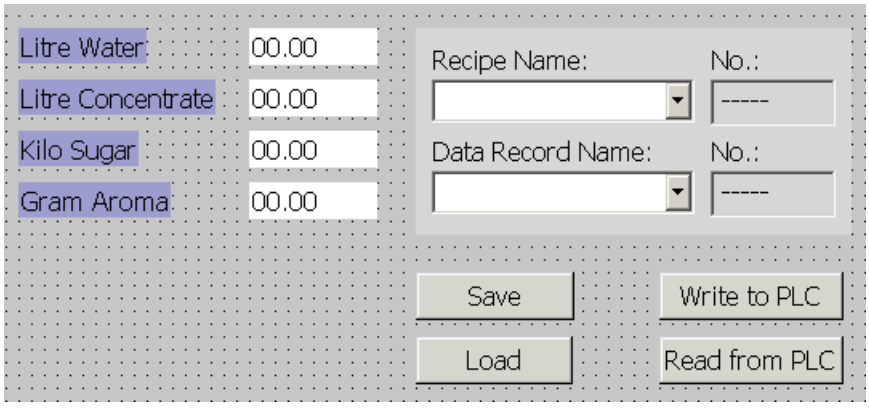
12.5.5.2 Example of configuring a recipe screen

Task

In this example, you create a recipe screen for the visualization of values of the fruit juice mixing machine. You use a recipe view to select the recipes and their associated recipe data records.

You should be able to use the following functions with the buttons:

- "Load" Button
  The selected recipe data record is loaded from the recipe memory on the HMI device and displayed in the recipe screen.
- "Save" Button
  The displayed recipe tags are saved in the recipe memory of the HMI device.
- "Data to PLC" button
  The displayed tags are transferred to the PLC.
- "Data from PLC" button
  The current recipe data record in the PLC is transferred to the recipe variables and displayed on the HMI device.



Requirement

- The "Creating a recipe" sample application has been carried out.
- You have created and opened the "Fruit juice mixing plant" screen.

Settings

In this example, you need the following tags and buttons with the indicated settings:

Tags:

Name	PLC connection	Type
RecipeNumber	No	Integer
Data record number	No	Integer

Buttons:

Labeling	Configured event	System function
Load	Click	LoadDataRecord
Save	Click	SaveDataRecord
Data to PLC	Click	SetDataRecordTagsToPLC
Data from PLC	Click	GetDataRecordTagsFromPLC

Procedure

To configure a recipe screen, proceed as follows:

1. Drag-and-drop the "Liter water", "Liter concentrate", "Kilo sugar" and "Gram flavoring" tags from the object view to the "Fruit juice mixing machine" process screen.
   Four IO fields are created and linked by the specified tags.
2. Add a recipe view containing selection fields for the recipe name and data record name only.
   Make the following settings in the Inspector window for the recipe view:
   - Select the "Advanced view" display type under "General".
   - Deselect "Edit mode" under "Properties > General" and "Display table" under "Properties > Table".
   - Connect the "Recipe tag" field to the "RecipeNumber" tag.
   - Connect the "Tag" field to the "DataRecordNumber" tag.
   - Disable all the buttons under "Properties > Buttons".
3. Assign the above settings to four buttons. Transfer each "Recipe number" and "Data record number" tag as a parameter for the recipe number and recipe data record number.

Result

You can select the recipe and the associated recipe data record from the recipe view and modify the recipe values at runtime. You can load, save and transfer the recipe data records using the buttons.

12.5.5.3 Scenario for Entering Recipe Data Records in Runtime

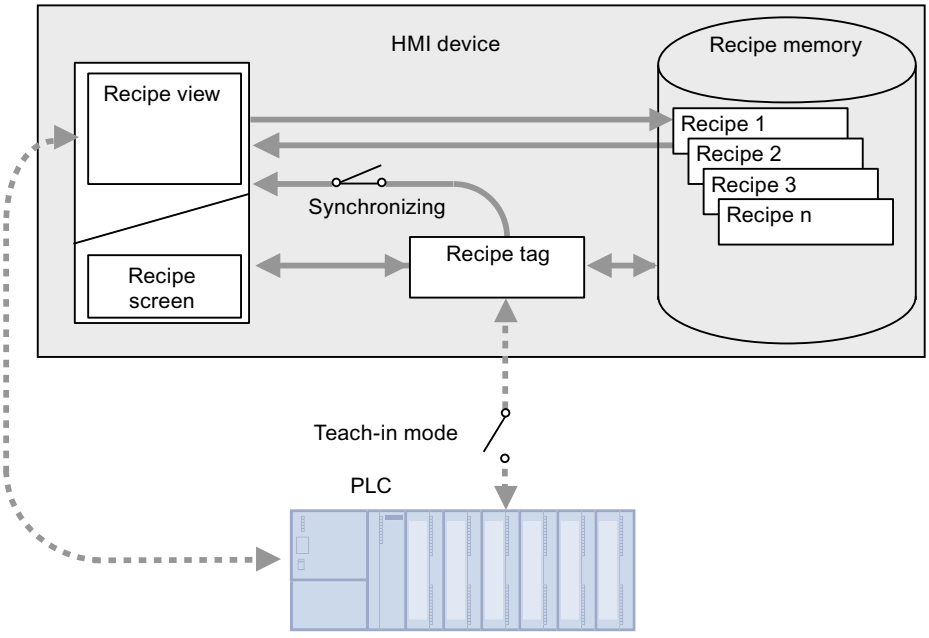
Objective

You want to enter production data on the HMI device without disturbing the process that is currently underway. Therefore, the production data should not be transferred to the PLC.

Requirement

- The recipe has been created.
- The recipe has the following settings:
  - "Synchronizing recipe view and recipe tags" is activated or deactivated.
  - If "Synchronize recipe view and recipe tags" is activated, "Manual transfer of individual modified values (teach-in mode)" has to be activated.
    This will prevent the recipe tags being transferred automatically between the HMI device and PLC.
- A recipe screen or a screen with recipe view is available.
- There is an operating element for saving the recipe data records.

Sequence



1. Enter the production data in the recipe view or recipe screen.
2. Save the modified recipe data record.

Transfer the recipe data to the PLC

The configuration may provide operating elements for transferring recipe data to the PLC.

12.5.5.4 Scenario for a manual production sequence

Objective

A reading device connected to the PLC reads a bar code on the work piece to be processed. The recipe data record names correspond to the respective bar code names. This will enable the PLC to load the necessary recipe data record from the storage medium of the HMI device. The recipe data record is displayed for inspection on screen.

You want to be able to correct the transferred production data online, if necessary.

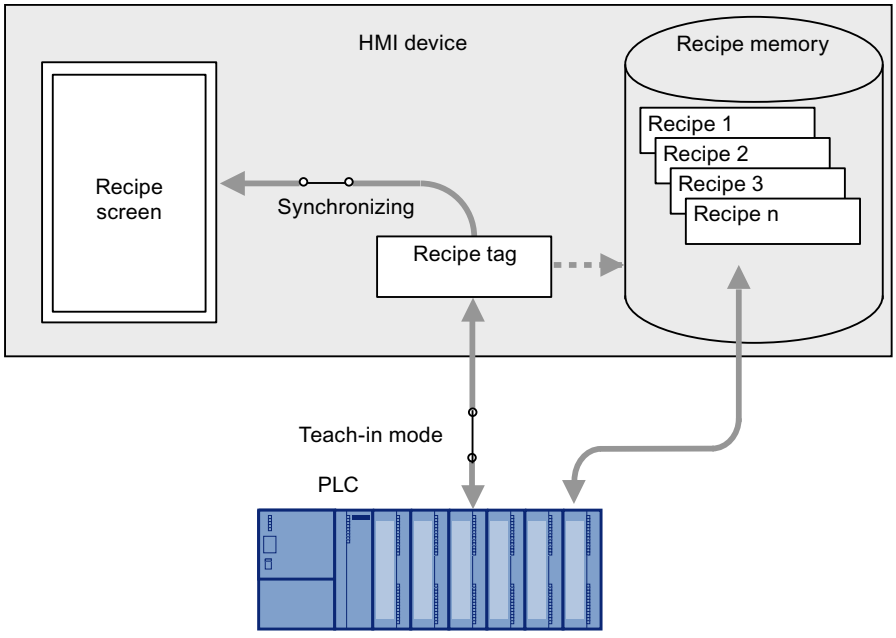
Requirement

- You have created the recipe.
- The recipe has the following settings:
  - "Synchronizing recipe view and recipe tags" is activated.
  - "Manual transfer of individual modified values (teach-in mode)" is deactivated.

  Note

  The changes are immediately transferred to the PLC
- There is a recipe screen available.
  There may also be an operating element for saving the recipe data records in the recipe screen.

Sequence



Behavior when the recipe view is used

If the recipe view is used, it is not possible to transfer changes immediately. You must use the operating element to transfer the recipe data record to the PLC.

12.5.5.5 Scenario for an Automatic Production Sequence

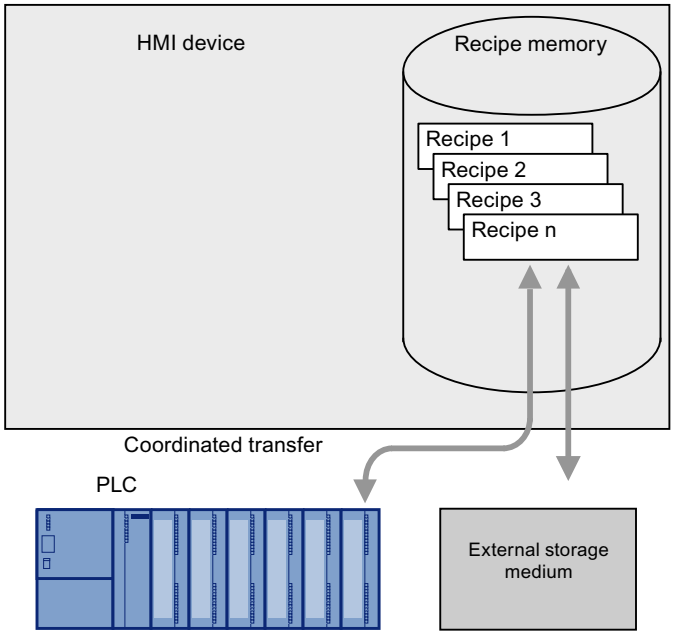
Objective

You want production to be executed automatically. The production data is to be transferred directly to the PLC, either from the recipe memory in the HMI device or from an external storage medium. The screen display is not necessary.

Requirement

- You have created the recipe.
- The recipe has the following settings:
  - "Coordinated transfer of data records" is activated.
    The production data is transferred to the PLC, so a coordinated transfer with the PLC in necessary to prevent the data from being accidentally overwritten.

Sequence



Implementation

You can control the flow of data in the following ways:

- The control program controls the automatic transfer via control jobs or via WinCC system functions, if necessary.
  The sequence is controlled via the status information in the mailbox and via return values from the functions used.
- One or more scripts control the automatic transfer via WinCC system functions.
  The sequence can be checked using the values returned by the functions used.

You can implement the automatic production sequence with available system functions:

- "ImportDataRecords"
  This function loads data records from a *.CSV file into the recipe memory of the HMI device.
- "SetDataRecordToPLC"
  This function transfers a data record from the HMI device's recipe memory to the PLC.

12.6 Working with reports

12.6.1 Basics

12.6.1.1 Reports

Introduction

Reports are used to record process data and processed production cycles. You have the opportunity, for example, to create regular shift reports, output batch data, or record the production process for quality control (QC).

Creating reports

A report is created and edited in the "Reports" editor. In this editor, you configure the following report items:

- Formal appearance
  Specify the formal layout of the report in the Inspector window. In this window, for example, you specify the page format, page margins, title page, back page, headers, or footers for the report.
- Contents
  In the work area, specify the content of the report, for example, the alarms of a shift. To this purpose you insert the corresponding objects into the report.

The modular structure lets you configure reports that suit all of your applications.

Note

The "Reports" editor is not available at HMI devices that do not support reporting.

Report output

In runtime, the configured reports are printed on the default printer of the HMI device.

Reports are output to the printer in graphic mode. The use of a serial printer is not recommended because of the accumulated data volume.

In runtime, report output is event or time-controlled.

- Time-controlled: Automatic print at specific dates, times or intervals.
- Event-driven: Printing is initiated by specific events, e.g. click on a button, or when a limit is exceeded.

For proper output, the printer must support the paper format and page layout of the report.

Note

The value of a tag in the report is read and output at the moment of printing. A substantial period of time may elapse between printing out the first and the last page of a report consisting of several pages. This may lead to the same tag being output with a different value on the last page than on the first page.

See also

Principles for preparation of reports (Page 4496)

Printing reports (Page 4501)

Structure of reports (Page 4494)

Inserting an object (Page 4502)

Printing reports (Page 4516)

Audit report (Page 4516)

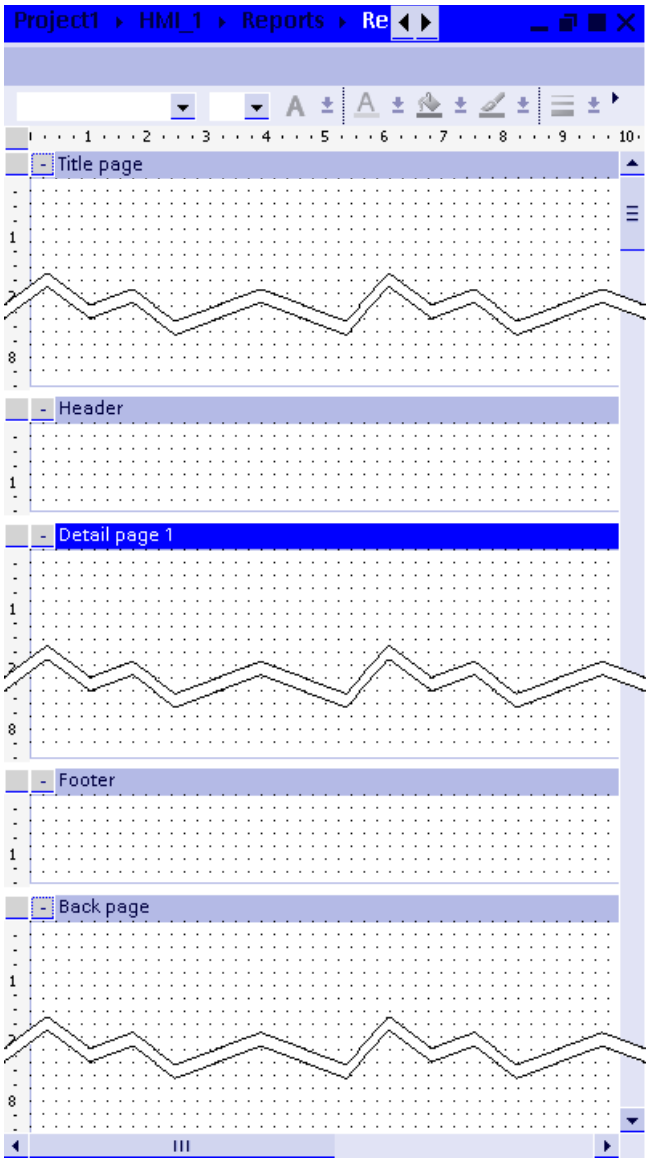
12.6.1.2 Structure of reports

Introduction

A report in WinCC consists of several sections that can be enabled or disabled, as required.

Sections of a report

The following figure shows an example of the different sections of a report in the "Reports" editor.



Title page and back page

The title page contains important information about the report content. The back page is used, for example, as Impressum, on which you provide contact information of shift managers or service technicians. The title page and back page are output separately on a single page. Each of them consist of exactly one page and page breaks are not used.

Detail page

Configure the output of runtime data such as recipe or alarm reports on the detail pages of the report.

Use the shortcut menu on the detail page to insert additional detail pages or change their order.

Header and footer

The header and footer are output on each detail page of the report. You typically insert the page numbers or the date in the header or footer.

See also

Reports (Page 4493)

Principles for preparation of reports (Page 4496)

12.6.2 Working with reports

12.6.2.1 Creating reports

Principles for preparation of reports

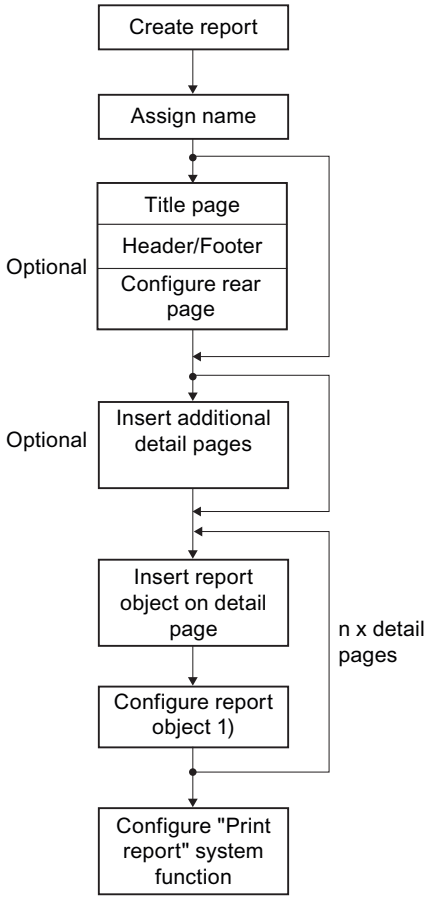
Introduction

A report in WinCC consists of several sections that can be enabled or disabled, as required.

- Title page
- Back page
- Detail pages
- Headers and footers for the detail pages

Procedure

The following figure shows the general procedure for creating a report:



1) Settings depend on the report object used.

Tools for the design of reports

The tools that you can use to design reports are available in the "Tools" task card.

- For *graphic design* insert "basic objects", "elements" and "graphics" in the different sections of the report. For example, for a Logo in the header, separation lines and a page number in the footer.
- Using the "Controls" in the detail pages, configure the *output of runtime data*.

Note

The "Screens" editor provides many objects for designing a report – however, with restricted functionality. Data input properties are not available. The I/O fields in the reports, for example, serve only to output data.

Position and size of objects

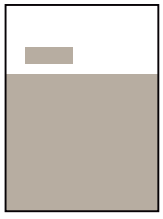
Insert the objects at the position where you want them to be output in the report. The objects are output in their configured size and with the following special features:

On the detail pages, configure "recipe reports" or "alarm reports" that serve to output runtime data in tabular format.

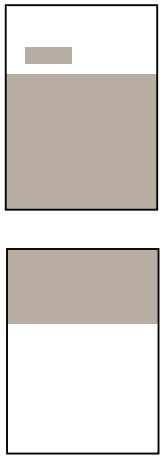
- The *width* of the table is set up automatically to fit the width of the detail page. You cannot modify the width.
- WinCC automatically wraps the *height* of the table to fit the data content for output in the report. The table can be continued on the next page.
  In the detail page, WinCC automatically extends tables of dynamic length to the bottom margin.

The following figure shows an example of a detail page in the report and its output in Runtime:

Detail page in the report



Report output in Runtime



Additional detail pages

Use the shortcut menu on the detail page to insert additional detail pages or change their order.

See also

Objects in reports (Page 4516)

Structure of reports (Page 4494)

Administration of detail pages (Page 4500)

Create a report

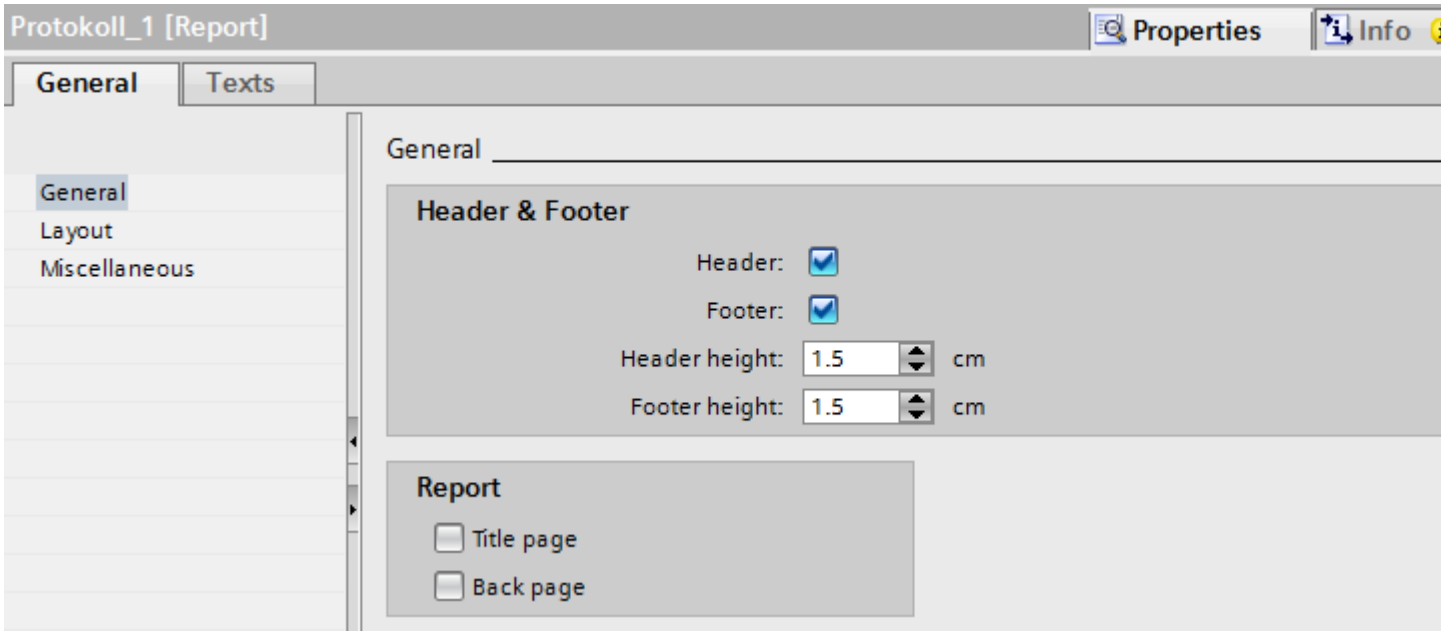
Requirement

A project with an HMI device is open.

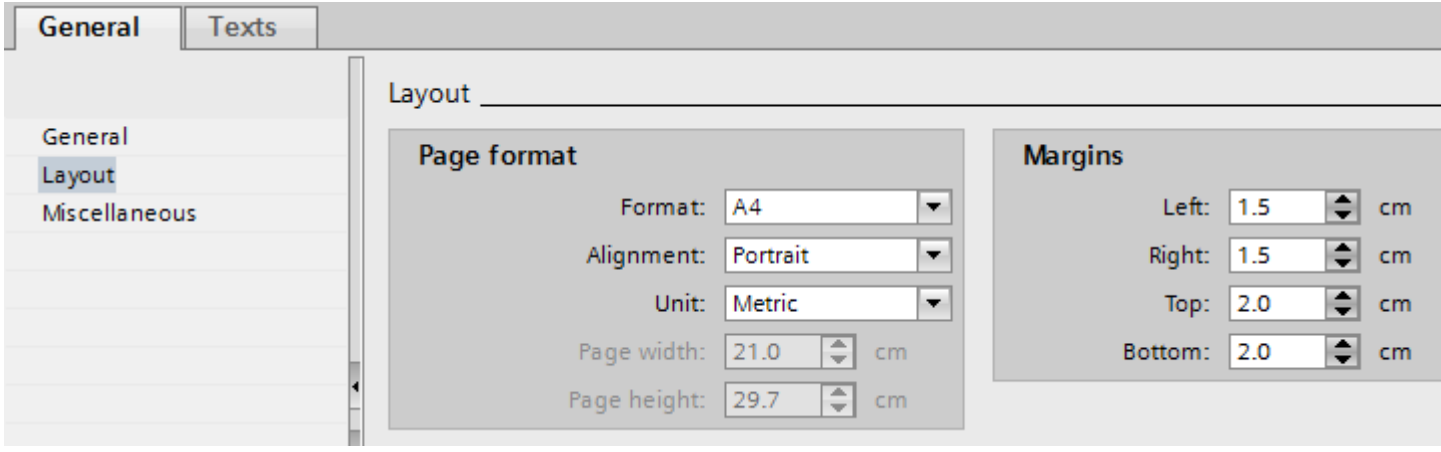
Procedure

To create a report, proceed as follows:

1. Double-click "Add new report" under "Reports" in the project navigation.
   A new blank report is displayed in the "Reports" editor.
2. Select the "Report properties" command in the shortcut menu of the report.
3. In the "Properties > Properties > General" area of the Inspector window, specify whether you want to configure the "Title page", "Back page", "Header" and "Footer" in the report.
   The report sections are updated accordingly.



4. Configure the format, the page layout, and the page margins of the report under "Properties > Properties > Layout."



5. Enter a meaningful report name under "Properties > Properties > Miscellaneous."

6. Design the report sections as required.
   Drag and drop the necessary basic objects, elements, graphic images and controls from the "Tools" task card to the required position.
   Alternatively, you can also copy or move objects already configured from a screen to the report.
7. Configure the objects in the Inspector window:

See also

Objects in reports (Page 4516)

Administration of detail pages

Adding a detail page

1. Open the shortcut menu of a detail page.
2. Select the command "Pages > Insert page before" or "Pages > Insert page after."
   Depending on the selected command, the new detail page is inserted either before or after the existing detail page.

Deleting a detail page

1. Open the shortcut menu of the detail page that you want to remove.
2. Select the command "Pages > Delete detail page".
   The detail page is deleted.

Sorting detail pages

1. Open the shortcut menu of the detail page that you want to move.
2. Select the "Pages > Move one page up" or "Pages > Move one page down" command.
   Depending on the selected command, the detail page is moved either upward or downward.

Showing and hiding sections

1. To show or hide a specific section, click on the plus or minus sign in the title bar of the section in the working area.
2. If you want to show or hide all sections of a report, select the command "Show all pages" or "Hide all pages" in the report shortcut menu.

See also

Structure of reports (Page 4494)

Principles for preparation of reports (Page 4496)

12.6.2.2 Printing reports

Introduction

In Runtime, report output is event-driven or time-driven.

Event-driven output

The report is output after an event was generated.

Examples:

- A tag value changes or exceeds a limit.
- Alarm is incoming, outgoing or acknowledged
- Action by the operator, for example, clicking a button

Time-driven output

The report is output automatically.

Examples:

- Once at a specified date, e.g. on December 31, 2010
- At intervals, e.g. daily at 8 am, or on Mondays at 6 pm

Configuration steps

Configuring event-driven output:

- An event is configured on a button or tag with the "PrintReport" system function.

Configuring time-driven output:

- In the scheduled tasks, configure a "Print job" task and assign it to the desired report. In the task properties, specify the time and frequency of report output.

Output in Runtime

On the control panel of the HMI device, the report is output to the printer that is specified as the default printer.

The default printer enabled for a HMI device can be found in the "Printer list". For further information about the "Printer list", refer to the Internet page of Siemens Customer Support and the Article ID "11376409".

See also

Planning jobs (Page 5985)

Printing reports (Page 4516)

12.6.2.3 Working with objects

Inserting an object

Introduction

In the "Screens" or "Reports" editor, insert the objects to the "Toolbox" task card. Use the mouse to drag the objects into the work area. You either keep the objects in their original size, or scale them up or down when you paste them.

In addition, you can copy or move objects via the clipboard from one editor to another, for example to transfer a screen object to a report. Alternatively, you can also use the mouse instead of the clipboard for copying and moving:

- Copying: <Ctrl + Drag&Drop>
- Moving: Drag&drop

Note

Basic Panels

The "Reports" editor is not available for Basic Panels.

Requirement

The "Tools" task card is open.

Inserting objects in their standard size

1. In the "Toolbox" task card, select the desired graphic object or the desired graphic in the WinCC graphics folder.
   When you move the cursor across the work area, it turns into a crosshair with an appended object icon.
2. Click the location in the work area where you want to insert the object or graphic.
   The object is inserted with its standard size at the desired position in the work area.

Alternatively, double-click the object in the "Toolbox" task card.

Copying an object

1. Select the desired object.
2. Select "Copy" in the shortcut menu.
3. Click the desired location and select "Paste" in the shortcut menu.

WinCC inserts a copy of the object at the desired location. You can only change the properties that are appropriate for the relevant context.

Example: In the "Screens" editor, you can set for I/O fields and the mode for input and output. In the "Reports" editor, the mode is set to "Output".

Original and copy are not interconnected and are configured independently from one another.

Inserting lines

1. Select the desired graphic object in the "Tools" task card.
2. Click on a location in the work area. A line in the standard size is inserted.

Inserting a polygon or polyline

1. Select the desired object "Polyline" or "Polygon" in the "Tools" task card.
2. Click on a location in the work area. This fixes the starting point of the object.
3. Click another location in the work area. A corner point is set.
4. For every additional corner point, click on the corresponding location in the work area.
5. Double-click on a location in the work area. The last corner point is set.
   This fixes all the points of the polygon or polyline.

Note

Basic Panels

The "Polyline" and "Polygon" objects are not available for Basic Panels.

Note

If you want to insert several objects of the same type, use the "Stamp" function. This avoids having to reselect the object in the "Tools" task card every time before inserting it. To do so, select the icon in the toolbar of the "Tools" task card.

See also

Deleting an object (Page 4503)

Inserting multiple objects of the same type (stamping tool) (Page 4504)

Positioning objects (Page 4505)

Resizing objects (Page 4508)

Moving an object forwards or backwards (Page 4507)

Flipping objects (Page 4515)

Reports (Page 4493)

Deleting an object

Introduction

You can delete objects individually or with a multiple selection.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to delete.
   To delete multiple objects, keep the <Shift> key pressed and select the objects to be deleted one after the other. Alternatively, drag and maximize an area around the desired objects with the mouse.
2. Select "Delete" from the shortcut menu.

Result

The selected objects are deleted.

Inserting multiple objects of the same type (stamping tool)

Introduction

WinCC offers the possibility to "stamp" several objects of the same type directly one after the other, i.e. paste without having to reselect the object every time. In addition you have the possibility of multiplying an object that has already been inserted.

Requirement

The "Tools" task card is open.

Inserting several objects of one type

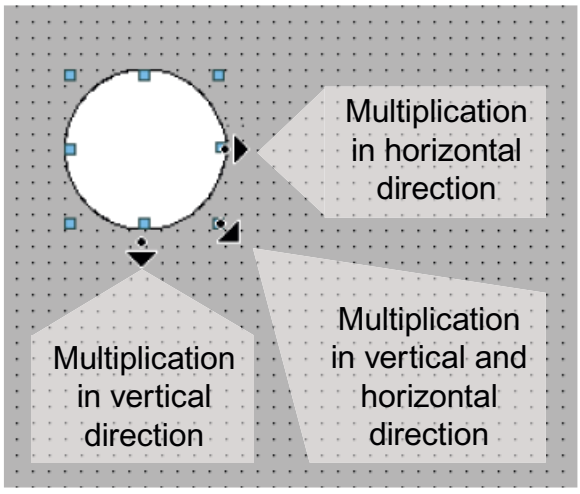
1. Select the object that you want to insert in the "Tools" task card.
2. Click the icon in the toolbar of the "Tools" task card.
   The "Stamp" function is activated.
3. To insert the object with its standard size, click the relevant insertion position in the work area.
   To insert the object with another size, position the mouse pointer at the desired location in the work area. Press the left mouse button and drag the object to the required size.
   The object is inserted in the work area as soon as you release the mouse button.
4. Repeat step 3 to insert further objects of the same type.
5. Click the icon again.
   The "Stamp" function is deactivated.

Note

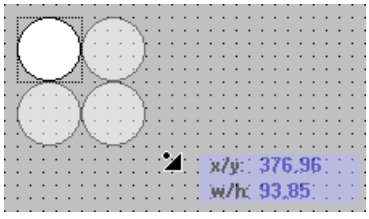
You can copy existing objects using the drag-and-drop +<CTRL> function. The existing object is not moved in this case. You paste a copy of this object into the new position instead.

Inserting and multiplying an object

1. Insert the desired object from the "Tools" task card.
2. Press the <Ctrl> key and position the cursor on one of the handles displayed in the figure shown below.



3. Drag the handles to the right and/or down while keeping the left mouse button pressed.
4. The object is multiplied depending on available space if you keep moving the cursor.



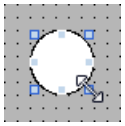
Result

You have pasted and stamped an object in a screen.

Positioning objects

Introduction

When you select an object, it is enclosed by a rectangle with resizing handles. This rectangle is the rectangle which surrounds the object. The position of an object is defined by the coordinates of the top left corner of the rectangle surrounding the object.



Note

If the position is outside the work area the object is not displayed in Runtime.

Position and align

You have the possibility of having a grid displayed in the work area. You have three options for easier positioning of objects:

- "Snap to grid" When you resposition objects, they are automatically snapped and pasted to the grid. If you hold down the <Alt> key, the object is no longer snapped to the grid.
- "Snap to objects" When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.
- "None": You position the objects in any position.

You activate and deactivate the grid and options as follows:

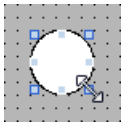
- In the "Options > Settings > Visualization > Screens" menu
- In the "Layout > Grid" task card

Requirement

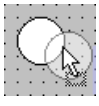
You have opened the work area containing at least one object.

Procedure

1. Select the object you want to move.
   The selected object is framed by a rectangle with resizing handles.



2. Left-click the object and keep the mouse button pressed.
3. Move the mouse pointer onto the new position.
   The contour of the object moves with the mouse and displays the new position for the object.



   The object initially remains at its original position.
4. Now release the mouse button.
   The object is moved into the position indicated by the contour of the selection rectangle.

Alternative procedure

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the X and Y values for the position under "Position & Size".

Result

The object appears at its new position.

See also

Rotating objects (Page 4513)

Moving an object forwards or backwards

Introduction

You can use the "Order" functions in the shortcut menu of a selected object or in the toolbar to move a selected object in front of or behind other objects within an object layer.

Note

ActiveX controls are always positioned in front of an object layer (.NET property).

Requirement

You have opened a screen which contains a layer with multiple objects.

Procedure

1. Select the object you want to move forward or backward.
2. Select the "Sort" command and one of the following commands from the shortcut menu:

Icon	Description
	Moves the selected object before all the other objects of the same layer
	Moves the selected object to the lowest position in the same layer
	Moves the selected object up by one position
	Moves the selected object down by one position

Alternative procedure

1. Open the "Layers" palette of the "Layout" task card.
2. Navigate to the required object.

3. Hold down the mouse button, and drag the object in the tree topology to the required position in the layer.
4. Now release the mouse button.

Result

The object is moved up or down.

See also

Inserting an object (Page 4502)

Resizing objects

Introduction

When you select an object, it is enclosed by a rectangle with handles. You have the following options of resizing an object:

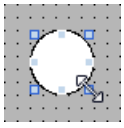
- Drag the handles using the mouse.
- Modify the "Size" property in the Inspector window.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object you want to resize.
   The selection rectangle appears. The following figure shows a selected object:



2. Drag a resizing contact of the rectangle to a new position.
   The size of the object changes.
   - The size of the object is aligned to the grid pattern, provided the "Snap to grid" function is set.
   - Press <ALT> to disable this function while you drag the object.
     In order to scale the object proportionally, keep the <Shift> key pressed while changing the size with the mouse.

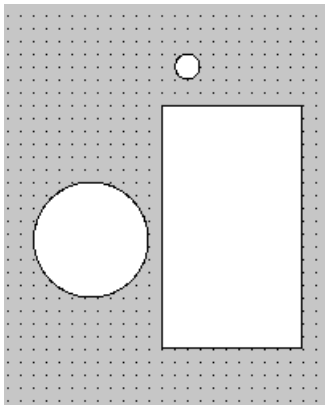
Alternative procedure

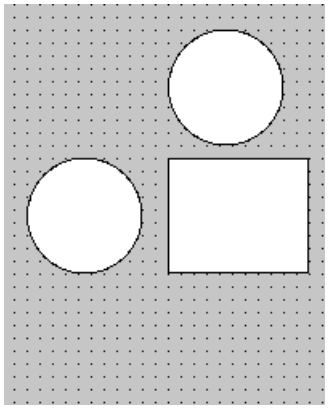
1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the size of the object under "Position & Size".

Harmonizing the object size

1. Select the objects.
2. Now, click one of the following buttons: or or
   The size of the selected objects is matched to each other.

The following screen shows how the selected objects are adapted to the height of the reference object:





Icon	Description
	Aligns the selected objects to the width of the reference object.
	Aligns the selected objects to the height of the reference object.
	Aligns the selected objects to the width and height of the reference object.

Result

The object now appears with its new size.

Selecting multiple objects

Introduction

Select all objects you want to align with each other or to change global properties. This procedure is called "multiple selection."

The Inspector window shows all the properties of the selected objects.

You now have several options of selecting multiple objects:

- Draw a selection frame around the objects.
- Hold down the <Shift> key, and click the required objects.

Selection frame of a multiple selection

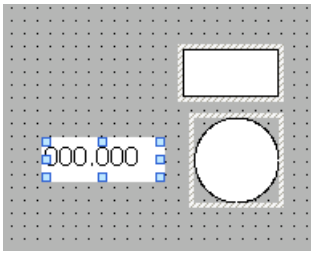
The selection frame surrounds all objects of a multiple selection. The selection frame is comparable with the rectangle that surrounds an individual object.

The selection frame is not visible. When you have made your multiple selection, the following frame is displayed:

- The reference object is indicated by the rectangle around it.
- The other selected objects are indicated by a dashed-line frame.

Specifying a reference object

The reference object is the object upon which the other objects are oriented. The reference object is framed by a rectangle with handles. The following figure shows a reference object with two other selected objects:



You have the following options to specify the reference object:

- Select the objects via multiple selection. The object selected first is then the reference object.
- Draw a selection frame around the objects. The reference object compiled automatically. If you wish to specify a different object within the selection as the reference object, click on the desired object. This action does not cancel your multiple selection.

Requirement

You have opened the work area containing at least two objects.

Selecting multiple objects with a selection frame

1. Position the mouse pointer in the work area close to one of the objects to be selected.
2. Hold down the mouse button, and draw a selection frame around the objects to be selected.

Or:

1. Hold down the <Shift> key.
2. Click the relevant objects, working in succession.
   All the selected objects are identified by frames.
   The object selected first is identified as reference object.

Note

To remove an object from the multiple selection, press <SHIFT>, hold it down and then click the relevant object once again.

Result

Multiple objects are selected. One of those is identified as the reference object. You can now perform the following steps:

- Changing the object properties of all the objects
- Resizing all the objects by the same ratio, by dragging the selection frame to increase or reduce the size
- Moving all the objects in one group
- Aligning the objects to the reference object

Repositioning and resizing multiple objects

Possible modifications

After you have selected multiple objects, you edit them:

- Shift using the mouse
  - To change the absolute position of the marked objects, position the mouse pointer over an object, and shift the multiple selection with the mouse button pressed.
  - To resize all the objects by the same ratio, grab the resizing handles of the reference object.
- Move over the work area with the icons of the toolbar
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects
- Moving with the shortcut menu commands of the work area
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects

Aligning objects

Procedure

1. Select the objects via multiple selection.
2. Specify an object as the reference object.
3. Select the desired command in the toolbar or the shortcut menu - see table below.
   The selected objects will be aligned.

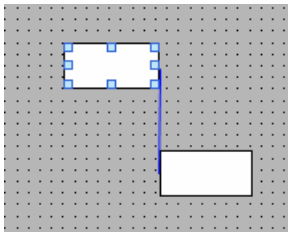
Aligning objects flush

The selected objects will be aligned flush to the reference object.

Icon	Description
	Aligns the selected objects to the left edge of the reference object.
	Aligns the selected objects to the vertical center axis of the reference object.
	Aligns the selected objects to the right edge of the reference object.
	Aligns the selected objects to the upper edge of the reference object.
	Aligns the selected objects to the horizontal center axis of the reference object.
	Aligns the selected objects to the lower edge of the reference object.
	Centers the selected objects to the center points of the reference object.
	Centers the selected objects vertically in the screen.

Snap to object

When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.



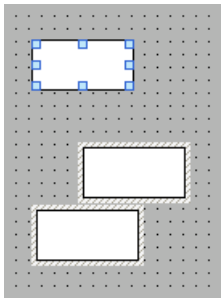
If you are working with the keyboard, press the Alt key. When you move the selected object with the arrow keys, the next anchor point is displayed.

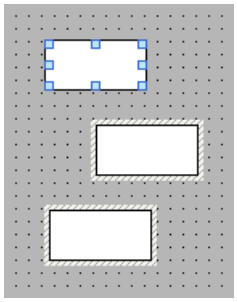
Distributing objects evenly

You need at least three selected objects. A reference object is not required.

1. Select the objects.
2. Click one of the buttons "Distribute horizontally equal" or "Distribute vertically equal".
   The selected objects are distributed at equal distances.

The following screen shows how you align the vertical spacing of the selected objects:





Icon	Description
	Aligns the horizontal distance between the objects. The position of the objects on the extreme left and right side remains unchanged. All other objects are distributed evenly between them.
	Aligns the vertical distance between the objects. The position of the objects at the extreme top and bottom (right and left) remains unchanged. All other objects are distributed evenly between them.

Rotating objects

Introduction

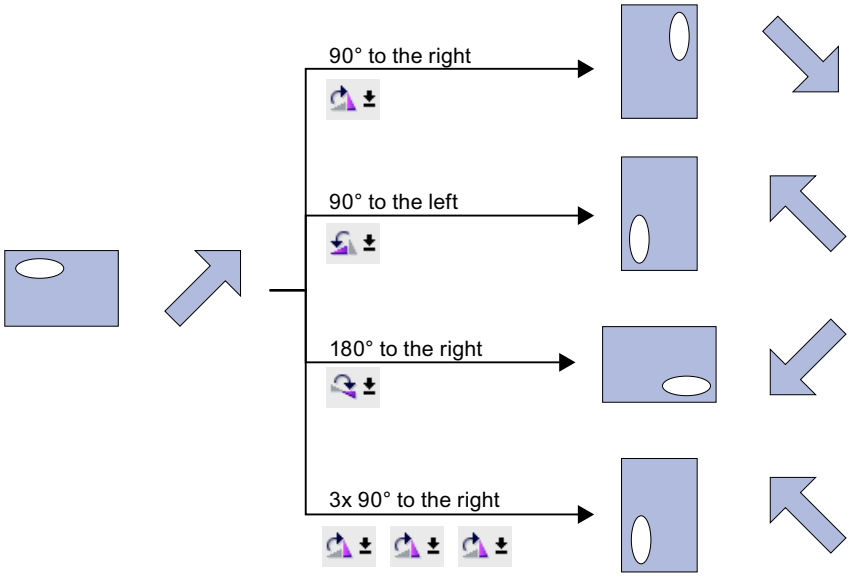
You can rotate a suitable object clockwise or counterclockwise around its center axis in steps of 90°.

Note

Not all the objects can be rotated. Some objects that can be rotated in screens cannot be rotated in reports.

You can also rotate multiple objects using the multiple selection function. Certain WinCC objects such as buttons cannot be rotated.

The alignment of elements in an object will change in a rotated object. The following figure shows how a rectangle and an ellipse behave under the different commands for rotating an object:



Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to rotate.
2. Click one of the following toolbar icons:
   , to rotate the object clockwise around its center point. The angle of rotation is 90°.
   , to rotate the object counterclockwise around its center point. The angle of rotation is 90°.
   , to rotate the object clockwise by 180°.

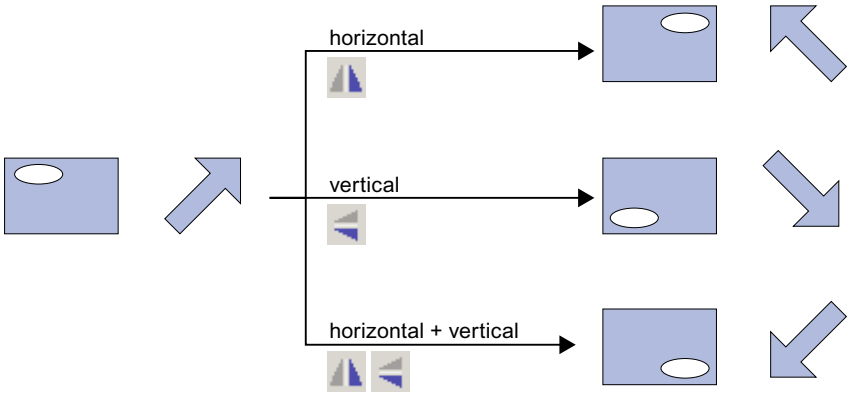
Result

The object is shown at its new angle.

Flipping objects

Introduction

You can flip an object along its vertical or horizontal center axis. The alignment of elements in an object will change when you flip an object. The following figure shows how a rectangle and an ellipse behave under the different commands for flipping an object.



Requirement

You have opened a screen which contains at least one object.

Procedure

1. Select the object that you want to flip.
2. Click the "Flip" command in the shortcut menu and select one of the options displayed:
   - , to flip the selected object along its vertical center axis.
   - , to flip the selected object along its horizontal center axis.

Result

The object is shown at its flipped position.

See also

Inserting an object (Page 4502)

12.6.3 Operation in Runtime

12.6.3.1 Printing reports

Printing a report in Runtime

If you have configured an object with the "PrintReport" system function, the operator can print out a report in Runtime. To do this, the operator uses the "Print report" button on a screen. The operator can set or change further settings for the report. The report is output to the default printer.

See also

Printing reports (Page 4501)

Principles for preparation of reports (Page 4496)

Reports (Page 4493)

12.6.4 Objects in reports

12.6.4.1 Audit report

Application

With the audit report, the content of the audit trail of a HMI device is output as a report.

RecordID	Timestamp	DeltaToUTC	UserID
0	18.11.2007 23:00 ObjectID: Application Description: This is an example description. It may not be cut and will be printed out completely.	-1:00	System

Requirements

The "audit report" object is available only if the following requirements are met:

- The HMI device supports a GMP-compliant configuration.
- GMP-compliant configuration is active in the Runtime settings of the HMI device.

Note that operator actions and system operations are only recorded if a corresponding log is configured.

Layout

In the Inspector window the position, shape, style, color and font types of the object are customized. In particular, the following can be adjusted:

- Visibility of the comment

See also

Enabling GMP compliant configuration (Page 7046)

Audit Trail (Page 7048)

Creating an audit trail (Page 7049)

Principles for preparation of reports (Page 4496)

Reporting an audit trail (Page 7056)

Audit Trail reporting (Page 7057)

Parameters for the audit trail report (Page 7058)

Printing out an audit trail report (Page 7061)

Date/time field (Page 4517)

I/O field (Page 4518)

Graphic view (Page 4520)

Graphic I/O field (Page 4521)

Alarm report (Page 4521)

Recipe report (Page 4524)

Page number (Page 4525)

Symbolic I/O field (Page 4526)

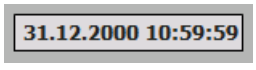
Text field (Page 4526)

Reports (Page 4493)

12.6.4.2 Date/time field

Application

The "Date/time field" object shows the system time of the HMI device or the date and / or time of a connected tag. The layout depends on the language setting on the HMI device.



Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. In the "Properties > Properties > General" dialog, you can adapt the following properties in particular:

- Long date/time format: This setting defines the format displayed for the data and / or time.
- System time: Specifies whether to use the system time of the HMI device, or the data and / or time of a connected tag.

Long date/time format

Option	Description
"Enabled"	Date and / or time is fully displayed, for example "Sunday, December 31, 2000 10:59:59 AM"
"Disabled"	Date and / or time is displayed in abbreviated form, e.g. "12/31/2000 10:59:59 AM"

System time

Option	Description
"Enabled"	The system time of the HMI device is displayed
"Disabled"	The date and / or time of the connected tag is displayed Select a tag of the "DateTime" data type, e.g. an internal tag. Information about data types which are available for connection to other PLCs can be found in the documentation about the respective communication drivers.

See also

Create a report (Page 4498)

Principles for preparation of reports (Page 4496)

Audit report (Page 4516)

12.6.4.3 I/O field

Application

The "I/O field" object is used to enter process values.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Display format: Specifies the format in which the values in the I/O field are output.

Note

The "I/O field" object is also available in "Screens" editor. In reports, the object only outputs data. Accordingly only configure the output of data for application in reports.

Display format

The "display format" for the output of values is specified in "Properties > Properties > General > Format", in the Inspector window.

Layout	
"Binary"	Output of values in binary form
"Date" *	Output of date specifications. The format depends on the language setting on the HMI device.
"Date/time" *	Output of date and time specifications. The format depends on the language setting on the HMI device.
"Decimal" *	Output of values in decimal form.
"Hexadecimal"	Output of values in hexadecimal form.
"Time" *	Output of time specifications. The format depends on the language setting on the HMI device.
"Character string"	Output of strings.
*: not in Runtime Professional	

Avoid overlaps with output fields

If several I/O fields are configured as output fields with a transparent background in a report, these I/O fields may overlap. The transparent part of the one field covers the digits of the other field. This may cause display problems in the report. In order to avoid such overlaps, set the margins of the I/O fields to zero in the object properties under "Properties > Properties > Appearance". Activate "Properties > Properties > Layout > Fit object to contents."

See also

Create a report (Page 4498)

Principles for preparation of reports (Page 4496)

Audit report (Page 4516)

12.6.4.4 Graphic view

application

The "Graphic view" object is used to display graphics.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Graphic: Specifies the graphic file that is displayed in the object.
- Stretch graphic: Specifies the automatic size stretching for objects with graphics.

Inserting graphics

The following graphic format is used in the "Graphic view" object: *.bmp, *.tif, *.png, *.ico, *.emf, *.wmf, *.gif, *.jpg or *.jpeg. You may also use graphics as OLE objects in the Graphic view .

1. In the Inspector window, select "Properties > Properties > General":
2. Select the graphic you want to add in the drop-down list under "Graphic".
   The graphic preview is shown in the right pane.
3. Click "Assign" to insert the graphic in the Graphic view .

Stretch graphic

Whether a graphic displayed in a Graphic view is stretched to the size of the Graphic view in runtime is specified in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Select one of the following options from the "Fit to size" area:
   - Fit object size to content
   - Fit content to object size

See also

Audit report (Page 4516)

12.6.4.5 Graphic I/O field

Application

With the "Graphic I/O field" object, the graphics of a graphics list are displayed, depending on the tag value. In the Inspector window, the tag and graphics list are configured, under "Properties > Properties > General":

Note

The "Graphic I/O field" object is also available in "Screens" editor. In reports, the object only outputs data. Accordingly only configure the output of data for application in reports.

Layout

Customize settings for color, frame, position and object size in the inspector window. In particular, the "Fit to size" is defined by specifying whether the object is adjusted to the graphics during output in the report or the graphic is adjusted in the object.

See also

Create a report (Page 4498)

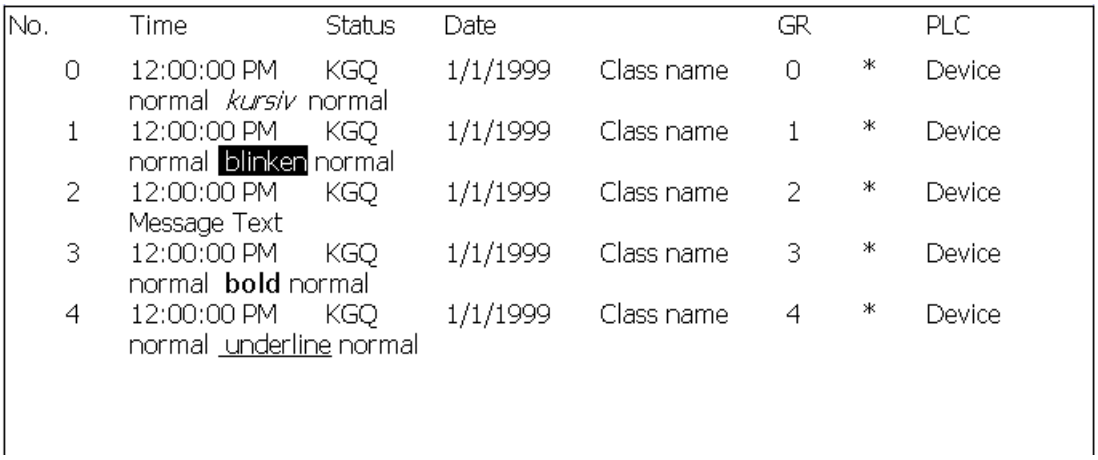
Audit report (Page 4516)

Objects in reports (Page 4516)

12.6.4.6 Alarm report

Application

Use the "Alarm report" object to output alarms of the selected alarm classes from the alarm buffer or alarm log to the report.



Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Source
- Alarm classes
- Time range
- Additional settings for the display
- Visible columns

Specifying the source

Under "Properties > Properties > General > Settings > Source", specify whether to display alarms from the alarm buffer, or from the alarm log. Select the alarm log in the "Alarm log" dialog.

Note

The "Alarm log" source is only available if the HMI device supports logs.

Specifying alarm classes

Under "Properties > Properties > General > Alarm classes, enable the alarm classes for the alarms output to the report.

Specifying the time range

If you want to restrict alarm output to a specific time range, select the start and end tags for the time range in the "Properties > Properties > General > Time range" dialog. The tag must be of the "DateTime" type.

Additional settings for the display

Under "Properties > Properties > General > Settings, specify the following settings for the display in the alarm report:

- Under "Sorting", specify whether to start the display with the oldest or most recent alarm.
- Under "Lines per entry", specify the number of lines to be available for each alarm. The required number of lines depends on the following factors:
  - Number and width of the selected columns for the output
  - Font size used
  - Paper format of the printer
- Select "Visible heading" to enable the display of column headers.
- Select "Display milliseconds" to enable the display of milliseconds for time data.

Visible columns

Under ""Properties > Properties > Layout > Visible columns", enable the columns to be displayed in the alarm report.

See also

Create a report (Page 4498)

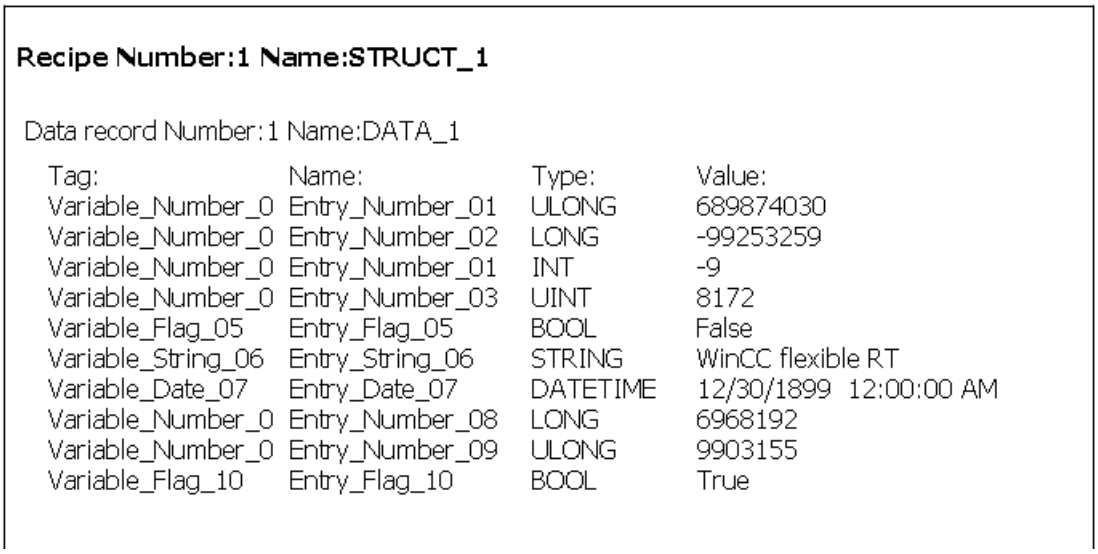
Principles for preparation of reports (Page 4496)

Audit report (Page 4516)

12.6.4.7 Recipe report

application

Use the "Recipe report" object to output the elements of recipe data records in the report.



Recipe Number:1 Name:STRUCT_1

Data record Number:1 Name:DATA_1

Tag:	Name:	Type:	Value:
Variable_Number_0	Entry_Number_01	ULONG	689874030
Variable_Number_0	Entry_Number_02	LONG	-99253259
Variable_Number_0	Entry_Number_01	INT	-9
Variable_Number_0	Entry_Number_03	UINT	8172
Variable_Flag_05	Entry_Flag_05	BOOL	False
Variable_String_06	Entry_String_06	STRING	WinCC flexible RT
Variable_Date_07	Entry_Date_07	DATETIME	12/30/1899 12:00:00 AM
Variable_Number_0	Entry_Number_08	LONG	6968192
Variable_Number_0	Entry_Number_09	ULONG	9903155
Variable_Flag_10	Entry_Flag_10	BOOL	True

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Recipe
- Data record
- Format
- Visible entries

Selecting a recipe

Specify the recipes to be output in the recipe report under "Properties > Properties > General > Recipe". You can define the recipes based on their recipe name, or on a range of recipe numbers. Under "First recipe" and "Last recipe", enter a value or select a tag.

You can also choose to display all recipes.

Selecting a data record

Under "Properties > Properties > General > Data record", specify which data records of the selected recipes are to be output in the recipe report. You can define the data records based on the data record name, or on a range of data record numbers. Under "First recipe" and "Last recipe", enter a value or select a tag.

You can also choose to display all data records of the selected recipes.

Format

In the "Format" field of the "Properties > Properties > Layout > Settings" dialog, specify whether to output the data records as a table in column format or line report format.

WinCC updates the preview in the detail page accordingly.

Visible entries

Under ""Properties > Properties > Layout > Visible entries", enable the columns to be displayed in the recipe report. Select "Show headings" to enable the display of column headers.

See also

Create a report (Page 4498)

Principles for preparation of reports (Page 4496)

Audit report (Page 4516)

12.6.4.8 Page number

application

Use the "Page number" object to output the current page number in the report.

Layout

In the inspector window, change the settings for color, font, position and object size. In particular, specify "Size adaptation":

Fit to size

Under "Properties > Properties > Layout > Fit to size", the "Fit object to content" option is used to specify whether WinCC adjusts to the object size of the field content:

- Enabled "Fit object to contents" option:
  WinCC automatically adjusts the size of the object to the configured format. The font size and field length are defined under "Properties > Properties > General > Text".
  The object can be moved in the working area, however, the size cannot be changed. During report output, the entire field content is output at the time of report output.
- Disabled "Fit object to contents" option:
  Customize the field size by yourself. WinCC does not resize the field for report output. It can be possible that not all content of the field is output in the report. Therefore, configure a field of sufficient size.

See also

Create a report (Page 4498)

Principles for preparation of reports (Page 4496)

Audit report (Page 4516)

12.6.4.9 Symbolic I/O field

application

With the "Symbolic I/O field" object, the content of a text list in reports is displayed, depending on the tag value. In the inspector window, the tag and text list is configured under "Properties > Properties > General":

Layout

In the inspector window, change the settings for color, frame, font, position and object size. In particular, the "Fit to size" is defined, by specifying whether the object size is adjusted to the text during output in the report or not.

Note

The "Symbolic I/O field" object is also available in "Screens" editor. In reports, the object only outputs data. Accordingly only configure the output of data for application in reports.

See also

Create a report (Page 4498)

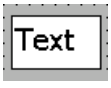
Principles for preparation of reports (Page 4496)

Audit report (Page 4516)

12.6.4.10 Text field

Application

The "Text field" is a closed object which you can fill with a color.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Text: Specifies the text for the text field.
- Size of text field: Defines whether the size of the object is adapted to the space required by the largest list entry.

Text

Specify the text for the text field in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Enter a text.
   For texts over several lines you can set a line break by pressing the key combination <Shift + Enter>.

Size of text field

In the Inspector window, you can define whether the size of the object is adapted to the space required by the largest list entry.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Activate "Resize > Fit to contents".

See also

Audit report (Page 4516)

12.7 Configuring user administration

12.7.1 Field of application of the user administration

Principle

The access protection controls access to data and functions in Runtime. This feature protects your applications against unauthorized operation. Safety-related operations are already limited to specific user groups when a project is being created. To this purpose you set up users and user groups that you equip with characteristic access rights, so-called authorizations. You then configure the authorizations required for operation of safety-related objects. Operators only have access, for example, to specific operator controls. Commissioners, for example, have unlimited access in Runtime.

Definition

You administer users, user groups and authorizations centrally in the user administration of WinCC. You transfer users and user groups together with the project to the HMI device. The users and passwords are managed on the HMI device in the User view.

Application example

You configure the "Service" authorization so that only service technicians have access to the configuration parameters. You assign the authorization to the "Service technician" user group. This allows all members of this group to set the protected configuration parameters.

Notice

Access protection does not protect against incorrect operations. It is your job to ensure that only authorized personnel with appropriate training will design, commission, operate and maintain plants and machines.

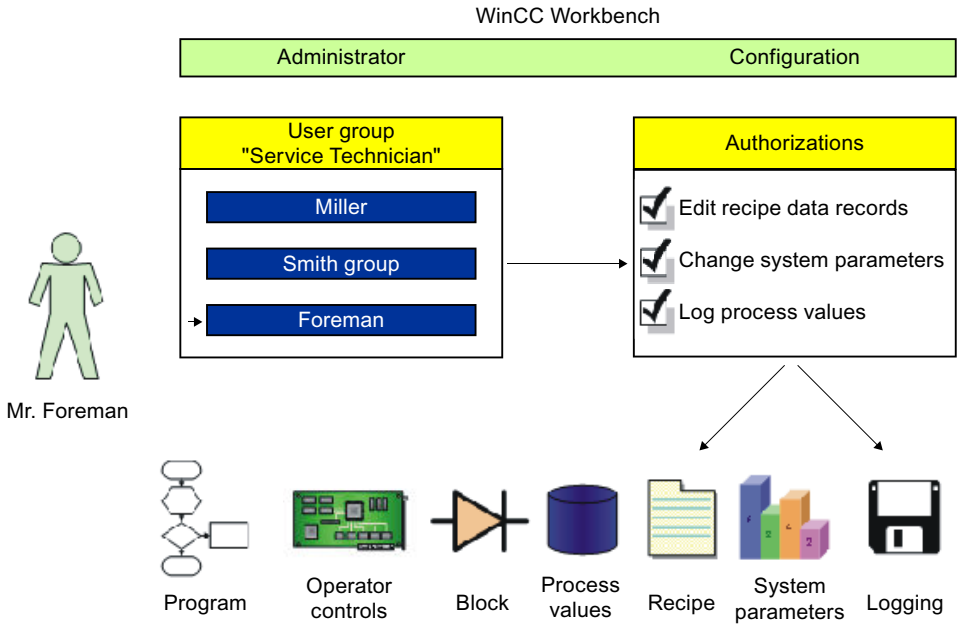
Access protection is not suitable for defining work routines and monitoring their observance.

12.7.2 Form of the user administration

Introduction

In case of a project in manufacturing engineering, the environment at the equipment manufacturer has to be differentiated from the environment at the end customer as plant operator.

The equipment manufacturer allows users, for example Mr. Foreman, a specific access within the application or HMI device. However, a user Foreman does not exist at the end customer. The machine manufacturer cannot know the end users and the tasks they have to perform for configuration. The final users are usually set after commissioning at the end customer.



Principle

To minimize the work required for management, authorizations are assigned via user groups and not directly to individual users.

A user group assembles configured authorizations according to common jobs. For example, all permissions required for a service job are collected in a "Service technician" group. When you create a user who should be responsible for servicing, you simply assign him to the "Service technician" group.

The user view enables user administration in Runtime. Use user view to create, delete and assign an authorization to users in Runtime.

The user administration separates the administration of the users from the configuration of the authorizations. This ensures flexibility at the access protection.

Defaults can be set for the user administration during the configuration phase in the Engineering System.

12.7.3 Basics

12.7.3.1 Users

Introduction

You can create users in the "Users" tab of the "User administration" editor and assign them to user groups. The "Users" tab is part of the user administration in WinCC.

Open

To open the "Users" tab, double-click "User administration" in the project window.

Work area

The users are managed in the work area:

- You create or delete users.
- You assign users to user groups.

Note

You can assign a user to exactly one user group.

Note

On an HMI device the number is limited to 100 users and one PLC user. This restriction does not apply to PCs. On a PC, the maximum number of users is restricted by the physical memory.

Inspector window

When you select a user, you can change the password in the "General" group. Under "Automatic logoff" you can specify if the user is to be automatically logged off by the HMI device when there is no operator activity after the specified time.

Backup and restore

The user data is encrypted and saved on the HMI device to protect it from loss due to power failure.

The users, passwords, group assignments and logoff times set up on the HMI device can be backed up and restored. This prevents you having to enter all of the data again on another HMI device.

Note

The currently valid user data is overwritten in the following cases:

- Depending on settings, upon a new download of the project
- Upon restore of a backed-up project
- Upon import of the user administration via an operator control. The newly downloaded or restored user data and passwords take effect immediately.

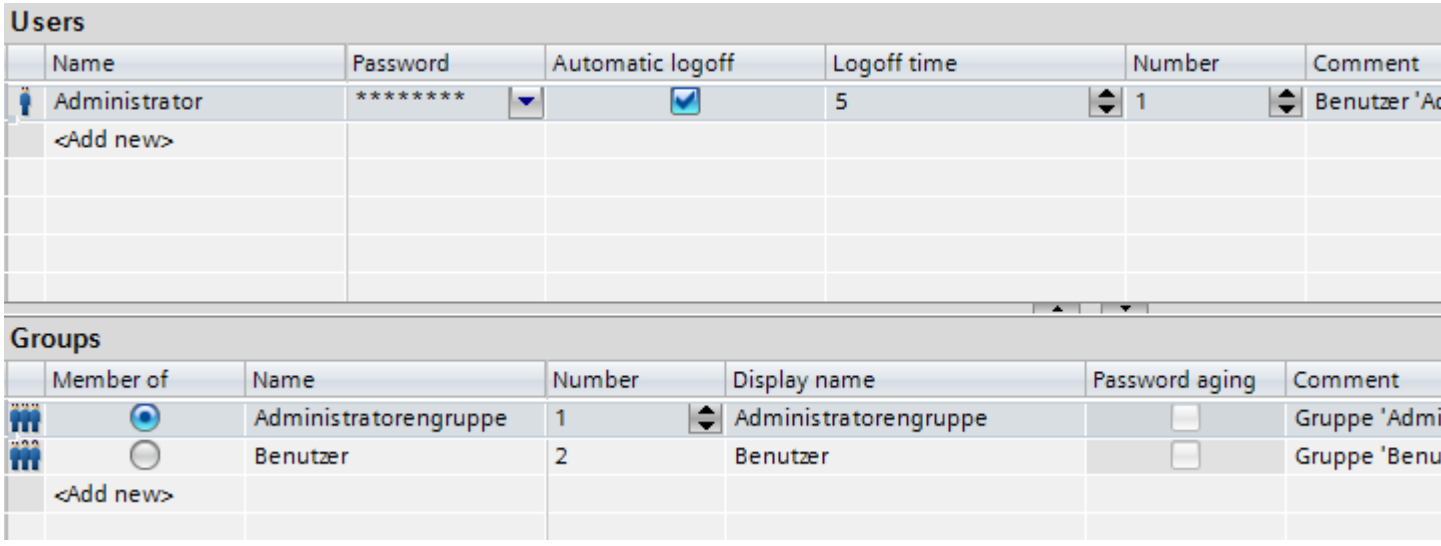
12.7.3.2 Users work area

Introduction

The "Users" work area lists the users and user groups in table form. You administrate the users and assign them to a user group.

Principle

The work area consists of the "Users" and "Groups" tables.



The "Users" table shows the existing users. When you select a user in this table, the "Groups" table shows the user group to which the user belongs.

Note

For the user "Administrator", the default password is "administrator". For security reasons, you should change the password of this user.

12.7.3.3 User groups

Introduction

You can create users and authorizations in the "User groups" tab of the "User Administration" editor. The "User groups" tab is part of the user administration in WinCC.

Open

Double-click the "User administration" in the project window. Open the "User groups" tab.

Work area

The user groups and authorizations are managed in the work area:

- You create new user groups and authorizations or delete them.
- You assign the authorizations to the user groups.

Inspector window

When a user group or an authorization is selected, you can edit the name in the "General" group. You can also enter a brief description in the "Comment" group.

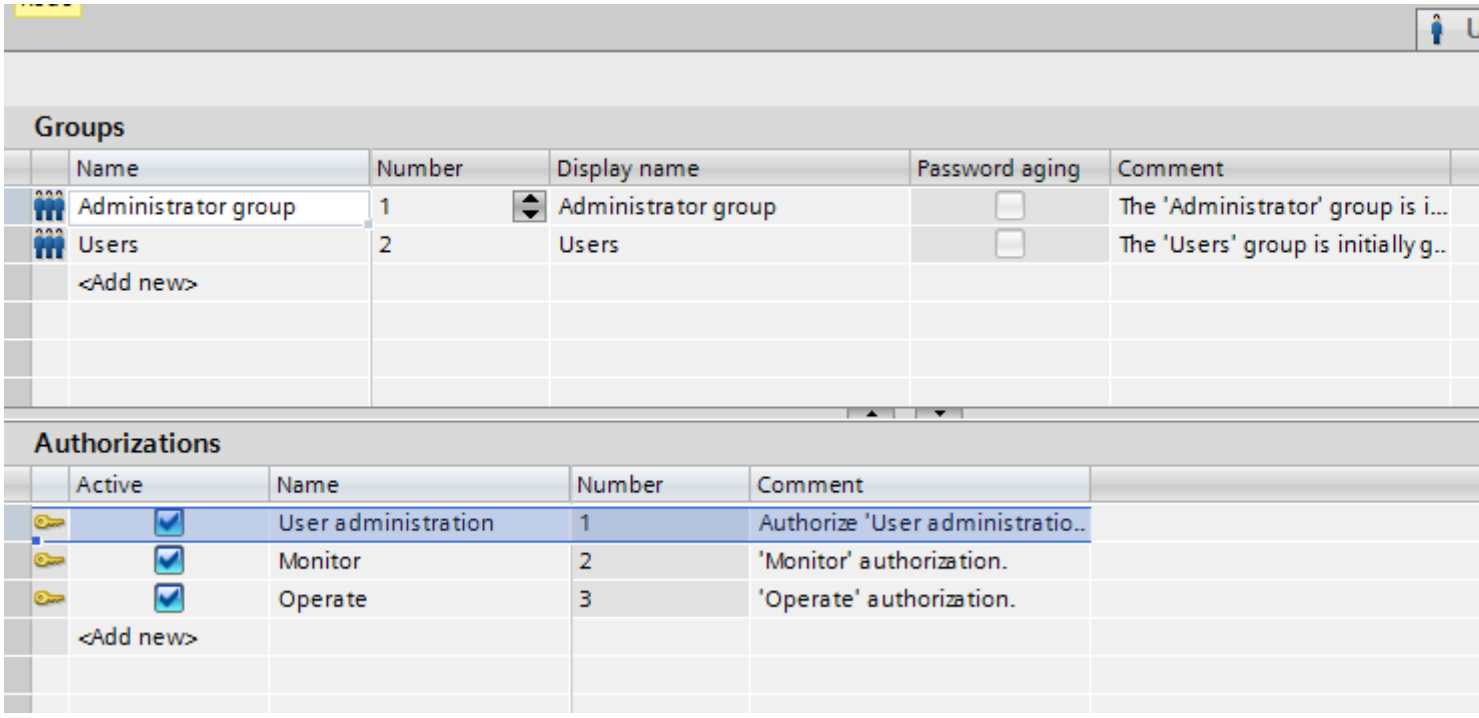
12.7.3.4 User groups work area

Introduction

The "User groups" work area shows a table of the groups and their authorizations. You administer the user groups and assign authorizations to them.

Principle

The work area consists of the "Groups" and "Authorizations" tables.



The "Groups" table shows the existing user groups. When you select a user group in this table, the "Active" column of the "Authorizations" table shows the authorizations which were assigned to the user group.

The number of the user group and of the authorization is assigned by the user administration. The designations and descriptions are assigned by you.

The number of predefined authorizations are fixed. Authorizations that you create can be freely edited. Ensure that the assigned numbers are unique.

12.7.3.5 Settings for the user administration

Introduction

In the "Runtime settings > User administration" editor, configure the security settings for users and their passwords in runtime.

Open

Double-click the "Runtime settings" editor in the project window. Click "User administration".

Work area

You carry out the settings for the validity of passwords in runtime in the work area. You determine the complexity of the password, for example.

Effects in Runtime

The security settings have the following effects in runtime, depending on the configuration.

- "General" group
  - "Enable limit for logon attempts" check box selected
    The number entered in the "Number of incorrect logon attempts" box defines the number of logon attempts a user is allowed before being assigned to the "Unauthorized" group.
    "Enable limit for logon attempts" check box not selected
    The user has an unlimited number of logon attempts in runtime.
  - "Number of incorrect logon attempts" field
    If you enter "4" in the field, for example, and the fourth logon attempt fails, the user is automatically assigned to the "Unauthorized" group.
    You can specify 1 to 9 attempts.
  - "Logon only with password" check box
    When the check box is selected, the user will be authenticated by the password. The user name is not required.
    To match users to passwords, you cannot configure passwords more than once.
- "Hierarchy level" group
  - "Group-specific rights for user administration" check box
    When this check box is selected, administrators only manage users whose group number is less than or equal to their own.
    For example, an administrator whose group number is 5 can only manage users whose group number is less than or equal to 5. This means that the administrator can assign users only to groups with a group number less than or equal to 5.

- "Password" group
  - "Enable password aging" checkbox selected
    The password expires after the number of days set in the "Validity of the password (days)" field.
    The "Password aging" column is selected in the "User groups" editor. This enables you to specify group-by-group, if the passwords should expire or if the password generations should be saved. Passwords never expire for groups for which password aging is not enabled.
  - "Prewarning time (days)" field
    The user is informed that the password will expire the specified number of days before the password expires.
  - "Password generations" field
    If the user changes the password, the new password must be different from the specified number of previous passwords. The number of password generations ranges from 1 to 5.
- "Password complexity" group
  - "Must include special characters" check box selected
    The user must enter a password containing at least one special character at any position.
  - "Must include number" check box selected
    The user must enter a password containing at least one number at any position.
  - "Minimum password length" field
    The user must enter a password with a minimum length, as specified in the "Minimum password length" field.
    The minimum length of the password is 3 characters.

12.7.3.6 Settings for the user administration

Introduction

In the "Runtime settings > User administration" editor, configure the security settings for users and their passwords in runtime.

Open

Double-click the "Runtime settings" editor in the project window. Click "User administration".

Work area

You carry out the settings for the validity of passwords in runtime in the work area. You determine the complexity of the password, for example.

Effects in Runtime

The security settings have the following effects in runtime, depending on the configuration.

- "General" group
  - "Change initial password" check box selected
    The user must change the administrator-assigned password when logging on for the first time.
  - "Change logoff time" checkbox selected
    Simple user rights are sufficient for changing the logoff time.
    The logoff time is the period after which the user administration automatically logs off a user when no operator activity is detected in the system.
    The logoff time for the SIMATIC Logon user corresponds to the logoff time of the default user "Administrator".
    Any user changes of the logoff time are logged in the Audit Trail.
  - "Enable limit for logon attempts" check box selected
    The number entered in the "Number of incorrect logon attempts" box defines the number of logon attempts a user is allowed before being assigned to the "Unauthorized" group.
    "Enable limit for logon attempts" check box not selected
    The user has an unlimited number of logon attempts in runtime.
  - "Number of incorrect logon attempts" field
    If you enter "4" in the field, for example, and the fourth logon attempt fails, the user is automatically assigned to the "Unauthorized" group.
    You can specify 1 to 9 attempts.
  - "Logon only with password" check box
    When the check box is selected, the user will be authenticated by the password. The user name is not required.
    To match users to passwords, you cannot configure passwords more than once.
- "Hierarchy level" group
  - "Group-specific rights for user administration" check box
    When this check box is selected, administrators only manage users whose group number is less than or equal to their own.
    For example, an administrator whose group number is 5 can only manage users whose group number is less than or equal to 5. This means that the administrator can assign users only to groups with a group number less than or equal to 5.

- "Password" group
  - "Enable password aging" checkbox selected
    The password expires after the number of days set in the "Validity of the password (days)" field.
    The "Password aging" column is selected in the "User groups" editor. This enables you to specify group-by-group, if the passwords should expire or if the password generations should be saved. Passwords never expire for groups for which password aging is not enabled.
  - "Prewarning time (days)" field
    The user is informed that the password will expire the specified number of days before the password expires.
  - "Password generations" field
    If the user changes the password, the new password must be different from the specified number of previous passwords. The number of password generations ranges from 1 to 5.
- "Password complexity" group
  - "Must include special characters" check box selected
    The user must enter a password containing at least one special character at any position.
  - "Must include number" check box selected
    The user must enter a password containing at least one number at any position.
  - "Minimum password length" field
    The user must enter a password with a minimum length, as specified in the "Minimum password length" field.
    The minimum length of the password is 3 characters.
- "SIMATIC Logon" group
  - "Activate SIMATIC Logon" check box selected
    A connection is established to the server. Authorization is performed via SIMATIC Logon.
  - "Encrypted transfer" checkbox selected
    The data is encrypted before it is transferred to the server.

12.7.4 Building up and structuring a user administration

12.7.4.1 Basics of user administration

Principle

This section addresses four target groups. The topics are organized correspondingly. The target groups serve as examples for different groups of persons who use the user administration.

1. Administrator OEM
2. Administrator RT

3. Planners
4. Operator

As Administrator OEM you create the user groups, users and authorizations for Runtime in the Engineering System of, for example, an equipment manufacturer.

As Administrator RT you administer users in Runtime by means of the "User view."

As the project engineer you assign the authorizations to the user groups in the Engineering System. In addition, you configure the authorizations for objects.

As Operator you log on in runtime. You can only access a protected object if you have the required authorization.

Note

The Administrator RT target group already exists in the Runtime user administration as the predefined user group "Administrator group." For a better understanding the predefined user groups and authorizations are not used below.

12.7.4.2 Administering users for Runtime

Creating an authorization

Introduction

You create an authorization and assign it to one or more user groups.

Requirements

The "User groups" work area is open.

Procedure

1. Double-click "Add" in the "Authorizations" table.
2. Enter "Stop runtime" as the name of the authorization.
3. Enter a brief description as the "Comment".

See also

Creating a user group (Page 4538)

Creating a user group

Introduction

User groups are created so that you do not have to assign an authorization to every single user. You create a user group, assign authorizations and then assign users to it.

Note

The name of the user group has to be unique within the project. Otherwise the input is not accepted.

Note

Using SIMATIC Logon

Ensure that the names of the user groups in Windows and WinCC are identical.

Requirements

The "User groups" work area is open.

Procedure

1. Double-click "Add" in the "Groups" table.
2. Enter "Operators" as the "Name" of the user group.
3. Change the "Number" of the user group as required.
4. Enter "Display name" of the "Operators" user group.
5. Enter a brief description as the "Comment".

In runtime, the user view shows the display name of the user group. The display name of the user group depends on the language. You can specify the name in several languages and switch between languages in runtime.

See also

Creating an authorization (Page 4537)

Example: Configuring a button with logon dialog box (Page 4559)

Example: Logging the logon and logoff events (Page 4560)

Assigning an authorization

Introduction

When you allocate an authorization to a user group, all assigned users have this authorization.

Requirements

- A "Stop runtime" authorization has been created.
- An "Operators" user group has been created.
- The "User groups" work area is open.

Procedure

1. Click on the "Operators" user group in the "Groups" table. The "Authorizations" table shows all authorizations.
2. Enable the "Stop runtime" authorization in the "Authorizations" table.

Note

The "Stop runtime" authorization is only a designation and does not have any relation to the function "StopRuntime". You have to establish this relation yourself. To do so, configure the "StopRuntime" system functions at a button and select "Stop runtime" as the "Authorization".

Creating users

Introduction

You create a user so that users can log on with their user names in runtime after loading to the device.

As an alternative, you can create and change users by means of the "User view" in Runtime.

In order for a created user to have authorizations you have to assign him to a user group and allocate authorizations to the user group.

The logon is successful when the user name entered during the logon matches a user in Runtime. In addition, the entered password must agree with the stored password of the user.

Note

Note that the entry is case-sensitive.

Requirements

The "Users" work area is open.

Procedure

1. Double-click "Add" in the "Users" table.
2. Enter "Foreman" as the user name.

   Note

   The user name must be unique within the project. Otherwise the input is not accepted.

   Note

   Do not use special characters such as / " § $ % ? when you enter a user name and a password. ' &.

3. Click the button in the "Password" column. A dialog box for entering the password is displayed.
4. Enter the password of the user.
5. To confirm the password enter it a second time in the lower field.
6. Close the dialog box using the icon.
7. If a user is to be logged off after a specific time period, activate "Automatic logoff".
8. Click in the "Logoff time" column. The preset value for "Logoff time" is 5 minutes.
9. Enter a brief description as the "Comment".

Assigning a user to a user group

Introduction

When you assign a user to a user group, the user has the authorizations of the user group.

Note

You have to assign a user to exactly one user group. The assignment is checked during the consistency check and generation of the project.

Requirements

- The user "Foreman" has been created.
- An "Operators" user group has been created.
- The "Users" work area is open.

Procedure

1. Click on the "Foreman" user in the "Users" table. The "Groups" table shows all user groups.
2. Activate the "Operators" user group in the "Groups" table.

Managing users

Introduction

In the work area, you can administer users and assign them to user groups.

Requirements

The "Users" work area is open.

Changing the user name

1. In the "Users" table, double-click the field in the "Name" column to change the user name.
2. Change the user name.
3. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the user name under "Properties > Properties > General" in the Inspector window.

Changing the password of the user

1. Click the button in the "Password" column of the "Users" table. A dialog for entering the password opens.
2. In the "Enter password" field, enter the new password.
3. Reenter the new password in the "Confirm password" field.
4. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the password under "Properties > Properties > General" in the Inspector window.

Displaying invisible columns

1. Position the mouse cursor on the title of the "Users" table.
2. Right-click to open the shortcut menu and enable the display of the "Logoff time" column, for example.

Changing the logoff time of the user

1. In the "Users" area, double-click on the field in the "Logoff time" column to change the logoff time.
2. Change the logoff time.
3. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the logoff time under "Properties > Properties > Automatic logoff" in the Inspector window.

Deleting a user

1. Select the line of the user to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

Note

Predefined users cannot be deleted.

Managing user groups

Introduction

In the workplace you administer user groups and assign authorizations for use in runtime.

Requirements

The "User groups" work area is open.

Changing the name of the user group

1. In the "Groups" table, double-click the field in the "Name" column to change the name of the user group.
2. Change the name of the user group.
3. Confirm your entry with <Return>.

Alternatively, select the user group in the work area. Change the name under "Properties > Properties > General" in the Inspector window.

Note

Predefined user groups cannot be deleted.

Displaying invisible columns

1. Position the mouse cursor on the title of the "Users" table.
2. Right-click to open the shortcut menu and enable the display of the "Display name" column, for example.

Changing the displayed name of the user group

1. In the "Groups" table, double-click the field in the "Display name" column to change the display name of the user group.
2. Change the displayed name of the user group.
3. Confirm your entry with <Return>.

Alternatively, select the user group in the work area. Change the display name under "Properties > Properties > General" in the Inspector window.

Deleting a user group

1. Mark the line of the user group to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

Note

Predefined user groups cannot be deleted.

Changing the name of the authorization

1. In the "Authorizations" table, double-click the field in the "Name" column to change the name of the authorization.
2. Change the name of the authorization.
3. Confirm your entry with <Return>.

Alternatively, select the authorization in the work area. Change the name under "Properties > Properties > General" in the Inspector window.

Deleting authorizations

1. Mark the line of the authorization to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

Note

Predefined authorizations cannot be deleted.

12.7.4.3 Managing users on the server

Central user administration using SIMATIC Logon

Introduction

To manage users and user groups for several applications or HMI devices, activate SIMATIC Logon.

Principle

SIMATIC Logon is a tool for system-wide user administration. If you use SIMATIC Logon, the users are centrally managed outside the application or HMI device.

You configure the user groups and their permissions as you usually do with the local user administration in WinCC. You assign identical names to the user groups on the server and in WinCC. Authorizations are assigned in runtime to a user group when the names are identical.

You do not have to create users in WinCC because they are taken dynamically from Windows user management during the logon process. The application or HMI device forwards each logon or password change to SIMATIC Logon for processing.

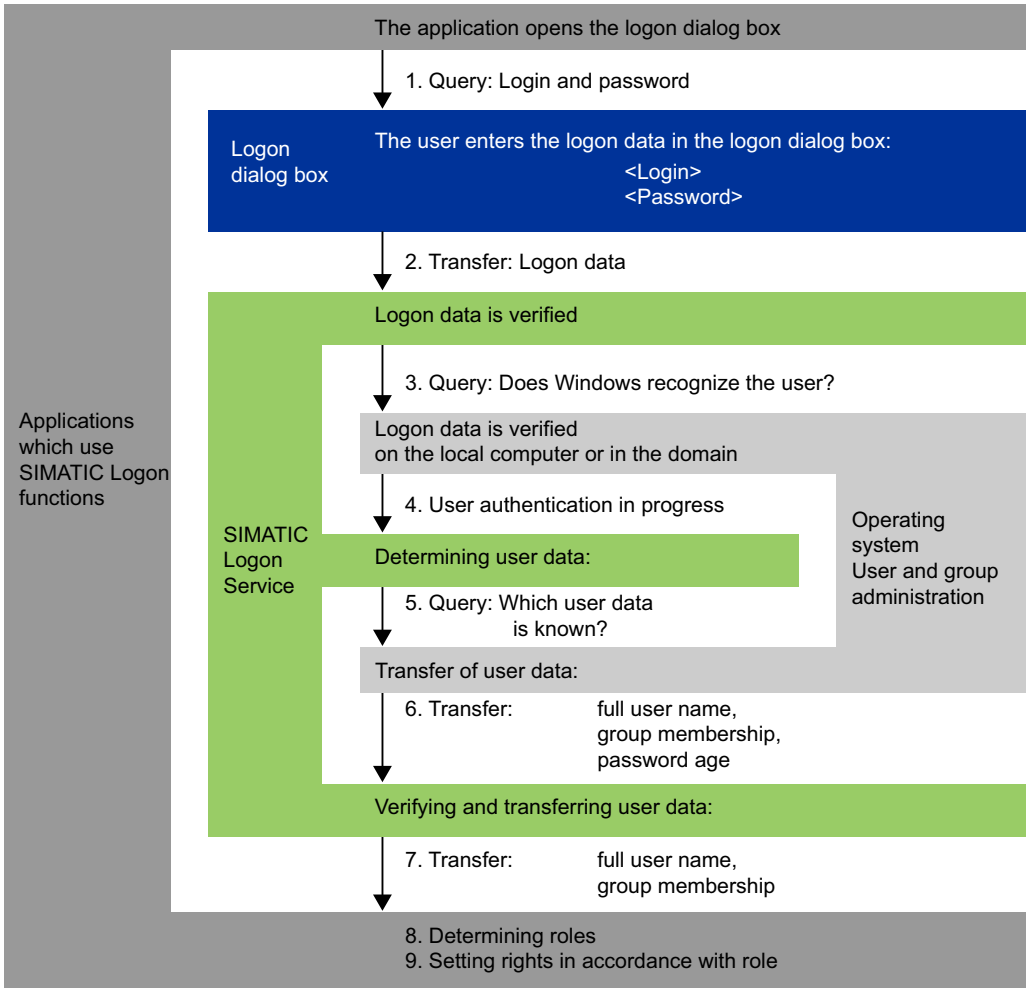
Note

SIMATIC Logon is a product requiring a license. For more information on SIMATIC Logon go to <http://support.automation.siemens.com>.

Enter the ID "34519648" in the search field and start searching. The "SIMATIC Logon - Electronic Signature" manual is available to download.

Log on process via SIMATIC Logon Service

The following diagram illustrates the process that runs automatically when a user logs onto Runtime.



Results: The user can implement the role functions.

Making settings on the server

Introduction

Perform the following steps on your sever to ensure correct operation of SIMATIC Logon.

Procedure

1. Install SIMATIC Logon Services.
2. Create the user group in the user administration of the operating system.

3. Create the users in the user administration of the operating system.

   Note

   Users of SIMATIC Logon must be members of a user group of the operating system. Members of a subgroup cannot be logged on.

4. Assign the new users to a group.
5. Transfer the licenses for each HMI device on the server.

Note

You can find more detailed information in the documentation supplied with SIMATIC Logon.

Note

The password policies stored on the server are valid if users are authorized by means of SIMATIC Logon.

Note

You can configure the logoff time for the SIMATIC Logon user with "Configure SIMATIC Logon" on the server.

You set the automatic logoff time identical for all users, regardless of the panels.

Logging on using SIMATIC Logon

Requirements

- User administration exists.
- The groups created in WinCC agree with the Windows groups on the server.
- There is a server with SIMATIC Logon Service installed.

Note

A user may only be a member of one user group in WinCC. When a user is a member of several user groups on the server, only one of these user groups can be made known in WinCC.

"Administrators" and "Users" are predefined groups in both Windows and WinCC. Any new user created in Windows is automatically a member of the "Users" group. A user will be a member of two groups if assigned to a new group, for example, to the "SL user" group.

Remove this user from the "Users" group of the operating system to enable logon of this user using SIMATIC Logon.

A system message in runtime always confirms the successful logon of a user on the server.

Note

When you use SIMATIC Logon to manage access to a panel or a device with WinCC Runtime Advanced, you must note that the characters '/' and '\' cannot be used in the names of Windows user groups or Windows users.

Procedure

1. Open the "Runtime settings > User administration" editor in the Project window.
2. Select "Enable SIMATIC Logon".
3. Select "Windows domain" or "Windows computer", depending on where you administer your users.
4. Enter the name or IP address of the SIMATIC Logon server in the "Server name" text box.

   Note

   Make sure your SIMATIC Logon Server always has the same IP address if you use IP addresses.

5. Enter the port number for TCP/IP communication in the "Port number" text box. The valid range of values lies between 1024 and 49151. The preset port number is "16389" and used by SIMATIC Logon as the default.
6. Enter the Windows domain or the logon server with the user data in the "Windows domain" text box.
   Enter the name of the sever to access users on the logon server.
   Enter the name of the domain if the logon sever is in a domain and you wish to access users of the domain.

   Note

   Local Windows users are not accepted when the logon server is in a domain.

7. Activate the "Encrypted transfer" check box in order to encrypt the data before sending it to the server.

Emergency user

If the server cannot be accessed, all local users that were created in WinCC user administration can also be used as emergency users.

Emergency users have the rights of the user group to which they were assigned.

Note

For additional information, refer to the included SIMATIC Logon documentation.

This documentation is available at: [SIMATIC Logon installation directory] \manuals

12.7.4.4 Administering users in Runtime

Users in runtime

Principle

You create users and user groups and assign authorizations to them. You configure objects with authorizations. After download to the HMI device, all objects which were configured with an authorization are protected against unauthorized access in Runtime.

User view

When you configure a user view in the Engineering System, you administer users in this user view following download to the HMI device.

Notice

Changes in the user view are effective immediately in Runtime. Changes in runtime are not updated in the engineering system. When downloading the user administration to the HMI device, all changes in the user view are overwritten after a security prompt and based on the settings.

Users who have a "User administration" authorization have unlimited access to the user view. This allows them to administer all users. Any other user has only limited access to the user view for self administration.

See also

User view (Page 4548)

User view

Purpose

You configure a user view in the engineering system to also administer the users in Runtime.

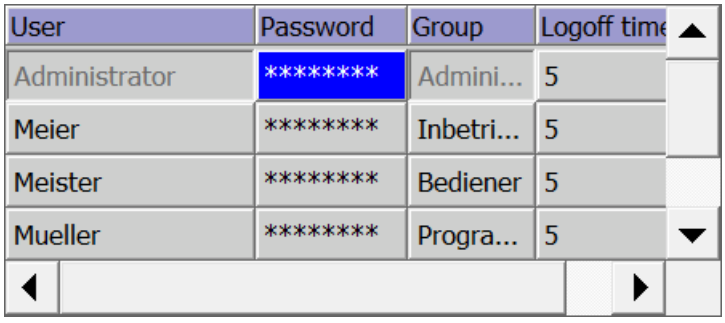
Structure

The user view shows the following in each line:

- The user
- The encrypted password
- The corresponding user group
- The logoff time

If no user is logged on, the user view is empty. The content of the individual fields is displayed after logon.

User view of an administrator



When an administrator is logged on, the user view shows all the users. The administrator changes the user name and the password. The administrator creates new users and assigns them to an existing user group.

User view of a user

When no administrator is logged on, the user view shows only the logged-on user. Users can change their own passwords and logoff time.

See also

Users in runtime (Page 4548)

Configuring a user view

Introduction

You configure a user view in the Engineering System to also administer users in Runtime.

Requirements

A screen has been created.

Procedure

1. Select the "User view" object from the "Controls" category in the toolbox.
2. Drag-and-drop the "User view" object into the screen.
3. Click on "Properties > Properties" in the Inspector window.
4. Specify the appearance of the "User view".
5. You can, for example, select " "Display mode > Fit to size > Fit object to contents".

Result

You have created a user view in the screen.

Creating users

Introduction

You create a user so that users can log on under their user name in runtime.

As an alternative, you can create users in the engineering system and download them to the HMI device.

The logon is successful only when the user name entered during the logon matches a user in runtime. In addition, the password entered at logon has to match that of the user.

Note

Note that the entry is case-sensitive.

Note

Do not use special characters such as / " § $ % ? when you enter a user name and a password. ' &.

You assign the user to a user group. The user then has the authorizations of the user group.

Note

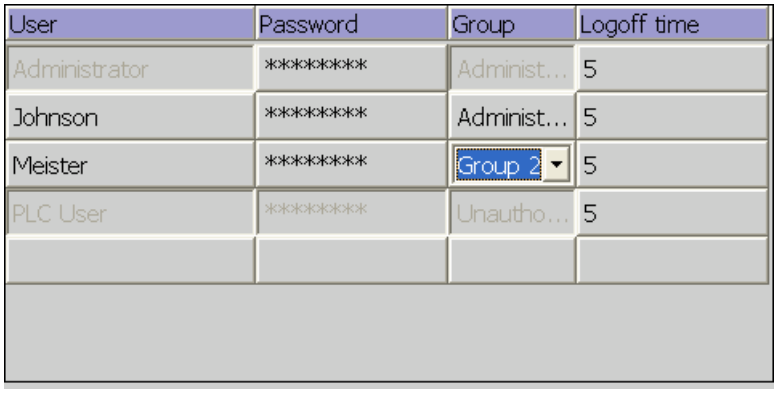
Runtime users must be assigned to a user group. The user group is created in the engineering system. The designation of the user group is language-dependent.

Requirements

- The user view is open.
- A "Group 2" user group has been created.

Procedure

1. Click "<New User>" in the user view. A dialog opens.
2. Enter "Foreman" as the user name.
3. Press the <Return> button.
4. Click "Password."
5. Enter the password of the user.
6. Press the <Return> button. The password is hidden.
7. Click in the "Group" column.
8. Select "Group 2" as the "Group".



9. Press the <Return> button.
10. Click in the "Logoff time" column.
11. Enter the time after which the user is logged off automatically.

Managing users in the user view

Introduction

If you have configured a user view in the engineering system, the users and user groups can be managed in runtime.

Notice

Changes in the user view are effective immediately in runtime. Changes in runtime are not updated in the engineering system. When downloading the user administration to the HMI device, all changes in the user view are overwritten after a security prompt and based on the settings.

Requirements

- Runtime is enabled.
- A complex user view has been created.

- The screen with the user view is open.
- You have the default "User administration" authorization.

Note

If you do not have a "User administration" authorization, you can only change your own password and your logoff time.

Changing the user name

1. Enter a new user name in the "Users" column in the user display.
2. Confirm your entry with <Return>.

Note

The user can then no longer log on with his previous password in runtime. If you delete the name and press <Return>, the user is deleted.

Changing the password of the user

Availability of complex user display in selected devices only.

1. Enter a new password in the "Password" column in the user display.
2. Confirm your entry with <Return>.

Note

The user can then no longer log on with his previous password in runtime.

If you delete the password in the complex user view and press <Return>, the user will be deleted.

Changing the logoff time of the user

1. Enter a new logoff time in the "Logoff time" column in the user display.
2. Confirm your entry with <Return>.

Deleting a user

1. Click the name of the user to be deleted.
2. Delete the name.
3. Press the <Return> button.

Note

The user can no longer log on in runtime.

Assigning a user to a different user group

1. Activate the user group field for the corresponding user.
2. Select a user group.
3. Confirm your selection with <Return>.

Unlock locked out users

Unlock locked out users

The check box "Activate limit for login attempts" is activated in the "Runtime settings > User administration".
The number 3 is entered in the field "Number of invalid login attempts".

If users have three failed attempts at login, e.g. by entering an incorrect password, they are assigned to the "Unauthorized" group. The user loses all authorizations. The user can still log on, but no longer has any authorizations. Only a user with administrator rights re-assigns the unauthorized user to a user group.

Exporting the user administration

Introduction

The users and user groups are transferred to the HMI device from the Engineering System. If you have configured a user view, you can administer the users at the HMI device via the user view.

If a user has access to several HMI devices, the same user and the same password must be configured on each HMI device. Create a standard for user administration of the different HMI devices. Export the users and passwords on an HMI device to a storage medium, such as a floppy, memory card or a network drive. You import the users and passwords from this storage medium at all the other HMI devices.

You program a function which exports the user data at a button.

The system function "ExportImportUserAdministration" is not available on all HMI devices.

Notice

When exporting, the user data are encrypted and written from the HMI device to the target file. The target file is overwritten

Requirements

- An HMI device has been created.
- A screen has been created.
- A button has been created in the screen.
- The Inspector window is open.

Procedure

1. Click the button in the screen.
2. Click "Properties > Events > Click" in the Inspector window.
3. Click the entry "Add function" in the function list.
4. Select the "ExportImportUserAdministration" system function.
5. Select "Export" as the "Direction". When exporting, the target file is overwritten with the user data.
6. Enter the full path of the destination file as the "File name", for example "a:\test \usersview.txt".

Note

Ensure that the file name is written correctly. If the directory, for example "\test\", does not exist, it will be created without a prompt.

Importing the user administration

Introduction

If a user has access to several HMI devices, the same user and the same password must be configured on each HMI device. Create a standard for user administration of the different HMI devices. Export the users and passwords on an HMI device to a storage medium, such as a floppy, memory card or a network drive. You import the users and passwords from this storage medium at all the other HMI devices.

You configure a function which imports the user data at a command button.

The system function "ExportImportUserAdministration" is not available on all HMI devices.

Notice

When importing, the user data are read in from the source file and written into the HMI device. The users and passwords currently valid in the HMI device are overwritten. The imported users and passwords are valid immediately.

Requirements

- A screen has been created.
- A command button has been created in the screen.
- The Inspector window is open.

Procedure

1. Click the button in the screen.
2. Click "Properties > Events > Click" in the Inspector window.

3. Click the entry "Add function" in the function list.
4. Select the "ExportImportUserAdministration" system function.
5. Select "Import" as the "Direction." When importing, the user data in the HMI device are overwritten.
6. Enter the full path of the source file as the "File name", for example "a:\test\usersview.txt".

Note

Ensure that the file name is written correctly. If the folder, for example "\test\", or the file does not exist, an error message will be displayed.

Logging on as a user

Introduction

As a rule you log-on as a user by means of a special button. The logon dialog box is displayed to this purpose.

The logon dialog box is displayed by default during access to a protected object if

- No user is logged on in Runtime.
- The logged-on user does not have the required authorization.

Note

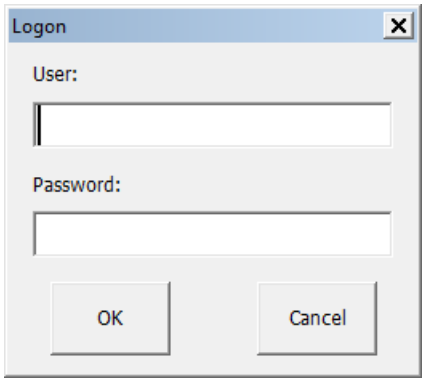
The system always opens the logon dialog on the OP 73, OP 77A, TP 177A and Basic Panels HMI devices when you press an access-protected button:

Requirements

- Under "Runtime settings > User administration" the
  - "Enable limit for logon attempts" check box has been selected.
  - The number 3 is entered in the field "Number of incorrect login attempts".
- The "ShowLogonDialog" system function is configured on a button called "Logon".

Procedure

1. Click the "Logon" button. The logon dialog box is displayed.



2. Enter your user name as it was specified in the user administration, for example "Foreman". If someone has logged on before you, the name of the user will be displayed.
3. Enter the corresponding password. The input is concealed.

   Note

   When entering the password, note that it is case sensitive.

4. Click "OK" to close the dialog box.

Logon was successful

If you have entered the user name "Foreman" and the entered password corresponds with the stored password, you are logged on as the user "Foreman" in Runtime. You have the authorizations of the user "Foreman".

When the user "Foreman" accesses a protected object such as the "Logging" button, access to this protected object will only be authorized if the user "Foreman" has the required authorization. The programmed function is executed immediately.

If you do not have the required authorization after the successful log-on, a corresponding error message is displayed. However, you remain logged on in Runtime.

Logon was unsuccessful

An error message is displayed.

In order to maintain security, you or the user logged-on before you no longer has any authorizations. However, access to unprotected objects remains possible. The user view does not, however, show any entries. The user view and the authorizations change after the next successful log-on.

If the third login attempt has failed, the user will be assigned to the "Unauthorized" group. For this reason, do not configure a user group with this display name.

If the "Log off" function is called up or the logoff time of the user has expired, the user is logged off.

12.7.4.5 Configuring access protection

Access protection

Introduction

You configure an authorization at an object in order to protect it against access. All logged-on users who have this authorization can then access the object. If a user does not have authorization to operate an object, the logon dialog is displayed automatically.

Note

Several system functions are available under "User administration" so that user, password, and user group can be edited, for example, in the PLC.

Configuring operating authorizations

Introduction

You configure the "Stop runtime" authorization on a button. Then only those users who have the appropriate authorization have access to this button, for example all the users of the "Operators" user group.

This ensures that access to the command button is protected. If a logged-on user who belongs to the "Operators" user group and has the required authorizations clicks the button, runtime is stopped.

An example describes in detail how to configure a command button with access protection.

Requirements

- The "Operators" user group has been created.
- The "Stop runtime" authorization has been created.
- A screen has been created and opened.
- The screen contains a button.

Procedure

1. Click the button in the screen.
2. Click "Properties > Properties > Security" in the Inspector window.
3. Select "Stop runtime" as the "Authorization".

4. In the Inspector window, select "Properties > Events > Click".
5. Select a system function from the function list, for example, "StopRuntime".

Note

The "Enable" and "Disable" events are only used to detect whether an object was selected or deselected. The events do not, however, trigger a password prompt.

Do not use the "Enable"or "Disable" event if you want to configure access protection at the function call of the object.

12.7.5 Reference

12.7.5.1 Objects with access protection

Introduction

The following objects can be configured with an authorization:

- Date/time field
- I/O field
- Graphic I/O field
- Recipe view
- Switch
- Button
- Symbolic I/O field
- System diagnostic view

12.7.5.2 Objects with access protection

Introduction

The following objects can be configured with an authorization:

- Date/time field
- I/O field
- f(x) trend view
- Graphic I/O field
- HTML Browser
- Alarm view

- Alarm window
- Recipe view
- Switch
- Button
- Slider
- Symbol library
- Symbolic I/O field
- System diagnostic view
- System diagnostic window

12.7.5.3 Default user groups and authorizations

Principle

The predefined user groups and authorizations have the following numbers:

User group	Number
"Administrator group"	1
"Users"	2

Authorization	Number
"User administration"	1
"Monitor"	2
"Operate"	3

12.7.6 Examples

12.7.6.1 Example: Configuring a button with logon dialog box

Task

In the following example, you configure the function "ShowLogonDialog" for a button. A different user can then log on in runtime when the shift changes, for example. In the process the user previously logged on is logged off.

Note

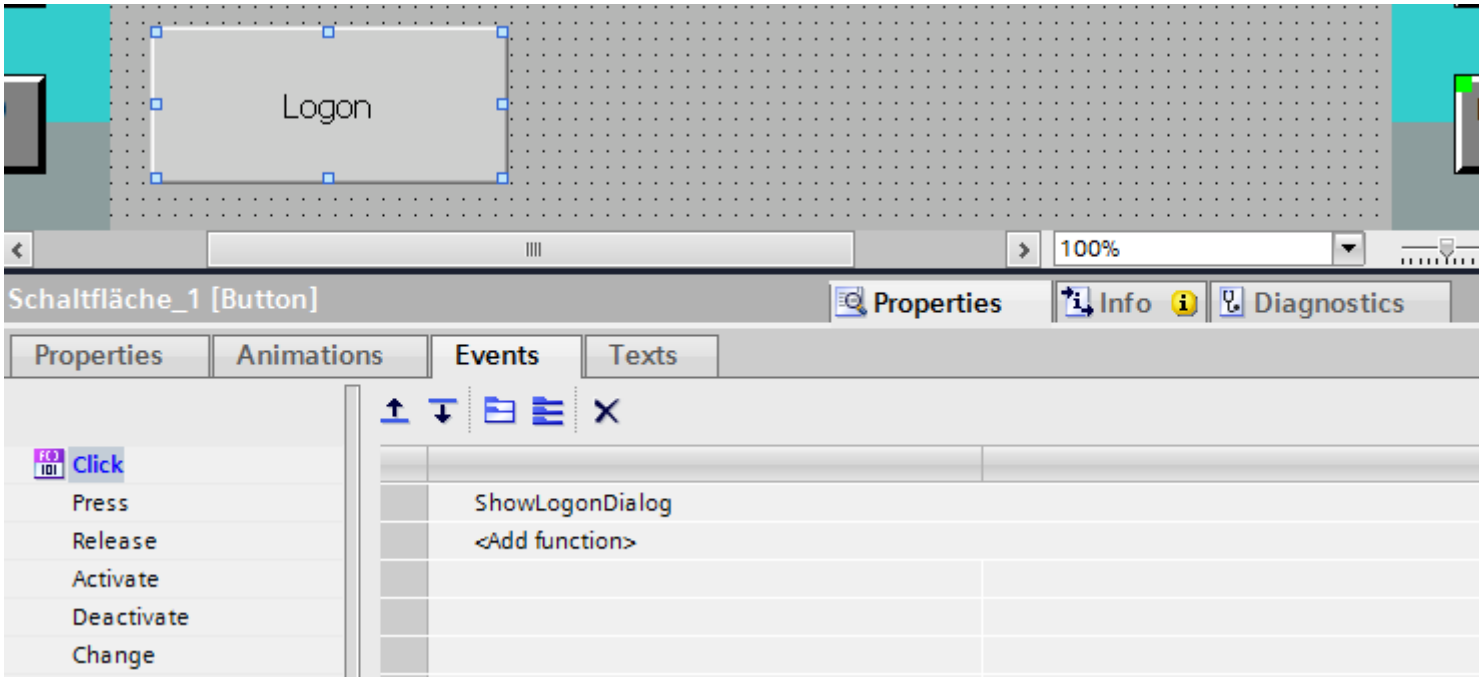
In runtime the logon dialog box is not displayed by default until you access a protected object. Either no user is logged on or the logged-on user does not have the required authorization.

Requirements

- A screen has been created.
- A button has been created in the screen.

Procedure

1. Click the button in the screen.
2. Click "Properties > Events > Release" in the Inspector window.
3. Click the entry "Add function" in the "Function list" table.
4. Select the system function "ShowLogonDialog" from the "User administration" group.



Result

If the user clicks on the button in runtime, the function "ShowLogonDialog" is called up. When the function "ShowLogonDialog" is called up, the logon dialog box is displayed. The user logs on with his user name and password.

12.7.6.2 Example: Logging the logon and logoff events

Task

In the following example, you configure the function "TraceUserChange" to the event "User change".

Principle

The "TraceUserChange" function is called when a user logs on or off. When a function is called up, a system message with information about the corresponding user is output.

This system message can be archived. When archiving, the system message is provided with a date stamp and time stamp. This ensures that you can track which user was logged on at the HMI device at which time and for how long.

Requirements

- You have created an HMI device with Runtime Advanced.
- The Inspector window is open.

Procedure

1. Double-click the "Scheduler" in the Project view.
2. Double-click "Add" in the table of the tasks.
3. Enter "Logon-Protocol" as the "Name".
4. Select "User change" as the "Trigger".
5. Open "Properties > Events" in the Inspector window.
6. Click the entry "Add function" in the "Function list" table.
7. Select the "TraceUserChange" system function.

Result

A system message is output when a user logs on or logs off.

12.7.6.3 Example of a user administration

Example: Structure of user administration

Task

In the following example you set up a user administration for different users and user groups. The example orientates itself to a typical requirement profile from manufacturing engineering.

Principle

Completely different groups of persons are involved in a plant or project. Each group of persons protects its respective data and functions against access by others. For this purpose, users are created and assigned to a user group.

You can reproduce different views through user groups.

Example:

- Organizational view: Commissioners, Operators, Shift I, Shift II
- Technological view: Axis control, Tool changers, Plant North, Plant South

The following example orientates itself to the organizational view.

Every user group has characteristic requirements regarding access protection: A user group has operating authorizations for specific application cases. A programmer changes, for example, recipe data records.

In the example the users Miller, Group Smith and Foreman are created and assigned to different user groups.

Ms. Miller works as a programmer in the engineering system. The Group Smith are commissioners. Mr. Foreman is an operator.

Requirements

- A new project has been created.
- The "User administration" editor is open.

Procedures overview

Working with user administration has the following procedure in the example:

1. Creating authorizations The planner specifies which authorizations are required for access protection.
2. Configuring authorizations: The planner specifies which objects may be operated and which functions may be executed.
3. Creating user groups and allocating authorizations: The administrator creates the user groups together with the planner. The planner uses the authorizations to specify who may operate objects and change parameters.
4. Creating users and assigning them to a user group: The administrator administers the users.

Result

The aim is the following structure of the user administration of users, user groups and authorizations:

Users			User groups	Authorizations			
Miller	Smith	Foreman	Roles	Changing recipe records	Changing system parameters	Changing process parameters	Managing
			Administrator group				x
X			Programmer	X			
	X		Commissioning engineers	X	X	X	
		X	Operators	x			

The user "Foreman" who belongs to the "Operators" user group has access to the configured "To Recipe view" button.

Note

Alternatively, you can create several users as operators with different operating authorizations, for example, Operator Level 1, Operator Level 2.

Example: Creating and configuring authorizations

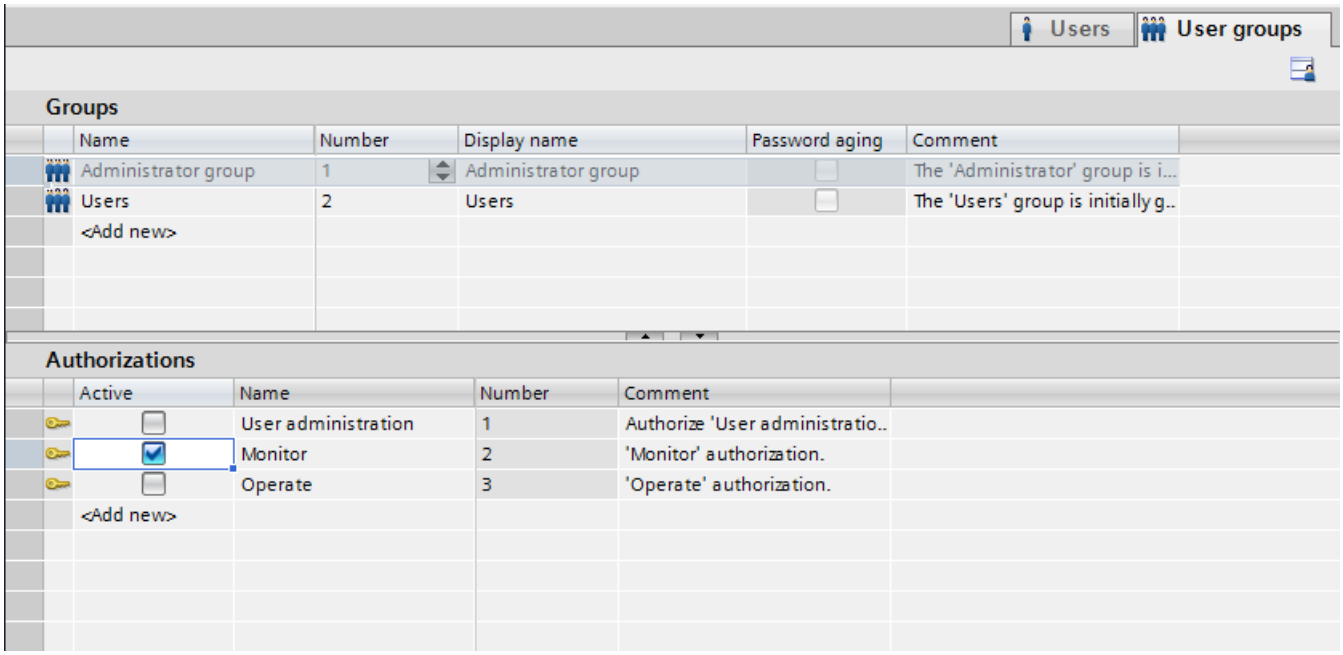
Task

The following example shows you how to create the authorizations

Procedure

1. Open the "User groups" work area.
2. Double-click "Add" in the "Authorizations" table.
3. Enter "Change recipe data records" as the "Name" of the authorization.
4. Repeat steps 2 and 3 to create additional authorizations: "Change system parameters", "Change process parameters".

Result



Example: Configuring a button with access protection

Task

In the following example you use a system function to create a button for a screen change. You protect the "To Recipe view" button against unauthorized operation. To do so, you configure the "Change recipe data records" authorization at the "To Recipe view" button.

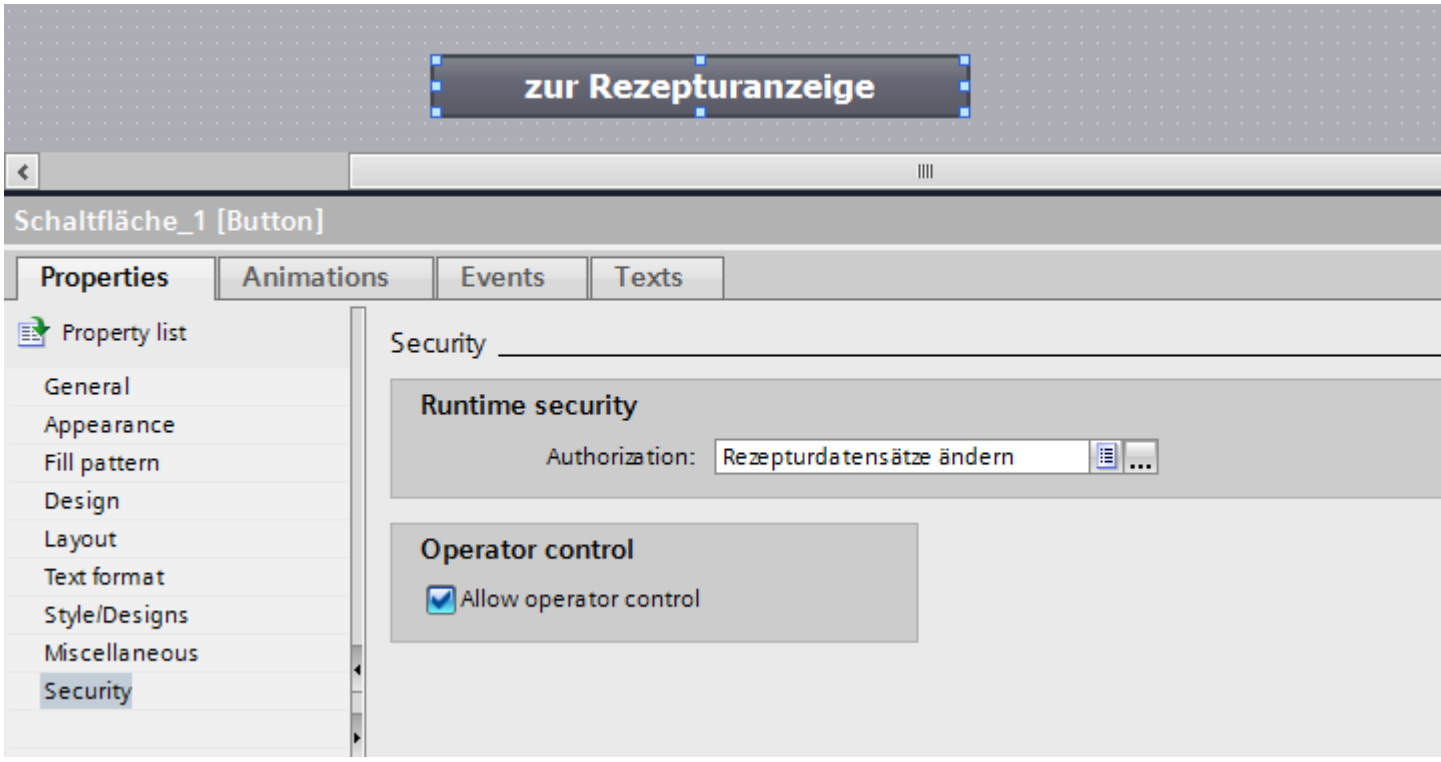
Requirements

- A "Change recipe data records" authorization has been created.
- A "Recipes" screen has been created.
- A "Start" screen has been created and opened.
- A button has been created and marked in the "Start" screen.

Procedure

1. Click "Properties > Properties > General" in the Inspector window.
2. Enter "To Recipe view" as the text.
3. Click "Properties > Events > Click" in the Inspector window.
4. Click the "Add function" entry in the first line of the "Function list" table.
5. Select the "ActivateScreen" system function in the "Screens" group.
6. Click the ... button in the "Screen name" field. A dialog box for selecting the screen opens.
7. Select the "Recipes" screen and use the button to close the dialog box.
8. Click "Properties > Properties > Security" in the Inspector window.
9. Select "Change recipe data records" as the "Authorization."

Result



Access to the "To Recipe view" button is protected. If, for example, the user "Smith" clicks the button in Runtime, the function "Recipe view" screen is called up. Prerequisite is that the user "Smith" has logged on correctly and has the required authorization. The "Recipes" screen contains a recipe view and other screen objects.

If the logged-on user does not have the required authorization or if no user is logged on, the "Logon dialog box" is displayed.

Example: Creating user groups and assigning authorizations:

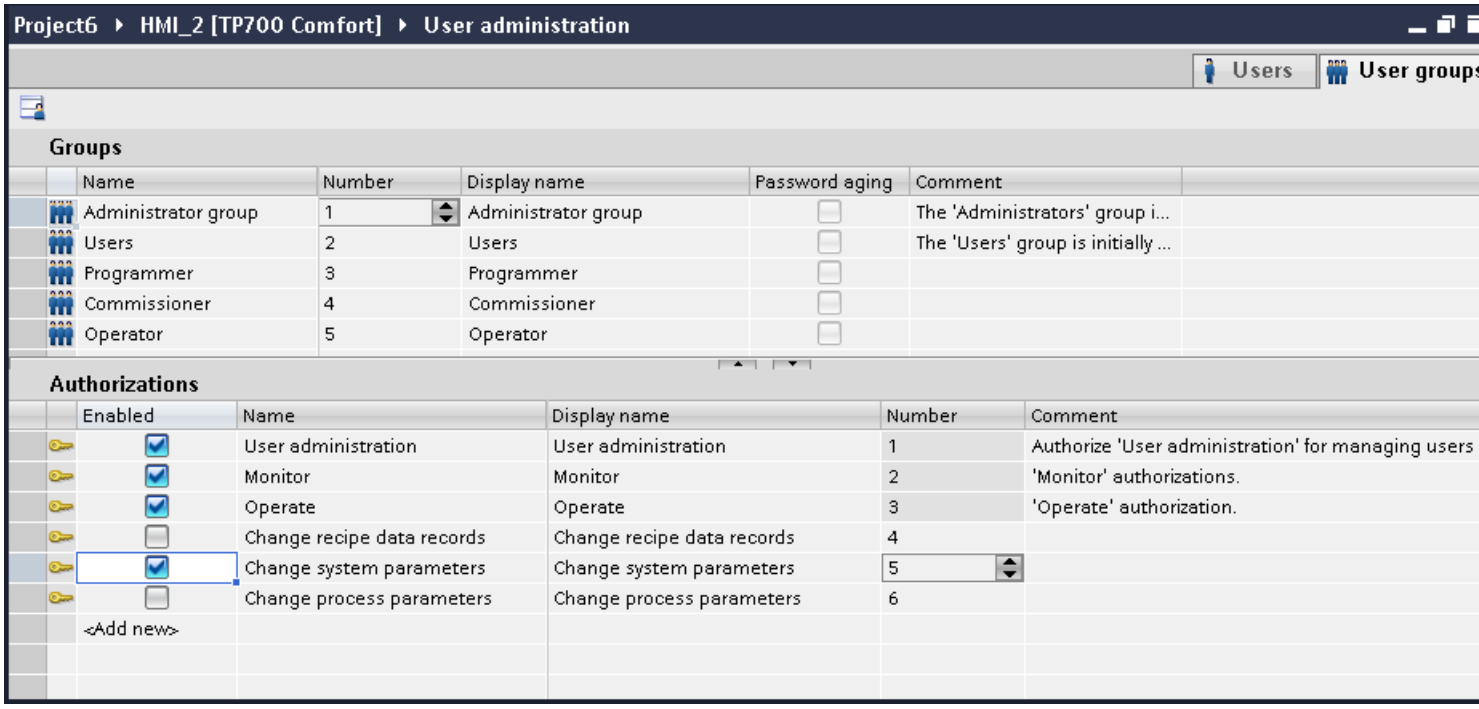
Task

In the following example you create the user groups and assign authorizations to them.

Procedure

1. Open the "User groups" work area.
2. Double-click "Add" in the "Groups" table.
3. Enter "Programmer" as the "Name".
4. Repeat steps 2 and 3 to create the "Commissioner" and "Operator" user groups.
5. Click "Administrator" in the "Groups" table.
6. Activate the "Change system parameters" authorization in the "Authorizations" table.

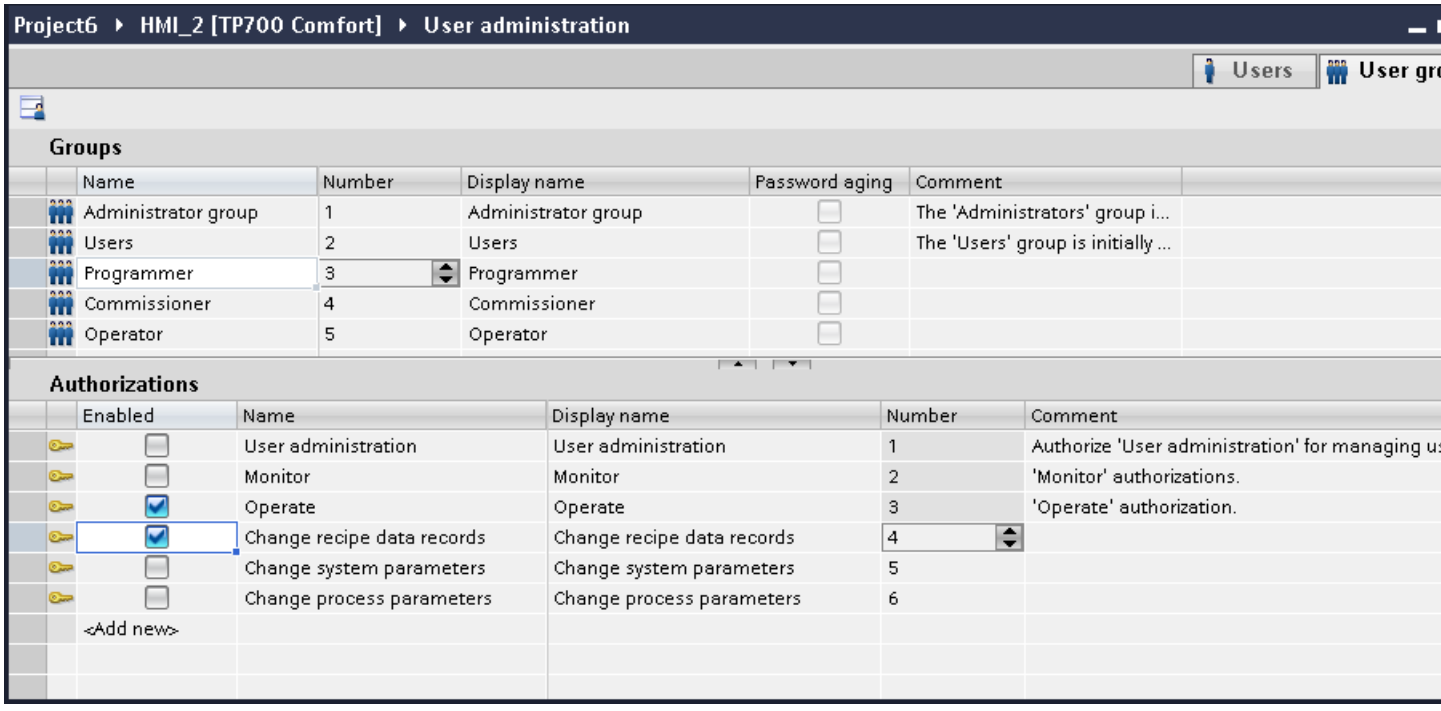
Interim result



Procedure

1. Click "Operator" in the "Groups" table.
2. Activate the "Change recipe data records" authorization in the "Authorizations" table.
3. Click "Commissioner" in the "Groups" table.
4. Activate the authorizations "Change recipe data records", "Change system parameters" and "Change process parameters" in the "Authorizations" table.
5. Click "Programmer" in the "Groups" table.
6. Activate the "Change recipe data records" authorization in the "Authorizations" table.

Result



Example: Creating users and assigning them to a user group

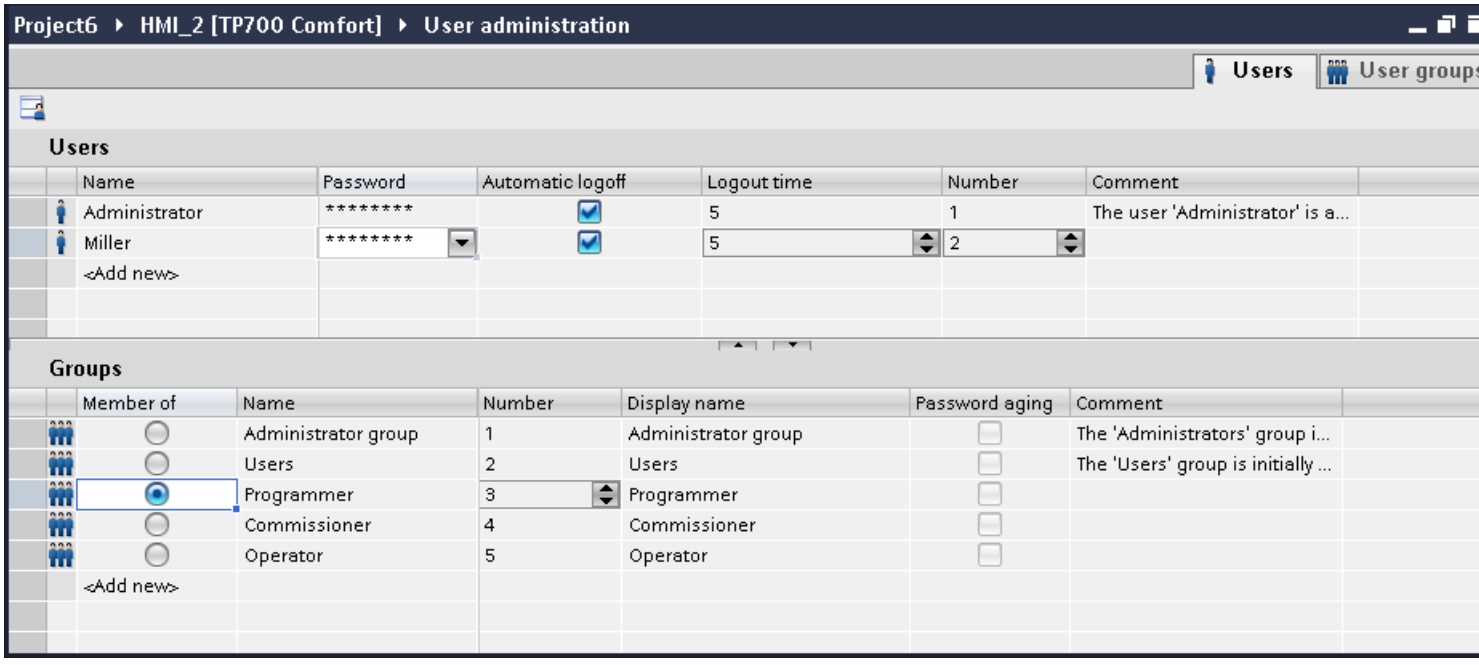
Task

In the following example you create the users and assign user groups to them. The users are sorted alphabetically immediately after the name has been entered.

Procedure

1. Open the "Users" work area.
2. Double-click "Add" in the "Users" table.
3. Enter "Miller" as the user name.
4. Click the button in the "Password" column. The dialog box for entering the password is displayed.
5. Enter "miller" as the password.
6. To confirm the password enter it a second time in the lower field.
7. Close the dialog box by using the icon.
8. Activate the "Programmer" user group in the "Groups" table.

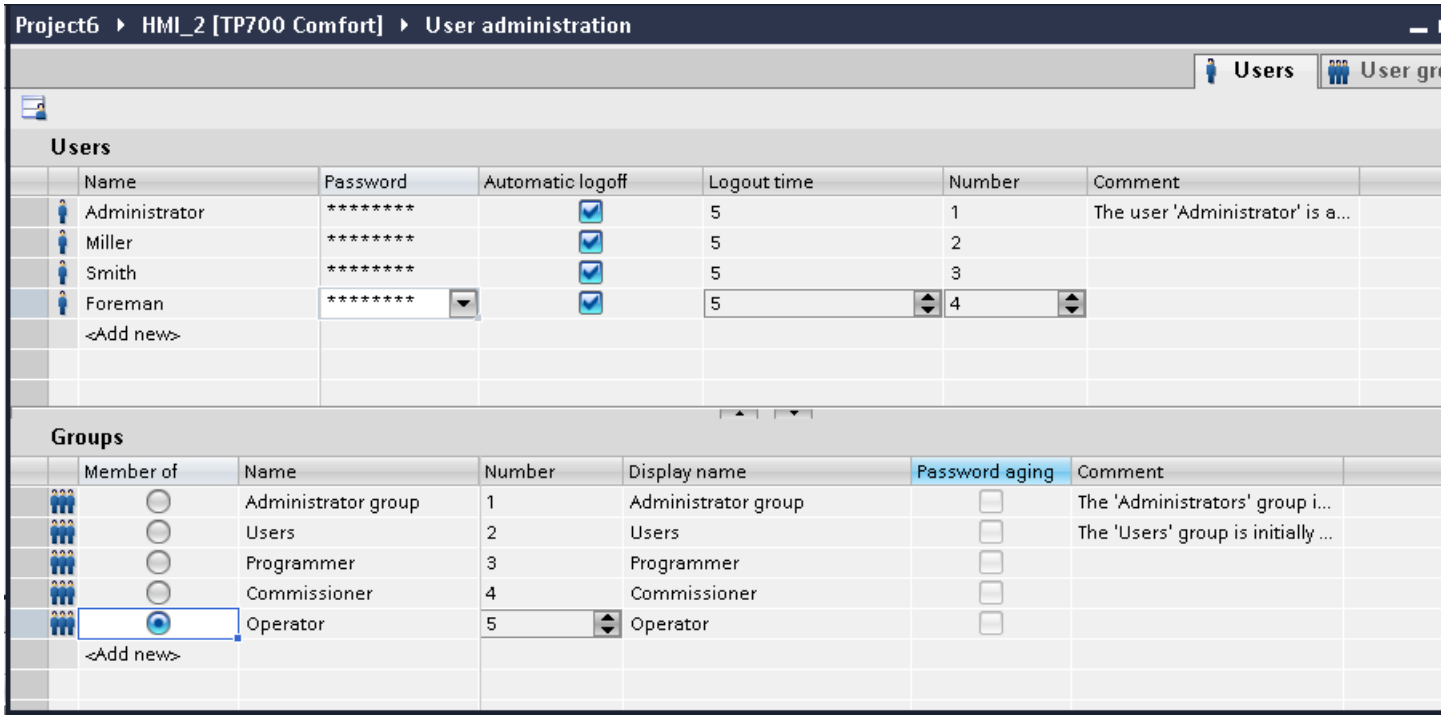
Interim result



Procedure

1. Double-click "Add" in the "Users" table.
2. Enter "Smith" as the user name.
3. Click the button in the "Password" column. The dialog box for entering the password is displayed.
4. Enter "smith" as the password.
5. To confirm the password enter it a second time in the lower field.
6. Close the dialog box by using the icon.
7. Activate the "Commissioner" user group in the "Groups" table.
8. Repeat steps 2 to 6 for the user "Foreman."
9. Activate the "Operators" user group in the "Groups" table.

Result



12.8 Working with system functions and Runtime scripting

12.8.1 Basics

12.8.1.1 Runtime scripting

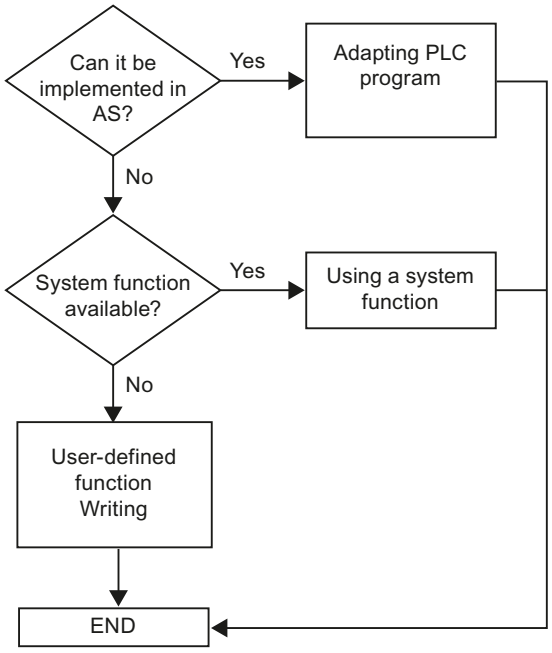
Term definition

The following terms are used in the documentation:

Term	
Runtime scripting	Generic term for all activities in user-defined functions.
Function	Generic term for system functions and user-defined functions.
System functions	System functions are all functions supplied with WinCC. The system functions are used either in function lists or user-defined functions.
User-defined functions	User-defined functions are functions written in the "Scripts" editor. For more precise specification, the term "User-defined VB function" is used in the documentation.
VBS / VBScript	Abbreviation for Visual Basic Script

Applying Runtime scripting

The following figure serves as a decision guide for programming tasks at hand:



12.8.1.2 System functions

Applications

You use system functions for the following:

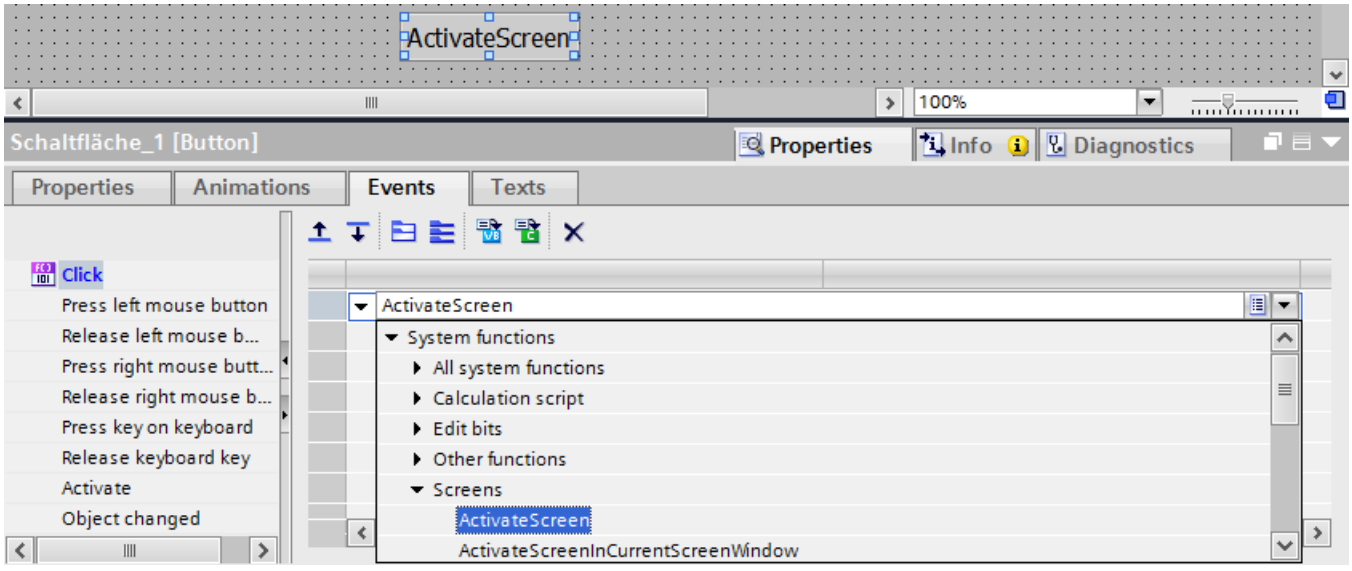
- Function lists
  A function list is executed from top to bottom in Runtime. If a function list includes system functions with longer runtime, these are processed asynchronously. For additional information, refer to "Runtime characteristics".
- User-defined functions
  If the HMI device supports customized functions, you use system functions in combination with instructions and conditions in the code of a customized function. In this way you execute a customized function depending on a specific system state. In addition, you can evaluate return values of system functions, for example. Depending on the return value, you then perform test functions, for example, that in turn affect the user-defined function flow.

Application examples of system functions

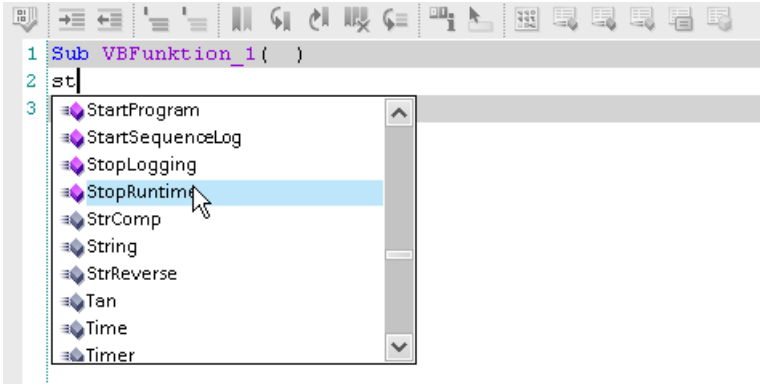
- Calculations, increase tag value by a defined or variable amount.
- Log function, e.g. to start a process value log.
- Settings, e.g. to switch the PLC or set a bit in the PLC.
- Alarms, e.g after changing the user.

Application

- Function list
  When configuring a function list, you select the system functions from a selection list that is sorted by categories:



- User-defined function
  If you are using a system function in a user-defined function, you choose it from a selection list. To open the selection list, use the key shortcut <Ctrl+Space> or click .



Language dependency

In the function list, the names of the system functions are dependent on the set project language. The functionality can then be recognized immediately by the project planner.

You use the English name of the system function in user-defined functions. You will find the English name of the system function in the reference.

Availability

Which system functions are available depends on the selected HMI device.

HMI device replacement

If you use system functions in a function list that are not available on the set HMI device, these system functions are marked in color.

If you use a system function in a user-defined function which is not available on the set HMI device, a warning is issued. In addition, the respective system function will be underlined with a wavy blue line.

See also

System functions for logs (Page 4259)

12.8.1.3 User-defined functions

Applications

You use customized functions for the following for example: :

- Configuring an extended function list
  In a customized function you have the option to have customized functions and system functions executed depending on conditions or executed repeatedly. You then add the customized function to a function list.
- Programming new functions
  Customized functions are only valid in the project in which they are defined. For user-defined functions you define transfer parameters and return values, for example, for conversion of values.

Properties of customized functions

A customized function has the following properties:

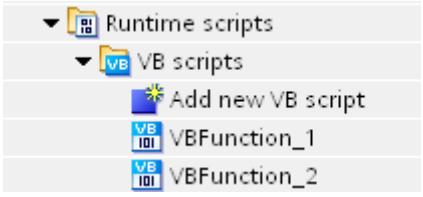
- Name
- Parameter (optional)
- Return value and type (optional)
- Can be modified centrally
- has no trigger. You have to call the customoized function explicitly, e.g. in a function list.
- behaves like a system function

Usage

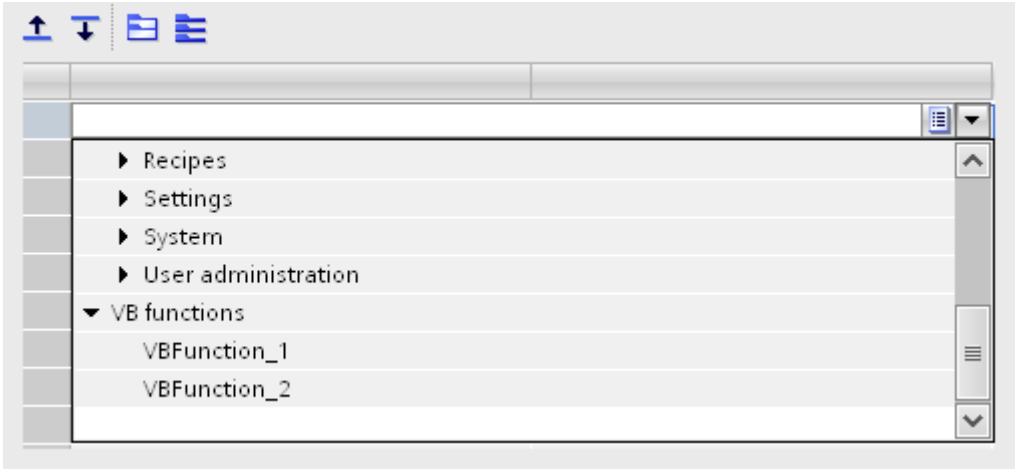
You create customized functions with the "Scripts" editor. See "Scripts" editor (Page 4578) for additional information.

Customized functions are saved in the project. To protect user-defined functions, set up know-how protection. See Auto-Hotspot for additional information.

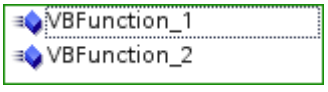
- Project tree
  The user-defined functions are listed in the project tree under "VB scripts".



- Function list
  The user-defined functions are listed in a function list under "VB functions".



- User-defined function
  If you are using a system function in a user-defined function, you choose it from a selection list. To open the selection list, use the key shortcut <Ctrl+Space> or click .



Recursion level

The recursion level in user-defined functions is limited by the stack size of the HMI device. If the number is exceeded, a system error message is output in Runtime. This means you should limit the number of recursions in a user-defined function.

12.8.2 Working with function lists

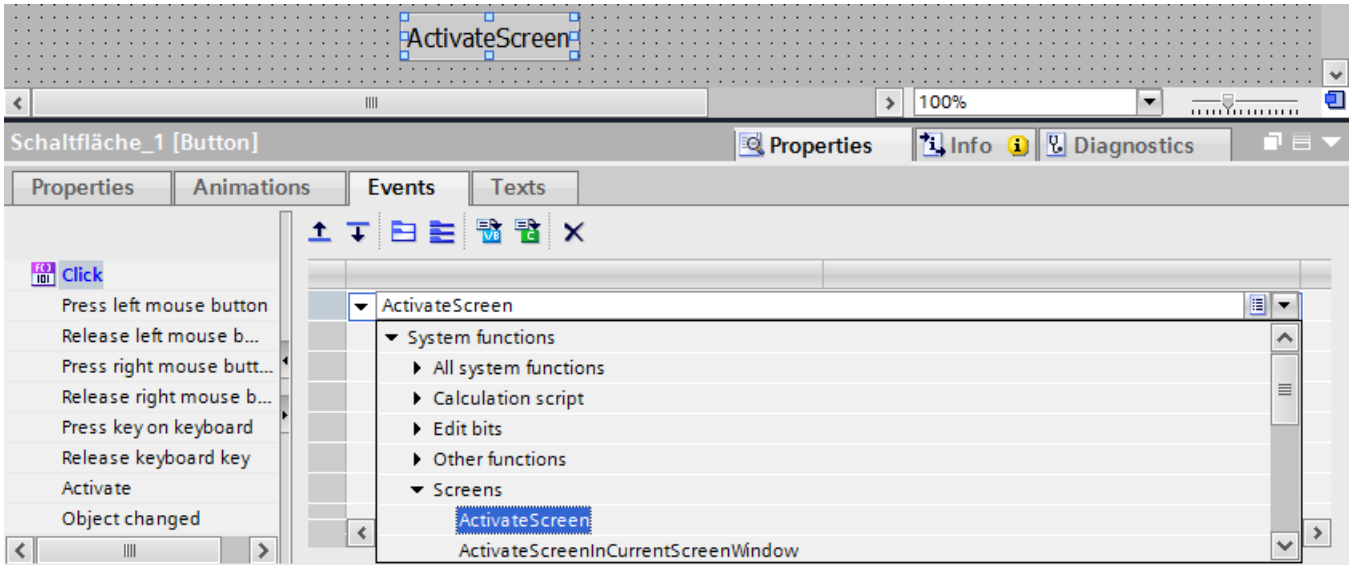
12.8.2.1 Basics of the function list

Introduction

A function list performs one or more system functions and user-defined functions when the configured event occurs.

Principle

The function list is configured for an event of an object, for example, for a screen object or a task. You can configure a function list precisely on every event. The events which are available depend on the selected object and the HMI device.



Events occur only when the project is in Runtime. Events include:

- Execution of a task
- Pressing of a button
- Acknowledging an alarm

Note

The choice of configurable system functions in a function list is dependent on the selected HMI device. For additional information, refer to "Availability for specific devices of system functions".

12.8.2.2 Properties of a function list

Status information

During configuration the project data is tested in the background.

With the following causes the function list is not executed in Runtime and the incorrect entries are marked red:

- At least one system function or one user-defined function is not completely supplied with parameters.
- At least one system function or one user-defined function is contained which is not supported by the selected HMI device, e.g. by changing the device type.

Processing of system functions and user-defined functions (WinCC Runtime Advanced and Panels)

A function list is executed from top to bottom in Runtime. If a function list includes system functions with longer runtime, these are processed asynchronously. You can find additional information about this in the sections "Calling system functions" and "Calling user-defined functions".

See also

Executing a function list in Runtime (Page 4600)

Processing sequence for user-defined functions and system functions (Page 4602)

12.8.2.3 Configuring a function list

Introduction

You configure a function list by selecting the system functions and customized functions from a drop-down list. The system functions are arranged in the selection list according to categories.

You only use customized functions of a programming language in a function list. Your selection of the first user-defined function determines whether user-defined VB functions or user-defined C functions can be selected in the function list. Which programming languages are available depends on the selected HMI device.

If you have created a user-defined VB function, for example, the user-defined function is available under the entry "VB functions".

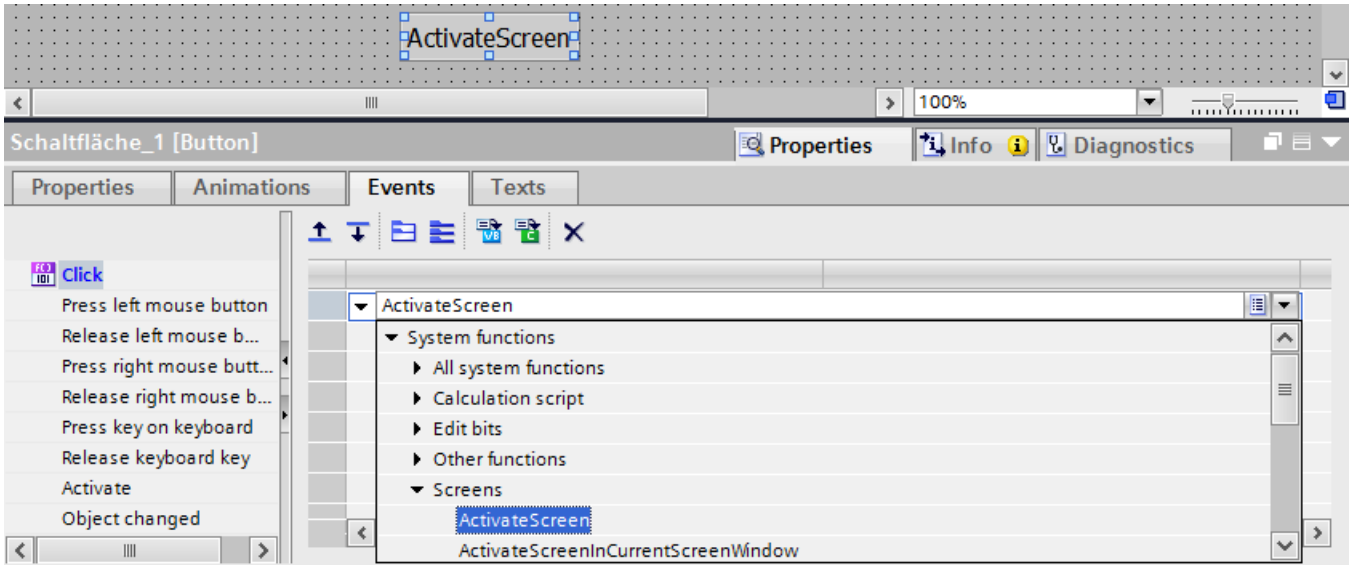
Requirement

Object has at least one configurable event.

Procedure

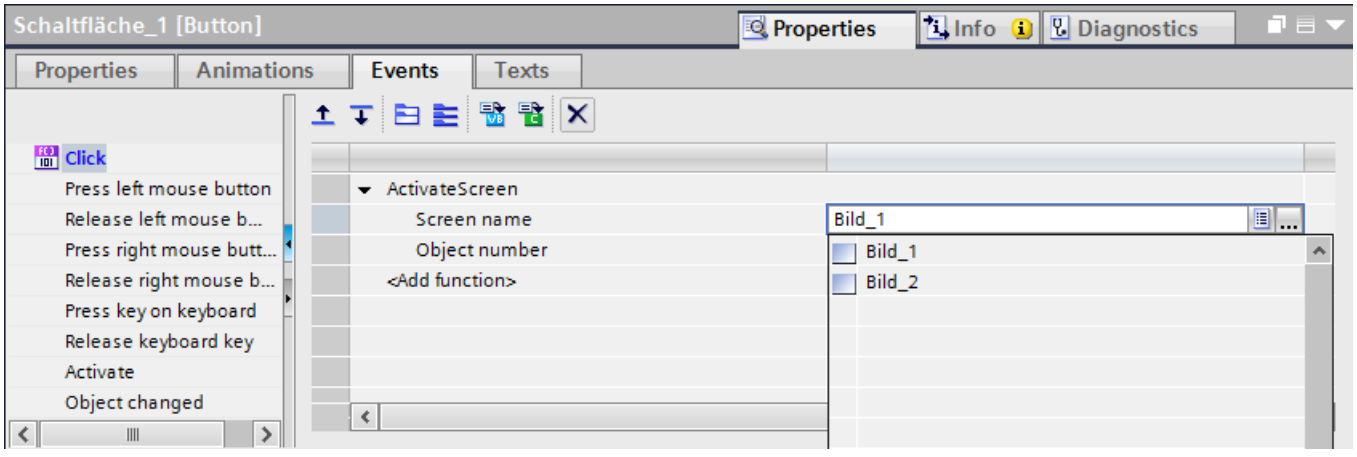
Proceed as follows to configure a function list:

1. Open the editor in which the object is located.
2. Select the object.
3. Click "Properties > Events" in the Inspector window. Choose the event for which you want to configure the function list.
4. Mark the "<Add function>" entry in the inspector window in the drop-down list.
5. Select the desired system function or the desired user-defined function from the selection list. You can also enter the name of the system function or the user-defined function.



The system function or the customized function is entered in the function list.

6. If the system function or the customized function has parameters, select the appropriate values for the parameters.



7. If you want to add other system functions or user-defined functions to the function list, repeat steps 4) to 6).

Result

The function list is configured. In addition, to the configured event, the status of the function list is displayed in the Inspector window. When the configured event occurs in Runtime, the function list is completed from top to bottom.

12.8.2.4 Editing a function list

Introduction

A function list can be edited as follows:

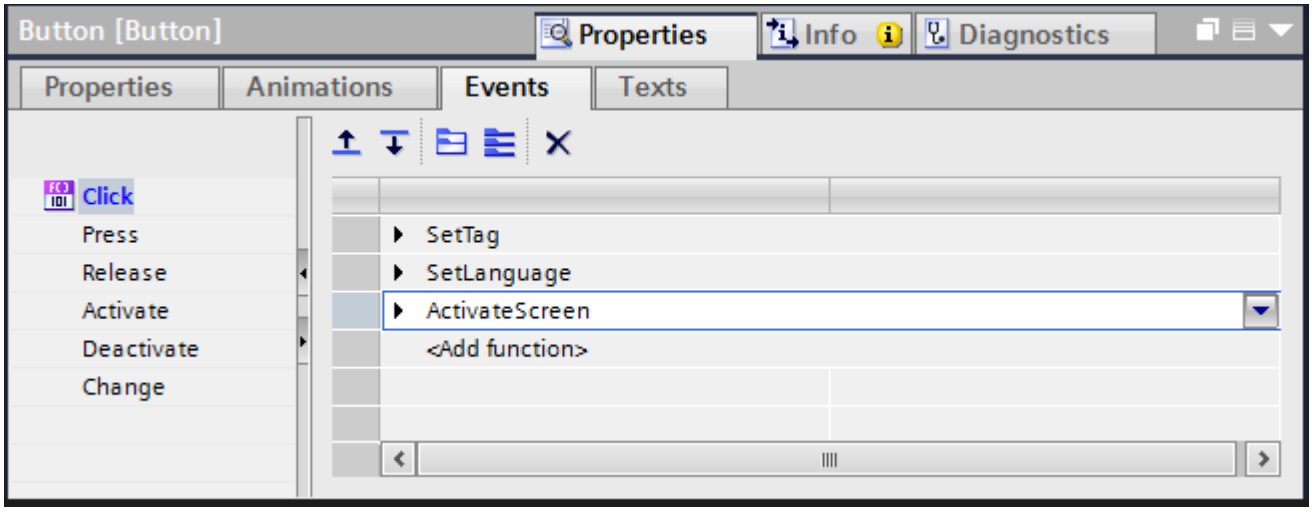
- Change the order of system functions and user-defined functions.
- Remove system functions or user-defined functions.

Requirement

The function list is configured and opened.

Changing the order of a system function or a customized function

1. Select the desired system function or customized function in the function list.
2. Then click the appropriate direction arrow in the inspector window until the system function or customized function is in the desired position.



Changing the order of several system functions and customized functions

1. Hold down the <Shift> key.
2. Click desired system functions or customized functions one after another with the mouse.
3. Move the selection to the desired position by drag&drop.

Removing a system function or customized function

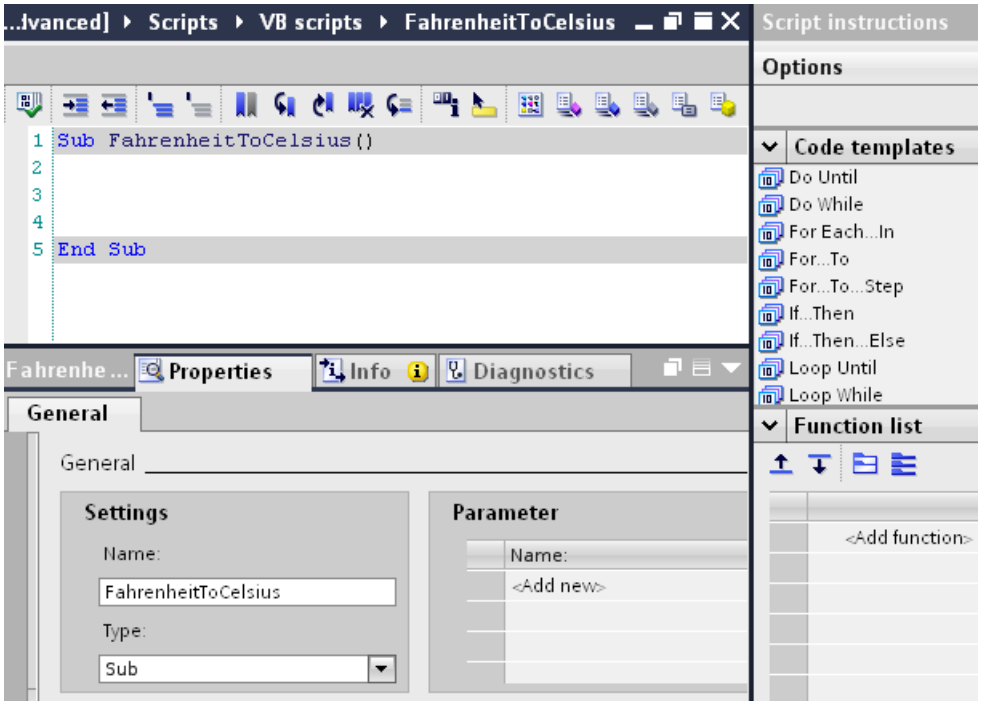
1. Select the desired system function or customized function in the function list.
2. Select "Delete" in the shortcut menu.

12.8.3 Working with user-defined VB functions

12.8.3.1 "Scripts" editor

Introduction

The "Scripts" editor supports you in the creating of user-defined functions with functionalities such as auto-completion and syntax highlighting. You can insert code templates for frequently used instructions with the "Instructions" task card for example.



Autocompletion and parameter entry

The "Scripts" editor provides you context-dependent programming support with autocompletion parameter entry support.

When you use system functions, the parameters of the function are shown as a tooltip.

<Ctrl+J>

The object list can be called context-specifically by using the shortcut <Ctrl+J>. Use the object list to supply values to system functions, methods, and properties: For example, you select screens, screen objects, tags, or colors. You transfer a selected object from the object list into the code by double-clicking it.

Note

Tags and screens can also be inserted using a drag-and-drop operation. Use, for example, drag-and-drop to assign values to parameters.

<Ctrl+Space>

With the key combination <Ctrl+Space> you call a list with the following contents:

- Constants and functions of the programming language used in the "Scripts" editor.
- User-defined functions
- System functions

"Instructions" task card

The "Instructions" task card contains the "Code templates" and "Function list" palettes.

- Code templates
  To insert frequently used instructions, double click the desired instruction in the "Code templates" palette. Replace the space holders with the used expressions.
- Function list
  In the function list you select the system functions and user-defined functions from a drop-down list. Proceed as for configuring a function list. To transfer the function list to the code, click the "Transfer" button. The function list is automatically converted into the correct syntax.

  Note

  Not all system functions are available in the "Scripts" editor. For additional information, refer to the reference.

Edit functions

Use the buttons of the toolbar "Advanced Edit" to make the code more readable by means of indentation and the use of comments.

Bookmarks help you keep an overview, even of many lines of code.

Further information regarding each button can be obtained in the accompanying direct help.

Synchronization of tags and objects

If you change tag names or object names in the project which you use in a user-defined function, the names are adapted automatically in the user-defined function.

If you change the tag name or object name in the function, the tag names or object names are not adapted in the project. An appropriate error message is output when executing the user-defined function.

Syntax highlighting

Keywords in the "Scripts" editor are highlighted using different colors. Unknown words are underlined with a red wavy line.

The table shows the preset colors for the most important keywords:

Color	Meaning	Example
Blue	Keyword	If, Then
Dark violet	Function	FahrenheitToCelsius
Chocolate	System function	SetTag
Orange	HMI tag	Tag_1
Green	Comment	'This is a comment'

Adapting display properties

When you have opened a user-defined function, you can customize the display properties in the editor. Select the "Options > Settings" menu command. Click "General > Text editors" in the "Settings" editor. For example, you can change the colors for syntax highlighting or adapt the indents for code in the work area.

See also

Create customized VB functions (Page 4585)

12.8.3.2 Access to HMI tags

Introduction

In user-defined VB functions, you have access to HMI tags which you have created in the project. Change or read the tag value with a user-defined VB function in Runtime.

In addition, you can create local tags as counters or buffers in the user-defined VB function. You must declare each tag to avoid errors due to misspelled tags.

HMI tags

The user-defined VB function accesses the tag value which is saved in Runtime memory. Next, the tag value will be updated to the set cycle time. The user-defined VB function first accesses the tag values which were read at the previous cycle time.

If the tag name matches the VBS name conventions in the project, you can use the HMI tag directly in the function.

```
'VBS_Example_03
```

```
If BeltDriveOilTemperature > 100 Then [instruction]
```

If the tag name in the project does not match the VBS name conventions, you must reference the HMI tag using the "SmartTags list". In the following example the tag name contains the "&" character that is not permitted in accordance with the VBS name conventions:

```
'VBS_Example_04
SmartTags("Test&Trial")= 2005
```

Local tags

You define local tags with the Dim instruction. You can only use local tags within user-defined VB functions. They therefore do not appear in the "HMI Tags" editor. You use a local tag in the user-defined VB function, for example, as a counter in a For instruction.

```
'VBS_Example_05
Dim intCount
For intCount = 1 To 10 [instruction] Next
```

Access to HMI tags with a dynamic name

The user-defined VB function accesses the tag value using the tag name. You can specify the tag name in such a way that the tag name is composed at the runtime of the user-defined function.

If you only call the user-defined VB function in a screen in which the HMI tag is also used elsewhere, for example in an I/O field, you should for performance reasons configure the HMI tag with "Cyclic in operation" acquisition mode.

If the user-defined VB function is accessed and the HMI tag is not being used in the screen currently displayed, configure the HMI tag with "Cyclic continuous" acquisition mode. This ensures that the current value of the tag is always available.

See also

SmartTags (Page 4981)

12.8.3.3 Access to objects

Introduction

The objects of the VBS object model with the appropriate properties and methods are available in user-defined VB functions.

The object properties can be read and changed in Runtime.

Referencing objects

In customized VB functions you reference the objects by the corresponding list. To identify the object, use its name or the position number within the list.

If you access the properties of an object more often, create an object reference. You can access object properties with and without object reference.

With the following instruction the first object is referenced in the "MainScreen":

```
'VBS_Example_01
Dim objObject
'Change to Screen "MainScreen"
HMIRuntime.BaseScreenName = "MainScreen"
Set objObject = HMIRuntime.Screens(1).ScreenItems(1)
```

With the following instruction an object is referenced by its name and an object property is changed. You must have created the object with this name in the screen.

```
'VBS_Example_02
Dim objCircle
HMIRuntime.BaseScreenName = "MainScreen"
Set objCircle = HMIRuntime.Screens(1).ScreenItems("Circle_01")
objCircle.BackColor = vbGreen
```

12.8.3.4 Calling system functions

Introduction

You can insert system functions in a user-defined VB function.

This can be done in the following ways:

- With <Ctrl+Space> or with
- Direct input
- With the "Function list" palette

With <Ctrl+Space> or with

Open the list of system functions with <Ctrl+Space> or with and select the desired system function.

Direct input

You enter the system functions directly into the code. You use the English name of the system function. You can find the English name of the system function in the system function reference under "Syntax". The editing language setting is not taken into consideration.

References to objects, e.g. screens, connections and logs are transferred in inverted commas:

Calling system functions without return value in VBS

```
SetTag "Tag1",64
```

Calling system functions with return value in VBS

```
InverseLinearScaling "XValue","Tag1", 0, 1
```

With the "Function list" palette

Select the system functions from a selection list that is sorted by categories in the "Function list" palette. Proceed as for configuring a function list.

To transfer the list to the code, click the "Transfer" button. The list is automatically converted into the correct syntax.

HMI device replacement

The code of a customized function depends on the selected HMI device. If you use system functions in the customized function which are not supported by the selected HMI device, you receive an error message in the output window. You will find additional information under "Device dependency of system functions".

See also

Configuring a function list (Page 4575)

12.8.3.5 Calling user-defined VB functions

Introduction

You can only insert user-defined VB functions in a user-defined VB function.

This can be done in the following ways:

- With <Ctrl+Space> or with
- Direct input
- With the "Function list" palette

With <Ctrl+Space> or with

Open the list of user-defined VB functions with <Ctrl+Space> or with and select the desired user-defined VB function.

Direct input

You enter the user-defined VB function directly into the code. References to objects, e.g. screens, connections and logs are transferred in inverted commas.

```
Function Average(ByVal Var1, ByVal Var2)
  Average = (Var1+Var2)*(1/2)
End Function
```

Calling user-defined VB functions without return value in VBS

```
Average 4,10
```

Calling user-defined VB functions with return value in VBS

```
Dim ValueA
ValueA = Average (4,10)
```

If you do not want to evaluate the return value, use the call as for a user-defined VB function without return value.

With the "Function list" palette

Select the desired user-defined VB function from a selection list in the "Function list" palette.

Proceed as for configuring a function list. You will find additional information under "Configuring a function list".

To transfer the list to the code, click the "Transfer" button. The list is automatically converted into the correct syntax.

12.8.3.6 Transfer and return of values in VBS

Transfer of a value

The following options are available for transferring parameters.

- "Call by Value" - ByVal
  ByVal transfers the parameter value. If you transfer a tag as a parameter, the tag value is transferred to the user-defined function when the user-defined function is executed.
- "Call by Reference" - ByRef
  ByRef transfers the address of the parameter. If you transfer a tag as a parameter, the tag address is transferred to the user-defined function when the user-defined function is executed.
  When user-defined functions and system functions are called in user-defined functions, the parameter is transferred as ByRef.

Return of a value

Return values can return the result of a calculation (e.g., average value of two numbers). But a return value can also give information about whether an instruction was executed correctly.

Therefore, the system functions that perform file operations such as "Delete" also have return values.

For a user-defined function to return a value, you must set the "Function" type. You assign the function name to the return value in the user-defined function:

```
1 Function Average( Value1 , Value2 )
2 'Check if Parameters are not numeric:
3 If IsNumeric (Value1) = False Then Value1 = 1
4 If IsNumeric (Value2) = False Then Value2 = 1
5 Average = (Value1+Value2)/2
6
7 End Function
```

To form an average value of two numbers, call the user-defined VB function Average and transfer the result to the HMI tag for example AverageValue.

- In a customized VB function

```
SmartTags("AverageValue") = Average ("4", "10")
```

- In the function list

Average	
Return value	AverageValue
Value1	10
Value2	4

You can output the value of the HMI tag AverageValue in an I/O field.

See also

Create customized VB functions (Page 4585)

12.8.3.7 Create customized VB functions

Introduction

When you create a customized VB function, you define the following settings:

- The name with which you call the user-defined VB function.
- The type of user-defined VB function.
- The parameters which are transferred to the user-defined VB function in Runtime.

Use only the following characters for the name:

- "A" to "Z"
- "a" to "z"
- "0" to "9"
- "_" as a separator

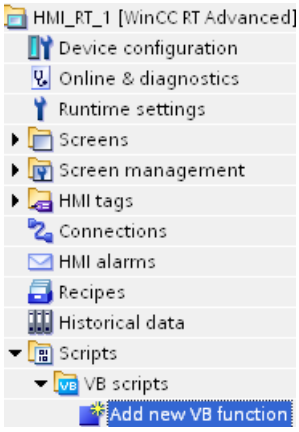
Note

When you change the parameters or the function type, the change is color marked at the application point, e.g. in the function list.

Procedure

To create a user-defined VB function, follow these steps:

1. Open "Scripts" in the project tree.
2. Double click "Add new VB function" in the project tree under "VB scripts".



The user-defined VB function is created as a new tab in the work area. The input mask for the configuration settings of the user-defined VB function opens in the Inspector window.

3. Configure the user-defined VB function in the Inspector window "Properties > General > General".
4. Enter a descriptive "Name" for the user-defined VB function.
5. Select the "Type" of the user-defined VB function.
   - "Function" have a return value.
   - "Sub" have no return value.

6. Click "<Add>" in the "Parameters" list to add parameters. Enter the parameter name and specify the parameter type.
   Further information can be found in "Transfer and return of values in VBS (Page 4584)".
   - "ByVal" transfers the parameter value.
   - "ByRef" transfers the address of the parameter.
7. Enter its code in the work area.
   You will find further information about this in ""Scripts" Editor" (Page 4578)".

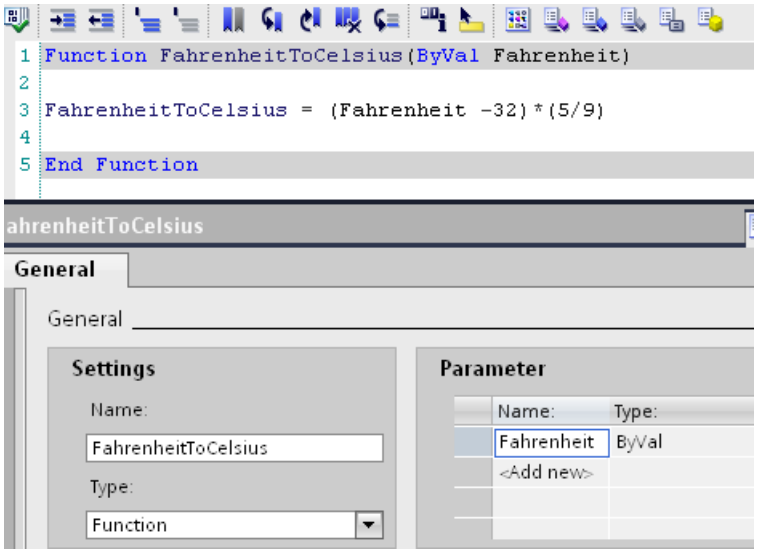
Result

The user-defined VB function has been created. The type, name, and function parameters are displayed in the title bar.

The start and end are inserted automatically when the user-defined function is created.

If you have protected the user-defined VB functions by a password, it is saved encrypted in the project. For additional information, refer to "Protecting user-defined functions".

The figure below shows a user-defined VB function which converts the temperature from "Fahrenheit" to "Degrees Celsius":



12.8.3.8 Testing the syntax of customized functions

Introduction

During programming, the code is tested in the background. Syntactical errors are marked with a wavy red line. The customized function is checked for correct syntax and valid object references.

Check the syntax in the "Scripts" editor to identify all the errors in the code and output the appropriate error messages. In this case, error messages generated by the script parser are output.

Use a separate debugger to check customized VB functions for logical programming errors.

Error types

The following error types are detected in the syntax test and output in the "output window":

- Unknown string (e.g., not a keyword)
- Assignment of value to a constant
- The name of a local tag created with the Dim instruction is already known, e.g.: as a parameter, object, or component of the object model
- Syntax error

Requirements

- Customized function is open.

Procedure

Proceed as follows to check the syntax:

1. Click to start the syntax check.
   The syntax is checked. The result of the syntax check is displayed in the Inspector window under "Info > Compilation." Syntax errors are output with the line number.
2. Correct the errors in the customized function if necessary.

Special consideration in the syntax checking of customized functions

Customized functions are not interpreted until in Runtime. When an HMI device is being compiled, the system checks its user-defined functions for correct syntax. However, runtime errors may occur in certain circumstances.

See also

Configuring a function list (Page 4575)

12.8.3.9 Renaming customized VB functions

Renaming a user-defined VB function in the project tree

Proceed as follows to rename a user-defined VB function in the project tree:

1. Select the desired user-defined VB function in the project tree.
2. Select the "Rename" (F2) command in the shortcut menu of the user-defined VB function.
3. Enter a new name for the user-defined VB function.

Renaming a user-defined VB function in the Inspector window

Proceed as follows to rename a user-defined VB function in the Inspector window:

1. Open the user-defined VB function in the "Scripts" editor.
2. In the Inspector window, click "Properties > Properties > General".
3. Enter the new function name.

Result

The user-defined VB function is renamed. The application points are adapted automatically.

12.8.3.10 Executing customized VB functions

Introduction

In WinCC you have the following options to execute a customized function:

- Function list
- Customized VB function

Calling a customized VB function in a function list

1. Add the user-defined VB function to a function list like a system function. You will find the customized VB functions in the drop-down list under "VB functions". You will find further information in "Configuring a function list (Page 4575)".
2. If the VB function has parameters, supply the parameters with static values or HMI tags.
3. If the VB function has a return value, select an HMI tag. The value to be processed is transferred to the HMI tag.

Calling a customized VB function in a customized VB function

1. Open the user-defined VB function in which you want to call the user-defined VB function.
2. Call the customized VB function in the syntax of the programming language. You will find further information in "Calling user-defined VB functions (Page 4583)".
3. If the VB function has a return value, assign an expression, e.g. a local tag, to the VB function.

See also

Configuring a function list (Page 4575)

Calling system functions (Page 4582)

Calling user-defined VB functions (Page 4583)

12.8.3.11 Protecting user-defined functions

Setting up know-how protection

Procedure

To set up know-how protection for user-defined functions, follow these steps:

1. Select the user-defined function without know-how protection that you want to protect.
2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.
3. Click "Define".
   The "Define password" dialog box opens.
4. Enter a password in the "New" field.
5. Enter the same password in the "Confirm" field.
6. Confirm your entries with "OK".
7. Close the "Know-how protection" dialog by clicking on "OK".

Result

The user-defined function receives know-how protection. User-defined functions with know-how protection are marked with a lock in the project tree. The password entered is valid for all user-defined functions selected.

Opening user-defined functions with know-how protection

You can only open multiple user-defined functions with know-how protection at once if they are protected with the same password.

Procedure

To create a user-defined function with know-how protection, follow these steps:

1. Double-click the user-defined function you want to open.
   The "Access protection" dialog will open.
2. Enter the password for the user-defined function with know-how protection.
3. Confirm your entry with "OK".

Result

The user-defined function with know-how protection will open provided you have entered the correct password. However, the function will remain know-how protected.

Once you have opened the user-defined function, you can edit it for as long as the user-defined function or TIA portal is open. You must enter the password again the next time you open the user-defined function. If you close the "Access protection" dialog with "Cancel" or "Close", the

user-defined function is opened but the code is not displayed. You cannot edit the user-defined function.

Removing know-how protection

Procedure

To remove the know-how protection of a user-defined function, follow these steps:

1. Select the user-defined function for which you want to remove know-how protection.

   Note

   To remove know-how protection for several user-defined functions at once, all selected user-defined functions must have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.
3. Deactivate the check box "Hide code (know-how protection)".
4. Enter the password.
5. Confirm your entries with "OK".

Result

Know-how protection will be disabled for the selected user-defined function.

Changing a password

Procedure

To change the password, follow these steps:

1. Select the user-defined function with know-how protection for which you want to change the password.

   Note

   To change the password for several user-defined functions at once, all selected user-defined functions must have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.
3. Click the "Change" button.
4. Enter the old password in the "Old" field.
5. Enter the new password in the "New" field.
6. Enter the new password again in the "Confirm" field.

7. Confirm your entries with "OK".
8. Close the "Know-how protection" dialog by clicking on "OK".

12.8.4 Debugging user-defined VB functions

12.8.4.1 Debugging user-defined VB functions

Introduction

Debugging allows you to test the user-defined VB functions for logical programming errors in Runtime. For example, you can test whether the proper values were transferred to the tags, and whether cancellation terms are realized correctly.

The following VBScript-capable debuggers have been tested and approved:

- "Microsoft Script Debugger"
- "Microsoft Script Editor": The debugger is included in Microsoft Office XP.

"Microsoft Script Debugger" provides the following functionality:

- Viewing the source code of the function you are debugging
- Checking the step-by-step execution of the functions
- Viewing and editing tags and property values

Note

Difference with regard to VBScript for Windows and Windows CE

The debugger checks the syntax for VBScript for Windows. If the function contains a VBScript function for Windows CE, a corresponding error message is output. Some VBScript functions are different, e.g., CreateObject. You can find a list of these VBScript functions under "VBScript for Windows CE (Page 4952)".

Microsoft Script Debugger

If you want to use the Microsoft Script Debugger for debugging, the English version "scd10en.exe" must be used. The German version "scd10de.exe" may not be installed.

"Microsoft Script Debugger" provides the following functionality:

- Viewing the source code of the function you are debugging
- Checking the step-by-step execution of the functions
- Viewing and editing tags and property values

Note

For tips and tricks on debugging, common sources of errors, and additional information, refer to the online help for the Microsoft Script Debugger.

The Microsoft Script Debugger is located in the Microsoft Download Center under the following URL:

- http://www.microsoft.com/downloads/Search.aspx?displaylang=en833a6a92-961e-4ce1-9069-528d22605127

Use the "Search" field to search for "Script Debugger" and select the required download.

Basic procedure

The debugger is only intended for identifying error locations in the user-defined VB function. For this purpose, for example, you use break points or execute the user-defined VB function step-by-step. For detailed information, refer to the documentation of the debugger you are using. In order to correct errors, you change to the "Scripts" editor in WinCC. After having recompiled and reloaded the script, you test the user-defined VB function once again using the debugger.

Note

Your code is displayed in the debugger, but is write-protected. You cannot edit the code directly in the debugger; you can only test the required changes.

Error types

The following error types are distinguished when debugging:

- Logical error
  A logical error occurs when the event you are expecting does not take place, e.g., because a condition was checked incorrectly. To debug logical errors, you go through the user-defined VB function step by step. This way, you identify the portion of the user-defined VB function that is not working.
- Runtime error
  A runtime error is generated if an attempt is made to execute an invalid or faulty instruction, e.g., if a tag is not defined.
  To intercept runtime errors, use the "On Error Resume Next" instruction in the user-defined VB function. With this instruction, the next instruction is executed after a runtime error. You check the error code with the Err object in the next line. To stop the handling of runtime errors in the user-defined VB function again, use the "On Error Goto 0" instruction.
  Example for error handling

```
Sub OnClick(ByVal Item)
'VBS27
  Dim objScreenItem
'
  'Activation of errorhandling:
  On Error Resume Next
  For Each objScreenItem In ScreenItems
    If "HMIRectangle" = objScreenItem.Type Then
    '
      '=== Property "RoundCornerHeight" only available for RoundRectangle
      objScreenItem.RoundCornerHeight = objScreenItem.RoundCornerHeight * 2
      If 0 <> Err.Number Then
        HMIRuntime.Trace objScreenItem.Name & ": no RoundedRectangle" & vbCrLf
        '
        'Delete error message
        Err.Clear
      End If
    End If
  Next
  On Error Goto 0  'Deactivation of errorhandling
End Sub
```

Screen change while debugging

If you change to another screen while debugging, the user-defined VB function of the previous screen stays open but is no longer valid. The debugger could return invalid error messages in this case, as the objects called will no longer exist after the screen change.

12.8.4.2 Integrating the Debugger

Installing a script debugger for WinCC

A VBScript-capable debugger must be installed to search for errors in user-defined VB functions.

The following VBScript-capable debuggers have been tested and released:

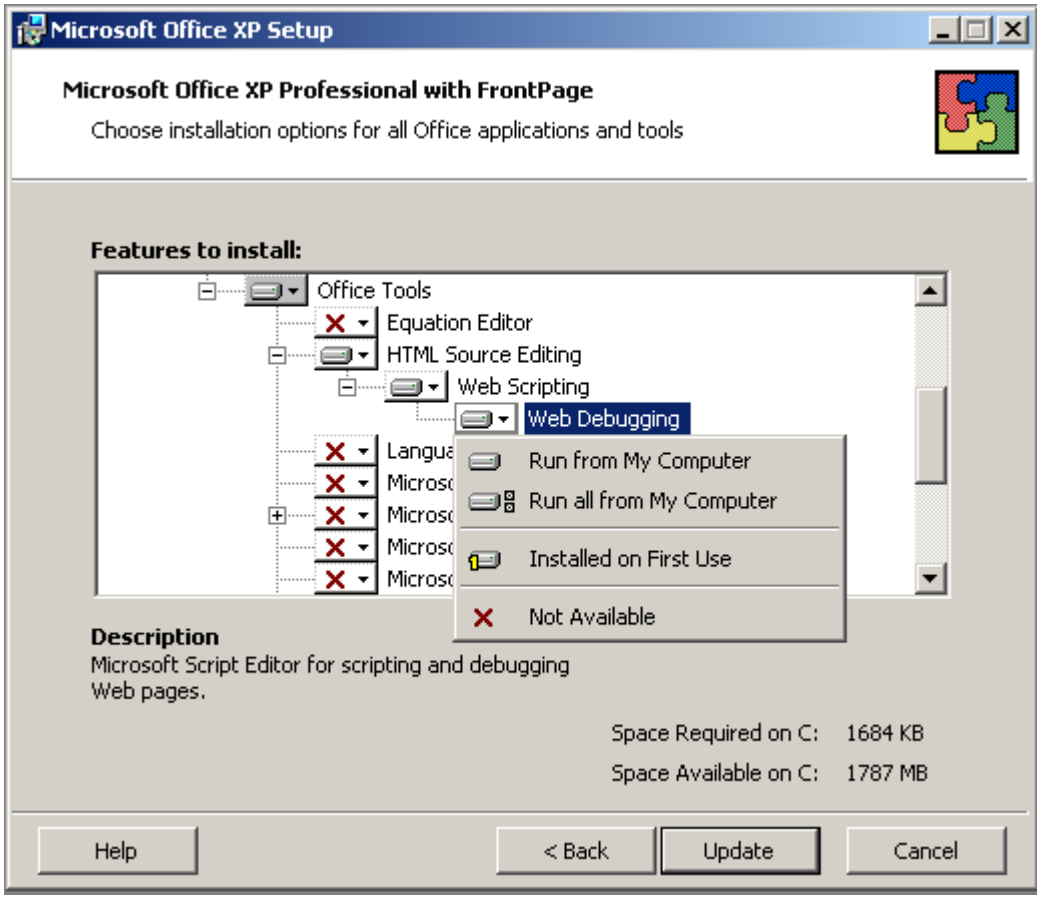
- Microsoft Script Editor by Office XP
- Microsoft Script Debugger: The English version "scd10en.exe" must be used. The German version "scd10de.exe" may not be installed.

An installed debugger is started as follows:

- If a runtime error occurs during execution
- Via "Online > Simulate Runtime > With script debugger"

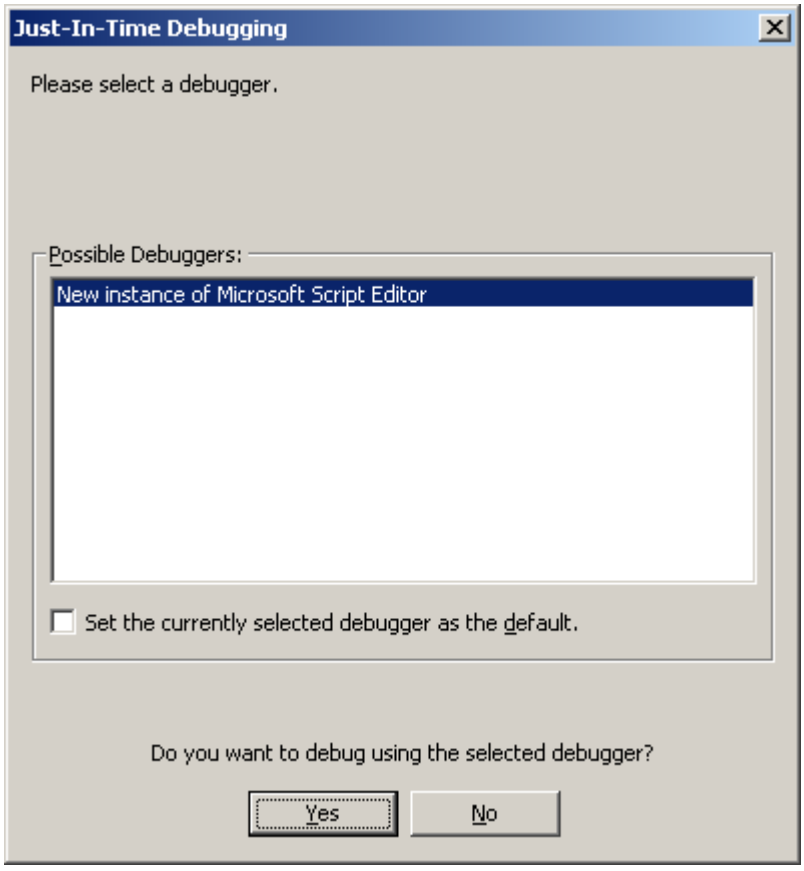
Microsoft Script Editor

The Microsoft Office XP component "Microsoft Script Editor" contains a Script Debugger. If the default configuration was installed by Microsoft Office Setup, the "Microsoft Script Editor" component is not installed until called the first time ("Installed on First Use"). If you wish to explicitly install these components, you must specify it in the Microsoft Office setup. Select the component selection menu, click"Web Debugging" component and then select the "Run from My Computer" option.

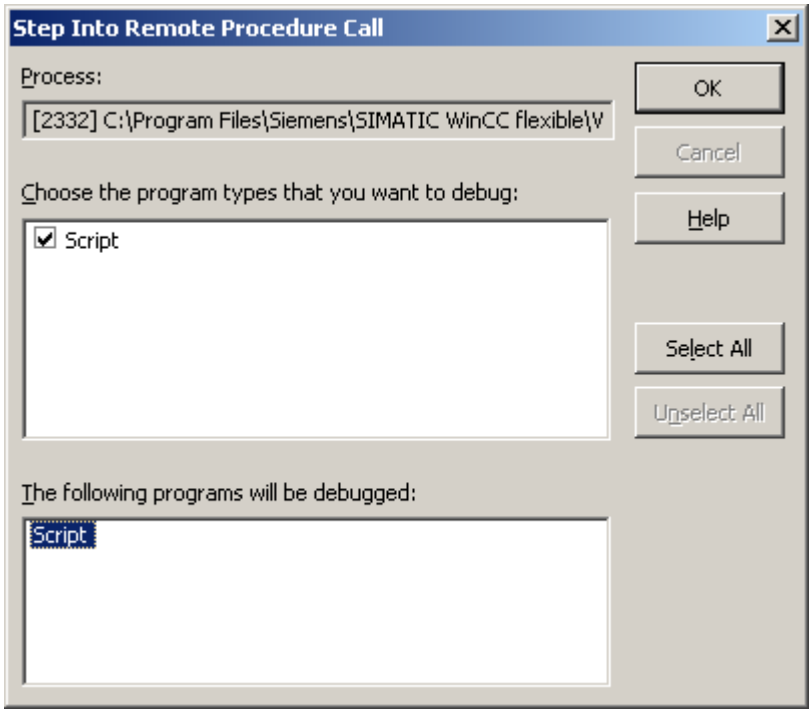


If a project is activated in WinCC via "Online > Simulate Runtime > With script debugger", a dialog containing the list of available script debuggers appears when the first user-defined function is run.

Other installed script debuggers such as "Microsoft Visual Interdev" or "Microsoft Visual Studio .NET" may also appear in the list. Select "Microsoft Script Editor" and confirm your selection with "Yes".



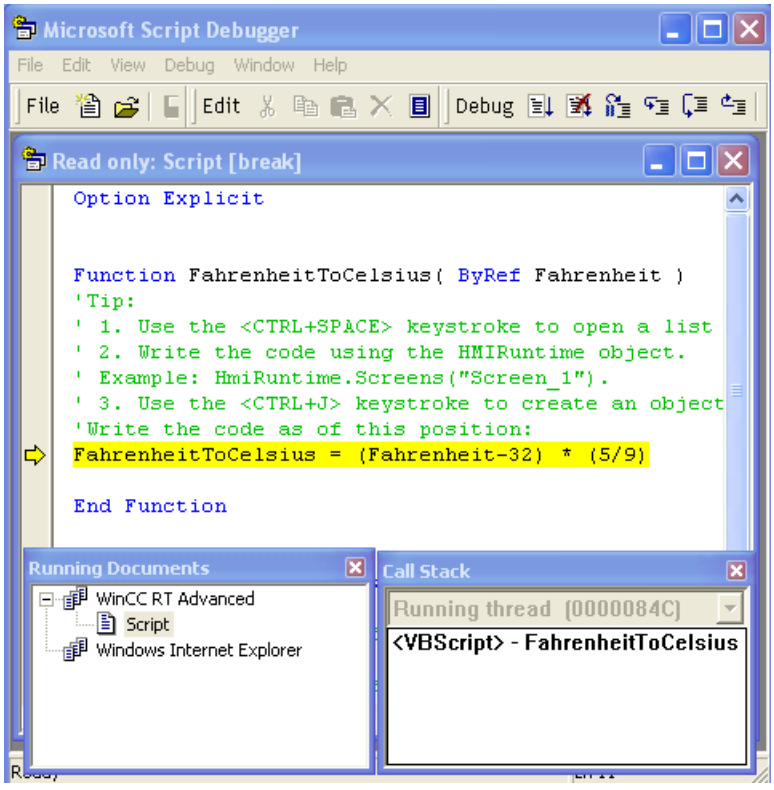
Activate the "Script" program object in the "Step Into Remote Procedure Call" dialog and confirm your selection with "OK".



The "Microsoft Script Editor" is launched. The execution pauses at the first line of the first function.

Microsoft Script Debugger

If a script debugger is not available, download "Microsoft Script Debugger" (scd10en.exe) free of charge from Microsoft (www.microsoft.com). It will be started automatically in WinCC once it is installed.



Note

The "Microsoft Script Debugger" can no longer be called or installed when another VBScript-capable script debugger is available on your computer.

12.8.4.3 Registering Microsoft Script Debugger

Introduction

Under Windows 7 / Windows 8, the Microsoft Script Debugger starts in the simulation only after it has been automatically entered once in the registry as the default script debugger.

Requirement

- Microsoft Script Debugger is installed on the configuration PC.
- You have administrator rights on the configuration PC.

Procedure

1. Open a command prompt with "Start > Run > cmd".
2. Enter the command: [File path]/Regserver, e.g.: "c:\Program Files\Microsoft Script Debugger\msscrdbg.exe/Regserver".

Once the Microsoft Script Debugger has been successfully registered as the default script debugger, you will see the following entries in the registry:

- "HKEY_CLASSES_ROOT\CLSID\{834128A2-51F4-11D0-8F20-00805F2CD064}]
  (Default)="ScriptDebugSvc Class"
  "AppID"="{A87F84D0-7A74-11D0-B216-080000185165}"
- [HKEY_CLASSES_ROOT\CLSID
  \{834128A2-51F4-11D0-8F20-00805F2CD064}\LocalServer32]
  (Default)="c:\\Program Files\\Microsoft Script Debugger\\msscrdbg.exe"
- [HKEY_CLASSES_ROOT\CLSID\{834128A2-51F4-11D0-8F20-00805F2CD064}\ProgID]
  (Default)="ScriptDebugSvc.ScriptDebugSvc.1"
- [HKEY_CLASSES_ROOT\CLSID
  \{834128A2-51F4-11D0-8F2000805F2CD064}\VersionIndependentProgID]
  (Default)="ScriptDebugSvc.ScriptDebugSvc"

The "LocalServer32" file path must point to the installation folder of Microsoft Script Debugger. If you have installed Microsoft Script Debugger in a different folder, adapt the path information.

Result

You have registered the Microsoft Script Debugger as the default script debugger. The Microsoft Script Debugger now starts automatically when Runtime is started.

12.8.4.4 Starting and stopping the debugger

Requirement

- VBScript-capable debugger (e.g., MS Script Debugger) and WinCC Runtime are installed on the configuration PC.
- Project is open.

Starting the debugger

To start the debugger, proceed as follows:

1. Select the HMI device and then select the "Online > Simulation > With script debugger" menu command.
   The Runtime software searches for debuggers installed on the configuration PC.
2. If several debuggers are found click on the one required.

Or:

1. Click "Start > Run".
2. Enter the following command: "HmiRtm /ScriptDebug /<ProjectFile>".
   Further information about the project file can be found in "Starting Runtime Advanced and Panels (Page 6953)".

Or:

1. Select the "debug" command in the project file shortcut menu in the Windows Explorer.

Result

The debugger is connected automatically with the runtime software.

Stopping the debugger

The debugger is not closed automatically when you close the Runtime software. Therefore, close the debugger separately.

12.8.5 Runtime behavior in Runtime

12.8.5.1 Executing a function list in Runtime

Principle

A function list is executed from top to bottom in Runtime. A distinction is made between synchronous and asynchronous execution, so that no waiting periods ensue during execution. The distinction is made by the system by evaluating the different runtimes of the system functions. User-defined functions are always executed synchronously independent of the runtime. If a system function returns an error status, the execution of the function list is cancelled.

Synchronous execution

During synchronous execution, the system functions in a function list are executed one after the other. The previous system function must be finished before the next system function can be executed.

Asynchronous execution

System functions that perform file operations such as saving and reading have a longer runtime than system functions that, for example, set a tag value.

Therefore, system functions with longer runtimes are executed asynchronously. For example, while a system function is writing a recipe data record to a storage medium, the next system function is already being executed. Due to the parallel execution of system functions, waiting periods at the HMI device are avoided.

12.8.5.2 Executing user-defined functions in Runtime

Principle

Only one user-defined function at a time is executed in Runtime. If several user-defined functions are waiting to be executed, they are lined up in a queue and executed one after the other.

Note

A loop in a user-defined function therefore blocks the execution of the other functions in the queue even if the functions were initiated asynchronously.

WinCC supports a maximum nesting depth of eight user-defined functions. Note that the nesting depth is not checked.

Note

If a user-defined function is configured for the "Runtime stop" event, the only system functions that may be used in this user-defined function are those which are available at the "Runtime stop" event.

Ensure that the ending of the Runtime is not interfered with by the execution of the user-defined function.

Note

Configuration of user-defined functions

During configuration make sure that not too many user-defined functions are activated at the same time. Avoid a continuous system load of 100%.

User-defined functions are processed at a lower priority so as not to interfere with the display of values and operability. If system utilization is extreme, the user-defined functions to be executed are therefore first only reserved for execution. The maximum size of the reservation list is dependent on the HMI device and is limited by the maximum permitted number of user-defined functions. For additional information, see the performance features. If more user-defined functions are activated at one time than can be reserved, excess calls are discarded and a system alarm displayed.

HMI device changeover

If you use system functions in a customized function which are not available on the set HMI device, you get a warning. In addition, the corresponding system function in the user-defined function will be underlined with a wavy blue line.

See also

Runtime Stop (Page 4855)

12.8.5.3 Processing sequence for user-defined functions and system functions

Warning

Adherence to the expected processing sequence for user-defined functions and system functions in Runtime cannot always be guaranteed. This fact may lead to unexpected operating states.

Please note the following description in which the correlations are explained. Instructions for a solution are available under "Remedy".

Use of user-defined functions for events

It is not always possible to adhere to the processing sequence for user-defined functions and system functions expected in the configuration, for example, when the processing of a user-defined function lasts beyond the trigger time of a subsequent event. In this case, system functions that are configured for the following event may be executed before the system functions that were configured with the user-defined function for the same event.

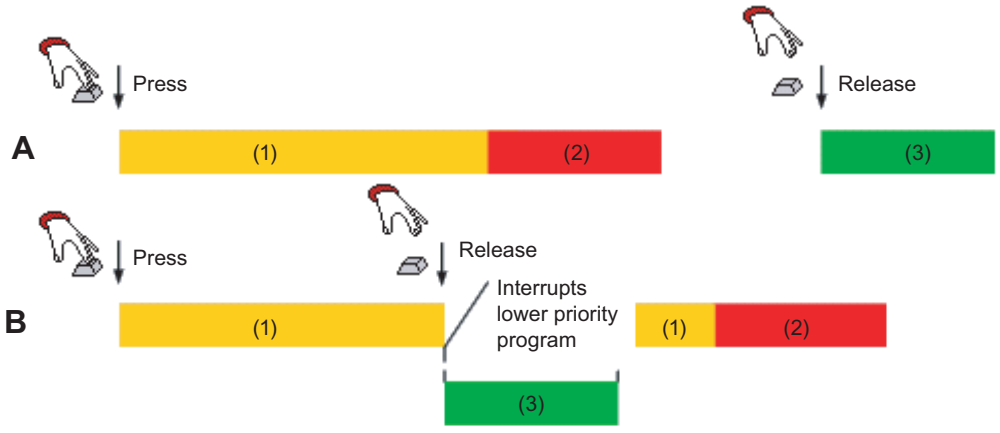
Background

User-defined functions are arranged in a low-priority queue. The queue is processed in sequence. If system functions and user-defined functions are configured for an event, all system functions and user-defined functions configured for this event are executed in this queue.

Example

On a button, you have configured:

	Event	Functions
	Press	User-defined function (1) SetBit(Tag) (2)
	Release	ResetBit(Tag) (3)



(1) Execution time of the user-defined function

(2) Execution time of system function SetBit(Tag)

(3) Execution time of system function ResetBit(Tag)

A: Expected execution of the user-defined function and system functions in the configuration.

When you press the button, user-defined function (1) and then system function SetBit(Tag) (2) are executed. When you release the key, system function ResetBit(Tag) (3) is then executed.

B: Processing of the user-defined function is not yet complete when you release the key.

When you press the key, the user-defined function (1) is then executed. The execution of the user-defined function (1) is not yet complete and you have already released the key. System function ResetBit(Tag) (3) interrupts the execution of the function. In this case, system function ResetBit(Tag) (3) is executed before system function SetBit(Tag) (2). As a result, the bit remains placed.

System function "SetBitWhileKeyPressed" shows the same behavior, if the system function is configured together with user-defined functions.

Remedy

The processing sequence of the previous Example A is adhered to, if you additionally configure a user-defined function for the "Release" event. The user-defined function does not require functionality for this purpose.

	Event	Functions
	Press	User-defined function (1) SetBit(Tag) (2)
	Release	User-defined function_2 (3) ResetBit(Tag) (4)

Note

Make sure that the user-defined function queue does not overflow; otherwise the system function for the following event may not be executed.

An overflow is recognizable by "Overload: Script <name> is rejected" or "Adherence to the expected processing order for user-defined functions and system functions in Runtime cannot always be ensured".

12.8.5.4 Making object properties dynamic in Runtime

Introduction

You can use user-defined functions to access object properties of screen objects as well as tags in Runtime. If you use a user-defined function to change values of object properties, this will not affect the project data.

Changes to object properties

If you use a user-defined function in Runtime to change an object property of a screen element, this change will only remain effective while the screen is active. If you change to the screen, or reload the screen, the configured object properties are displayed once more.

Language switching

If you switch the language in Runtime, the foreign language labels are loaded from the project data. If you use a user-defined function to change texts, the texts changed by the user-defined function are overwritten again.

12.8.6 Examples

12.8.6.1 Example: Converting Fahrenheit into degrees Celsius

Scheduled task

In this example, create a user-defined VB function which converts the values of a temperature sensor from Fahrenheit to degrees Celsius. The temperature is displayed in an output field on the HMI device.

Settings

For the example you need two HMI tags and a user-defined VB function with the following settings:

HMI tags:

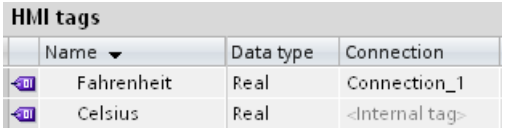
Name	PLC connection	Type
Fahrenheit	Yes	Real
Centigrade	No	Real

User-defined VB function:

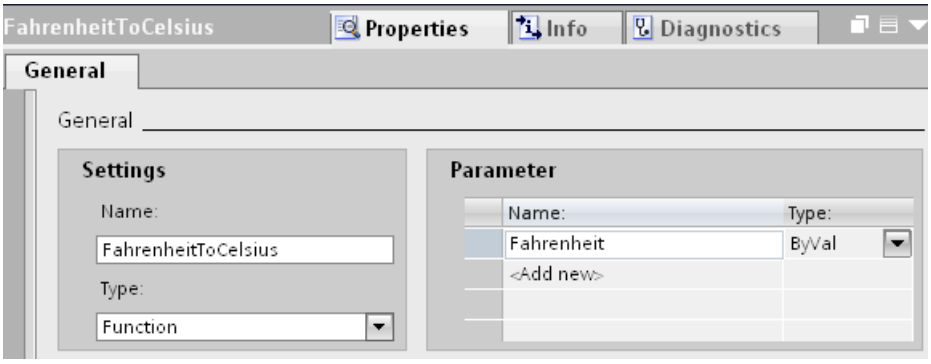
Name	Type	Parameters
FahrenheitToCelsius	Function	Fahrenheit

Procedure

1. Create the two HMI-tags "Fahrenheit" and "Celsius" with the settings named above.

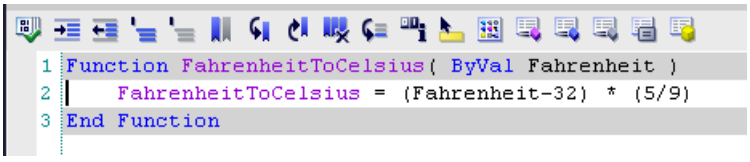


2. Create the user-defined VB function "FahrenheitToCelsius" with the settings given above.



3. Write the following VBS code:

```
FahrenheitToCelsius = (Fahrenheit-32) * (5/9)
```



Interim result

The HMI tags and the VB user-defined function are created.

Procedure

1. Open the "HMI Tag"editor and click on the "Fahrenheit" tag.
2. Click the "Properties > Events" tab in the Inspector window. Select the "Value change" event.

3. Configure the user-defined VB function "FahrenheitToCelsius" to the "Value change" event.
4. Select the tag "Fahrenheit" for the parameter "Fahrenheit".
5. Select the HMI-tag "Celsius" for the return value "Fahrenheit to Celsius".
6. Configure an output field in a screen and connect it with the tag "Celsius".

Alternative procedure

Instead of using a user-defined VB function of the "Function" type, you can also use the "Sub" type. In this case, assign the converted value directly to the HMI-tag "Celsius".

```
SmartTags("Celsius") = (Fahrenheit-32) * (5/9)
```

In this case the return value of the user-defined VB function is not applicable.

Result

When the tag value of "Fahrenheit" changes in Runtime, the user-defined VB function "Fahrenheit to Celsius" is performed. The converted temperature value is returned to the HMI-tag "Celsius" and displayed in the output field.

12.8.6.2 Example: Converting inches into meters

Scheduled task

In this example you create a user-defined VB function that converts the value of a length measuring system from inches into meters. The length in meters is displayed on the HMI device in an output field.

Settings

For the example you need two HMI tags and a user-defined VB function with the following settings:

HMI tags:

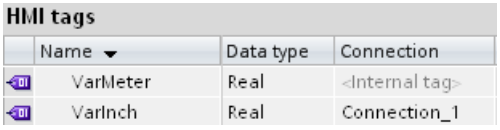
Name	PLC connection	Type
VarInch	Yes	Real
VarMeter	No	Real

Customized VB function

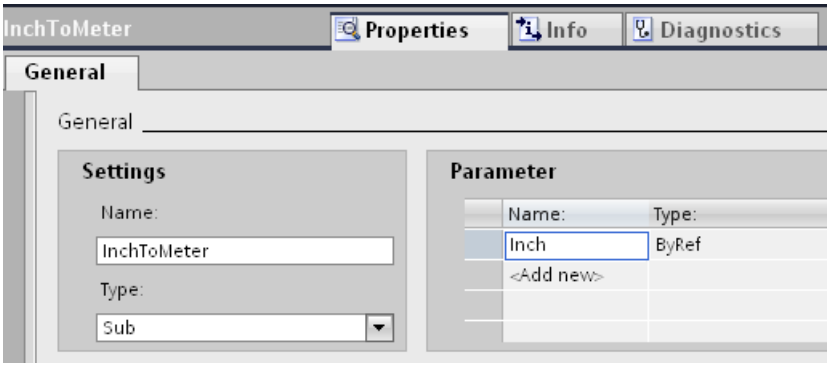
Name	Type	Parameters
InchToMeter	Sub	Inch

Procedure

1. Create the two HMI tags "VarInch" and "VarMeter" with the above settings.

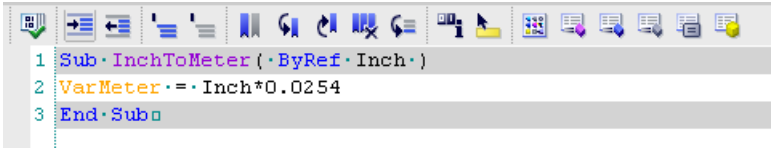


2. Create the user-defined VB function "InchToMeter" with the settings given above.



3. Write the following VBS code:

```
VarMeter = Inch*0.0254
```



Interim result

The tags and the user-defined VB function are created.

Procedure

1. Open the "HMI tags" editor and click the "VarInch" tag.
2. Click "Properties> Events" in the Inspector window. Select the "Value change" event.
3. Configure the user-defined VB function "InchToMeter" to the "Value change" event.
4. Select the "VarInch" HMI tag for the "Inch" parameter.
5. Configure an output field in a screen and connect it with the "VarMeter" HMI tag.

Result

When the tag value of "VarInch" changes in Runtime, the "InchToMeter" user-defined VB function is performed. The calculated value is written into the "VarMeter" HMI tag and displayed in the output field.

12.8.6.3 Example: Changing the operating mode on the HMI device with the current display

Scheduled task

In this example, you use the "SetDeviceMode" system function to switch between the "online" and "offline" modes on the HMI device. You also display the current set operating mode on the HMI device.

Requirements

A screen has been created.

Settings

For this example you require a HMI-tag and a text list with the following settings:

HMI tag:

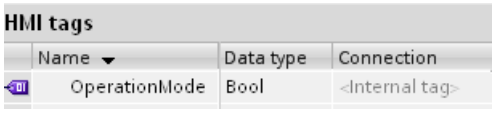
Name	PLC connection	Type
OperatingMode	No	Bool

Text list:

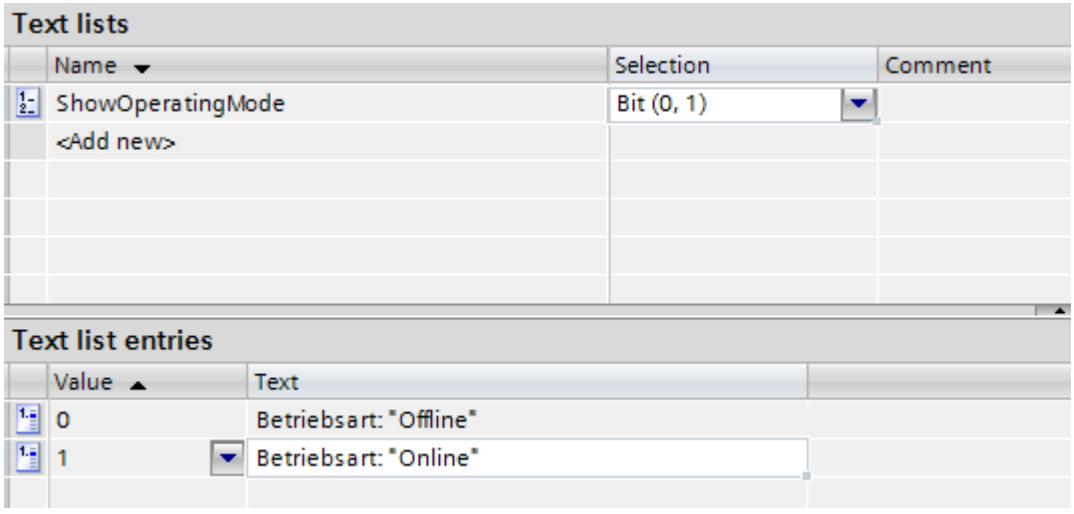
Name	Contains	Values
ShowOperatingMode	Bit (0/1)	1: Operating mode: "Online" 0: Operating mode: "Offline"

Procedure

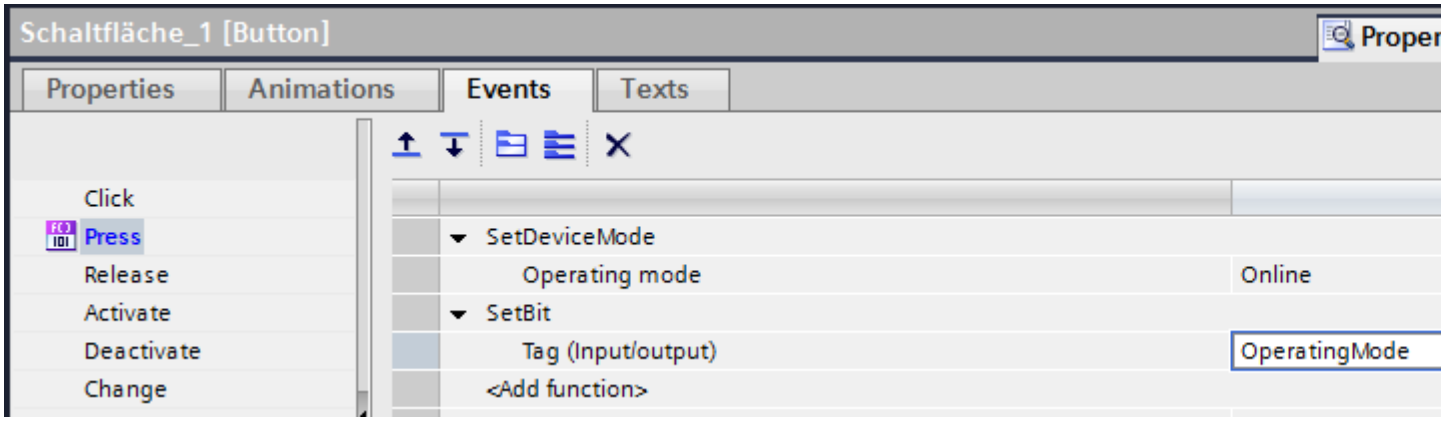
1. Create the "OperatingMode" HMI-tag indicated above.



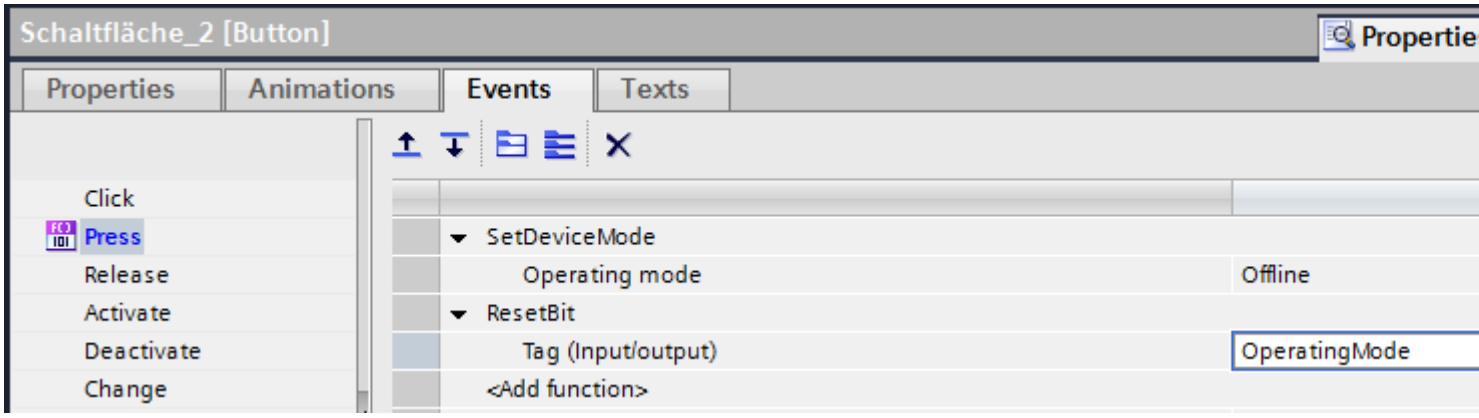
2. Create the "ShowOperatingMode" text list indicated above.



3. Open the screen and insert a button for which you configure the operating mode change to "online".
4. Click "Properties> Events" in the Inspector window. Select the "Press" event.
5. Configure the "SetDeviceMode" system function for the "Press" event. The system function is found in the selection list under "Settings".
6. For the "Mode" parameter, select the "Online" entry.
7. Configure the system function "SetBit" on the event "Press". The system function is found in the selection list under "Bit processing".
8. Select the HMI-tag "Operating mode" from the selection list for the parameter "Tag".



9. Add a button in the process screen for which you configure the operating mode change to "offline".
10. Repeat steps four to seven. For the "Mode" parameter, select the "Offline" entry. Configure the system function "ResetBit" in place of the system function "SetBit."



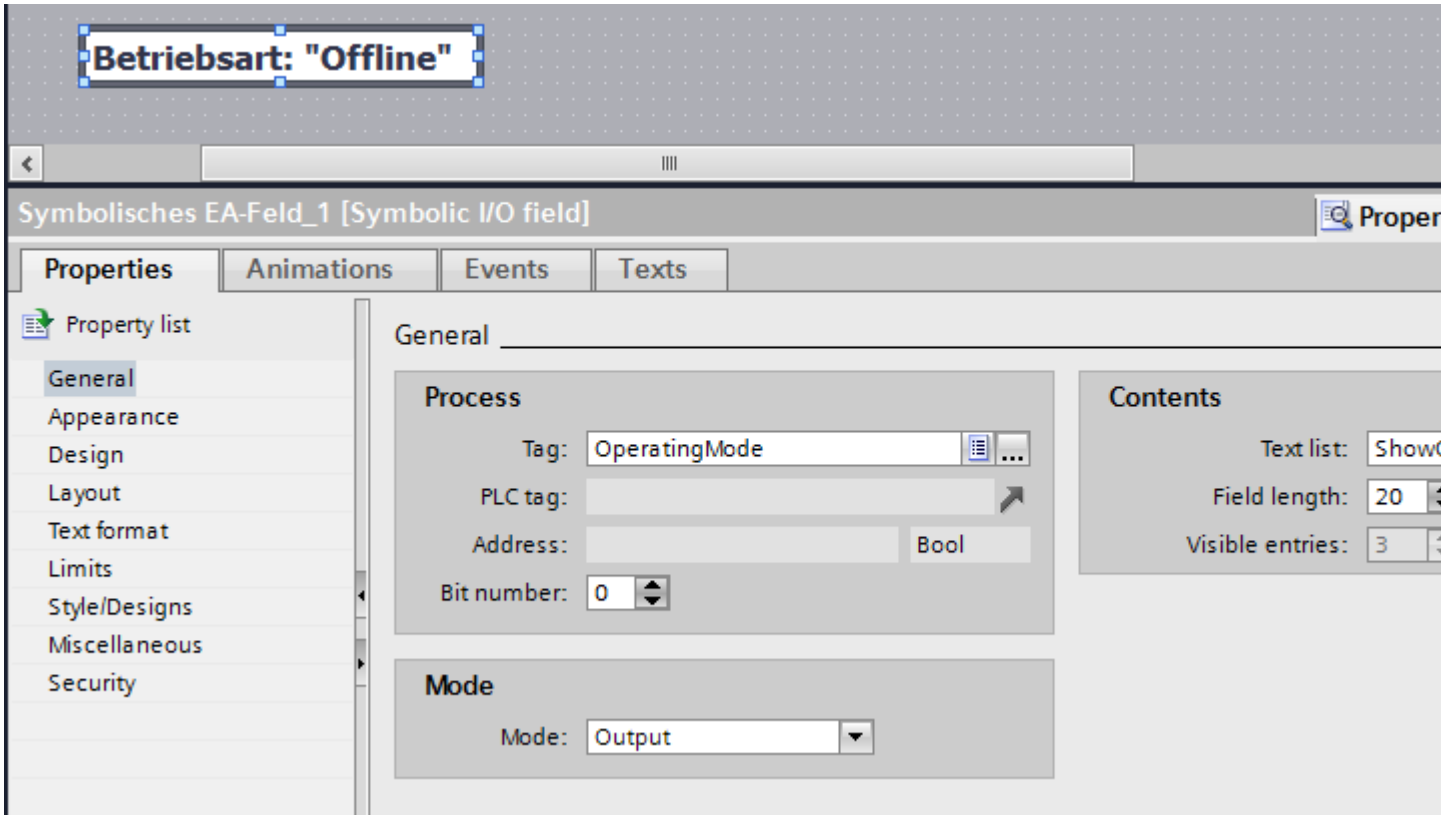
Interim result

You can toggle the operating mode of the HMI device with the two buttons in Runtime.

You want to display the current set operating mode in an output field on the HMI device.

Procedure

1. Create a "Symbolic I/O field" in the process image. Click "Properties > Properties" in the Inspector window.
2. Make the following settings in the "General" group:
   - Select "Output" as the "Mode".
   - Select the text list "Show operating mode" as "Text list".
   - Select "Operating mode" as "Tag".



Result

When you change the operating mode with the buttons, the currently set operating mode on the HMI device is always shown.

12.8.7 Reference

12.8.7.1 Function list

Availability for specific devices of system functions

System functions for Basic Panels

Availability of system functions

The following table shows the availability of the system functions on the Basic Panels.

Technical data subject to change.

Overview

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
User-defined functions	No	No	No	No	No
Logoff (Page 4657)	Yes	Yes	Yes	Yes	Yes
ActivateScreen (Page 4657)	Yes	Yes	Yes	Yes	Yes
ActivateScreenByNumber (Page 4659)	Yes	Yes	Yes	Yes	Yes
ActivateCleanScreen (Page 4660)	No	No	Yes	Yes	Yes
ActivateSystemDiagnosticsView (Page 4796)	No	No	No	No	No
ActivatePreviousScreen (Page 4660)	Yes	Yes	Yes	Yes	Yes
UpdateTag (Page 4661)	Yes	Yes	Yes	Yes	Yes
AdjustContrast (Page 4662)	Yes	No	Yes [1)]	Yes	Yes
Logon (Page 4663)	Yes	Yes	Yes	Yes	Yes
ArchiveLogFile (Page 4663)	No	No	No	No	No
LogTag (Page 4665)	No	No	No	No	No
EditAlarm (Page 4666)	Yes	Yes	Yes	Yes	No
ScreenObjectCursorDown (Page 4666)	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorUp (Page 4667)	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorLeft (Page 4668)	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorRight (Page 4668)	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageDown (Page 4669)	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageUp (Page 4670)	Yes	Yes	Yes	Yes	Yes
Encode (Page 4671)	No	No	No	No	No
EncodeEx (Page 4672)	No	No	No	No	No

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
Direct key (Page 4673)	No	No	No	No	No
DirectKeyScreenNumber (Page 4675)	No	No	No	No	No
PrintScreen (Page 4676)	No	No	No	No	No
PrintReport (Page 4677)	No	No	No	No	No
NotifyUserAction (Page 4677)	No	No	No	No	No
IncreaseFocusedValue (Page 4679)	Yes	Yes	Yes	Yes	No
IncreaseTag (Page 4679)	Yes	Yes	Yes	Yes	Yes
ExportDataRecords (Page 4680)	Yes	Yes	Yes	Yes	Yes
ExportDataRecordsWithChecksum (Page 4683)	No	No	No	No	No
ExportImportUserAdministration (Page 4685)	No	No	No	No	No
GoToHome (Page 4686)	Yes	Yes	Yes	Yes	No
GoToEnd (Page 4686)	Yes	Yes	Yes	Yes	No
SafelyRemoveHardware (Page 4687)	No	No	No	No	No
HTMLBrowserStop (Page 4688)	No	No	No	No	No
HTMLBrowserScrollDown (Page 4688)	No	No	No	No	No
HTMLBrowserRefresh (Page 4689)	No	No	No	No	No
HTMLBrowserScrollUp (Page 4689)	No	No	No	No	No
HTMLBrowserForward (Page 4695)	No	No	No	No	No
HTMLBrowserBack (Page 4695)	No	No	No	No	No
HTMLBrowserZoomIn (Page 4690)	No	No	No	No	No
HTMLBrowserZoomOut (Page 4691)	No	No	No	No	No
HTMLBrowserScrollLeft (Page 4691)	No	No	No	No	No
HTMLBrowserScrollRight (Page 4692)	No	No	No	No	No
HTMLBrowserPageUp (Page 4693)	No	No	No	No	No
HTMLBrowserPageDown (Page 4693)	No	No	No	No	No
HTMLBrowserHome (Page 4694)	No	No	No	No	No
ImportDataRecords (Page 4696)	Yes	Yes	Yes	Yes	Yes
ImportDataRecordsWithChecksum (Page 4698)	No	No	No	No	No
InvertBit (Page 4699)	Yes	Yes	Yes	Yes	Yes
InvertBitInTag (Page 4700)	Yes	Yes	Yes	Yes	Yes
InvertLinearScaling (Page 4701)	Yes	Yes	Yes	Yes	Yes
CalibrateTouchScreen (Page 4703)	No	Yes	Yes	Yes	Yes
CopyLog (Page 4704)	No	No	No	No	No
TrendViewScrollForward (Page 4705)	Yes	Yes	Yes	Yes	Yes
TrendViewScrollBack (Page 4705)	Yes	Yes	Yes	Yes	Yes
TrendViewExtend (Page 4706)	Yes	Yes	Yes	Yes	Yes
TrendViewCompress (Page 4706)	Yes	Yes	Yes	Yes	Yes

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
TrendViewRulerLeft (Page 4707)	Yes	Yes	Yes	Yes	Yes
TrendViewRulerRight (Page 4708)	Yes	Yes	Yes	Yes	Yes
TrendViewSetRulerMode (Page 4708)	Yes	Yes	Yes	Yes	Yes
TrendViewStartStop (Page 4709)	Yes	Yes	Yes	Yes	Yes
TrendViewBackToBeginning (Page 4709)	Yes	Yes	Yes	Yes	Yes
LoadDataRecord (Page 4710)	Yes	Yes	Yes	Yes	Yes
GetUserName (Page 4711)	Yes	Yes	Yes	Yes	Yes
GetDataRecordFromPLC (Page 4712)	Yes	Yes	Yes	Yes	Yes
GetDataRecordName (Page 4713)	No	No	No	No	No
GetDataRecordTagsFromPLC (Page 4715)	Yes	Yes	Yes	Yes	Yes
GetGroupNumber (Page 4716)	Yes	Yes	Yes	Yes	Yes
GetBrightness (Page 4716)	No	No	No	No	No
GetPassword (Page 4717)	Yes	Yes	Yes	Yes	Yes
LinearScaling (Page 4718)	Yes	Yes	Yes	Yes	Yes
ClearLog (Page 4720)	No	No	No	No	No
DeleteDataRecord (Page 4721)	Yes	Yes	Yes	Yes	Yes
ClearDataRecordMemory (Page 4722)	No	No	No	No	No
ClearAlarmBuffer (Page 4723)	Yes	Yes	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 4724)	Yes	Yes	Yes	Yes	Yes
AlarmViewUpdate (Page 4725)	No	No	No	No	No
AlarmViewLoopInAlarm (Page 4725)	Yes	Yes	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 4726)	Yes	Yes	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 4727)	Yes	Yes	Yes	Yes	Yes
OpenAllLogs (Page 4728)	No	No	No	No	No
OpenScreenKeyboard (Page 4729)	No	No	No	No	No
OpenControlPanelDialog (Page 4730)	Yes [2)]	Yes [2)]	Yes [2)]	Yes [2)]	Yes [2)]
OpenCommandPrompt (Page 4731)	No	No	No	No	No
OpenInternetExplorer (Page 4732)	No	No	No	No	No
OpenControlPanel (Page 4733)	No	No	No	No	No
OpenTaskManager (Page 4733)	No	No	No	No	No
AcknowledgeAlarm (Page 4744)	Yes	Yes	Yes	Yes	No
PDFScrollDown (Page 4734)	No	No	No	No	No
PDFScrollUp (Page 4735)	No	No	No	No	No
PDFFitToWidth (Page 4735)	No	No	No	No	No
PDFFitToHeight (Page 4736)	No	No	No	No	No
PDFGoToFirstPage (Page 4737)	No	No	No	No	No
PDFGoToLastPage (Page 4737)	No	No	No	No	No
PDFGoToNextPage (Page 4738)	No	No	No	No	No
PDFGoToPage (Page 4738)	No	No	No	No	No

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
PDFGoToPreviousPage (Page 4739)	No	No	No	No	No
PDFZoomIn (Page 4740)	No	No	No	No	No
PDFZoomOut (Page 4740)	No	No	No	No	No
PDFScrollLeft (Page 4741)	No	No	No	No	No
PDFScrollRight (Page 4741)	No	No	No	No	No
PDFZoomOriginal (Page 4742)	No	No	No	No	No
RecipeViewNewDataRecord (Page 4744)	Yes	Yes	Yes	Yes	Yes
RecipeViewGetDataRecordFromPLC (Page 4745)	Yes	Yes	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 4745)	Yes	Yes	Yes	Yes	Yes
RecipeViewMenu (Page 4746)	Yes	Yes	Yes	Yes	Yes
RecipeViewOpen (Page 4746)	Yes	Yes	Yes	Yes	Yes
RecipeViewSetDataRecordToPLC (Page 4747)	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 4748)	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 4748)	Yes	Yes	Yes	Yes	Yes
RecipeViewSynchronizeDataRecordWith-Tags (Page 4749)	Yes	Yes	Yes	Yes	Yes
RecipeViewRenameDataRecord (Page 4749)	Yes	Yes	Yes	Yes	Yes
RecipeViewShowOperatorNotes (Page 4750)	Yes	Yes	Yes	Yes	Yes
RecipeViewBack (Page 4750)	Yes	Yes	Yes	Yes	Yes
ResetBit (Page 4751)	Yes	Yes	Yes	Yes	Yes
ResetBitInTag (Page 4752)	Yes	Yes	Yes	Yes	Yes
PressButton (Page 4753)	Yes	Yes	Yes	Yes	No
ReleaseButton (Page 4754)	Yes	Yes	Yes	Yes	No
ShiftAndMask (Page 4755)	Yes	Yes	Yes	Yes	Yes
CloseAllLogs (Page 4757)	No	No	No	No	No
SetDataRecordToPLC (Page 4758)	Yes	Yes	Yes	Yes	Yes
SetDataRecordTagsToPLC (Page 4759)	Yes	Yes	Yes	Yes	Yes
PageDown (Page 4759)	Yes	Yes	Yes	Yes	No
PageUp (Page 4760)	Yes	Yes	Yes	Yes	No
SendEMail (Page 4761)	No	No	No	No	No
SetAcousticSignal (Page 4762)	No	No	No	No	No
SetDisplayMode (Page 4762)	No	No	No	No	No
SetDeviceMode (Page 4763)	Yes	Yes	Yes	Yes	Yes
SetBit (Page 4764)	Yes	Yes	Yes	Yes	Yes
SetBitInTag (Page 4765)	Yes	Yes	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 4767)	Yes	Yes	Yes	Yes	Yes

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
SetBacklightColor (Page 4768)	Yes	Yes	No	No	No
SetBrightness (Page 4769)	No	Yes	Yes	No	No
SetScreenKeyboardMode (Page 4770)	No	No	No	No	No
SetAlarmReportMode (Page 4772)	No	No	No	No	No
SetRecipeTags (Page 4772)	Yes	Yes	Yes	Yes	Yes
SetDaylightSavingTime (Page 4774)	No	No	No	No	No
SetLanguage (Page 4775)	Yes	Yes	Yes	Yes	Yes
SetTag (Page 4777)	Yes	Yes	Yes	Yes	Yes
SetConnectionMode (Page 4778)	Yes	Yes	Yes	Yes	Yes
SetWebAccess (Page 4780)	No	No	No	No	No
BackupRAMFileSystem (Page 4780)	No	No	No	No	No
SimulateSystemKey (Page 4781)	Yes	Yes	Yes	Yes	No
SimulateTag (Page 4782)	Yes	Yes	Yes	Yes	Yes
SmartClientViewRefresh (Page 4783)	No	No	No	No	No
SmartClientViewReadOnlyOff (Page 4784)	No	No	No	No	No
SmartClientViewReadOnlyOn (Page 4784)	No	No	No	No	No
SmartClientViewDisconnect (Page 4785)	No	No	No	No	No
SmartClientViewConnect (Page 4785)	No	No	No	No	No
SmartClientViewLeave (Page 4786)	No	No	No	No	No
SaveDataRecord (Page 4787)	Yes	Yes	Yes	Yes	Yes
StartLogging (Page 4788)	No	No	No	No	No
StartNextLog (Page 4789)	No	No	No	No	No
StartProgram (Page 4789)	No	No	No	No	No
StatusForceGetValues (Page 4791)	No	No	No	No	No
StatusForceSetValues (Page 4792)	No	No	No	No	No
ControlSmartServer (Page 4792)	No	No	No	No	No
ControlWebServer (Page 4793)	No	No	No	No	No
StopLogging (Page 4794)	No	No	No	No	No
StopRuntime (Page 4795)	Yes	Yes	Yes	Yes	Yes
LookupText (Page 4799)	No	No	No	No	No
ResetTagToHandWheel (Page 4800)	No	No	No	No	No
SetTagToHandWheel (Page 4800)	No	No	No	No	No
TraceUserChange (Page 4801)	Yes	Yes	Yes	Yes	Yes
DecreaseFocusedValue (Page 4801)	Yes	Yes	Yes	Yes	No
DecreaseTag (Page 4802)	Yes	Yes	Yes	Yes	Yes
ChangeConnection (Page 4803)	Yes	Yes	Yes	Yes	Yes
WinACMPUpdateStartupCharacteristics (Page 4811)	No	No	No	No	No
WinACMPUpdateKeySwitchSetting (Page 4812)	No	No	No	No	No

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
WinACMPUpdateBUSF1LED (Page 4813)	No	No	No	No	No
WinACMPUpdateBUSF2LED (Page 4814)	No	No	No	No	No
WinACMPUpdateAverageExecTime (Page 4815)	No	No	No	No	No
WinACMPUpdateAverageCycleTime (Page 4816)	No	No	No	No	No
WinACMPUpdateEXTFLED (Page 4817)	No	No	No	No	No
WinACMPUpdateHMIEnableTime (Page 4818)	No	No	No	No	No
WinACMPUpdateINTFLED (Page 4819)	No	No	No	No	No
WinACMPUpdateLastCycleTime (Page 4820)	No	No	No	No	No
WinACMPUpdateMaximumCycleTime (Page 4821)	No	No	No	No	No
WinACMPUpdateMinimumCycleTime (Page 4822)	No	No	No	No	No
WinACMPUpdatePowerLED (Page 4823)	No	No	No	No	No
WinACMPUpdateSleepTime (Page 4824)	No	No	No	No	No
WinACMPUpdateRUNLED (Page 4825)	No	No	No	No	No
WinACMPUpdateSTOPLED (Page 4826)	No	No	No	No	No
WinACMPArchive (Page 4827)	No	No	No	No	No
WinACMPGetStartMode (Page 4828)	No	No	No	No	No
WinACMPGetVersion (Page 4828)	No	No	No	No	No
WinACMPClearCycleTimeBuffer (Page 4829)	No	No	No	No	No
WinACMPSetStartAtBoot (Page 4829)	No	No	No	No	No
WinACMPSetKeySwitch (Page 4830)	No	No	No	No	No
WinACMPSetHMIExecutionTime (Page 4831)	No	No	No	No	No
WinACMPSetResetMethod (Page 4831)	No	No	No	No	No
WinACMPSetSleeptime (Page 4832)	No	No	No	No	No
WinACMPSetStartMode (Page 4832)	No	No	No	No	No
WinACMPStartHistogram (Page 4833)	No	No	No	No	No
WinACMPControl (Page 4834)	No	No	No	No	No
WinACMPStopHistogram (Page 4834)	No	No	No	No	No
WinACMPRestore (Page 4835)	No	No	No	No	No
ShowLogonDialog (Page 4804)	Yes	Yes	Yes	Yes	Yes
ShowOperatorNotes (Page 4805)	Yes	Yes	Yes	Yes	Yes
ShowAlarmWindow (Page 4806)	Yes	Yes	Yes	Yes	Yes
ShowPopupScreen (Page 4807)	No	No	No	No	No
ShowSlideInScreen (Page 4808)	No	No	No	No	No
ShowSoftwareVersion (Page 4809)	No	No	No	No	No

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
ShowSystemDiagnosticsWindow (Page 4809)	No	No	No	No	No
ShowSystemEvent (Page 4810)	No	No	No	No	No

1) For KTP600 Basic mono PN only
2) Only for editing the IP settings

See also

ReadPLCMode (Page 4718)

TerminatePROFIsafe (Page 4743)

EstablishPROFIsafe (Page 4743)

System functions for Basic Panels 2nd Generation

Availability of system functions

The following table shows the availability of the system functions on the Basic Panels.

Technical data subject to change.

Overview

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic DP	KTP1200 Basic PN / DP
User-defined functions	No	No	No	No
Logoff (Page 4657)	Yes	Yes	Yes	Yes
ActivateScreen (Page 4657)	Yes	Yes	Yes	Yes
ActivateScreenByNumber (Page 4659)	Yes	Yes	Yes	Yes
ActivateCleanScreen (Page 4660)	Yes	Yes	Yes	Yes
ActivatePreviousScreen (Page 4660)	Yes	Yes	Yes	Yes
UpdateTag (Page 4661)	Yes	Yes	Yes	Yes
AdjustContrast (Page 4662)	No	No	No	No
Logon (Page 4663)	Yes	Yes	Yes	Yes
ArchiveLogFile (Page 4663)	No	No	No	No
LogTag (Page 4665)	Yes	Yes	Yes	Yes
EditAlarm (Page 4666)	Yes	Yes	Yes	Yes
ScreenObjectCursorDown (Page 4666)	Yes	Yes	Yes	Yes
ScreenObjectCursorUp (Page 4667)	Yes	Yes	Yes	Yes

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic DP	KTP1200 Basic PN / DP
ScreenObjectCursorLeft (Page 4668)	Yes	Yes	Yes	Yes
ScreenObjectCursorRight (Page 4668)	Yes	Yes	Yes	Yes
ScreenObjectPageDown (Page 4669)	Yes	Yes	Yes	Yes
ScreenObjectPageUp (Page 4670)	Yes	Yes	Yes	Yes
Encode (Page 4671)	No	No	No	No
EncodeEx (Page 4672)	No	No	No	No
Direct key (Page 4673)	No	No	No	No
DirectKeyScreenNumber (Page 4675)	No	No	No	No
PrintScreen (Page 4676)	No	No	No	No
PrintReport (Page 4677)	No	No	No	No
NotifyUserAction (Page 4677)	No	No	No	No
IncreaseFocusedValue (Page 4679)	Yes	Yes	Yes	Yes
IncreaseTag (Page 4679)	Yes	Yes	Yes	Yes
ExportDataRecords (Page 4680)	Yes	Yes	Yes	Yes
ExportDataRecordsWithChecksum (Page 4683)	No	No	No	No
ExportImportUserAdministration (Page 4685)	No	No	No	No
GoToHome (Page 4686)	Yes	Yes	Yes	Yes
GoToEnd (Page 4686)	Yes	Yes	Yes	Yes
SafelyRemoveHardware (Page 4687)	Yes	Yes	Yes	Yes
HTMLBrowserStop (Page 4688)	Yes	Yes	Yes	Yes
HTMLBrowserScrollDown (Page 4688)	Yes	Yes	Yes	Yes
HTMLBrowserRefresh (Page 4689)	Yes	Yes	Yes	Yes
HTMLBrowserScrollUp (Page 4689)	Yes	Yes	Yes	Yes
HTMLBrowserZoomIn (Page 4690)	Yes	Yes	Yes	Yes
HTMLBrowserZoomOut (Page 4691)	Yes	Yes	Yes	Yes
HTMLBrowserScrollLeft (Page 4691)	Yes	Yes	Yes	Yes
HTMLBrowserScrollRight (Page 4692)	Yes	Yes	Yes	Yes
HTMLBrowserPageUp (Page 4693)	Yes	Yes	Yes	Yes
HTMLBrowserPageDown (Page 4693)	Yes	Yes	Yes	Yes
HTMLBrowserStartPage (Page 4694)	Yes	Yes	Yes	Yes
HTMLBrowserForward (Page 4695)	Yes	Yes	Yes	Yes
HTMLBrowserBack (Page 4695)	Yes	Yes	Yes	Yes
ImportDataRecords (Page 4696)	Yes	Yes	Yes	Yes
ImportDataRecordsWithChecksum (Page 4698)	No	No	No	No
InvertBit (Page 4699)	Yes	Yes	Yes	Yes
InvertBitInTag (Page 4700)	Yes	Yes	Yes	Yes
InvertLinearScaling (Page 4701)	Yes	Yes	Yes	Yes
CalibrateTouchScreen (Page 4703)	Yes	Yes	Yes	Yes
CopyLog (Page 4704)	No	No	No	No
TrendViewScrollForward (Page 4705)	Yes	Yes	Yes	Yes

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic DP	KTP1200 Basic PN / DP
TrendViewScrollBack (Page 4705)	Yes	Yes	Yes	Yes
TrendViewExtend (Page 4706)	Yes	Yes	Yes	Yes
TrendViewCompress (Page 4706)	Yes	Yes	Yes	Yes
TrendViewRulerLeft (Page 4707)	Yes	Yes	Yes	Yes
TrendViewRulerRight (Page 4708)	Yes	Yes	Yes	Yes
TrendViewSetRulerMode (Page 4708)	Yes	Yes	Yes	Yes
TrendViewStartStop (Page 4709)	Yes	Yes	Yes	Yes
TrendViewBackToBeginning (Page 4709)	Yes	Yes	Yes	Yes
LoadDataRecord (Page 4710)	Yes	Yes	Yes	Yes
GetUserName (Page 4711)	Yes	Yes	Yes	Yes
GetDataRecordFromPLC (Page 4712)	Yes	Yes	Yes	Yes
GetDataRecordName (Page 4713)	No	No	No	No
GetDataRecordTagsFromPLC (Page 4715)	Yes	Yes	Yes	Yes
GetGroupNumber (Page 4716)	Yes	Yes	Yes	Yes
GetBrightness (Page 4716)	No	No	No	No
GetPassword (Page 4717)	Yes	Yes	Yes	Yes
LinearScaling (Page 4718)	Yes	Yes	Yes	Yes
ClearLog (Page 4720)	Yes	Yes	Yes	Yes
DeleteDataRecord (Page 4721)	Yes	Yes	Yes	Yes
ClearDataRecordMemory (Page 4722)	No	No	No	No
ClearAlarmBuffer (Page 4723)	Yes	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 4724)	Yes	Yes	Yes	Yes
AlarmViewUpdate (Page 4725)	No	No	No	No
AlarmViewLoopInAlarm (Page 4725)	No	No	No	No
AlarmViewAcknowledgeAlarm (Page 4726)	Yes	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 4727)	Yes	Yes	Yes	Yes
OpenAllLogs (Page 4728)	Yes	Yes	Yes	Yes
OpenScreenKeyboard (Page 4729)	No	No	No	No
OpenControlPanelDialog (Page 4730)	Yes [1]	Yes [1]	Yes [1]	Yes [1]
OpenCommandPrompt (Page 4731)	No	No	No	No
OpenInternetExplorer (Page 4732)	No	No	No	No
OpenControlPanel (Page 4733)	No	No	No	No
OpenTaskManager (Page 4733)	No	No	No	No
PDFScrollDown (Page 4734)	No	No	No	No
PDFFitToWidth (Page 4735)	No	No	No	No
PDFFitToHeight (Page 4736)	No	No	No	No
PDFScrollUp (Page 4735)	No	No	No	No
PDFGoToFirstPage (Page 4737)	No	No	No	No
PDFGoToLastPage (Page 4737)	No	No	No	No
PDFGoToNextPage (Page 4738)	No	No	No	No

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic DP	KTP1200 Basic PN / DP
PDFGoToPreviousPage (Page 4739)	No	No	No	No
PDFGoToPage (Page 4738)	No	No	No	No
PDFZoomIn (Page 4740)	No	No	No	No
PDFZoomOut (Page 4740)	No	No	No	No
PDFScrollLeft (Page 4741)	No	No	No	No
PDFScrollRight (Page 4741)	No	No	No	No
PDFZoomOriginal (Page 4742)	No	No	No	No
AcknowledgeAlarm (Page 4744)	Yes	Yes	Yes	Yes
RecipeViewNewDataRecord (Page 4744)	Yes	Yes	Yes	Yes
RecipeViewGetDataRecordFromPLC (Page 4745)	Yes	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 4745)	Yes	Yes	Yes	Yes
RecipeViewMenu (Page 4746)	Yes	Yes	Yes	Yes
RecipeViewOpen (Page 4746)	Yes	Yes	Yes	Yes
RecipeViewSetDataRecordToPLC (Page 4747)	Yes	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 4748)	Yes	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 4748)	Yes	Yes	Yes	Yes
RecipeViewSynchronizeDataRecordWithTags (Page 4749)	Yes	Yes	Yes	Yes
RecipeViewRenameDataRecord (Page 4749)	Yes	Yes	Yes	Yes
RecipeViewShowOperatorNotes (Page 4750)	Yes	Yes	Yes	Yes
RecipeViewBack (Page 4750)	Yes	Yes	Yes	Yes
ResetBit (Page 4751)	Yes	Yes	Yes	Yes
ResetBitInTag (Page 4752)	Yes	Yes	Yes	Yes
PressButton (Page 4753)	Yes	Yes	Yes	Yes
ReleaseButton (Page 4754)	Yes	Yes	Yes	Yes
ShiftAndMask (Page 4755)	Yes	Yes	Yes	Yes
CloseAllLogs (Page 4757)	Yes	Yes	Yes	Yes
SetDataRecordToPLC (Page 4758)	Yes	Yes	Yes	Yes
SetDataRecordTagsToPLC (Page 4759)	Yes	Yes	Yes	Yes
PageDown (Page 4759)	Yes	Yes	Yes	Yes
PageUp (Page 4760)	Yes	Yes	Yes	Yes
SendEMail (Page 4761)	No	No	No	No
SetAcousticSignal (Page 4762)	No	No	No	No
SetDisplayMode (Page 4762)	No	No	No	No
SetDeviceMode (Page 4763)	Yes	Yes	Yes	Yes
SetBit (Page 4764)	Yes	Yes	Yes	Yes
SetBitInTag (Page 4765)	Yes	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 4767)	Yes	Yes	Yes	Yes
SetBacklightColor (Page 4768)	No	No	No	No
SetBrightness (Page 4769)	Yes	Yes	No	No

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic DP	KTP1200 Basic PN / DP
SetScreenKeyboardMode (Page 4770)	No	No	No	No
SetAlarmReportMode (Page 4772)	No	No	No	No
SetRecipeTags (Page 4772)	Yes	Yes	Yes	Yes
SetDaylightSavingTime (Page 4774)	No	No	No	No
SetLanguage (Page 4775)	Yes	Yes	Yes	Yes
SetTag (Page 4777)	Yes	Yes	Yes	Yes
SetConnectionMode (Page 4778)	Yes	Yes	Yes	Yes
SetWebAccess (Page 4780)	No	No	No	No
BackupRAMFileSystem (Page 4780)	No	No	No	No
SimulateSystemKey (Page 4781)	Yes	Yes	Yes	Yes
SimulateTag (Page 4782)	Yes	Yes	Yes	Yes
SmartClientViewRefresh (Page 4783)	No	No	No	No
SmartClientViewReadOnlyOff (Page 4784)	No	No	No	No
SmartClientViewReadOnlyOn (Page 4784)	No	No	No	No
SmartClientViewDisconnect (Page 4785)	No	No	No	No
SmartClientViewConnect (Page 4785)	No	No	No	No
SmartClientViewLeave (Page 4786)	No	No	No	No
SaveDataRecord (Page 4787)	Yes	Yes	Yes	Yes
StartLogging (Page 4788)	Yes	Yes	Yes	Yes
StartNextLog (Page 4789)	No	No	No	No
StartProgram (Page 4789)	No	No	No	No
StatusForceGetValues (Page 4791)	No	No	No	No
StatusForceSetValues (Page 4792)	No	No	No	No
ControlSmartServer (Page 4792)	No	No	No	No
ControlWebServer (Page 4793)	No	No	No	No
StopLogging (Page 4794)	Yes	Yes	Yes	Yes
StopRuntime (Page 4795)	Yes	Yes	Yes	Yes
ActivateSystemDiagnosticsView (Page 4796)	No	No	No	No
LookupText (Page 4799)	No	No	No	No
ResetTagToHandWheel (Page 4800)	No	No	No	No
SetTagToHandWheel (Page 4800)	No	No	No	No
TraceUserChange (Page 4801)	Yes	Yes	Yes	Yes
DecreaseFocusedValue (Page 4801)	Yes	Yes	Yes	Yes
DecreaseTag (Page 4802)	Yes	Yes	Yes	Yes
ChangeConnection (Page 4803)	Yes	Yes	Yes	Yes
WinACMPUpdateStartupCharacteristics (Page 4811)	No	No	No	No
WinACMPUpdateKeySwitchSetting (Page 4812)	No	No	No	No
WinACMPUpdateBUSF1LED (Page 4813)	No	No	No	No
WinACMPUpdateBUSF2LED (Page 4814)	No	No	No	No

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic DP	KTP1200 Basic PN / DP
WinACMPUpdateAverageExecTime (Page 4815)	No	No	No	No
WinACMPUpdateAverageCycleTime (Page 4816)	No	No	No	No
WinACMPUpdateEXTFLED (Page 4817)	No	No	No	No
WinACMPUpdateHMIEnableTime (Page 4818)	No	No	No	No
WinACMPUpdateINTFLED (Page 4819)	No	No	No	No
WinACMPUpdateLastCycleTime (Page 4820)	No	No	No	No
WinACMPUpdateMaximumCycleTime (Page 4821)	No	No	No	No
WinACMPUpdateMinimumCycleTime (Page 4822)	No	No	No	No
WinACMPUpdatePowerLED (Page 4823)	No	No	No	No
WinACMPUpdateSleepTime (Page 4824)	No	No	No	No
WinACMPUpdateRUNLED (Page 4825)	No	No	No	No
WinACMPUpdateSTOPLED (Page 4826)	No	No	No	No
WinACMPArchive (Page 4827)	No	No	No	No
WinACMPGetStartMode (Page 4828)	No	No	No	No
WinACMPGetVersion (Page 4828)	No	No	No	No
WinACMPClearCycleTimeBuffer (Page 4829)	No	No	No	No
WinACMPSetStartAtBoot (Page 4829)	No	No	No	No
WinACMPSetKeySwitch (Page 4830)	No	No	No	No
WinACMPSetHMIExecutionTime (Page 4831)	No	No	No	No
WinACMPSetResetMethod (Page 4831)	No	No	No	No
WinACMPSetSleeptime (Page 4832)	No	No	No	No
WinACMPSetStartMode (Page 4832)	No	No	No	No
WinACMPStartHistogram (Page 4833)	No	No	No	No
WinACMPControl (Page 4834)	No	No	No	No
WinACMPStopHistogram (Page 4834)	No	No	No	No
WinACMPRestore (Page 4835)	No	No	No	No
ShowLogonDialog (Page 4804)	Yes	Yes	Yes	Yes
ShowOperatorNotes (Page 4805)	Yes	Yes	Yes	Yes
ShowAlarmWindow (Page 4806)	Yes	Yes	Yes	Yes
ShowPopupScreen (Page 4807)	No	No	No	No
ShowSlideInScreen (Page 4808)	No	No	No	No
ShowSoftwareVersion (Page 4809)	No	No	No	No
ShowSystemDiagnosticsWindow (Page 4809)	No	No	No	No
ShowSystemEvent (Page 4810)	No	No	No	No

1) Only for editing the IP settings

System functions for Panels

Availability of system functions

The following table shows the availability of the system functions on the Panels.

Technical data subject to change.

Overview

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
User-defined functions	No	No	No	No	No	Yes
Logoff (Page 4657)	Yes	Yes	Yes	Yes	Yes	Yes
ActivateScreen (Page 4657)	Yes	Yes	Yes	Yes	Yes	Yes
ActivateScreenByNumber (Page 4659)	Yes	Yes	Yes	Yes	Yes	Yes
ActivateCleanScreen (Page 4660)	No	No	No	Yes	Yes [4)]	Yes [2)]
ActivateSystemDiagnosticsView (Page 4796)	No	No	No	No	No	No
ActivatePreviousScreen (Page 4660)	Yes	Yes	Yes	Yes	Yes	Yes
UpdateTag (Page 4661)	Yes	Yes	No	Yes	Yes	Yes
AdjustContrast (Page 4662)	Yes	Yes	Yes	Yes	Yes	No
Logon (Page 4663)	Yes	Yes	Yes	Yes	Yes	Yes
ArchiveLogFile (Page 4663)	No	No	No	No	No	Yes
LogTag (Page 4665)	No	No	No	No	No	Yes
EditAlarm (Page 4666)	Yes	Yes	Yes	No	Yes	Yes [1)]
ScreenObjectCursorDown (Page 4666)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorUp (Page 4667)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorLeft (Page 4668)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorRight (Page 4668)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageDown (Page 4669)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageUp (Page 4670)	Yes	Yes	Yes	Yes	Yes	Yes
Encode (Page 4671)	No	No	Yes	Yes	Yes	Yes
EncodeEx (Page 4672)	No	No	Yes	Yes	Yes	Yes
DirectKey (Page 4673)	No	No	No	No	Yes [3)]	Yes [2)]
DirectKeyScreenNumber (Page 4675)	No	No	No	No	Yes [3)]	Yes [2)]
PrintScreen (Page 4676)	No	No	Yes	No	Yes	Yes
PrintReport (Page 4677)	No	No	Yes	No	Yes	Yes

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
NotifyUserAction (Page 4677)	No	No	No	No	No	Yes
IncreaseFocusedValue (Page 4679)	Yes	Yes	Yes	No	Yes [1)]	Yes [1)]
IncreaseTag (Page 4679)	Yes	Yes	Yes	Yes	Yes	Yes
ExportDataRecords (Page 4680)	No	No	Yes	No	Yes	Yes
ExportDataRecordsWithChecksum (Page 4683)	No	No	No	No	No	Yes
ExportImportUserAdministration (Page 4685)	No	No	Yes	No	Yes	Yes
GoToHome (Page 4686)	Yes	Yes	Yes	No	Yes [1)]	Yes [1)]
GoToEnd (Page 4686)	Yes	Yes	Yes	No	Yes [1)]	Yes [1)]
SafelyRemoveHardware (Page 4687)	No	No	No	No	Yes [5)]	No
HTMLBrowserStop (Page 4688)	No	No	No	No	No	No
HTMLBrowserRefresh (Page 4689)	No	No	No	No	No	No
HTMLBrowserForward (Page 4695)	No	No	No	No	No	No
HTMLBrowserBack (Page 4695)	No	No	No	No	No	No
ImportDataRecords (Page 4696)	No	No	Yes	No	Yes	Yes
ImportDataRecordsWithChecksum (Page 4698)	No	No	No	No	No	Yes
InvertBit (Page 4699)	Yes	Yes	Yes	Yes	Yes	Yes
InvertBitInTag (Page 4700)	Yes	Yes	Yes	Yes	Yes	Yes
InvertLinearScaling (Page 4701)	Yes	Yes	Yes	Yes	Yes	Yes
CalibrateTouchScreen (Page 4703)	No	No	No	Yes	Yes	Yes [2)]
CopyLog (Page 4704)	No	No	No	No	No	Yes
TrendViewScrollForward (Page 4705)	No	No	No	Yes	Yes	Yes
TrendViewScrollBack (Page 4705)	No	No	No	Yes	Yes	Yes
TrendViewExtend (Page 4706)	No	No	No	Yes	Yes	Yes
TrendViewCompress (Page 4706)	No	No	No	Yes	Yes	Yes
TrendViewRulerLeft (Page 4707)	No	No	No	Yes	Yes	Yes
TrendViewRulerRight (Page 4708)	No	No	No	Yes	Yes	Yes
TrendViewSetRulerMode (Page 4708)	No	No	No	Yes	Yes	Yes
TrendViewStartStop (Page 4709)	No	No	No	Yes	Yes	Yes
TrendViewBackToBeginning (Page 4709)	No	No	No	Yes	Yes	Yes
LoadDataRecord (Page 4710)	No	No	No	No	Yes	Yes
GetUserName (Page 4711)	Yes	Yes	Yes	Yes	Yes	Yes
GetDataRecordFromPLC (Page 4712)	No	No	No	No	Yes	Yes
GetDataRecordName (Page 4713)	No	No	No	No	Yes	Yes

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
GetDataRecordTagsFromPLC (Page 4715)	No	No	No	No	Yes	Yes
GetGroupNumber (Page 4716)	Yes	Yes	Yes	Yes	Yes	Yes
GetBrightness (Page 4716)	No	No	No	No	No	No
GetPassword (Page 4717)	Yes	Yes	Yes	Yes	Yes	Yes
LinearScaling (Page 4718)	Yes	Yes	Yes	Yes	Yes	Yes
ClearLog (Page 4720)	No	No	No	No	No	Yes
DeleteDataRecord (Page 4721)	No	No	No	No	Yes	Yes
ClearDataRecordMemory (Page 4722)	No	No	No	No	Yes	Yes
ClearAlarmBuffer (Page 4723)	Yes	Yes	Yes	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 4724)	Yes	Yes	Yes	No	Yes	Yes
AlarmViewUpdate (Page 4725)	No	No	Yes	No	Yes	Yes
AlarmViewLoopInAlarm (Page 4725)	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 4726)	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 4727)	Yes	Yes	Yes	Yes	Yes	Yes
OpenAllLogs (Page 4728)	No	No	No	No	No	Yes
OpenScreenKeyboard (Page 4729)	No	No	No	No	No	Yes
OpenControlPanelDialog (Page 4730)	No	No	No	No	No	No
OpenCommandPrompt (Page 4731)	No	No	No	No	No	Yes
OpenInternetExplorer (Page 4732)	No	No	No	No	Yes [3]	Yes
OpenControlPanel (Page 4733)	No	No	No	No	Yes	Yes
OpenTaskManager (Page 4733)	No	No	No	No	No	Yes
AcknowledgeAlarm (Page 4744)	Yes	Yes	Yes	No	Yes [1] [4]	Yes [1]
RecipeViewNewDataRecord (Page 4744)	No	Yes	Yes	Yes	Yes	Yes
RecipeViewGetDataRecord-FromPLC (Page 4745)	No	Yes	Yes	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 4745)	No	Yes	Yes	Yes	Yes	Yes
RecipeViewMenu (Page 4746)	No	Yes	Yes	Yes	Yes	Yes
RecipeViewOpen (Page 4746)	No	Yes	Yes	Yes	Yes	Yes
RecipeViewSetDataRecordToPLC (Page 4747)	No	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 4748)	No	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 4748)	No	Yes	Yes	Yes	Yes	Yes

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
RecipeViewSynchronizeDataRecordWithTags (Page 4749)	No	No	No	No	Yes	Yes
RecipeViewRenameDataRecord (Page 4749)	No	Yes	Yes	Yes	Yes	Yes
RecipeViewShowOperatorNotes (Page 4750)	No	Yes	Yes	Yes	Yes	Yes
RecipeViewBack (Page 4750)	No	Yes	Yes	Yes	Yes	Yes
ResetBit (Page 4751)	Yes	Yes	Yes	Yes	Yes	Yes
ResetBitInTag (Page 4752)	Yes	Yes	Yes	Yes	Yes	Yes
PressButton (Page 4753)	Yes	Yes	Yes	No	Yes)	Yes)
ReleaseButton (Page 4754)	Yes	Yes	Yes	No	Yes	Yes 1)
ShiftAndMask (Page 4755)	Yes	Yes	Yes	Yes	Yes	Yes
CloseAllLogs (Page 4757)	No	No	No	No	No	Yes
SetDataRecordToPLC (Page 4758)	No	No	No	No	Yes	Yes
SetDataRecordTagsToPLC (Page 4759)	No	No	No	No	Yes	Yes
PageDown (Page 4759)	Yes	Yes	Yes	No	Yes 1) 4)	Yes 1)
PageUp (Page 4760)	Yes	Yes	Yes	No	yes 1).4)	Yes 1)
SendEMail (Page 4761)	Yes	Yes	Yes	No	Yes 3)	Yes
SetAcousticSignal (Page 4762)	No	No	No	No	No	No
SetDisplayMode (Page 4762)	No	No	No	No	No	No
SetDeviceMode (Page 4763)	Yes	Yes	Yes	Yes	Yes	Yes
SetBit (Page 4764)	Yes	Yes	Yes	Yes	Yes	Yes
SetBitInTag (Page 4765)	Yes	Yes	Yes	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 4767)	Yes	Yes	Yes	Yes	Yes	Yes
SetBacklightColor (Page 4768)	No	No	No	No	No	No
SetBrightness (Page 4769)	No	No	No	No	No	No
SetScreenKeyboardMode (Page 4770)	No	No	No	No	No	Yes
SetAlarmReportMode (Page 4772)	No	No	Yes	No	Yes	Yes
SetRecipeTags (Page 4772)	No	No	Yes	No	Yes	Yes
SetDaylightSavingTime (Page 4774)	No	No	No	No	No	No
SetLanguage (Page 4775)	Yes	Yes	Yes	Yes	Yes	Yes
SetTag (Page 4777)	Yes	Yes	Yes	Yes	Yes	Yes
SetConnectionMode (Page 4778)	Yes	Yes	Yes	Yes	Yes	Yes
SetWebAccess (Page 4780)	No	No	No	No	Yes	Yes
BackupRAMFileSystem (Page 4780)	No	No	No	No	No	No
SimulateSystemKey (Page 4781)	No	No	Yes	No	Yes 1) 4)	Yes 1)
SimulateTag (Page 4782)	Yes	Yes	Yes	Yes	Yes	Yes

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
SmartClientViewRefresh (Page 4783)	No	No	No	No	Yes [3]	Yes
SmartClientViewReadOnlyOff (Page 4784)	No	No	No	No	Yes [3]	Yes
SmartClientViewReadOnlyOn (Page 4784)	No	No	No	No	Yes [3]	Yes
SmartClientViewDisconnect (Page 4785)	No	No	No	No	Yes [3]	Yes
SmartClientViewConnect (Page 4785)	No	No	No	No	Yes [3]	Yes
SmartClientViewLeave (Page 4786)	No	No	No	No	Yes [3]	Yes
SaveDataRecord (Page 4787)	No	No	No	No	Yes	Yes
StartLogging (Page 4788)	No	No	No	No	No	Yes
StartNextLog (Page 4789)	No	No	No	No	No	Yes
StartProgram (Page 4789)	No	No	No	No	No	Yes
StatusForceGetValues (Page 4791)	No	No	No	No	Yes	Yes
StatusForceSetValues (Page 4792)	No	No	No	No	Yes	Yes
ControlSmartServer (Page 4792)	No	No	No	No	Yes [3]	Yes
ControlWebServer (Page 4793)	No	No	No	No	Yes [3]	Yes
StopLogging (Page 4794)	No	No	No	No	No	Yes
StopRuntime (Page 4795)	Yes	Yes	Yes	Yes	Yes	Yes
LookupText (Page 4799)	No	No	Yes	No	Yes	Yes
ResetTagToHandWheel (Page 4800)	No	No	No	No	No	No
SetTagToHandWheel (Page 4800)	No	No	No	No	No	No
TraceUserChange (Page 4801)	Yes	Yes	Yes	Yes	Yes	Yes
DecreaseFocusedValue (Page 4801)	Yes	Yes	Yes	No	yes [1][4]	Yes [1]
DecreaseTag (Page 4802)	Yes	Yes	Yes	Yes	Yes	Yes
ChangeConnection (Page 4803)	Yes	Yes	Yes	Yes	Yes	Yes
WinACMPUpdateControllerForStartAtBoot (Page 4811)	No	No	No	No	No	No
WinACMPUpdateKeySwitchSetting (Page 4812)	No	No	No	No	No	No
WinACMPUpdateBUSF1LED (Page 4813)	No	No	No	No	No	No
WinACMPUpdateBUSF2LED (Page 4814)	No	No	No	No	No	No
WinACMPUpdateAverageExecTime (Page 4815)	No	No	No	No	No	No
WinACMPUpdateAverageCycleTime (Page 4816)	No	No	No	No	No	No
WinACMPUpdateEXTFLED (Page 4817)	No	No	No	No	No	No

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
WinACMPUpdateHMIEnableTime (Page 4818)	No	No	No	No	No	No
WinACMPUpdateINTFLED (Page 4819)	No	No	No	No	No	No
WinACMPUpdateLastCycleTime (Page 4820)	No	No	No	No	No	No
WinACMPUpdateMaximumCycle-Time (Page 4821)	No	No	No	No	No	No
WinACMPUpdateMinimumCycle-Time (Page 4822)	No	No	No	No	No	No
WinACMPUpdatePowerLED (Page 4823)	No	No	No	No	No	No
WinACMPUpdateSleepTime (Page 4824)	No	No	No	No	No	No
WinACMPUpdateRUNLED (Page 4825)	No	No	No	No	No	No
WinACMPUpdateSTOPLED (Page 4826)	No	No	No	No	No	No
WinACMPArchive (Page 4827)	No	No	No	No	No	No
WinACMPGetStartMode (Page 4828)	No	No	No	No	No	No
WinACMPGetVersion (Page 4828)	No	No	No	No	No	No
WinACMPClearCycleTimeBuffer (Page 4829)	No	No	No	No	No	No
WinACMPSetStartAtBoot (Page 4829)	No	No	No	No	No	No
WinACMPSetKeySwitch (Page 4830)	No	No	No	No	No	No
WinACMPSetHMIEnableTime (Page 4831)	No	No	No	No	No	No
WinACMPSetResetMethod (Page 4831)	No	No	No	No	No	No
WinACMPSetSleepTime (Page 4832)	No	No	No	No	No	No
WinACMPSetStartMode (Page 4832)	No	No	No	No	No	No
WinACMPStartHistogram (Page 4833)	No	No	No	No	No	No
WinACMPControl (Page 4834)	No	No	No	No	No	No
WinACMPStopHistogram (Page 4834)	No	No	No	No	No	No
WinACMPRestore (Page 4835)	No	No	No	No	No	No
ShowLogonDialog (Page 4804)	Yes	Yes	Yes	Yes	Yes	Yes
ShowOperatorNotes (Page 4805)	Yes	Yes	Yes	Yes	Yes	Yes
ShowAlarmWindow (Page 4806)	Yes	Yes	Yes	Yes	Yes	Yes

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
ShowSoftwareVersion (Page 4809)	No	No	No	No	No	Yes
ShowSystemDiagnosticsWindow (Page 4809)	No	No	No	No	No	No
ShowSystemAlarm (Page 4810)	No	No	Yes	No	Yes	Yes

1) only with keyboard units
2) only with touch screen devices
3) only with PROFINET Bus (PN devices)
4) only with TP 177B
5) Only for TP 177B 4" PN/DP

System functions for Multi Panels

Availability of system functions

The following table shows the availability of the system functions on the Multi Panels.

Technical data subject to change.

Overview

	MP 177	MP 277	MP 377
User-defined functions	No	Yes	Yes
Logoff (Page 4657)	Yes	Yes	Yes
ActivateScreen (Page 4657)	Yes	Yes	Yes
ActivateScreenByNumber (Page 4659)	Yes	Yes	Yes
ActivateCleanScreen (Page 4660)	Yes 2)	Yes 2)	Yes 2)
ActivatePreviousScreen (Page 4660)	Yes	Yes	Yes
UpdateTag (Page 4661)	Yes	Yes	Yes
AdjustContrast (Page 4662)	No	No	No
Logon (Page 4663)	Yes	Yes	Yes
ArchiveLogFile (Page 4663)	No	Yes	Yes
LogTag (Page 4665)	No	Yes	Yes
EditAlarm (Page 4666)	No	Yes 1)	No
ScreenObjectCursorDown (Page 4666)	Yes	Yes	Yes
ScreenObjectCursorUp (Page 4667)	Yes	Yes	Yes
ScreenObjectPageDown (Page 4669)	Yes	Yes	Yes
ScreenObjectPageUp (Page 4670)	Yes	Yes	Yes
Encode (Page 4671)	Yes	Yes	Yes
EncodeEx (Page 4672)	Yes	Yes	Yes

	MP 177	MP 277	MP 377
DirectKey (Page 4673)	Yes [2]	Yes [2]	Yes [2]
DirectKeyScreenNumber (Page 4675)	Yes [2]	Yes [2]	Yes [2]
PrintScreen (Page 4676)	Yes	Yes	Yes
PrintReport (Page 4677)	Yes	Yes	Yes
NotifyUserAction (Page 4677)	No	Yes	Yes
IncreaseFocusedValue (Page 4679)	Yes [1]	Yes [1]	Yes [1]
IncreaseTag (Page 4679)	Yes	Yes	Yes
ExportDataRecords (Page 4680)	Yes	Yes	Yes
ExportDataRecordsWithChecksum (Page 4683)	Yes	Yes	Yes
ExportImportUserAdministration (Page 4685)	Yes	Yes	Yes
GoToHome (Page 4686)	Yes [1]	Yes [1]	Yes [1]
GoToEnd (Page 4686)	Yes [1]	Yes [1]	Yes [1]
SafelyRemoveHardware (Page 4687)	No	Yes	Yes
HTMLBrowserStop (Page 4688)	No	No	No
HTMLBrowserRefresh (Page 4689)	No	No	No
HTMLBrowserForward (Page 4695)	No	No	No
HTMLBrowserBack (Page 4695)	No	No	No
ImportDataRecords (Page 4696)	Yes	Yes	Yes
ImportDataRecordsWithChecksum (Page 4698)	Yes	Yes	Yes
InvertBit (Page 4699)	Yes	Yes	Yes
InvertBitInTag (Page 4700)	Yes	Yes	Yes
InvertLinearScaling (Page 4701)	Yes	Yes	Yes
CalibrateTouchScreen (Page 4703)	Yes [2]	Yes [2]	Yes [2]
CopyLog (Page 4704)	No	Yes	Yes
TrendViewScrollForward (Page 4705)	Yes	Yes	Yes
TrendViewScrollBack (Page 4705)	Yes	Yes	Yes
TrendViewExtend (Page 4706)	Yes	Yes	Yes
TrendViewCompress (Page 4706)	Yes	Yes	Yes
TrendViewRulerLeft (Page 4707)	Yes	Yes	Yes
TrendViewRulerRight (Page 4708)	Yes	Yes	Yes
TrendViewSetRulerMode (Page 4708)	Yes	Yes	Yes
TrendViewStartStop (Page 4709)	Yes	Yes	Yes
TrendViewBackToBeginning (Page 4709)	Yes	Yes	Yes
LoadDataRecord (Page 4710)	Yes	Yes	Yes
GetUserName (Page 4711)	Yes	Yes	Yes
GetDataRecordFromPLC (Page 4712)	Yes	Yes	Yes
GetDataRecordName (Page 4713)	Yes	Yes	Yes

	MP 177	MP 277	MP 377
GetDataRecordTagsFromPLC (Page 4715)	Yes	Yes	Yes
GetGroupNumber (Page 4716)	Yes	Yes	Yes
GetBrightness (Page 4716)	No	No	No
GetPassword (Page 4717)	Yes	Yes	Yes
LinearScaling (Page 4718)	Yes	Yes	Yes
ClearLog (Page 4720)	No	Yes	Yes
DeleteDataRecord (Page 4721)	Yes	Yes	Yes
ClearDataRecordMemory (Page 4722)	Yes	Yes	Yes
ClearAlarmBuffer (Page 4723)	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 4724)	Yes	Yes	Yes
AlarmViewUpdate (Page 4725)	Yes	Yes	Yes
AlarmViewLoopInAlarm (Page 4725)	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 4726)	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 4727)	Yes	Yes	Yes
OpenAllLogs (Page 4728)	No	Yes	Yes
OpenScreenKeyboard (Page 4729)	No	Yes	Yes
OpenControlPanelDialog (Page 4730)	No	No	No
OpenCommandPrompt (Page 4731)	No	Yes	Yes
OpenInternetExplorer (Page 4732)	No	Yes	Yes
OpenControlPanel (Page 4733)	Yes	Yes	Yes
OpenTaskManager (Page 4733)	No	Yes	Yes
AcknowledgeAlarm (Page 4744)	Yes [1)]	Yes [1)]	Yes [1)]
RecipeViewNewDataRecord (Page 4744)	Yes	Yes	Yes
RecipeViewGetDataRecordFromPLC (Page 4745)	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 4745)	Yes	Yes	Yes
RecipeViewMenu (Page 4746)	Yes	Yes	Yes
RecipeViewOpen (Page 4746)	Yes	Yes	Yes
RecipeViewSetDataRecordToPLC (Page 4747)	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 4748)	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 4748)	Yes	Yes	Yes
RecipeViewSynchronizeDataRecord-WithTags (Page 4749)	Yes	Yes	Yes
RecipeViewRenameDataRecord (Page 4749)	Yes	Yes	Yes

	MP 177	MP 277	MP 377
RecipeViewShowOperatorNotes (Page 4750)	Yes	Yes	Yes
RecipeViewBack (Page 4750)	Yes	Yes	Yes
ResetBit (Page 4751)	Yes	Yes	Yes
ResetBitInTag (Page 4752)	Yes	Yes	Yes
PressButton (Page 4753)	Yes	Yes	Yes)
ReleaseButton (Page 4754)	Yes)	Yes)	Yes)
ShiftAndMask (Page 4755)	Yes	Yes	Yes
CloseAllLogs (Page 4757)	No	Yes	Yes
SetDataRecordToPLC (Page 4758)	Yes	Yes	Yes
SetDataRecordTagsToPLC (Page 4759)	Yes	Yes	Yes
PageDown (Page 4759)	Yes 1)	Yes 1)	Yes 1)
PageUp (Page 4760)	Yes 1)	Yes 1)	Yes 1)
SendEMail (Page 4761)	Yes	Yes	Yes
SetAcousticSignal (Page 4762)	No	Yes 2)	Yes 2)
SetDisplayMode (Page 4762)	No	No	No
SetDeviceMode (Page 4763)	Yes	Yes	Yes
SetBit (Page 4764)	Yes	Yes	Yes
SetBitInTag (Page 4765)	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 4767)	Yes	Yes 1)	Yes 1)
SetBacklightColor (Page 4768)	No	No	No
SetBrightness (Page 4769)	No	No	No
SetScreenKeyboardMode (Page 4770)	No	Yes	Yes
SetAlarmReportMode (Page 4772)	Yes	Yes	Yes
SetRecipeTags (Page 4772)	Yes	Yes	Yes
SetDaylightSavingTime (Page 4774)	No	Yes	Yes
SetLanguage (Page 4775)	Yes	Yes	Yes
SetAndGetBrightness (Page 4776)	No	No	Yes3)
SetTag (Page 4777)	Yes	Yes	Yes
SetConnectionMode (Page 4778)	Yes	Yes	Yes
SetWebAccess (Page 4780)	Yes	Yes	Yes
BackupRAMFileSystem (Page 4780)	No	Yes	Yes
SimulateSystemKey (Page 4781)	No	Yes 1)	Yes 1)
SimulateTag (Page 4782)	Yes	Yes	Yes
SmartClientViewRefresh (Page 4783)	Yes	Yes	Yes
SmartClientViewReadOnlyOff (Page 4784)	Yes	Yes	Yes
SmartClientViewReadOnlyOn (Page 4784)	Yes	Yes	Yes
SmartClientViewDisconnect (Page 4785)	Yes	Yes	Yes

	MP 177	MP 277	MP 377
SmartClientViewConnect (Page 4785)	Yes	Yes	Yes
SmartClientViewLeave (Page 4786)	Yes	Yes	Yes
SaveDataRecord (Page 4787)	Yes	Yes	Yes
StartLogging (Page 4788)	No	Yes	Yes
StartNextLog (Page 4789)	No	Yes	Yes
StartProgram (Page 4789)	Yes	Yes	Yes
StatusForceGetValues (Page 4791)	Yes	Yes	Yes
StatusForceSetValues (Page 4792)	Yes	Yes	Yes
ControlSmartServer (Page 4792)	Yes	Yes	Yes
ControlWebServer (Page 4793)	Yes	Yes	Yes
StopLogging (Page 4794)	No	Yes	Yes
StopRuntime (Page 4795)	Yes	Yes	Yes
ActivateSystemDiagnosticsView (Page 4796)	No	No	No
LookupText (Page 4799)	Yes	Yes	Yes
ResetTagToHandWheel (Page 4800)	No	No	No
SetTagToHandWheel (Page 4800)	No	No	No
TraceUserChange (Page 4801)	Yes	Yes	Yes
DecreaseFocusedValue (Page 4801)	Yes [1)]	Yes [1)]	Yes [1)]
DecreaseTag (Page 4802)	Yes	Yes	Yes
ChangeConnection (Page 4803)	Yes	Yes	Yes
WinACMPUpdateControllerForStartAt-Boot (Page 4811)	Yes	Yes	Yes
WinACMPUpdateKeySwitchSetting (Page 4812)	Yes	Yes	Yes
WinACMPUpdateBUSF1LED (Page 4813)	Yes	Yes	Yes
WinACMPUpdateBUSF2LED (Page 4814)	Yes	Yes	Yes
WinACMPUpdateAverageExecTime (Page 4815)	Yes	Yes	Yes
WinACMPUpdateAverageCycleTime (Page 4816)	Yes	Yes	Yes
WinACMPUpdateEXTFLED (Page 4817)	Yes	Yes	Yes
WinACMPUpdateHMIEnableTime (Page 4818)	Yes	Yes	Yes
WinACMPUpdateINTFLED (Page 4819)	Yes	Yes	Yes
WinACMPUpdateLastCycleTime (Page 4820)	Yes	Yes	Yes
WinACMPUpdateMaximumCycleTime (Page 4821)	Yes	Yes	Yes

	MP 177	MP 277	MP 377
WinACMPUpdateMinimumCycleTime (Page 4822)	Yes	Yes	Yes
WinACMPUpdatePowerLED (Page 4823)	Yes	Yes	Yes
WinACMPUpdateSleepTime (Page 4824)	Yes	Yes	Yes
WinACMPUpdateRUNLED (Page 4825)	Yes	Yes	Yes
WinACMPUpdateSTOPLED (Page 4826)	Yes	Yes	Yes
WinACMPArchive (Page 4827)	Yes	Yes	Yes
WinACMPGetStartMode (Page 4828)	Yes	Yes	Yes
WinACMPGetVersion (Page 4828)	Yes	Yes	Yes
WinACMPClearCycleTimeBuffer (Page 4829)	Yes	Yes	Yes
WinACMPSetStartAtBoot (Page 4829)	Yes	Yes	Yes
WinACMPSetKeySwitch (Page 4830)	Yes	Yes	Yes
WinACMPSetHMIEnableTime (Page 4831)	Yes	Yes	Yes
WinACMPSetResetMethod (Page 4831)	Yes	Yes	Yes
WinACMPSetSleepTime (Page 4832)	Yes	Yes	Yes
WinACMPSetStartMode (Page 4832)	Yes	Yes	Yes
WinACMPStartHistogram (Page 4833)	Yes	Yes	Yes
WinACMPControl (Page 4834)	Yes	Yes	Yes
WinACMPStopHistogram (Page 4834)	Yes	Yes	Yes
WinACMPRestore (Page 4835)	Yes	Yes	Yes
ShowLogonDialog (Page 4804)	Yes	Yes	Yes
ShowOperatorNotes (Page 4805)	Yes	Yes	Yes
ShowAlarmWindow (Page 4806)	Yes	Yes	Yes
ShowSoftwareVersion (Page 4809)	No	Yes	Yes
ShowSystemDiagnosticsWindow (Page 4809)	No	No	No
ShowSystemAlarm (Page 4810)	Yes	Yes	Yes

1) only with keyboard units
2) only with touch screen devices
3) only with MP 377 Touch daylight readable

System functions for Comfort Panels

Availability of system functions

The following table shows the availability of the system functions on the Comfort Panels.

Technical data subject to change.

Overview

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
User-defined functions	Yes	Yes	Yes
Logoff (Page 4657)	Yes	Yes	Yes
ActivateScreen (Page 4657)	Yes	Yes	Yes
ActivateScreenByNumber (Page 4659)	Yes	Yes	Yes
ActivateCleanScreen (Page 4660)	Yes [1)]	Yes	No
ActivatePreviousScreen (Page 4660)	Yes	Yes	Yes
UpdateTag (Page 4661)	Yes	Yes	Yes
AdjustContrast (Page 4662)	No	No	No
Logon (Page 4663)	Yes	Yes	Yes
ArchiveLogFile (Page 4663)	Yes	Yes	Yes
LogTag (Page 4665)	Yes	Yes	Yes
EditAlarm (Page 4666)	Yes	No	Yes
ScreenObjectCursorDown (Page 4666)	Yes	Yes	Yes
ScreenObjectCursorUp (Page 4667)	Yes	Yes	Yes
ScreenObjectPageDown (Page 4669)	Yes	Yes	Yes
ScreenObjectPageUp (Page 4670)	Yes	Yes	Yes
Encode (Page 4671)	Yes	Yes	Yes
EncodeEx (Page 4672)	Yes	Yes	Yes
DirectKey (Page 4673)	Yes [1)]	Yes	No
DirectKeyScreenNumber (Page 4675)	Yes [1)]	Yes	No
PrintScreen (Page 4676)	Yes	Yes	Yes
PrintReport (Page 4677)	Yes	Yes	Yes
NotifyUserAction (Page 4677)	Yes	Yes	Yes
IncreaseFocusedValue (Page 4679)	Yes	No	Yes
IncreaseTag (Page 4679)	Yes	Yes	Yes
ExportDataRecords (Page 4680)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
ExportDataRecordsWithChecksum (Page 4683)	Yes	Yes	Yes
ExportImportUserAdministration (Page 4685)	Yes	Yes	Yes
GoToHome (Page 4686)	Yes	No	Yes
GoToEnd (Page 4686)	Yes	No	Yes
SafelyRemoveHardware (Page 4687)	Yes	Yes	Yes
HTMLBrowserStop (Page 4688)	No	Yes	Yes
HTMLBrowserScrollDown (Page 4688)	No	Yes	Yes
HTMLBrowserRefresh (Page 4689)	No	Yes	Yes
HTMLBrowserScrollUp (Page 4689)	No	Yes	Yes
HTMLBrowserForward (Page 4695)	No	Yes	Yes
HTMLBrowserBack (Page 4695)	No	Yes	Yes
HTMLBrowserZoomIn (Page 4690)	No	Yes	Yes
HTMLBrowserZoomOut (Page 4691)	No	Yes	Yes
HTMLBrowserScrollLeft (Page 4691)	No	Yes	Yes
HTMLBrowserScrollRight (Page 4692)	No	Yes	Yes
HTMLBrowserPageDown (Page 4693)	No	Yes	Yes
HTMLBrowserPageUp (Page 4693)	No	Yes	Yes
HTMLBrowserStartPage (Page 4694)	No	Yes	Yes
ImportDataRecords (Page 4696)	Yes	Yes	Yes
ImportDataRecordsWithChecksum (Page 4698)	Yes	Yes	Yes
InvertBit (Page 4699)	Yes	Yes	Yes
InvertBitInTag (Page 4700)	Yes	Yes	Yes
InvertLinearScaling (Page 4701)	Yes	Yes	Yes
CalibrateTouchScreen (Page 4703)	Yes [1)]	Yes	No
CopyLog (Page 4704)	Yes	Yes	Yes
TrendViewScrollForward (Page 4705)	Yes	Yes	Yes
TrendViewScrollBack (Page 4705)	Yes	Yes	Yes
TrendViewExtend (Page 4706)	Yes	Yes	Yes
TrendViewCompress (Page 4706)	Yes	Yes	Yes
TrendViewRulerLeft (Page 4707)	Yes	Yes	Yes
TrendViewRulerRight (Page 4708)	Yes	Yes	Yes
TrendViewSetRulerMode (Page 4708)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
TrendViewStartStop (Page 4709)	Yes	Yes	Yes
TrendViewBackToBeginning (Page 4709)	Yes	Yes	Yes
LoadDataRecord (Page 4710)	Yes	Yes	Yes
GetUserName (Page 4711)	Yes	Yes	Yes
GetDataRecordFromPLC (Page 4712)	Yes	Yes	Yes
GetDataRecordName (Page 4713)	Yes	Yes	Yes
GetDataRecordTagsFromPLC (Page 4715)	Yes	Yes	Yes
GetGroupNumber (Page 4716)	Yes	Yes	Yes
GetBrightness (Page 4716)	Yes	Yes	Yes
GetPassword (Page 4717)	Yes	Yes	Yes
LinearScaling (Page 4718)	Yes	Yes	Yes
ClearLog (Page 4720)	Yes	Yes	Yes
DeleteDataRecord (Page 4721)	Yes	Yes	Yes
ClearDataRecordMemory (Page 4722)	Yes	Yes	Yes
ClearAlarmBuffer (Page 4723)	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 4724)	Yes	Yes	Yes
AlarmViewUpdate (Page 4725)	Yes	Yes	Yes
AlarmViewLoopInAlarm (Page 4725)	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 4726)	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 4727)	Yes	Yes	Yes
OpenAllLogs (Page 4728)	Yes	Yes	Yes
OpenScreenKeyboard (Page 4729)	Yes	Yes	Yes
OpenControlPanelDialog (Page 4730)	Yes	Yes	Yes
OpenCommandPrompt (Page 4731)	Yes	Yes	Yes
OpenInternetExplorer (Page 4732)	Yes	Yes	Yes
OpenControlPanel (Page 4733)	Yes	Yes	Yes
OpenTaskManager (Page 4733)	Yes	Yes	Yes
AcknowledgeAlarm (Page 4744)	Yes	No	Yes
PDFScrollDown (Page 4734)	Yes	Yes	Yes
PDFScrollUp (Page 4735)	Yes	Yes	Yes
PDFFitToWidth (Page 4735)	Yes	Yes	Yes
PDFFitToHeight (Page 4736)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
PDFGoToFirstPage (Page 4737)	Yes	Yes	Yes
PDFGoToLastPage (Page 4737)	Yes	Yes	Yes
PDFGoToNextPage (Page 4738)	Yes	Yes	Yes
PDFGoToPage (Page 4738)	Yes	Yes	Yes
PDFGoToPreviousPage (Page 4739)	Yes	Yes	Yes
PDFZoomIn (Page 4740)	Yes	Yes	Yes
PDFZoomOut (Page 4740)	Yes	Yes	Yes
PDFScrollLeft (Page 4741)	Yes	Yes	Yes
PDFScrollRight (Page 4741)	Yes	Yes	Yes
PDFZoomOriginal (Page 4742)	Yes	Yes	Yes
RecipeViewNewDataRecord (Page 4744)	Yes	Yes	Yes
RecipeViewGetDataRecordFromPLC (Page 4745)	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 4745)	Yes	Yes	Yes
RecipeViewMenu (Page 4746)	Yes	Yes	Yes
RecipeViewOpen (Page 4746)	Yes	Yes	Yes
RecipeViewSetDataRecordToPLC (Page 4747)	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 4748)	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 4748)	Yes	Yes	Yes
RecipeViewSynchronizeDataRecord-WithTags (Page 4749)	Yes	Yes	Yes
RecipeViewRenameDataRecord (Page 4749)	Yes	Yes	Yes
RecipeViewShowOperatorNotes (Page 4750)	Yes	Yes	Yes
RecipeViewBack (Page 4750)	Yes	Yes	Yes
ResetBit (Page 4751)	Yes	Yes	Yes
ResetBitInTag (Page 4752)	Yes	Yes	Yes
PressButton (Page 4753)	Yes	Yes	Yes
ReleaseButton (Page 4754)	Yes	Yes	Yes
ShiftAndMask (Page 4755)	Yes	Yes	Yes
CloseAllLogs (Page 4757)	Yes	Yes	Yes
SetDataRecordToPLC (Page 4758)	Yes	Yes	Yes
SetDataRecordTagsToPLC (Page 4759)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
PageDown (Page 4759)	Yes	No	Yes
PageUp (Page 4760)	Yes	No	Yes
SendEMail (Page 4761)	Yes	Yes	Yes
SetAcousticSignal (Page 4762)	Yes [1)]	Yes	No
SetDisplayMode (Page 4762)	No	No	No
SetDeviceMode (Page 4763)	Yes	Yes	Yes
SetBit (Page 4764)	Yes	Yes	Yes
SetBitInTag (Page 4765)	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 4767)	Yes	Yes	Yes
SetBacklightColor (Page 4768)	No	No	No
SetBrightness (Page 4769)	Yes	Yes	Yes
SetScreenKeyboardMode (Page 4770)	Yes	Yes	Yes
SetAlarmReportMode (Page 4772)	Yes	Yes	Yes
SetRecipeTags (Page 4772)	Yes	Yes	Yes
SetDaylightSavingTime (Page 4774)	Yes	Yes	Yes
SetLanguage (Page 4775)	Yes	Yes	Yes
SetTag (Page 4777)	Yes	Yes	Yes
SetConnectionMode (Page 4778)	Yes	Yes	Yes
SetWebAccess (Page 4780)	Yes	Yes	Yes
BackupRAMFileSystem (Page 4780)	Yes	Yes	Yes
SimulateSystemKey (Page 4781)	Yes	No	Yes
SimulateTag (Page 4782)	Yes	Yes	Yes
SmartClientViewRefresh (Page 4783)	Yes	Yes	Yes
SmartClientViewReadOnlyOff (Page 4784)	Yes	Yes	Yes
SmartClientViewReadOnlyOn (Page 4784)	Yes	Yes	Yes
SmartClientViewDisconnect (Page 4785)	Yes	Yes	Yes
SmartClientViewConnect (Page 4785)	Yes	Yes	Yes
SmartClientViewLeave (Page 4786)	Yes	Yes	Yes
SaveDataRecord (Page 4787)	Yes	Yes	Yes
StartLogging (Page 4788)	Yes	Yes	Yes
StartNextLog (Page 4789)	Yes	Yes	Yes
StartProgram (Page 4789)	Yes	Yes	Yes
StatusForceGetValues (Page 4791)	Yes	Yes	Yes
StatusForceSetValues (Page 4792)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
ControlSmartServer (Page 4792)	Yes	Yes	Yes
ControlWebServer (Page 4793)	Yes	Yes	Yes
StopLogging (Page 4794)	Yes	Yes	Yes
StopRuntime (Page 4795)	Yes	Yes	Yes
ActivateSystemDiagnosticsView (Page 4796)	Yes	Yes	Yes
LookupText (Page 4799)	Yes	Yes	Yes
ResetTagToHandWheel (Page 4800)	No	No	No
SetTagToHandWheel (Page 4800)	No	No	No
TraceUserChange (Page 4801)	Yes	Yes	Yes
DecreaseFocusedValue (Page 4801)	Yes	No	Yes
DecreaseTag (Page 4802)	Yes	Yes	Yes
ChangeConnection (Page 4803)	Yes	Yes	Yes
WinACMPUpdateControllerForStartAtBoot (Page 4811)	No	No	No
WinACMPUpdateKeySwitchSetting (Page 4812)	No	No	No
WinACMPUpdateBUSF1LED (Page 4813)	No	No	No
WinACMPUpdateBUSF2LED (Page 4814)	No	No	No
WinACMPUpdateAverageExecTime (Page 4815)	No	No	No
WinACMPUpdateAverageCycleTime (Page 4816)	No	No	No
WinACMPUpdateEXTFLED (Page 4817)	No	No	No
WinACMPUpdateHMIEnableTime (Page 4818)	No	No	No
WinACMPUpdateINTFLED (Page 4819)	No	No	No
WinACMPUpdateLastCycleTime (Page 4820)	No	No	No
WinACMPUpdateMaximumCycleTime (Page 4821)	No	No	No
WinACMPUpdateMinimumCycleTime (Page 4822)	No	No	No
WinACMPUpdatePowerLED (Page 4823)	No	No	No
WinACMPUpdateSleepTime (Page 4824)	No	No	No

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
WinACMPUpdateRUNLED (Page 4825)	No	No	No
WinACMPUpdateSTOPLED (Page 4826)	No	No	No
WinACMPArchive (Page 4827)	No	No	No
WinACMPGetStartMode (Page 4828)	No	No	No
WinACMPGetVersion (Page 4828)	No	No	No
WinACMPClearCycleTimeBuffer (Page 4829)	No	No	No
WinACMPSetStartAtBoot (Page 4829)	No	No	No
WinACMPSetKeySwitch (Page 4830)	No	No	No
WinACMPSetHMIEnableTime (Page 4831)	No	No	No
WinACMPSetResetMethod (Page 4831)	No	No	No
WinACMPSetSleepTime (Page 4832)	No	No	No
WinACMPSetStartMode (Page 4832)	No	No	No
WinACMPStartHistogram (Page 4833)	No	No	No
WinACMPControl (Page 4834)	No	No	No
WinACMPStopHistogram (Page 4834)	No	No	No
WinACMPRestore (Page 4835)	No	No	No
ShowLogonDialog (Page 4804)	Yes	Yes	Yes
ShowOperatorNotes (Page 4805)	Yes	Yes	Yes
ShowAlarmWindow (Page 4806)	Yes	Yes	Yes
ShowPopupScreen (Page 4807)	Yes	Yes	Yes
ShowSlideInScreen (Page 4808)	Yes	Yes	Yes
ShowSoftwareVersion (Page 4809)	Yes	Yes	Yes
ShowSystemDiagnosticsWindow (Page 4809)	Yes	Yes	Yes
ShowSystemAlarm (Page 4810)	Yes	Yes	Yes

[1] Only on KTP 400 Comfort

System functions for Mobile Panels

Availability of system functions

The following table shows the availability of the system functions on the mobile panels.

Technical data subject to change.

Overview

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
User-defined functions	No	No	Yes	Yes	Yes	Yes
Logoff (Page 4657)	Yes	Yes	Yes	Yes	Yes	Yes
ActivateScreen (Page 4657)	Yes	Yes	Yes	Yes	Yes	Yes
ActivateScreenByNumber (Page 4659)	Yes	Yes	Yes	Yes	Yes	Yes
ActivateCleanScreen (Page 4660)	No	No	No	No	No	No
ActivatePreviousScreen (Page 4660)	Yes	Yes	Yes	Yes	Yes	Yes
UpdateTag (Page 4661)	Yes	Yes	Yes	Yes	Yes	Yes
AdjustContrast (Page 4662)	Yes	Yes	No	No	No	No
Logon (Page 4663)	Yes	Yes	Yes	Yes	Yes	Yes
ArchiveLogFile (Page 4663)	No	No	Yes	Yes	Yes	Yes
LogTag (Page 4665)	No	No	Yes	Yes	Yes	Yes
EditAlarm (Page 4666)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorDown (Page 4666)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorUp (Page 4667)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageDown (Page 4669)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageUp (Page 4670)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorLeft (Page 4668)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorRight (Page 4668)	Yes	Yes	Yes	Yes	Yes	Yes
Encode (Page 4671)	Yes	Yes	Yes	Yes	Yes	Yes
EncodeEx (Page 4672)	Yes	Yes	Yes	Yes	Yes	Yes
DirectKey (Page 4673)	Yes	Yes	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
DirectKeyScreenNumber (Page 4675)	Yes	Yes	Yes	Yes	Yes	Yes
PrintScreen (Page 4676)	No	Yes	Yes	Yes	Yes	Yes
PrintReport (Page 4677)	No	Yes	Yes	Yes	Yes	Yes
NotifyUserAction (Page 4677)	No	No	Yes	Yes	Yes	Yes
IncreaseFocusedValue (Page 4679)	Yes	Yes	Yes	Yes	Yes	Yes
IncreaseTag (Page 4679)	Yes	Yes	Yes	Yes	Yes	Yes
ExportDataRecords (Page 4680)	Yes	Yes	Yes	Yes	Yes	Yes
ExportDataRecordsWithChecksum (Page 4683)	No	No	Yes	Yes	Yes	Yes
ExportImportUserAdministration (Page 4685)	Yes	Yes	Yes	Yes	Yes	Yes
GoToHome (Page 4686)	Yes	Yes	Yes	Yes	Yes	Yes
GoToEnd (Page 4686)	Yes	Yes	Yes	Yes	Yes	Yes
SafelyRemoveHardware (Page 4687)	No	No	Yes	Yes	Yes	Yes
HTMLBrowserStop (Page 4688)	No	No	No	No	Yes	Yes
HTMLBrowserScrollDown (Page 4688)	No	No	No	No	Yes	Yes
HTMLBrowserRefresh (Page 4689)	No	No	No	No	Yes	Yes
HTMLBrowserScrollUp (Page 4689)	No	No	No	No	Yes	Yes
HTMLBrowserZoomIn (Page 4690)	No	No	No	No	Yes	Yes
HTMLBrowserZoomOut (Page 4691)	No	No	No	No	Yes	Yes
HTMLBrowserScrollLeft (Page 4691)	No	No	No	No	Yes	Yes
HTMLBrowserScrollRight (Page 4692)	No	No	No	No	Yes	Yes
HTMLBrowserPageDown (Page 4693)	No	No	No	No	Yes	Yes
HTMLBrowserPageUp (Page 4693)	No	No	No	No	Yes	Yes
HTMLBrowserStartPage (Page 4694)	No	No	No	No	Yes	Yes
HTMLBrowserForward (Page 4695)	No	No	No	No	Yes	Yes
HTMLBrowserBack (Page 4695)	No	No	No	No	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
ImportDataRecords (Page 4696)	Yes	Yes	Yes	Yes	Yes	Yes
ImportDataRecordsWithChecksum (Page 4698)	No	No	Yes	Yes	Yes	Yes
InvertBit (Page 4699)	Yes	Yes	Yes	Yes	Yes	Yes
InvertBitInTag (Page 4700)	Yes	Yes	Yes	Yes	Yes	Yes
InvertLinearScaling (Page 4701)	Yes	Yes	Yes	Yes	Yes	Yes
CalibrateTouchScreen (Page 4703)	Yes	Yes	Yes [2)]	Yes	Yes	Yes
CopyLog (Page 4704)	No	No	Yes	Yes	Yes	Yes
TrendViewScrollForward (Page 4705)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewScrollBack (Page 4705)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewExtend (Page 4706)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewCompress (Page 4706)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewRulerLeft (Page 4707)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewRulerRight (Page 4708)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewSetRulerMode (Page 4708)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewStartStop (Page 4709)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewBackToBeginning (Page 4709)	Yes	Yes	Yes	Yes	Yes	Yes
LoadDataRecord (Page 4710)	Yes	Yes	Yes	Yes	Yes	Yes
GetUserName (Page 4711)	Yes	Yes	Yes	Yes	Yes	Yes
GetDataRecordFromPLC (Page 4712)	Yes	Yes	Yes	Yes	Yes	Yes
GetDataRecordName (Page 4713)	Yes	Yes	Yes	Yes	Yes	Yes
GetDataRecordTagsFromPLC (Page 4715)	Yes	Yes	Yes	Yes	Yes	Yes
GetGroupNumber (Page 4716)	Yes	Yes	Yes	Yes	Yes	Yes
GetBrightness (Page 4716)	No	No	No	Yes	Yes	Yes
GetPassword (Page 4717)	Yes	Yes	Yes	Yes	Yes	Yes
LinearScaling (Page 4718)	Yes	Yes	Yes	Yes	Yes	Yes
ClearLog (Page 4720)	No	No	Yes	Yes	Yes	Yes
DeleteDataRecord (Page 4721)	Yes	Yes	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
ClearDataRecordMemory (Page 4722)	Yes	Yes	Yes	Yes	Yes	Yes
ClearAlarmBuffer (Page 4723)	Yes	Yes	Yes	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 4724)	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewUpdate (Page 4725)	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewLoopInAlarm (Page 4725)	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 4726)	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 4727)	Yes	Yes	Yes	Yes	Yes	Yes
OpenAllLogs (Page 4728)	No	No	Yes	Yes	Yes	Yes
OpenScreenKeyboard (Page 4729)	No	No	Yes	Yes	Yes	Yes
OpenCommandPrompt (Page 4731)	No	No	Yes	Yes	Yes	Yes
OpenControlPanelDialog (Page 4730)	No	No	No	Yes	Yes	Yes
OpenInternetExplorer (Page 4732)	No	Yes	Yes	Yes	Yes	Yes
OpenTaskManager (Page 4733)	No	No	Yes	Yes	Yes	Yes
OpenControlPanel (Page 4733)	No	No	Yes	Yes	Yes	Yes
AcknowledgeAlarm (Page 4744)	Yes	Yes	Yes	Yes	Yes	Yes
PDFScrollDown (Page 4734)	No	No	No	Yes	Yes	Yes
PDFScrollUp (Page 4735)	No	No	No	Yes	Yes	Yes
PDFFitToWidth (Page 4735)	No	No	No	Yes	Yes	Yes
PDFFitToHeight (Page 4736)	No	No	No	Yes	Yes	Yes
PDFGoToFirstPage (Page 4737)	No	No	No	Yes	Yes	Yes
PDFGoToLastPage (Page 4737)	No	No	No	Yes	Yes	Yes
PDFGoToNextPage (Page 4738)	No	No	No	Yes	Yes	Yes
PDFGoToPage (Page 4738)	No	No	No	Yes	Yes	Yes
PDFGoToPreviousPage (Page 4739)	No	No	No	Yes	Yes	Yes
PDFZoomIn (Page 4740)	No	No	No	Yes	Yes	Yes
PDFZoomOut (Page 4740)	No	No	No	Yes	Yes	Yes
PDFScrollLeft (Page 4741)	No	No	No	Yes	Yes	Yes
PDFScrollRight (Page 4741)	No	No	No	Yes	Yes	Yes
PDFZoomOriginal (Page 4742)	No	No	No	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
RecipeViewNewDataRecord (Page 4744)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewGetDataRecord-FromPLC (Page 4745)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 4745)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewMenu (Page 4746)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewOpen (Page 4746)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSetDataRecord-ToPLC (Page 4747)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 4748)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 4748)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSynchronizeDataR-ecordWithTags (Page 4749)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewRenameDataRe-cord (Page 4749)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewShowOperator-Notes (Page 4750)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewBack (Page 4750)	Yes	Yes	Yes	Yes	Yes	Yes
ResetBit (Page 4751)	Yes	Yes	Yes	Yes	Yes	Yes
ResetBitInTag (Page 4752)	Yes	Yes	Yes	Yes	Yes	Yes
PressButton (Page 4753)	Yes	Yes	Yes	Yes	Yes	Yes
ReleaseButton (Page 4754)	Yes	Yes	Yes	Yes	Yes	Yes
ShiftAndMask (Page 4755)	Yes	Yes	Yes	Yes	Yes	Yes
CloseAllLogs (Page 4757)	No	No	Yes	Yes	Yes	Yes
SetDataRecordToPLC (Page 4758)	Yes	Yes	Yes	Yes	Yes	Yes
SetDataRecordTagsToPLC (Page 4759)	Yes	Yes	Yes	Yes	Yes	Yes
PageDown (Page 4759)	Yes	Yes	Yes	Yes	Yes	Yes
PageUp (Page 4760)	Yes	Yes	Yes	Yes	Yes	Yes
SendEMail (Page 4761)	No	Yes	Yes	Yes	Yes	Yes
SetAcousticSignal (Page 4762)	No	No	No	Yes	Yes	Yes
SetDisplayMode (Page 4762)	No	No	No	No	No	No
SetDeviceMode (Page 4763)	Yes	Yes	Yes	Yes	Yes	Yes
SetBit (Page 4764)	Yes	Yes	Yes	Yes	Yes	Yes
SetBitInTag (Page 4765)	Yes	Yes	Yes	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 4767)	Yes	Yes	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
SetBacklightColor (Page 4768)	No	No	No	No	No	No
SetBrightness (Page 4769)	No	No	No	Yes	Yes	Yes
SetScreenKeyboardMode (Page 4770)	No	No	Yes	Yes	Yes	Yes
SetAlarmReportMode (Page 4772)	No	No	Yes	Yes	Yes	Yes
SetRecipeTags (Page 4772)	Yes	Yes	Yes	Yes	Yes	Yes
SetDaylightSavingTime (Page 4774)	Yes	Yes	Yes	Yes	Yes	Yes
SetLanguage (Page 4775)	Yes	Yes	Yes	Yes	Yes	Yes
SetTag (Page 4777)	Yes	Yes	Yes	Yes	Yes	Yes
SetConnectionMode (Page 4778)	Yes	Yes	Yes	Yes	Yes	Yes
SetWebAccess (Page 4780)	No	Yes	Yes	Yes	Yes	Yes
BackupRAMFileSystem (Page 4780)	No	No	Yes	Yes	Yes	Yes
SimulateSystemKey (Page 4781)	Yes	Yes	Yes	Yes	Yes	Yes
SimulateTag (Page 4782)	Yes	Yes	Yes	Yes	Yes	Yes
SmartClientViewRefresh (Page 4783)	No	Yes	Yes	Yes	Yes	Yes
SmartClientViewReadOnlyOff (Page 4784)	No	Yes	Yes	Yes	Yes	Yes
SmartClientViewReadOnlyOn (Page 4784)	No	Yes	Yes	Yes	Yes	Yes
SmartClientViewDisconnect (Page 4785)	No	Yes	Yes	Yes	Yes	Yes
SmartClientViewConnect (Page 4785)	No	Yes	Yes	Yes	Yes	Yes
SmartClientViewLeave (Page 4786)	No	Yes	Yes	Yes	Yes	Yes
SaveDataRecord (Page 4787)	Yes	Yes	Yes	Yes	Yes	Yes
StartLogging (Page 4788)	No	No	Yes	Yes	Yes	Yes
StartNextLog (Page 4789)	No	No	Yes	Yes	Yes	Yes
StartProgram (Page 4789)	No	No	Yes	Yes	Yes	Yes
StatusForceGetValues (Page 4791)	Yes	Yes	Yes	Yes	Yes	Yes
StatusForceSetValues (Page 4792)	No	Yes	Yes	Yes	Yes	Yes
ControlSmartServer (Page 4792)	No	Yes	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
ControlWebServer (Page 4793)	No	Yes	Yes	Yes	Yes	Yes
StopLogging (Page 4794)	No	No	Yes	Yes	Yes	Yes
StopRuntime (Page 4795)	Yes	Yes	Yes	Yes	Yes	Yes
LookupText (Page 4799)	Yes	Yes	Yes	Yes	Yes	Yes
ActivateSystemDiagnosticsView (Page 4796)	No	No	No	Yes	Yes	Yes
ResetTagToHandWheel (Page 4800)	Yes	Yes	Yes	Yes	Yes	Yes
SetTagToHandWheel (Page 4800)	Yes	Yes	Yes	Yes	Yes	Yes
TraceUserChange (Page 4801)	Yes	Yes	Yes	Yes	Yes	Yes
DecreaseFocusedValue (Page 4801)	Yes	Yes	Yes	Yes	Yes	Yes
DecreaseTag (Page 4802)	Yes	Yes	Yes[1)]	Yes	Yes	Yes
ChangeConnection (Page 4803)	Yes	Yes	Yes	Yes	Yes	Yes
WinACMPUpdateControllerForStartAtBoot (Page 4811)	No	No	No	No	No	No
WinACMPUpdateKeySwitchSetting (Page 4812)	No	No	No	No	No	No
WinACMPUpdateBUSF1LED (Page 4813)	No	No	No	No	No	No
WinACMPUpdateBUSF2LED (Page 4814)	No	No	No	No	No	No
WinACMPUpdateAverageExecTime (Page 4815)	No	No	No	No	No	No
WinACMPUpdateAverageCycleTime (Page 4816)	No	No	No	No	No	No
WinACMPUpdateEXTFLED (Page 4817)	No	No	No	No	No	No
WinACMPUpdateHMIEnableTime (Page 4818)	No	No	No	No	No	No
WinACMPUpdateINTFLED (Page 4819)	No	No	No	No	No	No
WinACMPUpdateLastCycleTime (Page 4820)	No	No	No	No	No	No
WinACMPUpdateMaximumCycleTime (Page 4821)	No	No	No	No	No	No
WinACMPUpdateMinimumCycleTime (Page 4822)	No	No	No	No	No	No
WinACMPUpdatePowerLED (Page 4823)	No	No	No	No	No	No

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
WinACMPUpdateSleepTime (Page 4824)	No	No	No	No	No	No
WinACMPUpdateRUNLED (Page 4825)	No	No	No	No	No	No
WinACMPUpdateSTOPLED (Page 4826)	No	No	No	No	No	No
WinACMPArchive (Page 4827)	No	No	No	No	No	No
WinACMPGetStartMode (Page 4828)	No	No	No	No	No	No
WinACMPGetVersion (Page 4828)	No	No	No	No	No	No
WinACMPClearCycleTimeBuffer (Page 4829)	No	No	No	No	No	No
WinACMPSetStartAtBoot (Page 4829)	No	No	No	No	No	No
WinACMPSetKeySwitch (Page 4830)	No	No	No	No	No	No
WinACMPSetHMIEnableTime (Page 4831)	No	No	No	No	No	No
WinACMPSetResetMethod (Page 4831)	No	No	No	No	No	No
WinACMPSetSleepTime (Page 4832)	No	No	No	No	No	No
WinACMPSetStartMode (Page 4832)	No	No	No	No	No	No
WinACMPStartHistogram (Page 4833)	No	No	No	No	No	No
WinACMPControl (Page 4834)	No	No	No	No	No	No
WinACMPStopHistogram (Page 4834)	No	No	No	No	No	No
WinACMPRestore (Page 4835)	No	No	No	No	No	No
ShowLogonDialog (Page 4804)	Yes	Yes	Yes	Yes	Yes	Yes
ShowOperatorNotes (Page 4805)	Yes	Yes	Yes	Yes	Yes	Yes
ShowAlarmWindow (Page 4806)	Yes	Yes	Yes	Yes	Yes	Yes
ShowPopupScreen (Page 4807)	No	No	No	No	No	No
ShowSlideInScreen (Page 4808)	No	No	No	Yes	Yes	Yes
ShowSoftwareVersion (Page 4809)	No	No	Yes	Yes	Yes	Yes
ShowSystemDiagnosticsWindow (Page 4809)	No	No	No	Yes	Yes	Yes
ShowSystemAlarm (Page 4810)	Yes	Yes	Yes	Yes	Yes	Yes

[1] only for Mobile Panel 277
[2] Only for Mobile Panel 277, Mobile Panel 277 IWLAN

System functions for Runtime Advanced

Availability of system functions

The following table shows the availability of the system functions for WinCC RT Advanced.

Technical data subject to change.

Overview

	WinCC RT Advanced
User-defined functions	Yes
Logoff (Page 4657)	Yes
ActivateScreen (Page 4657)	Yes
ActivateScreenByNumber (Page 4659)	Yes
ActivateCleanScreen (Page 4660)	Yes
ActivatePreviousScreen (Page 4660)	Yes
UpdateTag (Page 4661)	Yes
AdjustContrast (Page 4662)	No
Logon (Page 4663)	Yes
ArchiveLogFile (Page 4663)	Yes
LogTag (Page 4665)	Yes
EditAlarm (Page 4666)	Yes
ScreenObjectCursorDown (Page 4666)	Yes
ScreenObjectCursorUp (Page 4667)	Yes
ScreenObjectCursorLeft (Page 4668)	Yes
ScreenObjectCursorRight (Page 4668)	Yes
ScreenObjectPageDown (Page 4669)	Yes
ScreenObjectPageUp (Page 4670)	Yes
Encode (Page 4671)	Yes
EncodeEx (Page 4672)	Yes
DirectKey (Page 4673)	No
DirectKeyScreenNumber (Page 4675)	No
PrintScreen (Page 4676)	Yes
PrintReport (Page 4677)	Yes
NotifyUserAction (Page 4677)	Yes
IncreaseFocusedValue (Page 4679)	Yes
IncreaseTag (Page 4679)	Yes
ExportDataRecords (Page 4680)	Yes

	WinCC RT Advanced
ExportDataRecordsWithChecksum (Page 4683)	No
ExportImportUserAdministration (Page 4685)	Yes
GoToHome (Page 4686)	Yes
GoToEnd (Page 4686)	Yes
SafelyRemoveHardware (Page 4687)	No
HTMLBrowserStop (Page 4688)	Yes
HTMLBrowserScrollDown (Page 4688)	Yes
HTMLBrowserRefresh (Page 4689)	Yes
HTMLBrowserScrollUp (Page 4689)	Yes
HTMLBrowserZoomIn (Page 4690)	Yes
HTMLBrowserZoomOut (Page 4691)	Yes
HTMLBrowserScrollLeft (Page 4691)	Yes
HTMLBrowserScrollRight (Page 4692)	Yes
HTMLBrowserPageDown (Page 4693)	Yes
HTMLBrowserPageUp (Page 4693)	Yes
HTMLBrowserStartPage (Page 4694)	Yes
HTMLBrowserForward (Page 4695)	Yes
HTMLBrowserBack (Page 4695)	Yes
ImportDataRecords (Page 4696)	Yes
ImportDataRecordsWithChecksum (Page 4698)	No
InvertBit (Page 4699)	Yes
InvertBitInTag (Page 4700)	Yes
InvertLinearScaling (Page 4701)	Yes
CalibrateTouchScreen (Page 4703)	No
CopyLog (Page 4704)	Yes
TrendViewScrollForward (Page 4705)	Yes
TrendViewScrollBack (Page 4705)	Yes
TrendViewExtend (Page 4706)	Yes
TrendViewCompress (Page 4706)	Yes
TrendViewRulerLeft (Page 4707)	Yes
TrendViewRulerRight (Page 4708)	Yes
TrendViewSetRulerMode (Page 4708)	Yes
TrendViewStartStop (Page 4709)	Yes
TrendViewBackToBeginning (Page 4709)	Yes
LoadDataRecord (Page 4710)	Yes
GetUserName (Page 4711)	Yes
GetDataRecordFromPLC (Page 4712)	Yes
GetDataRecordName (Page 4713)	Yes
GetDataRecordTagsFromPLC (Page 4715)	Yes
GetGroupNumber (Page 4716)	Yes
GetBrightness (Page 4716)	No

	WinCC RT Advanced
GetPassword (Page 4717)	Yes
LinearScaling (Page 4718)	Yes
ClearLog (Page 4720)	Yes
DeleteDataRecord (Page 4721)	Yes
ClearDataRecordMemory (Page 4722)	No
ClearAlarmBuffer (Page 4723)	Yes
ClearAlarmBufferProTool (Page 4724)	Yes
AlarmViewUpdate (Page 4725)	Yes
AlarmViewLoopInAlarm (Page 4725)	Yes
AlarmViewAcknowledgeAlarm (Page 4726)	Yes
AlarmViewShowOperatorNotes (Page 4727)	Yes
OpenAllLogs (Page 4728)	Yes
OpenScreenKeyboard (Page 4729)	Yes
OpenControlPanelDialog (Page 4730)	No
OpenCommandPrompt (Page 4731)	No
OpenInternetExplorer (Page 4732)	No
OpenControlPanel (Page 4733)	No
OpenTaskManager (Page 4733)	Yes
AcknowledgeAlarm (Page 4744)	Yes
PDFScrollDown (Page 4734)	Yes
PDFScrollUp (Page 4735)	Yes
PDFFitToWidth (Page 4735)	Yes
PDFFitToHeight (Page 4736)	Yes
PDFGoToFirstPage (Page 4737)	Yes
PDFGoToLastPage (Page 4737)	Yes
PDFGoToNextPage (Page 4738)	Yes
PDFGoToPage (Page 4738)	Yes
PDFGoToPreviousPage (Page 4739)	Yes
PDFZoomIn (Page 4740)	Yes
PDFZoomOut (Page 4740)	Yes
PDFScrollLeft (Page 4741)	Yes
PDFScrollRight (Page 4741)	Yes
PDFZoomOriginal (Page 4742)	Yes
RecipeViewNewDataRecord (Page 4744)	Yes
RecipeViewGetDataRecordFromPLC (Page 4745)	Yes
RecipeViewClearDataRecord (Page 4745)	Yes
RecipeViewMenu (Page 4746)	Yes
RecipeViewOpen (Page 4746)	Yes
RecipeViewSetDataRecordToPLC (Page 4747)	Yes
RecipeViewSaveDataRecord (Page 4748)	Yes
RecipeViewSaveAsDataRecord (Page 4748)	Yes

	WinCC RT Advanced
RecipeViewSynchronizeDataRecordWithTags (Page 4749)	Yes
RecipeViewRenameDataRecord (Page 4749)	Yes
RecipeViewShowOperatorNotes (Page 4750)	Yes
RecipeViewBack (Page 4750)	Yes
ResetBit (Page 4751)	Yes
ResetBitInTag (Page 4752)	Yes
PressButton (Page 4753)	Yes
ReleaseButton (Page 4754)	Yes
ShiftAndMask (Page 4755)	Yes
CloseAllLogs (Page 4757)	Yes
SetDataRecordToPLC (Page 4758)	Yes
SetDataRecordTagsToPLC (Page 4759)	Yes
PageDown (Page 4759)	Yes
PageUp (Page 4760)	Yes
SendEMail (Page 4761)	Yes
SetAcousticSignal (Page 4762)	No
SetDisplayMode (Page 4762)	Yes
SetDeviceMode (Page 4763)	Yes
SetBit (Page 4764)	Yes
SetBitInTag (Page 4765)	Yes
SetBitWhileKeyPressed (Page 4767)	Yes
SetBacklightColor (Page 4768)	No
SetBrightness (Page 4769)	No
SetScreenKeyboardMode (Page 4770)	Yes
SetAlarmReportMode (Page 4772)	Yes
SetRecipeTags (Page 4772)	Yes
SetDaylightSavingTime (Page 4774)	Yes
SetLanguage (Page 4775)	Yes
SetTag (Page 4777)	Yes
SetConnectionMode (Page 4778)	Yes
SetWebAccess (Page 4780)	Yes
BackupRAMFileSystem (Page 4780)	No
SimulateSystemKey (Page 4781)	Yes
SimulateTag (Page 4782)	Yes
SmartClientViewRefresh (Page 4783)	Yes
SmartClientViewReadOnlyOff (Page 4784)	Yes
SmartClientViewReadOnlyOn (Page 4784)	Yes
SmartClientViewDisconnect (Page 4785)	Yes
SmartClientViewConnect (Page 4785)	Yes
SmartClientViewLeave (Page 4786)	Yes
SaveDataRecord (Page 4787)	Yes

	WinCC RT Advanced
StartLogging (Page 4788)	Yes
StartNextLog (Page 4789)	Yes
StartProgram (Page 4789)	Yes
StatusForceGetValues (Page 4791)	Yes
StatusForceSetValues (Page 4792)	Yes
ControlSmartServer (Page 4792)	Yes
ControlWebServer (Page 4793)	Yes
StopLogging (Page 4794)	Yes
StopRuntime (Page 4795)	Yes
ActivateSystemDiagnosticsView (Page 4796)	No
LookupText (Page 4799)	Yes
ResetTagToHandWheel (Page 4800)	No
SetTagToHandWheel (Page 4800)	No
TraceUserChange (Page 4801)	Yes
DecreaseFocusedValue (Page 4801)	Yes
DecreaseTag (Page 4802)	Yes
ChangeConnection (Page 4803)	Yes
WinACMPUpdateControllerForStartAtBoot (Page 4811)	No
WinACMPUpdateKeySwitchSetting (Page 4812)	No
WinACMPUpdateBUSF1LED (Page 4813)	No
WinACMPUpdateBUSF2LED (Page 4814)	No
WinACMPUpdateAverageExecTime (Page 4815)	No
WinACMPUpdateAverageCycleTime (Page 4816)	No
WinACMPUpdateEXTFLED (Page 4817)	No
WinACMPUpdateHMIEnableTime (Page 4818)	No
WinACMPUpdateINTFLED (Page 4819)	No
WinACMPUpdateLastCycleTime (Page 4820)	No
WinACMPUpdateMaximumCycleTime (Page 4821)	No
WinACMPUpdateMinimumCycleTime (Page 4822)	No
WinACMPUpdatePowerLED (Page 4823)	No
WinACMPUpdateSleepTime (Page 4824)	No
WinACMPUpdateRUNLED (Page 4825)	No
WinACMPUpdateSTOPLED (Page 4826)	No
WinACMPArchive (Page 4827)	No
WinACMPGetStartMode (Page 4828)	No
WinACMPGetVersion (Page 4828)	No
WinACMPClearCycleTimeBuffer (Page 4829)	No
WinACMPSetStartAtBoot (Page 4829)	No
WinACMPSetKeySwitch (Page 4830)	No
WinACMPSetHMIEnableTime (Page 4831)	No
WinACMPSetResetMethod (Page 4831)	No

	WinCC RT Advanced
WinACMPSetSleepTime (Page 4832)	No
WinACMPSetStartMode (Page 4832)	No
WinACMPStartHistogram (Page 4833)	No
WinACMPControl (Page 4834)	No
WinACMPStopHistogram (Page 4834)	No
WinACMPRestore (Page 4835)	No
ShowLogonDialog (Page 4804)	Yes
ShowOperatorNotes (Page 4805)	Yes
ShowAlarmWindow (Page 4806)	Yes
ShowPopupScreen (Page 4807)	Yes
ShowSlideInScreen (Page 4808)	Yes
ShowSoftwareVersion (Page 4809)	Yes
ShowSystemDiagnosticsWindow (Page 4809)	No
ShowSystemAlarm (Page 4810)	Yes

System functions

Logoff

Description

Logs off the current user on the HMI device.

Use in the function list

Logoff

Use in user-defined functions

Logoff

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

ActivateScreen

Description

Performs a screen change to the given screen.

Use the "ActivateScreenByNumber" system function to change from the root screen to the permanent window or vice versa.

Use in the function list

ActivateScreen (Screen name, Object number)

Use in user-defined functions

ActivateScreen (Screen_name, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Name of the screen to which you change.

Object number

The operator control element which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent window when the system function is called up, the permanent window maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

Note

If the "Reach margin" event is assigned to the "ActivateScreen" system function, only the value "0" is valid for the "Object number" parameter. The active object is not defined by the object number, but rather by the X position it had prior to the screen change.

Example

The ActivateScreen function is used in the following program code to activate the "Screen_2" screen when you click a button.

```
{

// User defined code
// i.e. when pressing a button
ActivateScreen ("Screen_2", 0);
...
}
```

See also

ActivateScreenByNumber (Page 4659)

Availability for specific devices of system functions (Page 4612)

ActivateScreenByNumber

Description

Performs a screen change to a screen depending on a tag value.

The screen is identified by its screen number.

Use in the function list

ActivateScreenByNumber (Screen number, Object number)

Use in user-defined functions

ActivateScreenByNumber (Screen_number, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Screen number

Tag which contains the screen number of the destination screen.

When a change from the root screen to the permanent window is desired, "0" or "-1" is specified:

0 = Change from root screen to permanent window

-1 = Change from permanent window to root screen

Object number

The number of the screen object which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent window when the system function is called up, the permanent window maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

See also

ActivateScreen (Page 4657)

Availability for specific devices of system functions (Page 4612)

ActivateCleanScreen

Description

Activates the clean screen on the HMI device. The display of the HMI device is disabled for the given time period.

When the display of the HMI device is deactivated, it can be cleaned without triggering touch functions by mistake.

Use in the function list

ActivateCleanScreen (Time period)

Use in user-defined functions

--

Parameters

Time period

Time period for which the display is disabled. The time remaining is displayed as a progress bar.

Value range in seconds from 10 through 300.

Note

The system function ActivateCleanScreen cannot be simulated.

See also

Logoff (Page 4656)

Availability for specific devices of system functions (Page 4612)

ActivatePreviousScreen

Description

Performs a screen change to the screen which was activated before the current screen. The screen change is not performed if no screen was activated beforehand.

The last 10 screens that were called up are saved. A system alarm is output when you change to a screen which is no longer saved.

Note

If you want to use the system function, the screen to which you change has to be used in the navigation structure.

Use in the function list

ActivatePreviousScreen

Use in user-defined functions

ActivatePreviousScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

UpdateTag

Description

Reads the current value of the tag with the specified Update ID from the PLC.

Use in the function list

UpdateTag (Update ID)

Use in user-defined functions

-

Parameters

Update ID

Update ID assigned to the tag that will be updated.

See also

Availability for specific devices of system functions (Page 4612)

Logoff (Page 4656)

AdjustContrast

Description

Changes the contrast of the display one level on the HMI device.

Use in the function list

AdjustContrast (Adjust)

Use in user-defined functions

-

Parameters

Adjust

Specifies how the contrast is changed:

0 (hmiDecrease) = Decrease: Decreases the contrast one level.

1 (hmiIncrease) = Increase: Increases the contrast one level.

Application example

Objective

One button each for increasing and decreasing the screen contrast is desired.

Notes on configuring

Configure two buttons and configure the "AdjustContrast" system function on the "Press" event. The parameters "Increase" and "Decrease" are allocated.

Procedure on HMI device

When one of the two buttons is pressed in runtime, the contrast is increased or decreased one level.

See also

Logoff (Page 4656)

Availability for specific devices of system functions (Page 4612)

Logon

Description

Logs on the current user on the HMI device.

Use in the function list

Logon (Password, User name)

Use in user-defined functions

Logon (Password, User_name)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Password

The tag from which the password for the user logging on is read.

If the user is logged on, the password in the tag is deleted.

User name

The tag from which the user name for the user logging on is read.

See also

Availability for specific devices of system functions (Page 4612)

ArchiveLogFile

Description

This system function moves or copies a log to another storage location for long-term logging.

With Audit Trails, always use the "Move" (hmiMove)" mode, otherwise you will be violating the FDA guideline by duplicating the data stored.

Prior to "ArchiveLogFile" always run the system function "CloseAllLogs".

After this system function, run the "OpenAllLogs" system function.

In "Copy log" mode, the logs are not reopened until the log has been successfully copied, or unless a timeout occurs during the copy process. In "Move log" mode, the logs to be moved are renamed and the new logs are opened immediately. To move the renamed logs, a job is submitted which attempts to move them every 300 seconds if the server cannot be reached. The job is also retained after Runtime is restarted until it is executed.

Use in the function list

ArchiveLogFile (Log type, Log, Directory name, Mode)

Use in user-defined functions

ArchiveLogFile (Log_type, Log, Directory_name, Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (miAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail log available for GMP compliant projects. Additional information is available under "Activating GMP-compliant configuration".

Log

Name of the log being archived.

Directory name

Path for saving the log.

Mode

0 (hmiCopy) = Copy log

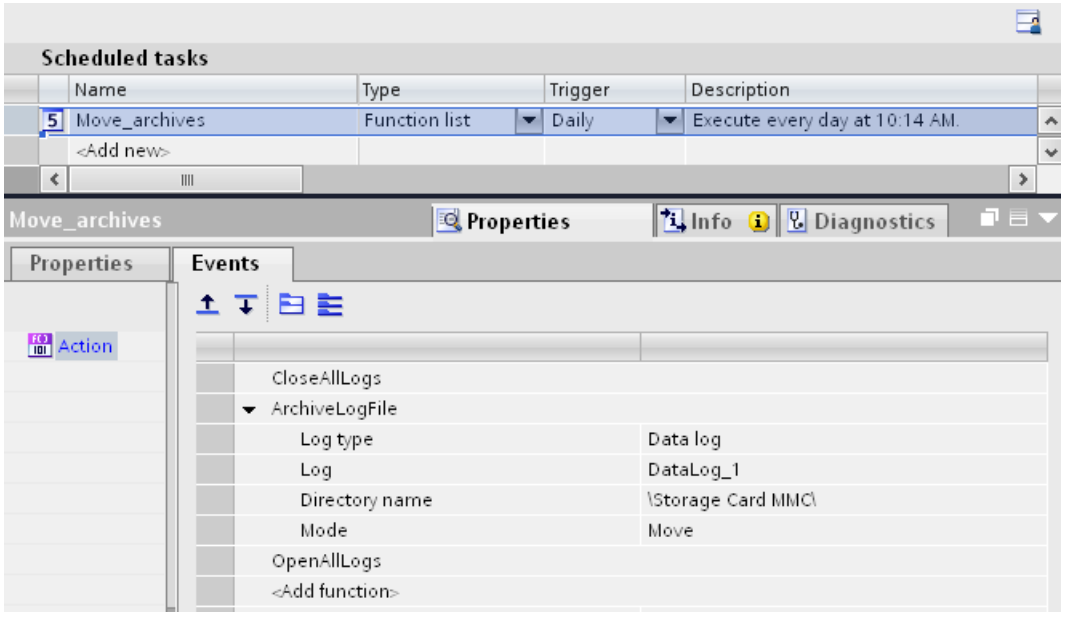
1 (hmiMove) = Move log

Application example

You want to move a log file from a local storage medium to the server in order to generate a backup copy of this file at cyclic intervals.

Notes on configuring

Set up a task in the scheduler which is executed at a specific time on a daily basis. Configure the following function list for the task:



Procedure on HMI device

- All log files will be closed.
- The log file you specified is moved to the server.
- All closed log files will be opened again.

See also

Availability for specific devices of system functions (Page 4612)

LogTag

Description

Saves the value of the given tags in the given data log.

This system function is used to archive a process value at a certain point in time.

Use in the function list

LogTag (Tag)

Use in user-defined functions

-

Parameters

Tag

The tag whose value is logged. The tag is stored in the log which is configured for the specified tag.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Basic Panels (Page 4612)

EditAlarm

Description

Triggers the "Edit" event for all selected alarms.

If the alarms to be edited have not yet been acknowledged, the acknowledgment takes place automatically when this system function is called up.

This system function can only be used for function keys.

Use in the function list

EditAlarm

Use in user-defined functions

EditAlarm

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

ScreenObjectCursorDown

Description

Results in a line-by-line, downward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view
- NC subprogram display

Use in the function list

ScreenObjectCursorDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

See also

Availability for specific devices of system functions (Page 4612)

ScreenObjectCursorUp

Description

Results in a line-by-line, upward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view
- NC subprogram display

Use in the function list

ScreenObjectCursorUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

See also

Availability for specific devices of system functions (Page 4612)

ScreenObjectCursorLeft

Description

Results in a line-by-line, cursor movement to the left in the specified screen object.

Use in the function list

ScreenObjectCursorLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Basic Panels (Page 4612)

System functions for Basic Panels 2nd Generation (Page 4618)

System functions for Panels (Page 4624)

System functions for Mobile Panels (Page 4643)

System functions for Runtime Advanced (Page 4651)

ScreenObjectCursorRight

Description

Results in a line-by-line, cursor movement to the right in the specified screen object.

Use in the function list

ScreenObjectCursorLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Basic Panels (Page 4612)

System functions for Basic Panels 2nd Generation (Page 4618)

System functions for Panels (Page 4624)

System functions for Mobile Panels (Page 4643)

System functions for Runtime Advanced (Page 4651)

ScreenObjectPageDown

Description

Results in a page-by-page, downward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view
- NC subprogram display

Use in the function list

ScreenObjectPageDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

See also

Availability for specific devices of system functions (Page 4612)

ScreenObjectPageUp

Description

Results in a page-by-page, upward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view
- NC subprogram display

Use in the function list

ScreenObjectPageUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

See also

Availability for specific devices of system functions (Page 4612)

Encode

Description

Adapts the "String" data type of a tag for transfer to the automation system (AS). The tag of WinCC data type "String" is converted into the AS data type "Array of byte". The result is written to a tag.

Use in the function list

Encode (Byte array, String, Coding)

Use in user-defined functions

Encode (Byte_array, String, Encoding)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Byte array

The tag that contains the converted value.

Note

The Byte array must be twice the size of the string length + 2. The Byte array must contain 242 array elements if the string has a length of 120 characters.

Characters are either truncated or not converted if the size is insufficient.

String

The tag of data type "String" that is converted.

Encode

0 (hmiEncodeUTF16LE) - String is encoded in UTF16LE (Unicode 16 Little Endian).

See also

Availability for specific devices of system functions (Page 4612)

EncodeEx

Description

Adapts the "String" data type of a tag for transfer to the automation system (AS). The tag of WinCC data type "String" is converted into the AS data type "Array of byte". The result is written to a tag.

By contrast to the Encode system function, you can define the Line break parameter. Using the Line break parameter you can delete line breaks or replace these with predefined characters.

Use in the function list

EncodeEx (Byte Array, String, Encode, Line break)

Use in user-defined functions

EncodeEx (Byte_array, String, Encoding, Line_break)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Byte array

The tag that contains the converted value.

Note

The byte array must be twice the size of the string length +2. If the string has a length of 120 characters, the byte array must contain 242 array elements.

Characters are either truncated or not converted if the size is insufficient.

String

The tag of data type "String" that is converted.

Encoding

0 (hmiEncodeUTF16LE) - String is encoded in UTF16LE (Unicode 16 Little Endian).

Line break

All line breaks are either deleted or replaced with predefined characters. Do not replace line breaks if set as the default value.

0 (replace with "\r\n' (0x000D, 0x000A)) - the line break is replaced with "\r\n".

1 (replace with "\n' (0x000A)) - the line break is replaced with "\n".

2 (do not replace) - the line break is not replaced.

3 (remove line breaks) - the line breaks are removed.

See also

Availability for specific devices of system functions (Page 4612)

DirectKey

Description

Allows rapid operation of keys on an HMI device without having communication caused delays occur.

Use this system function, for example. to set bits in the I/O area of a SIMATIC S7 directly from the HMI device.

Note

If you have integrated WinCC in SIMATIC STEP 7 and have configured a DP area in SIMATIC STEP 7: When the button on which the system function was configured is activated, a bit is set in the IO area of the CPU. Releasing the button resets the bit.

Notes on configuring the system function

When a button is configured with the "DirectKey" system function, a fixed area of the touch display is reserved for DirectKey operation.

Screen objects that are above the button with the "DirectKey" system function in this area in runtime will hide the button visually. However, the screen objects do not interfere with the triggering of the "DirectKey" system function.

Note

If an external application, such as Pocket Internet Explorer or Control Panel is started, it will run in the foreground and Runtime is placed in the background. The bit for the "DirectKeyScreenNumber" function is no longer set and the keys or buttons which are assigned the "DirectKey" function no longer to trigger the associated bit in the PLC.

Warning

Triggering the "DirectKey" system function by mistake will endanger personnel or result in damage to the machine.

In order to avoid this danger, the following must be observed:

- When the process screen is configured, the button with the "DirectKey" system function must not be covered by a screen object.
- The dynamic positioning or display (release) of a screen object in relation to process values must not cover the button with the "DirectKey" system function in runtime.

Note

Please observe this guideline during configuration. Check also existing configurations and adjust them immediately.

In the following cases, the covering of a button with the "DirectKey" system function does not lead to the problems described above:

- Operation of the alarm window
- Operation of the Recipe view
- Cancellation of the screen saver
- Use of the screen keyboard
- Running the "ActivateCleanScreen" system function

Actions of DP direct-keys in offline operation

When the "ChangeConnection" system function is performed with the "Offline" parameter on the HMI device, the connection to the specified PLC is disconnected.

Note

Please note the following on the SIMATIC STEP 7 side as well as the WinCC side:

The DP DirectKeys are still active in this case. If you activate a button with the "DirectKey" system function in "Offline" operating mode or activate the DirectKey on a keyboard device, the corresponding bit is set in the PLC.

Use in the function list

DirectKey (Bit)

Use in user-defined functions

-

Parameters

Bit

Specifies the bit which is set. Depending on the HMI device, the bit numbers 0 to 31 or 0 to 39 are possible.

Application example

Objective

You want to set bits in the IO area of a SIMATIC S7 directly from the HMI device using a button.

Requirement

WinCC integrated in SIMATIC STEP 7 is installed.

During operation, the HMI device is coupled to a SIMATIC S7 through PROFIBUS-DP.

WinCC integrated was installed when you compiled your project.

The bit area for DirectKeys is determined in SIMATIC STEP 7. For configuration see the "Communication" user's manual.

Notes on configuring

Configure the button to be used as the DirectKey. Assign the "DirectKey" system function to the button. As a parameter, enter the number of the bit which is set when the key is pressed.

Procedure on HMI device

When the DirectKey is touched, the bit is set, and reset when the key is released or the screen is exited.

See also

Availability for specific devices of system functions (Page 4612)

DirectKeyScreenNumber

Description

Sets the bit within the given bit area of a DirectKey and transfers it to the S7 controller to which the HMI device is connected. This ensures unambiguous allocation of a control bit to screen number at all times.

Without the use of the system function, the S7 controller must distinguish the respective functionality by means of the screen number. This delays the updating of the screen number after a screen change.

Note

If an external application, such as Pocket Internet Explorer or Control Panel is started, it will run in the foreground and Runtime is placed in the background. Nevertheless, the bit for the "DirectKeyScreenNumber" function is set and the keys or buttons with the configured "DirectKey" function continue to trigger the associated bit in the PLC.

Use in the function list

DirectKeyScreenNumber (Bit)

Use in user-defined functions

-

Parameters

Bit

Specifies the bit which is set. Depending on the HMI device, the bit numbers 0 to 31 or 0 to 39 are possible.

See also

Availability for specific devices of system functions (Page 4612)

PrintScreen

Description

Prints the screen currently being displayed on the HMI device from the printer which is connected to the HMI device.

Opened windows are also printed.

Note

The printer settings are taken over from the current settings in the Windows operation system.

Use in the function list

PrintScreen

Use in user-defined functions

PrintScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

PrintReport

Description

Prints the given report from the printer which is connected to the HMI device. The report is printed in the language which is set on the HMI device.

Note

If runtime is closed whilst log data are being printed using the system function, the report will cease to be supplied with data as soon as runtime stops.

Use in the function list

PrintReport (Report)

Use in user-defined functions

PrintReport (Report)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Report

Name of the report to be printed.

Note

If you have set up via the "Function list" dialog box a new report for the "PrintReport" function, when compiling, the following warning appears: "The report "Report_1" has no printed pages."

In order to remedy the warning, open the "Report_1" via the project view and recompile the project.

See also

Availability for specific devices of system functions (Page 4612)

NotifyUserAction

Description

This system function is used to log user actions that are not automatically logged in the audit trail. You can also use this system function to require the user to enter an acknowledgment or an electronic signature and a comment for the operator action. Requirement for the use of the system function is that the GMP-compliant configuration is activated under "Runtime settings".

If you use the "NotifyUserAction" system function in a function, the debugger may open if you cancel your input by clicking "Cancel". To compensate for this reaction, you can use the "On Error Resume Next" statement in a function. With this instruction, the next instruction is executed after a runtime error. If you use the "On Error Resume Next" statement, output of system events is also suppressed.

Use in the function list

NotifyUserAction (Confirmation type, Mandatory commenting, Category, Object name, Description)

Use in user-defined functions

NotifyUserAction (Confirmation_type, Mandatory_commenting, Category, Object_name, Description)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Confirmation type

Establishes how the action must be confirmed

0 = (None): No confirmation required, an entry is created in the audit trail

1 = (Acknowledgement): Acknowledgment, the user must acknowledge the action; an entry is created in the audit trail

2 = (Digital Signature): Electronic signature, a dialog window opens in which the user must enter his electronic signature - an entry is created in the audit trail

Mandatory commenting

Establishes if the user has to enter a comment. The comment is logged in the audit trail.

0 = (True): True; a dialog window opens in which the user must enter a comment

1 = (False): False; no mandatory commenting

Category

Category or class name of the modified object

Object name

Name of the modified object

Description

Text which describes the archiving user action.

See also

Availability for specific devices of system functions (Page 4612)

IncreaseFocusedValue

Description

Adds the given value to the value of the tag which is connected to the input field (drop-down list, graphic selection list, slider bar) which has the current focus.

This system function can only be used for function keys.

Use in the function list

IncreaseFocusedValue (Value)

Use in user-defined functions

-

Parameters

Value

The value which is added to the tag value.

See also

Availability for specific devices of system functions (Page 4612)

IncreaseTag

Description

Adds the given value to the value of the tags.

X = X + a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, help tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

IncreaseTag (Tag, Value)

Use in user-defined functions

IncreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the given value is added.

Value

The value that is added.

See also

Availability for specific devices of system functions (Page 4612)

SetTag (Page 4777)

ExportDataRecords

Description

Exports one or all data records of a recipe to a CSV file.

One file is created per recipe.

Use in the function list

ExportDataRecords (Recipe number/name, Data record number/name, File name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ExportDataRecords (Recipe_number/name,Data_record number/name, File_name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Recipe number/name

Number or name of the recipe from which the data records are exported. Specify "0" if recipe data records are to be exported from all available recipes.

File name

Name of the CSV file to which the recipe data records are exported. Enter the storage location and the file extension (*.csv), for example, "C:\TEMP\Orange.csv", for Basic Panels "\USB_X61.1\Orange.csv".

Note

Storage of the CSV file

Does not apply for Basic Panels:

- If a storage card is used as storage location, specify the storage location as follows: "\StorageCard\<File name>".
- If you only enter one file name and no path, the file will be saved in a system directory, for example, "C:\Documents and Settings\[User]".
- When only one path for the export is specified, the file name is automatically created from the respective recipe name. It is a requirement that the folder "D:\Temp\", for example, has been created. If the folder "D:\Temp" has not been created, the folder name will be used as the prefix of the file name, Temp_Recipe names.csv.

Data record number/name

Number or name of the recipe data record to be exported. Specify "0" if all recipe data records are to be exported.

Overwrite

Determines whether an existing CSV file with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: CSV file will not be overwritten. The export process will not be carried out.

1 (hmiOverwriteAlways) = Yes: CSV file is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithPrompting) = With confirmation: CSV file is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the export:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Export format of the recipe data records

If ".csv" is chosen as a file extension for the export file, then only valid characters from the ANSI character set will be supported. This also applies to separators in decimal numbers and list elements. The separators used are defined in the country/regional settings of the operating system of the exporting computer.

Does not apply for Basic Panels:

You may also set "Unicode text" (".txt") format for the export file. This file format supports the WinCC and WinCC Runtime character set. Again, the separators used are specified in the country/regional settings of the operating system of the exporting computer. This file format always uses tab separators in list elements.

The corresponding file import function also supports the ".csv" and ".txt" (Unicode) file formats.

Application example

You want to export all data records using a key.

Notes on configuring

Configure the "ExportDataRecords" system function on the "Press" event for the desired key. Transfer the following parameters:

- Recipe number/name = 1
- Data record number/name = 0
- File name = c:\temp\orange.csv, for Basic Panels "\USB_X61.1\orange.csv"
- Overwrite = 1
- Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data records are exported.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated. All data records of Recipe 1 are exported to the orange.csv file. If the file already exists, it will be overwritten.

After exporting the data records, a system alarm is output.

See also

Availability for specific devices of system functions (Page 4612)

ExportDataRecordsWithChecksum

Description

Exports one or all data records of a recipe to a CSV file and generates a checksum for each line in the file.

One file is created per recipe.

Use in the function list

ExportDataRecordsWithChecksum (Recipe number/name, Data record number/name, File name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ExportDataRecordsWithChecksum (Recipe_number/name, Data_record number/name, File_name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which the data records are exported. Specify "0" if recipe data records are to be exported from all available recipes.

Data record number/name

Number or name of the recipe data record to be exported. Specify "0" if all recipe data records are to be exported.

File name

Name of the CSV file to which the recipe data records are exported. Enter the path and the file extension e.g. "C:\TEMP\Orange.CSV".

If a storage card is used as storage location, specify the storage location as follows: "\StorageCard\<File name>".

If you define only a file name without specifying a path, the file is saved to the directory from which Runtime was started. If write access to this directory is not enabled in the Windows 7 operating system, the file is saved to the "VirtualStore" folder of the user directory.

When only one path for the export is specified, the file name is automatically created from the respective recipe name. This requires, for example, that the directory "D:\Temp\" has been created. If the directory "D:\Temp" does not exist, the directory name is used as the prefix for the file name, Temp_Recipe names.csv.

Overwrite

Determines whether an existing CSV file with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: CSV file will not be overwritten. The export process will not be carried out.

1 (hmiOverwriteAlways) = Yes: CSV file is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithConfirmation) = With confirmation: CSV file is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the export:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Export format of the recipe data records

If ".csv" is chosen as a file extension for the export file, then only valid characters from the ANSI character set will be supported. This also applies to separators in decimal numbers and list elements. The separators used are defined in the country/regional settings of the operating system of the exporting computer.

You may also set "Unicode text" (".txt") format for the export file. This file format supports the WinCC and WinCC Runtime character set. Again, the separators used are specified in the country/regional settings of the operating system of the exporting computer. This file format always uses tab separators in list elements.

The corresponding file import function also supports the ".csv" and ".txt" (Unicode) file formats.

Application example

Using a key, you want to export all data records and assign a checksum.

Notes on configuring

Configure the "ExportDataRecordsWithChecksum" system function on the "Press" event for the desired key. Transfer the following parameters:

- Recipe number/name = 1
- Data record number/name = 0
- File name = c:\temp\orange.csv
- Overwrite = 1
- Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data records are exported.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated. All data records of Recipe 1 are exported to the orange.csv file and assigned checksums. If the file already exists, it will be overwritten.

After exporting the data records, a system message is output.

See also

Availability for specific devices of system functions (Page 4612)

ExportImportUserAdministration

Description

Exports all users of the user administration of the currently active project to the given file or imports the users from the given file into the currently active project.

Users, their passwords and rights are saved in the user administration.

All users are overwritten when importing. The imported users are valid immediately.

Use in the function list

ExportImportUserAdministration (File name, Direction)

Use in user-defined functions

ExportImportUserAdministration (File_name, Direction)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

File name

Name of the file which contains the passwords or to which the passwords are written. Enter the file location and the file extension (*.txt), for example, "C:\TEMP\Passwords.txt".

Note

If a storage card is used as file location, specify the file location as follows: "\StorageCard\<File name>".

Direction

Specifies whether passwords are exported or imported:

0 (hmiExport) = Export: Passwords are exported.

1 (hmiImport) = Import: Passwords are imported.

See also

Availability for specific devices of system functions (Page 4612)

GoToHome

Description

Executes the key function <Home> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToHome

Use in user-defined functions

GoToHome

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

GoToEnd

Description

Executes the key function <End> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToEnd

Use in user-defined functions

GoToEnd

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

SafelyRemoveHardware

Description

Checks whether there is read or write access to the external storage medium. If there is no access, the external storage medium can be removed without data loss.

Use in the function list

SafelyRemoveHardware(Path, Result)

Use in user-defined functions

SafelyRemoveHardware(Path, Result)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Path

Path of storage medium , e.g. \Storage Card USB\

Result

The tag in which the result is entered.

TRUE : The storage medium can be removed safely. A corresponding system alarm will be output.

FALSE : The storage medium cannot be removed safely. A corresponding system alarm will be output.

See also

Availability for specific devices of system functions (Page 4612)

HTMLBrowserStop

Description

Performs the function "Stop" the HTML browser.

Use in the function list

HTMLBrowserStop (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is performed.

See also

Availability for specific devices of system functions (Page 4612)

HTMLBrowserScrollDown

Description

Scrolls down in the HTML browser.

Use in the function list

HTMLBrowserScrollDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Runtime Advanced (Page 4651)

System functions for Comfort Panels (Page 4636)

System functions for Mobile Panels (Page 4643)

System functions for Basic Panels 2nd Generation (Page 4618)

HTMLBrowserRefresh

Description

Performs the function "Refresh" of the HTML browser.

Use in the function list

HTMLBrowserRefresh (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is performed.

See also

Availability for specific devices of system functions (Page 4612)

HTMLBrowserScrollUp

Description

Scrolls up in the HTML browser.

Use in the function list

HTMLBrowserScrollUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

See also

System functions for Runtime Advanced (Page 4651)

System functions for Comfort Panels (Page 4636)

System functions for Mobile Panels (Page 4643)

System functions for Basic Panels 2nd Generation (Page 4618)

HTMLBrowserZoomIn

Description

Zooms in one zoom level in the HTML browser display.

Use in the function list

HTMLBrowserZooomIn (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Runtime Advanced (Page 4651)

System functions for Comfort Panels (Page 4636)

System functions for Mobile Panels (Page 4643)

System functions for Basic Panels 2nd Generation (Page 4618)

HTMLBrowserZoomOut

Description

Zooms out one zoom level in the HTML browser display.

Use in the function list

HTMLBrowserZoomOut (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Runtime Advanced (Page 4651)

System functions for Comfort Panels (Page 4636)

System functions for Mobile Panels (Page 4643)

System functions for Basic Panels 2nd Generation (Page 4618)

HTMLBrowserScrollLeft

Description

Scrolls left in the HTML browser.

Use in the function list

HTMLBrowserScrollLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Runtime Advanced (Page 4651)

System functions for Comfort Panels (Page 4636)

System functions for Mobile Panels (Page 4643)

System functions for Basic Panels 2nd Generation (Page 4618)

HTMLBrowserScrollRight

Description

Scrolls right in the HTML browser.

Use in the function list

HTMLBrowserScrollRight (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Runtime Advanced (Page 4651)

System functions for Comfort Panels (Page 4636)

System functions for Mobile Panels (Page 4643)

System functions for Basic Panels 2nd Generation (Page 4618)

HTMLBrowserPageUp

Description

Results in a page-by-page, upward cursor movement in the HTML browser.

Use in the function list

HTMLBrowserPageUp (screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Runtime Advanced (Page 4651)

System functions for Comfort Panels (Page 4636)

System functions for Mobile Panels (Page 4643)

System functions for Basic Panels 2nd Generation (Page 4618)

HTMLBrowserPageDown

Description

Results in a page-by-page, downward cursor movement in the HTML browser.

Use in the function list

HTMLBrowserPageDown (screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Runtime Advanced (Page 4651)

System functions for Comfort Panels (Page 4636)

System functions for Mobile Panels (Page 4643)

System functions for Basic Panels 2nd Generation (Page 4618)

HTMLBrowserStartPage

Description

Switches to the start page stored for the HTML browser.

Use in the function list

HTMLBrowserHome (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Runtime Advanced (Page 4651)

System functions for Comfort Panels (Page 4636)

System functions for Mobile Panels (Page 4643)

System functions for Basic Panels 2nd Generation (Page 4618)

HTMLBrowserForward

Description

Performs the function "Forward" of the HTML browser.

Use in the function list

HTMLBrowserForward (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is performed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Comfort Panels (Page 4636)

HTMLBrowserBack

Description

Performs the "Back" function of the HTML browser.

Use in the function list

HTMLBrowserBack (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is performed.

See also

Availability for specific devices of system functions (Page 4612)

ImportDataRecords

Description

Imports one or all data records of a recipe from a CSV file.

When a path is specified, all records of the given directory are imported.

Use in the function list

ImportDataRecords (File name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ImportDataRecords (File_name, Data_record number/name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

File name

Name of the CSV file from which the recipe data records are imported. Enter the storage location and the file extension (*.csv), for example, "C:\TEMP\Orange.csv", for Basic Panels "\USB_X61.1\Orange.csv".

Note

Does not apply for Basic Panels: If a storage card is used as file location, specify the file location as follows: "\StorageCard\<File name>".

Data record number/name

Number or name of the recipe data record to be imported. Specify "0" if all recipe data records are to be imported.

Overwrite

Determines whether existing recipe data records are to be overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data records are not overwritten. The import process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data records are overwritten without prompting.

2 (hmiOverwriteWithPrompting) = With confirmation: Recipe data records are not overwritten until confirmed.

Output status message

Determines whether a status message is output after the import:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Configurable objects

Object	Event
Tag	Value change Upper limit exceeded Lower limit violated
Function key (global)	Release Press
Function key (local)	Release Press
Screen	Loaded Cleared
Screen object	Press Release Click Change (or toggle for switch) Switch on Switch off Enable Disable
Scheduler	Time expired

See also

Availability for specific devices of system functions (Page 4612)

ImportDataRecordsWithChecksum

Description

Imports one or all data records of a recipe from a CSV file with a checksum and verifies the checksum.

Use in the function list

ImportDataRecordsWithChecksum (File name, Data record number/name, Overwrite, Output status message, Processing status, verify checksum)

Use in user-defined functions

ImportDataRecordsWithChecksum (File_name, Data_record number/name, Overwrite, Output_status_message, Processing_status, Verify_Checksum)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

File name

Name of the CSV file from which the recipe data records are imported. Enter the path and the file extension, for example, "C:\TEMP\Orange.CSV".

If a storage card is used as storage medium, specify the storage location as follows: "\StorageCard\<File name>".

If you specify a directory instead of an individual CSV file, all files in the specified directory will be imported.

Data record number/name

Number or name of the recipe data record to be imported. Specify "0" if all recipe data records are to be imported.

Overwrite

Determines whether existing recipe data records are to be overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data records are not overwritten. The import process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data records are overwritten without prompting.

2 (hmiOverwriteWithConfirmation) = With confirmation: Recipe data records are not overwritten until confirmed.

Output status message

Determines whether a status message is output after the import:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example to delay execution of other system functions, until this system function has been successfully completed.

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Verify checksum

Determines if the checksum should be verified during import:

0 (hmiFalse) = No: Checksum is not verified.

1 (hmiTrue) = Yes: Checksum is verified.

See also

Availability for specific devices of system functions (Page 4612)

InvertBit

Description

Inverts the value of the given tag of the "Bool" type:

- If the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

Use in the function list

InvertBit (Tag)

Use in user-defined functions

InvertBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag whose bit is set.

Example

The following program code inverts the value of the boolean tag b_value and outputs the result along with the original b_saved value.

```
{
BOOL b_value = 0;
BOOL b_saved = b_value;

//Invert variable
invertBit(b_value);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",b_value, b_saved);
  ...
}
```

See also

Availability for specific devices of system functions (Page 4612)

InvertBitInTag

Description

Inverts a bit in the given tag:

- If the bit in the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the bit in the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "InvertBit" system function instead.

Use in the function list

InvertBitInTag (Tag, Bit)

Use in user-defined functions

InvertBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which the given bit is set.

Bit

The number of the bit that is set.

When this system function is used in a user-defined function, the bits in a tag are counted from right to left. The counting begins with 0.

Example

The following program code inverts a bit at the specified bitposition in the bvalue tag and outputs the result along with the original bsaved value.

```
{
BYTE bvalue;
BYTE bsaved = bvalue;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bvalue, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bvalue, bsaved);
  ...
}
```

See also

Availability for specific devices of system functions (Page 4612)

InvertLinearScaling

Description

Assigns a value to the tag X, which is calculated from the value of the given tag Y using the linear function X = (Y - b) / a.

The tags X and Y must not be identical. This system function is the inverse of the "LinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

InvertLinearScaling (X, Y, b, a)

Use in user-defined functions

InvertLinearScaling (X, Y, b, a)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

X

The tag which is assigned the value calculated from the linear equation.

Y

The tag that contains the value used for calculation.

b

The value which is subtracted.

a

The value through which is divided.

Example

The following program code assigns a value to the varX tag by means of the InverseLinearScaling function.

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
 InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n, varX);
...
}
```

The saved return value can be processed in the following code.

See also

Availability for specific devices of system functions (Page 4612)

CalibrateTouchScreen

Description

Calls a program for calibrating the touch screen.

During the calibration process, there is a prompt to touch five positions on the screen display. Touch the screen display within 30 seconds, to confirm the calibration process. If the calibration is not completed within this time span, the calibration settings are discarded. The user prompt is in English.

Use this system function the first time you start the HMI device.

Use in the function list

CalibrateTouchScreen

Use in user-defined functions

CalibrateTouchScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Note

The CalibrateTouchScreen system function cannot be simulated.

See also

Availability for specific devices of system functions (Page 4612)

CopyLog

Description

Copies the contents of a log to another log. Tag values can only be copied to other data logs and alarms only to other alarm logs, however.

Note

If you copy a log using the "CopyLog" system function, it is possible that external applications will be unable to read certain country-specific special characters in the logged message text of the log copy. This does not apply to WinCC Runtime. WinCC Runtime can read the copied log files without errors.

Note

80% of the log entries are copied when copying the circular logs. 20% of entries are not copied because the space is reserved for buffer overflow by default.

Use in the function list

CopyLog (Log type, Destination log, Source log, Mode, Delete source log)

Use in user-defined functions

CopyLog (Log_type, Destination log, Source_log, Mode, Delete_source_log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

Destination log

Name of the log into which the entries are copied (Destination log).

Source log

Name of the log from which the entries are copied (Source log).

Mode

Determines what is done with copied entries in the destination log:

0 (hmiOverwrite) = Overwrite: Existing entries are overwritten.

2 (hmiAppend) = Append: The entries are inserted at the end of the destination log. When the configured size of the log has been reached, the destination log is treated like a circular log.

Delete source log

Determines whether the source log is deleted after copying.

0 (hmiNo) = No: Is not deleted.

1 (hmiYes) = Yes: Is deleted.

See also

Storage locations of logs (Page 4420)

Availability for specific devices of system functions (Page 4612)

TrendViewScrollForward

Description

Scrolls forward one display width in the Trend view.

Use in the function list

TrendViewScrollForward (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which is scrolled forward.

See also

Availability for specific devices of system functions (Page 4612)

TrendViewScrollBack

Description

Scrolls back one display width in the trend view.

Use in the function list

TrendViewScrollBack (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which is scrolled back.

See also

Availability for specific devices of system functions (Page 4612)

TrendViewExtend

Description

Reduces the time period which is displayed in the trend view.

Use in the function list

TrendViewExtend (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the displayed time period is reduced.

See also

Availability for specific devices of system functions (Page 4612)

TrendViewCompress

Description

Increases the time period which is displayed in the trend view.

Use in the function list

TrendViewCompress (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the displayed time period is increased.

See also

Availability for specific devices of system functions (Page 4612)

TrendViewRulerLeft

Description

Moves the read-line backwards (to the left) in the trend view.

Note

In order to be able to move the read-line, the read-line must have been switched on. This is done using the "TrendViewSetRulerMode" system function.

Use in the function list

TrendViewRulerLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is moved backwards.

See also

TrendViewSetRulerMode (Page 4708)

Availability for specific devices of system functions (Page 4612)

TrendViewRulerRight

Description

Moves the read-line forwards (to the right) in the trend view.

Note

In order to be able to move the read-line, the read-line must have been switched on. This is done using the "TrendViewSetRulerMode" system function.

Use in the function list

TrendViewRulerRight (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is moved forward.

See also

TrendViewSetRulerMode (Page 4708)

Availability for specific devices of system functions (Page 4612)

TrendViewSetRulerMode

Description

Hides or shows the read-line in the trend view. The read-line displays the Y value belonging to the X value.

Note

To ensure that the ruler is displayed, you have to activate the setting "Show ruler" in the trend view properties.

Use in the function list

TrendViewSetRulerMode (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is hidden or shown.

See also

TrendViewRulerRight (Page 4707)

TrendViewRulerLeft (Page 4706)

Availability for specific devices of system functions (Page 4612)

TrendViewStartStop

Description

Stops the trend recording or continues the trend recording in the trend view.

Use in the function list

TrendViewStartStop (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the recording of the trend is started or stopped.

See also

Availability for specific devices of system functions (Page 4612)

TrendViewBackToBeginning

Description

Scrolls back to the beginning of the display range in the trend view.

Use in the function list

TrendViewBackToBeginning (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which you scroll to the beginning of the display range.

See also

Availability for specific devices of system functions (Page 4612)

LoadDataRecord

Description

Loads the given recipe data record from the memory medium of the HMI device in the recipe tags. This system function is used, for example, to display a recipe data record in the recipe screen.

Activate the "Synchronize recipe view and recipe tags" option in the synchronization settings of the recipe. If this option is deactivated, the system function has no effect.

Use in the function list

LoadDataRecord (Recipe number/name, Data record number/name, Processing status)

Use in user-defined functions

LoadDataRecord (Recipe_number/name, Data_record_number/name, Confirmation, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which a recipe data record is loaded.

Data record number/name

Number or name of the recipe data record to be loaded.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

GetUserName

Description

Writes the user name of the user currently logged on to the HMI device in the given tag.

If the given tag has a control connection, the user name is also available in the PLC connection. This system function makes it possible, for example, to implement a user-dependent release of certain functionalities.

Use in the function list

GetUserName (Tag)

Use in user-defined functions

GetUserName (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the user name is written.

See also

Availability for specific devices of system functions (Page 4612)

GetDataRecordFromPLC

Description

Transfers the selected recipe data record from the PLC to the storage medium of the HMI device.

Use in the function list

GetDataRecordFromPLC (Recipe number/name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

GetDataRecordFromPLC (Recipe_number/name, Data_record_number/name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data records are transferred.

Data record number/name

Number or name of the recipe data record which is transferred from the PLC to the data medium of the HMI device.

Overwrite

Determines whether an existing recipe data record with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data record is not overwritten. The transfer process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data record is overwritten without prompting.

2 (hmiOverwriteWithPrompting) = With confirmation: Recipe data record is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the transfer:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Application example

You want to transfer a data record from the PLC to the data medium of the HMI device using a key.

Notes on configuring

Configure the "GetDataRecordFromPLC" system function on the "Press" event for the desired key. Transfer the following parameters:

Recipe number/name = 1

Data record number/name = 1

Overwrite = 1

Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data record is transferred from the PLC.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated and the first data record of Recipe 1 is transferred from the PLC to the data medium of the HMI device. If the recipe data record already exists, it will be overwritten.

A system message is output after the transfer.

See also

Availability for specific devices of system functions (Page 4612)

GetDataRecordName

Description

Writes the name of the given recipe and recipe data record to the given tags.

Note

If the recipe or the recipe data record do not exist, wildcards ("###") are written to the tags.

Use in the function list

GetDataRecordName (Recipe number, Data record number, Recipe name, Data record name, Processing status)

Use in user-defined functions

GetDataRecordName (Recipe_number, Data_record_number, Recipe_name, Data_record_name, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number

Number of the recipe whose name is written to the given tag.

Data record number

Number of the recipe data record whose name is written to the given tag.

Recipe name

The tag to which the recipe name is written. The tag must be of the STRING type.

Data record name

The tag to which the name of the recipe data record is written. The tag must be of the STRING type.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Application example

You want to output the names of the displayed recipes and the name of the displayed recipe data record on the HMI device.

Configure the following tags:

- "RecNumber" of type INTEGER
- "RecDataNumber" of type INTEGER
- "RecName" of type STRING
- "RecDataName" of type STRING

Configure a recipe view with the tags "RecNumber" for the recipe number and "RecDataNumber" for the data record number.

Configure the "GetDataRecordName" system function on the "Press" event for a button and pass the following parameters:

- Recipe number: RecNumber
- Data record number: RecDataNumber
- Recipe name: RecName
- Data record name: RecDataName

Configure two output fields and connect these to the "RecName" and "RecDataName" tags.

Select a recipe and a data record number from the recipe view. As soon as the button is activated, the system function is triggered and the name of the recipe and the recipe data record are displayed in both output fields.

See also

Availability for specific devices of system functions (Page 4612)

GetDataRecordTagsFromPLC

Description

Transfers the values of the recipe data record loaded in the PLC to the corresponding recipe tags.

This system function is used, for example, during teach-in mode on a machine.

Use in the function list

GetDataRecordTagsFromPLC (Recipe number/name, Processing status)

Use in user-defined functions

GetDataRecordTagsFromPLC (Recipe_number/name, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe data record whose values are written from the PLC to the tags.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

GetGroupNumber

Description

Reads the number of the group to which the user logged on to the HMI device belongs, and writes it to the given tag.

Use in the function list

GetGroupNumber (Tag)

Use in user-defined functions

GetGroupNumber (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the number of the group is written.

See also

Availability for specific devices of system functions (Page 4612)

GetBrightness

Description

Reads the value of the brightness.

Use in the function list

GetBrightness (Brightness)

Use in user-defined functions

GetBrightness (Brightness)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Brightness

The tag to which the value is written.

See also

Availability for specific devices of system functions (Page 4612)

GetPassword

Description

Writes the password of the user currently logged on to the HMI device in the given tag.

Note

Make sure that the value of the given tag is not displayed in another place in the project.

Note

The passwords of SIMATIC Logon users cannot be read.

Use in the function list

GetPassword (Tag)

Use in user-defined functions

GetPassword (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the password is written.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Runtime Advanced (Page 4651)

ReadPLCMode

Description

Evaluates the current state of the connected PLC.

The system function "ReadPLCMode" can only be configured for the following PLCs:

- SIMATIC S7-1200
- SIMATIC S7-1500

Use in the function list

ReadPLCMode (connection, mode)

Use in user-defined functions

GetPLCMode (Connection, Mode)

Parameters

Connection

Connection of PLC and HMI device.

Mode

Evaluates the mode of the connected PLC.
8: RUN = The PLC program is being executed.
4: STOP = The PLC program has been interrupted.
Select a suitable user variable for the evaluation.

LinearScaling

Description

Assigns a value to the tag Y, which is calculated from the value of the given tag X using the linear function Y= (a *X) + b.

The inverse of this function is the "InvertLinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

LinearScaling (Y, a, X, b)

Use in user-defined functions

LinearScaling (Y, a, X, b)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Y

The tag which is assigned the value calculated from the linear equation.

a

The value with which is multiplied.

X

The tag that contains the value used for calculation.

b

The value that is added.

Example

The following program code uses the LinearScaling function to assign a value to the Yvar tag.

```
{
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
 LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n, Yvar);
...
}
```

The saved return value can be processed in the following code.

See also

Availability for specific devices of system functions (Page 4612)

ClearLog

Description

Deletes all data records in the given log.

Use in the function list

ClearLog (Log type, Log)

Use in user-defined functions

ClearLog (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail log Available for GMP-compliant projects Additional information is available under "Activating GMP-compliant configuration".

Log

Name of the log from which all entries are deleted.

See also

Availability for specific devices of system functions (Page 4612)

DeleteDataRecord

Description

Deletes a recipe data record.

Several data records can be deleted from one or more recipes.

Use in the function list

DeleteDataRecord (Recipe number/name, Data record number/name, Confirmation, Output status message, Processing status)

Use in user-defined functions

DeleteDataRecord

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data records are deleted. "0" is specified if you want to delete recipe data records from all available recipes.

Data record number/name

Number or name of the recipe data record to be deleted. "0" is specified if you want to delete all recipe data records.

Confirmation

Determines whether the operator should confirm the deletion:

0 (hmiOff) = Off: Deletion is begun without confirmation.

1 (hmiOn) = On: The starting of the deletion process must be confirmed.

Output status message

Determines whether a status message is output after the deletion:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

ClearDataRecordMemory

Description

Deletes all recipes data records from the specified storage medium.

Use in the function list

ClearDataRecordMemory (Storage location, Confirmation, Output status message, Processing status)

Use in user-defined functions

ClearDataRecordMemory (Storage_location, Confirmation, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Storage location

Determines the storage location:

0 (hmiFlashMemory) = Flash memory: Internal flash memory of the HMI device

1 (hmiStorageCard) = Storage card

2 (hmiStorageCard2) = Storage card 2

3 (hmiStorageCard3) = Storage card MultiMediaCard

4 (hmiStorageCard4) = Storage card USB

Confirmation

Determines whether the operator should confirm the deletion:

0 (hmiOff) = Off: Deletion is begun without confirmation.

1 (hmiOn) = On: The starting of the deletion process must be confirmed.

Output status message

Determines whether a status message is output after the deletion:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

ClearAlarmBuffer

Description

Deletes alarms from the alarm buffer on the HMI device.

Note

Alarms which have not yet been acknowledged are also deleted.

Use in the function list

ClearAlarmBuffer (Alarm class number)

Use in user-defined functions

ClearAlarmBuffer (Alarm_class_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Alarm class number

Determines which alarms are to be deleted from the alarm buffer:

0 (hmiAll) = All alarms/events

1 (hmiAlarms) = Alarms of alarm class "Errors"

2 (hmiEvents) = Alarms of alarm class "Warnings"

3 (hmiSystem) = Alarms of alarm class "System"

4 (hmiS7Diagnosis) = Alarms of alarm class "Diagnosis Events"

Note

Device dependency

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

See also

Availability for specific devices of system functions (Page 4612)

ClearAlarmBufferProTool

Description

The system function exists to ensure compatibility.

It has the same functionality as the "ClearAlarmBuffer" system function, but uses the old ProTool numbering.

Use in the function list

ClearAlarmBufferProTool (Alarm class number)

Use in user-defined functions

ClearAlarmBufferProtoolLegacy (Alarm_class_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Alarm class number

Alarm class number whose messages are to be deleted:

-1 (hmiAllProtoolLegacy) = All alarms/events

0 (hmiAlarmsProtoolLegacy) = Alarms of alarm class "Errors"

1 (hmiEventsProtoolLegacy) = Alarms of alarm class "Warnings"

2 (hmiSystemProtoolLegacy) = Alarms of alarm class "System"

3 (hmiS7DiagnosisProtoolLegacy) = Alarms of alarm class "Diagnosis Events"

Note

Device dependency

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

See also

Availability for specific devices of system functions (Page 4612)

AlarmViewUpdate

Description

Updates the enhanced alarm view.

Use in the function list

AlarmViewUpdate (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view which is updated.

See also

Availability for specific devices of system functions (Page 4612)

AlarmViewLoopInAlarm

Description

Triggers the "Loop-in-Alarm" event for all alarms selected in the given alarm view.

This system function is used when the integrated button of the ActiveX control should not be used.

You can configure a system function for the "Loop-in-Alarm" event in turn. For example, it is possible to change to the process screen in which the alarm appeared.

Note

If the alarms to be edited have not yet been acknowledged, the acknowledgment takes place automatically when this system function is called up.

Use in the function list

AlarmViewLoopInAlarm (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

Note

The HMI devices listed below do not support this system function for the "screen" object: OP 73, OP 77A, TP 177A.

See also

Availability for specific devices of system functions (Page 4612)

AlarmViewAcknowledgeAlarm

Description

Acknowledges the alarms selected in the given alarm view.

This system function is used when the integrated button of the ActiveX control should not be used.

Use in the function list

AlarmViewAcknowledgeAlarm (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

Note

The HMI devices listed below do not support this system function for the "screen" object: OP 73, OP 77A, TP 177A.

See also

Availability for specific devices of system functions (Page 4612)

AlarmViewShowOperatorNotes

Description

Displays the configured tooltip of the alarm selected in the given alarm view.

Use in the function list

AlarmViewShowOperatorNotes (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

Note

The HMI devices listed below do not support this system function for the "screen" object: OP 73, OP 77A, TP 177A.

See also

Availability for specific devices of system functions (Page 4612)

OpenAllLogs

Description

Reestablishes the connection between WinCC and the logs. Logging can be continued.

Note

Run the "StartLogging" system function in order to restart logging.

Use in the function list

OpenAllLogs

Use in user-defined functions

OpenAllLogs

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "StopLogging" and "CloseAllLogs" system functions on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When the button "Close Archive" is pressed, the specified log is stopped and all open logs are closed. The data medium can be changed. You open all logs with the "Open Archive" button. Logging will be continued in the specified log.

See also

Availability for specific devices of system functions (Page 4612)

OpenScreenKeyboard

Description

Hides or shows the screen keyboard.

The screen keyboard remains open until the screen keyboard is explicitly closed. In this way, the screen keyboard can also be used in other applications.

Use in the function list

OpenScreenKeyboard (Layout)

Use in user-defined functions

OpenScreenKeyboard (Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Layout

Specifies whether the window is opened minimized or maximized with the screen keyboard:

0 (hmiScreenKeyboardMinimized) = Minimized

1 (hmiScreenKeyboardMaximized) = Maximized

See also

Availability for specific devices of system functions (Page 4612)

OpenFileBrowser

Description

Opens a file browser dialog.

Depending on the mode the user can select a file or a folder.

Use in the function list

OpenFileBrowser (Mask, Output Folder, File, Navigation Back, Navigation Up, New folder, Path, Status)

Use in user-defined functions

-

Parameters

Mask

Sets a mask for filtering the displayed files. This parameter is only evaluated in "File" mode. The filter criteria must be delimited by ; semicolon. Example: *.htm;*.html

Start folder

Specifies the folder with which the file selection dialog starts.

File

Specifies the mode in which the dialog is opened.

0 = folder search mode

1 = file search mode

Navigation Back

Specifies whether or not the "Back" button is activated. This button offers simple navigation to the last folder you opened.

Navigation Up

Specifies whether or not the "Up" button is activated. This button offers simple navigation to the parent folder.

New folder

Specifies whether or not the button can be used to create a new folder.

Path

Contains the path name of the last folder opened after you close the dialog.

Status

Returns information about the dialog closing mode

-1 = dialog was canceled

0 = dialog was closed with "OK".

OpenControlPanelDialog

Description

Opens a dialog that you can use to edit selected Control Panel settings.

This system function lets you set the following on the HMI device:

- Properties and value of the IP address
- User identification on the network
- WinCC Internet settings

Note

Project security

The "OpenControlPanelDialog" system function is used to bypass the SecureMode on the HMI device. Take the corresponding precautions to ensure the security of your project.

Use in the function list

OpenControlPanelDialog (Dialog)

Use in user-defined functions

-

Parameters

Dialog

Sets the Control Panel dialog to be opened.

- PROFINET_X1: Setting of the IP address and Ethernet parameters
- PROFINET_X3: Setting of the IP address and Ethernet parameters; only for Comfort Panel KP 1500, TP 1500; TP1900, TP2200
- WinCC Internet settings: Setting for Web Server, e-mail notification, provided the HMI device supports these functions
- Network ID: Setting for identification on the network, provided the HMI device supports these functions

See also

Availability for specific devices of system functions (Page 4612)

OpenCommandPrompt

Description

Opens a Windows system prompt.

This function is used, e.g., to copy files or to call up another application.

Use in the function list

OpenCommandPrompt

Use in user-defined functions

OpenCommandPrompt

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

OpenInternetExplorer

Description

Opens the Internet Explorer on the HMI device.

If the Internet Explorer is already open, it will be closed and reopened when you call the system function.

Note

The Internet Explorer saves data temporarily in the DRAM file system of the HMI device, e.g. the last web sites that were be called up.

This data can be saved using the "BackupRAMFileSystem" system function so that it is still available when the HMI device is restarted.

Use in the function list

OpenInternetExplorer (Start page)

Use in user-defined functions

OpenInternetExplorer (Start_page)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Start page

The page which is loaded when Internet Explorer is started, e.g. "www.siemens.com".

See also

Availability for specific devices of system functions (Page 4612)

OpenControlPanel

Description

Opens the window that displays the Windows CE control panel. You cannot use this system function on a PC.

This system function enables you to set the following on the Windows CE-based HMI device:

- Select printer
- Select transfer properties
- Configure screen saver
- Configure flash memory

Note

No backup or restore during Runtime

Backup and restore functions may only be executed if Runtime has been terminated. Otherwise, this could result in undesired effects, such as display errors.

Use in the function list

OpenControlPanel

Use in user-defined functions

OpenControlPanel

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

--

OpenTaskManager

Description

Shows the task manager.

The task manager allows changing to other open applications on the HMI device.

Note

The appearance of the task manager depends on the operating system installed.

Use in the function list

OpenTaskManager

Use in user-defined functions

OpenTaskManager

Can be used if the configured device supports user-defined scripts. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

PDFScrollDown

Description

Scrolls down in the PDF file in the PDF view.

Use in the function list

PDFScrollDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)
System functions for Mobile Panels (Page 4643)
System functions for Comfort Panels (Page 4636)
System functions for Runtime Advanced (Page 4651)

PDFScrollUp

Description

Scrolls up in the PDF file in the PDF view.

Use in the function list

PDFScrollUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object
Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)
System functions for Mobile Panels (Page 4643)
System functions for Comfort Panels (Page 4636)
System functions for Runtime Advanced (Page 4651)

PDFFitToWidth

Description

Fits the display of the PDF file to the height of the PDF view window.

Use in the function list

PDFFitToWidth (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Mobile Panels (Page 4643)

System functions for Comfort Panels (Page 4636)

System functions for Runtime Advanced (Page 4651)

PDFFitToHeight

Description

Fits the display of the PDF file to the height of the PDF view window.

Use in the function list

PDFFitToHeight (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Mobile Panels (Page 4643)

System functions for Comfort Panels (Page 4636)

System functions for Runtime Advanced (Page 4651)

PDFGoToFirstPage

Description

Switches to the first page of the PDF file in the PDF view.

Use in the function list

PDFGoToFirstPage (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Mobile Panels (Page 4643)

System functions for Comfort Panels (Page 4636)

System functions for Runtime Advanced (Page 4651)

PDFGoToLastPage

Description

Switches to the last page of the PDF file in the PDF view.

Use in the function list

PDFGoToLastPage (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Mobile Panels (Page 4643)

System functions for Comfort Panels (Page 4636)

System functions for Runtime Advanced (Page 4651)

PDFGoToNextPage

Description

Switches to the next page of the PDF file in the PDF view.

Use in the function list

PDFGoToNextPage (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Mobile Panels (Page 4643)

System functions for Comfort Panels (Page 4636)

System functions for Runtime Advanced (Page 4651)

PDFGoToPage

Description

Switches to the entered page of the PDF file in the PDF view.

Use in the function list

PDFGoToPage (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Mobile Panels (Page 4643)

System functions for Comfort Panels (Page 4636)

System functions for Runtime Advanced (Page 4651)

PDFGoToPreviousPage

Description

Switches to the previous page of the PDF file in the PDF view.

Use in the function list

PDFGoToPreviousPage (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Mobile Panels (Page 4643)

System functions for Comfort Panels (Page 4636)

System functions for Runtime Advanced (Page 4651)

PDFZoomIn

Description

Zooms in one zoom level in the display of the PDF file in the PDF view.

Use in the function list

PDFZoomIn (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Mobile Panels (Page 4643)

System functions for Comfort Panels (Page 4636)

System functions for Runtime Advanced (Page 4651)

PDFZoomOut

Description

Zooms out one zoom level in the display of the PDF file in the PDF view.

Use in the function list

PDFZoomOut (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)
System functions for Mobile Panels (Page 4643)
System functions for Comfort Panels (Page 4636)
System functions for Runtime Advanced (Page 4651)

PDFScrollLeft

Description

Scrolls to the left in the PDF file in the PDF view.

Use in the function list

PDFScrollLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object
Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)
System functions for Mobile Panels (Page 4643)
System functions for Comfort Panels (Page 4636)
System functions for Runtime Advanced (Page 4651)

PDFScrollRight

Description

Scrolls to the right in the PDF file in the PDF view.

Use in the function list

PDFScrollRight (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Mobile Panels (Page 4643)

System functions for Comfort Panels (Page 4636)

System functions for Runtime Advanced (Page 4651)

PDFZoomOriginal

Description

Switches the display of the PDF file in the PDF view to the original size.

Use in the function list

PDFZoomOriginal (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

See also

Availability for specific devices of system functions (Page 4612)

System functions for Mobile Panels (Page 4643)

System functions for Comfort Panels (Page 4636)

System functions for Runtime Advanced (Page 4651)

TerminatePROFIsafe

Description

Disconnects the PROFIsafe connection for fail-safe operation between a KTP Mobile Panel and the PLC.

After execution of the system function "TerminatePROFIsafe", the connector of the KTP Mobile Panel can be removed from the PLC without the system signaling an error.

Use in the function list

TerminatePROFIsafe

Use in user-defined functions

TerminatePROFIsafe

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

--

EstablishPROFIsafe

Description

Establishes the PROFIsafe connection for fail-safe operation between a KTP Mobile Panel and the PLC if the connection was previously disconnected by means of TerminatePROFIsafe.

Note

A connection for the PROFIsafe communication can only be established if ONLINE mode has been set on the HMI device. Use the "SetDeviceMode" system function for this purpose.

Use in the function list

EstablishPROFIsafe

Use in user-defined functions

EstablishPROFIsafe

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

--

AcknowledgeAlarm

Description

Acknowledges all selected alarms.

This system function is used when the HMI device does not have an ACK key or when the integrated key of the alarm view should not be used.

This system function can only be used for function keys.

Use in the function list

AcknowledgeAlarm

Use in user-defined functions

AcknowledgeAlarm

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewNewDataRecord

Description

Creates a new data record in the given recipe view.

Use in the function list

RecipeViewNewDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the new recipe data record is created.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewGetDataRecordFromPLC

Description

Transfers the data record that is currently loaded in the PLC to the HMI device and displays it in the recipe view.

Use in the function list

RecipeViewGetDataRecordFromPLC (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record from the PLC is displayed.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewClearDataRecord

Description

Deletes the data record which is currently displayed in the recipe view.

Use in the function list

RecipeViewClearDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the displayed recipe data record is deleted.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewMenu

Description

Opens the menu of the specified simple recipe view.

Only use this system function at a simple recipe view.

Use in the function list

RecipeViewMenu (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the menu is to be opened.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewOpen

Description

Displays the data record values in the given recipe view or changes to the next selection field. The system function has no effect if the selection field for the recipe data record values is displayed on the HMI device.

Operation sequence of the selection lists in runtime:

- Recipe name
- Data record name
- RecipeDataRecordValues

This system function is used when a simple recipe view has been configured. In the simple recipe view, only one selection list is displayed at a time on the HMI device. Use the "RecipeViewBack" system function to display the previous selection list.

Use in the function list

RecipeViewOpen (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the command is triggered.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewBack (Page 4750)

RecipeViewSetDataRecordToPLC

Description

Transfers the recipe data record which is currently displayed in the recipe view to the PLC.

Use in the function list

RecipeViewSetDataRecordToPLC (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view from which the recipe data record is transferred to the connected PLC.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewSaveDataRecord

Description

Saves the recipe data record which is currently displayed in the recipe view.

Use in the function list

RecipeViewSaveDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record is saved.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewSaveAsDataRecord

Description

Saves the data record currently being displayed in the recipe view under a new name.

Use in the function list

RecipeViewSaveAsDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Saves the data record currently being displayed in the recipe view under a new name and/or new number.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewSynchronizeDataRecordWithTags

Description

Synchronizes the values of the data record which is currently displayed in the recipe view with their recipe tags. Only use this system function at an advanced recipe view.

During synchronization, all values of the data record are written to their recipe tags.

Use in the function list

RecipeViewSynchronizeDataRecordWithTags (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the values are synchronized with their tags.

Note

The OP 77A and TP 177A HMI devices do not support this function for the screen object.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewRenameDataRecord

Description

Renames the selected data record in the given recipe view.

Use in the function list

RecipeViewRenameDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record is renamed.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewShowOperatorNotes

Description

Displays the configured tooltip of the specified recipe view.

Use in the function list

RecipeViewShowOperatorNotes (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view whose configured tooltip is displayed.

See also

Availability for specific devices of system functions (Page 4612)

RecipeViewBack

Description

Returns to the previous selection list in the given recipe view.

The system function has no effect if the selection field for the recipe is displayed on the HMI device. Operation sequence of the selection lists in runtime:

- Recipe name
- Data record name
- RecipeDataRecordValues

This system function is used when a simple recipe view has been configured. In the simple recipe view, only one selection list is displayed at a time on the HMI device. Use the "RecipeViewOpen" system function to display the recipe data record values or the next selection field.

Use in the function list

RecipeViewBack (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the command is triggered.

See also

Availability for specific devices of system functions (Page 4612)

ResetBit

Description

Sets the value of a "Bool" type tag to 0 (FALSE).

Use in the function list

ResetBit (Tag)

Use in user-defined functions

ResetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 0 (FALSE).

Example

The following program code resets the value of the boolean tag b_value to 0 by means of the ResetBit function and outputs the result along with the original b_saved value.

```
{
BOOL b_value = 1;
BOOL b_saved = b_value;

//Reset bit
  ResetBit (b_value);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",b_value, b_saved);
...
}
```

See also

ResetBitInTag (Page 4752)

Availability for specific devices of system functions (Page 4612)

ResetBitInTag

Description

Sets a bit in the specified tag to 0 (FALSE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "ResetBit" system function instead.

Use in the function list

ResetBitInTag (Tag, Bit)

Use in user-defined functions

ResetBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 0 (FALSE).

Bit

The number of the bit that is set to 0 (FALSE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Example

The following program code sets a bit at the specified bitposition in the bvalue tag to 0 and outputs the result along with the original bsaved value.

```
{
BYTE bvalue;
BYTE bsaved = bvalue;
BYTE bitposition = 2;

//Reset bit in bitposition
ResetBitInTag (bvalue, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bvalue, bsaved);
  ...
}
```

See also

Availability for specific devices of system functions (Page 4612)

ResetBit (Page 4750)

PressButton

Description

You configure the system function to the function keys of an HMI device. The system function "PressButton" triggers the function that is assigned to the event "PressKey" at the specified screen object.

Use this system function when you want to operate a button in a screen with a function key of the HMI device, for example.

Note

The "PressButton" and "ReleaseButton" system functions must always be configured together. If you configure the "PressButton" system function on the "Press key" event for a function key, the "ReleaseButton" system function is configured on the "Release" event for the same function key.

Use in the function list

PressButton (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object on which the event is triggered.

See also

Availability for specific devices of system functions (Page 4612)

ReleaseButton

Description

The system function can only be configured on the function keys of an HMI device and triggers the "Release button" event at the specified screen object.

Use this system function when you want to operate a button in a screen with a function key of the HMI device, for example.

Note

The "PressButton" and "ReleaseButton" system functions must always be configured together. If you configure the "PressButton" system function on the "Press key" event for a function key, then the "ReleaseButton" system function is configured on the "Release key" event for the same function key.

Use in the function list

ReleaseButton (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object on which the event is triggered.

See also

Availability for specific devices of system functions (Page 4612)

ShiftAndMask

Description

This system function converts the input bit pattern of the source tags into an output bit pattern of the target tags. This involves bit shifting and masking.

Note

If the source and target tag have a different number of bits, using the system function in the target tag can result in a violation of the value range.

Use in the function list

ShiftAndMask (Source tag, Target tag, Bits to shift, Bits to mask)

Use in user-defined functions

ShiftAndMask (Source_tag, Target_tag, Bits_to_shift, Bits_to_mask)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Source tag

The tag includes the input bit pattern. Integer-type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The actual value 72 is set at the 16-bit integer source tag: 0000000001001000.

Target tag

The output bit pattern is saved in the tag. Integer type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The shifted input bit pattern is multiplied by the bit mask, with bit-by-bit logical AND operation: 0000000000001001. The resultant decimal value "8" is saved to the target tag.

Please note the following:

- The source and target tags have the same number of bits.
- The number of bits to shift is less than the number of bits in the source tag and target tag.
- Bits to mask does not have more bits than the source tag and the target tag.

Bits to shift

Number of bits by which the input bit pattern is shifted right. A negative value shifts the input bit pattern to the left.

Example: "Bits to shift" has the value "+3". The input bit pattern is shifted right by three bits when the system function is called: 0000000000001001.
Bits to the left are padded with "0". Three bits are truncated on the right. The new decimal value is "9".

Note

The left bit is "1" in a source tag of the data type with negative signed integer. This sign bit is padded with "0" when the bits are shifted right. The sign changes to "+".

Bits to mask

An integer serves as bit mask. The bit pattern is used to multiply the shifted input bit pattern. Example: Integer "2478" with the bit pattern "0000100110101110".

You can enter the bit mask in three different ways:

- Hexadecimal: First enter the prefix "0h" or "0H", followed by an optional space for better readability. Then group the bit pattern in blocks of four (0000)(1001)(1010)(1110) and set each block in hexadecimal code: (0)(9)(A)(E). Only the characters 0-9, A-F, a-f are allowed: "0h 09AE".
- Binary: First enter the prefix "0b" or "0B", followed by an optional space for better readability. Then group the binary bit pattern into blocks of four 0000 1001 1010 1110 with spaces in between as a check. Only the characters "0" or "1" are allowed: "0b 0000 1001 1010 1110".
- Decimal: Enter the value "2478" directly, without a prefix.

See also

Availability for specific devices of system functions (Page 4612)

CloseAllLogs

Description

Disconnects the connection between WinCC and all logs.

Note

Before you close a log, the logging function must first be stopped in the log. Use the "StopLogging" system function.

Use in the function list

CloseAllLogs

Use in user-defined functions

CloseAllLogs

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "StopLogging" and "CloseAllLogs" system functions on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When the button "Close Archive" is pressed, the specified log is stopped and all open logs are closed. The data medium can be changed. You open all logs with the "Open Archive" button. The logging process will continue in the specified log.

See also

Availability for specific devices of system functions (Page 4612)

SetDataRecordToPLC

Description

Transfers the given recipe data record directly from the data medium of the HMI device to the PLC with which the HMI device is connected.

Note

The values of the recipe data record don't need to be displayed on the HMI device.

Use in the function list

SetDataRecordToPLC (Recipe number/name, Data record number/name, Output status message, Processing status)

Use in user-defined functions

SetDataRecordToPLC (Recipe_number/name, Data_record_number/name, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data record is transferred to the PLC.

Data record number/name

Number or name of the recipe data record to be transferred to the PLC.

Output status message

Determines whether a status message is output after the transfer:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

SetDataRecordTagsToPLC

Description

Transfers the values of the recipe tags to the PLC. The recipe tags contain the values of the data record which is displayed on the HMI device.

Use in the function list

SetDataRecordTagsToPLC (Recipe number/name, Processing status)

Use in user-defined functions

SetDataRecordTagsToPLC (Recipe_ number/name, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data record is transferred to the PLC.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

PageDown

Description

Executes the key function <Pagedown> on the HMI device.

This system function can only be used for function keys.

Use in the function list

PageDown

Use in user-defined functions

PageDown

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

-

See also

Availability for specific devices of system functions (Page 4612)

PageUp

Description

Executes the key function <PageUp> on the HMI device.

This system function can only be used for function keys and tasks with a time trigger.

Use in the function list

PageUp

Use in user-defined functions

PageUp

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

-

See also

Availability for specific devices of system functions (Page 4612)

SendEMail

Description

Sends an e-mail from the HMI device to the given addressee.

This system function is used, for example, when, in the case of service, the alarm is to be passed on directly to the service technician.

Note

To send alarms as e-mails, the HMI system must have an e-mail client at its disposal.

Use in the function list

SendEMail (Address, Subject, Text, Return address)

Use in user-defined functions

SendEMail (Address, Subject, Text, Return_address)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Address

The e-mail address of the addressee.

Subject

The subject line of the e-mail.

Text

The text sent in the e-mail.

Return address

The e-mail address to which the addressee of this e-mail should send the reply.

See also

Availability for specific devices of system functions (Page 4612)

SetAcousticSignal

Description

Configures the acoustic feedback of touch screen operation on the HMI device.

Note

The configuration that was set at Switch off is reestablished when restarting the HMI device.

Use in the function list

SetAcousticSignal (Volume)

Use in user-defined functions

SetAcousticSignal (Volume)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Volume

Determines whether and how loud an acoustic signal is emitted:

-1 (hmiToggle) = Toggle: Toggles the emission of the acoustic signal as follows: Muted > Quiet > Loud.

0 (hmiMuted) = Mute: no acoustic signal

1 (hmiQuiet) = Quiet: quiet acoustic signal

2 (hmiLoud) = Loud: loud acoustic signal

See also

Availability for specific devices of system functions (Page 4612)

SetDisplayMode

Description

Changes the settings of the screen in which the runtime software runs.

The runtime software runs in full-screen mode as default. The Windows task switch is disabled.

Use in the function list

SetDisplayMode (Layout)

Use in user-defined functions

SetDisplayMode (Display mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines the settings for the screen in which the runtime software runs.

1 (hmiScreenFull): Full-screen: Title bar of the screen is not visible

2 (hmiScreenMaximized): Maximized

3 (hmiScreenRestore): Restore: The last used screen setting is used. This layout can only be used when the window is displayed minimized or maximized.

4 (hmiScreenMinimized): Minimized

5 (hmiScreenAutoAdjust): Automatic: The size of the window is set so that all screen objects in it will be visible.

6 (hmiScreenOnTop): Foreground; either the window appears in the foreground or the program icon associated with the window flashes on the taskbar depending on the Windows setting. The setting can be changed in the Windows configuration and applies to all Windows applications.

See also

Availability for specific devices of system functions (Page 4612)

SetDeviceMode

Description

Toggles the operating mode on the HMI device. The following types of operation are possible: "Online", "Offline" and "Loading".

Use in the function list

SetDeviceMode (Operating mode)

Use in user-defined functions

SetDeviceMode (Operating_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Operating mode

Determines the operating mode of the HMI device:

0 (hmiOnline) = Online: The connection to the PLC is established. The configured connection status is always set in this process. The states that were last used in Runtime are not considered.

1 (hmiOffline) = Offline: The connection to the PLC is disconnected.

2 (hmiTransfer) = load: A project can be transferred from the configuration computer to the HMI device.

Note

If you use a PC as an HMI device, the runtime software will be exited when you switch operating mode after "Load".

See also

Availability for specific devices of system functions (Page 4612)

SetConnectionMode (Page 4778)

SetBit

Description

Sets the value of a "Bool" type tag to 1 (TRUE).

Use in the function list

SetBit (Tag)

Use in user-defined functions

SetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 1 (TRUE).

Example

The following program code sets the value of the boolean tag b_value to 1 by means of the SetBit function and outputs the result along with the original b_saved value.

```
{
BOOL b_value = 0;
BOOL b_saved = b_value;

//Set bit
  SetBit (b_value);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",b_value, b_saved);
...
}
```

See also

Availability for specific devices of system functions (Page 4612)

SetBitInTag

Description

Sets a bit in the given tag to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitInTag (Tag, Bit)

Use in user-defined functions

SetBitInTag(Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 1 (TRUE).

Bit

The number of the bit that is set to 1 (TRUE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an I/O field or assign the system function to a screen object, such as a button.

If you have configured a short event such as the activation of an alarm for the system function you can only access the actual process values by setting the tag for continuous reading.

Example

The following program code sets a bit to 1 at the specified bitposition in the bvalue tag and outputs the result along with the original bsaved value.

```
{
BYTE bvalue;
BYTE bsaved = bvalue;
BYTE bitposition = 2;

//Reset bit in bitposition
SetBitInTag (bvalue, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bvalue, bsaved);
  ...
}
```

See also

SetBit (Page 4763)

Availability for specific devices of system functions (Page 4612)

SetBitWhileKeyPressed

Description

Sets the bit of the given tag to 1 (TRUE) as long as the user keeps the configured key pressed.

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC. You should only access tags of the BOOL type with this system function to avoid problems with overlapping access to the same tag.

Note

All functions on the event "Release" are performed immediately by means of a screen change configured for a key, even if the key is kept pressed.

If the "SetBitWhileKeyPressed" system function is configured for a function key, the bit will be reset immediately following a screen change. This action is necessary since the key assignments change after the screen change.

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitWhileKeyPressed (Tag, Bit)

Use in user-defined functions

-

Parameters

Tag

The tag in which a bit is temporarily set to 1 (TRUE). Use only tags of the type BOOL, as far as allowed by the PLC.

Bit

The number of the bit that is temporarily set to 1 (TRUE).

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an IO field, or assign the function to a screen element such as a button.

If you configured a short event such as the activation of an alarm for the function you can only access the actual process values by setting the tag for continuous reading.

See also

SetBit (Page 4763)

Availability for specific devices of system functions (Page 4612)

SetBacklightColor

Description

Defines the background lighting of the button.

Note

The configuration that was set at Switch off is reestablished when restarting the HMI device.

Use in the function list

SetBacklightColor (Value)

Use in user-defined functions

-

Parameters

Value

Defines the background lighting of the button:

0 (hmiWhite) = White: No color

1 (hmiGreen) = Green: Green color

2 (hmiYellow) = Yellow: Yellow color

3 (hmiRed) = Red: Red color

See also

Availability for specific devices of system functions (Page 4612)

SetBrightness

Description

Determines the brightness of the display.

Note

The configuration that is set in the Control Panel / Start Center will be reestablished when you restart the HMI device.

For Basic Panels 2nd Generation, Mobile Panels and Comfort Panels:

The value for the system function "SetBrightness" can be set between 0% and 100%. The set value is transferred to the HMI device. The brightness settings on the HMI device can be viewed and edited in "Start Center > Settings > Display". The HMI devices support a brightness setting between 10% and 100%.

If the system function "SetBrightness" is assigned a value of 0%, the display of the HMI device is switched off by default in Runtime. If the operator touches the display, the display switches to the previous brightness setting.

If the system function "SetBrightness" is assigned a value between 1% and 10% and the operator opens the display settings in the Start Center, brightness is reset to 10%.

Use in the function list

SetBrightness (value)

Use in user-defined functions

SetBrightness (Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Value

New value for the brightness.

See also

Availability for specific devices of system functions (Page 4612)

SetScreenKeyboardMode

Description

Enables or disables the automatic display of the screen keyboard on the HMI device.

This system function is also used to prevent the display of the screen keyboard, e.g. because an external keyboard is connected to the HMI device.

Note

To enable the "SetScreenKeyboardMode" ("SetScreenKeyboardMode") system function on an HMI other than a Touch Panel device, set the "Use on-screen keyboard" check box in the "Runtime settings" dialog of the device settings.

Use in the function list

SetScreenKeyboardMode (Mode)

Use in user-defined functions

SetScreenKeyboardMode (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether the screen keyboard is hidden or shown:

0 (hmiOff) = Off: Screen keyboard is hidden

1 (hmiOn) = On: Screen keyboard is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes.

See also

Availability for specific devices of system functions (Page 4612)

SetPLCDateTime

Description

Changes the data and the time of the linked PLC

The system function "SetPLCDateTime" can only be configured for the following PLCs:

- SIMATIC S7 1200
- SIMATIC S7 1500

Use in the function list

SetPLCDateTime (connection, time)

Use in user-defined functions

SetPLCDateTime (Connection, Time)

Parameters

Connection

Connection of PLC and HMI device.

Time

Transfers the date and the time of the HMI device to the PLC. The PLC applies the date and the time of the HMI device.

SetPLCMode

Description

Switches the operating mode of the PLC to one of the following states:

- RUN
- STOP

The system function "SetPLCMode" can only be configured for the following PLCs:

- SIMATIC S7 1200
- SIMATIC S7 1500

Use in the function list

SetPLCMode (connection, mode)

Use in user-defined functions

SetPLCMode (Connection, Mode)

Parameters

Connection

Connection of PLC and HMI device.

Mode

Specifies the operating mode of the PLC:

RUN = the PLC is switched to the RUN state. The PLC program is executed.

STOP = the PLC is switched to the STOP state. The PLC program is interrupted.

SetAlarmReportMode

Description

Switches the automatic reporting of alarms on the printer on or off.

Use in the function list

SetAlarmReportMode (Mode)

Use in user-defined functions

SetAlarmReportMode (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether alarms are reported automatically on the printer:

0 (hmiDisablePrinting) = Reporting off: Alarms are not printed automatically.

1 (hmiEnablePrinting) = Reporting On: Alarms are printed automatically.

-1 (hmiToggle) = Toggle: Toggles between the two modes.

See also

Availability for specific devices of system functions (Page 4612)

SetRecipeTags

Description

Changes the status of the recipe tags from "Online" to "Offline" and vice versa.

This system function is used, for example, when recipe data record values are fine tuned when starting up a machine.

Use in the function list

SetRecipeTags (Recipe number/name, Status, Output status message, Processing status)

Use in user-defined functions

SetRecipeTags (Recipe_number/name, Status, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe in which the recipe data record is saved.

Status

Determines the status of the recipe tags:

0 (hmiOnline) = Online: Value changes of the recipe tags are transferred immediately to the PLC connected to the HMI device.

1 (hmiOffline) = Offline: Value changes to the recipe tags are only transferred to the PLC connected to the HMI device when, for example, the "SetDataRecordTagsToPLC" system function is executed.

Output status message

Determines whether a status message is output after saving:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

SetDaylightSavingTime

Description

The system function "SetDaylightSavingTime" changes the setting of the HMI device from daylight saving to standard time and vice versa.

Time settings will take place immediately following system function.

Note

The "SetDaylightSavingTime" system function does not support time zones without daylight saving time.

Note

Windows 7

The system function "SetDaylightSavingTime" is not supported for PC-based HMI devices under Windows 7.

Use in the function list

SetDaylightSavingTime(DaylightSavingTime)

Use in user-defined functions

SetDaylightSavingTime (Daylight_saving_time)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Daylight Saving Time

Determines whether Daylight Saving Time is set on the HMI device:

0 = Daylight Saving Time is not activated

1 = Daylight Saving Time is activated

See also

Availability for specific devices of system functions (Page 4612)

SetLanguage

Description

Toggles the language on the HMI device. All configured text and system events are displayed on the HMI device in the newly set language.

Use in the function list

SetLanguage (Language)

Use in user-defined functions

SetLanguage (Language)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Language

Determines which language is set on the HMI device. The following specifications are possible:

- -1 (hmiToggle) = Toggle: Changes to the next language. The sequence is determined during configuration in the "Project languages" editor.
- Number you have defined under "Languages and fonts" in the "Runtime Settings" editor. Changes to the language with the given number.
- Language you have defined under "Languages and fonts" in the "Runtime Settings" editor.
- Language abbreviation in accordance with the VBScript5 reference: This changes to the language corresponding to the specified language code, e.g. "de-DE" for German (Germany) or "en-US" for English (United States).
  An overview of the language abbreviations is available in the basic information of VBScript under the topic "Area diagram-ID (LCID) Diagram".

See also

Availability for specific devices of system functions (Page 4612)

SetAndGetBrightness

Description

Determines the brightness of the MP 377 Touch daylight readable display. The brightness value can be interpreted as absolute or relative to the current value.

Note

The configuration that is set in the Control Panel will be reestablished when you restart the HMI device.

Use in the function list

SetAndGetBrightness (Brightness, Mode, Current value)

Use in user-defined functions

SetAndGetBrightness (Brightness, Mode, Actual brightness)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Brightness

New value for the brightness.

Mode

Specifies if the new brightness value is set as absolute or relative to the current value.

Current value

Tag in which the current brightness value is stored.

See also

Availability for specific devices of system functions (Page 4612)

SetTag

Description

Assigns a new value to the given tag.

Note

This system function can be used to assign strings and numbers, depending on the type of tag.

Use in the function list

SetTag (Tag, Value)

Use in user-defined functions

SetTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag to which the given value is assigned.

Value

The value which the given tag is assigned.

Note

The "SetTag" system function is only executed after a connection has been established.

Example

The following program code uses the SetTag function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag.

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
bvalue = GetTagBit("gs_tag_bit");
  printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

The saved return value can be processed in the following code.

See also

Availability for specific devices of system functions (Page 4612)

SetConnectionMode

Description

Connects or disconnects the given connection.

Note

A connection to the PLC cannot be established until the operating mode ONLINE has been set on the HMI device. Use the "SetDeviceMode" system function for this purpose.

Use in the function list

SetConnectionMode (Mode, Connection)

Use in user-defined functions

SetConnectionMode (Mode, Connection)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether a connection to the PLC is established or disconnected:

0 (hmiOnline) = Online: Connection is established.

1 (hmiOffline) = Offline: Connection is disconnected.

Connection

The PLC to which the HMI device is connected. You specify the name of the PLC in the connection editor.

Multiple use of the system function in a user-defined function

If you use the "SetConnectionMode" system function for different connections, it may be possible that not all system functions are executed correctly. Proceed as follows to prevent this situation:

1. Create a "BOOL" type tag with the start value "0".
2. Configure the "SetConnectionMode" system function on the "Value change" event of the HMI tags. If you want to disconnect three connections, for example, you must configure the system function three times.
3. In the user-defined function, apply the "InvertBit" system function to the HMI tag.

Application example

Two typical application examples for this system function are as follows:

- Test
  As long as no PLC is connected to the HMI device, no error messages will be output during the test on the HMI device. If the HMI device is connected to a PLC, the connection to the PLC can be established by pressing a key.
- Commissioning
  Several PLCs are to be configured for a system. At first, all PLCs except one are configured "Offline". After commissioning of the first PLC, the connection to each of the other PLCs is established by pressing a key. In this way, the other PLCs are started up one after another.

See also

Availability for specific devices of system functions (Page 4612)

SetWebAccess

Description

Determines the access mode to the runtime application using the Internet.

Use in the function list

SetWebAccess (Access mode)

Use in user-defined functions

-

Parameters

Access mode

Determines the access mode to the runtime application:

-1 (hmiToggle) = Toggle: Toggles between the two modes.

0 (hmiReadOnly) = Read-only.

1 (hmiReadWrite) = Read-write.

See also

Availability for specific devices of system functions (Page 4612)

BackupRAMFileSystem

Description

Backs up the RAM file system in the memory medium of the HMI device.

After restarting the HMI device, the data is automatically reloaded in the RAM file system.

Applications such as the Internet Explorer save data (e.g. the last web sites called up) temporarily to the DRAM file system of the HMI device.

Use in the function list

BackupRAMFileSystem

Use in user-defined functions

BackupRAMFileSystem

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

SimulateSystemKey

Description

Simulates the behavior of a system key. Use this system function if a system key, such as the "ACK" key, "Input" key or the number pad is not available on the HMI device.

Use in the function list

SimulateSystemKey (System key)

Use in user-defined functions

-

Parameters

System key

System key whose behavior is to be simulated.

System key "+/-"

With the SimulateSystemKey system function, the system key "+/-" is only supported for the following HMI devices:

- KP300 Basic
- KP400 Basic
- KTP400 Basic mono PN
- KTP400 Basic color PN
- KTP400 Basic color PN Portrait
- KTP600 Basic mono PN
- KTP600 Basic color PN
- KTP600 Basic color DP

- KTP700 Basic PN / DP
- KTP900 Basic PN
- KTP1000 Basic PN
- KTP1000 Basic DP
- KTP1200 Basic PN / DP
- Comfort Panels

Use the system keys "+" and "-" separately for all other HMI devices.

See also

Availability for specific devices of system functions (Page 4612)

SimulateTag

Description

Simulates the behavior of tags and dynamic objects such as text lists, without having the HMI device connected to a PLC. You can, for example, configure the system function to the "Loaded" event of a screen.

This system function is used, for example, to demonstrate the functionality of a project.

Only tags of the data type Integer can be used for simulation. Tags of the data types Integer and Double Integer, however, can be used with OP 73, OP 77A, TP 177A.

Note

If you use the system function "SimulateTag" with a short cycle time on a Basic Panel, the HMI device may be overloaded.

Use in the function list

SimulateTag (Tag, Cycle, Maximum value, Minimum value, Value)

Use in user-defined functions

-

Parameter

Tag

The tag whose value is changed.

Cycle

The factor by which the basic cycle of 200 milliseconds is multiplied. The cycle defines when the tag value is changed by the specified value. Possible cycles between 1 and 32767.

Maximum value

The maximum value that the tag can assume during simulation. The maximum value must be greater than the minimum value but less than / equal to 32767.

Minimum value

The minimum value that the tag can assume during simulation. The minimum value must be lower than the maximum value but greater than / equal to -32768.

Value

The value by which the tag value is changed during each cycle. Possible values between -32768 and 32767.

- A positive value increases the tag value. When the maximum value is reached, the tag value is set to the minimum value after the next update cycle.
- A negative value reduces the tag value. When the minimum value is reached, the tag value is set to the maximum value after the next update cycle.

See also

Availability for specific devices of system functions (Page 4612)

SmartClientViewRefresh

Description

Updates the contents displayed in the Sm@rtClient view.

Use in the function list

SmartClientViewRefresh (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

See also

Availability for specific devices of system functions (Page 4612)

SmartClientViewReadOnlyOff

Description

Sets read-only access to "Off" in the Sm@rtClient view.

This setting allows a distant HMI device to be operated. The "SmartClientViewReadOnlyOn" system function is used to switch read-only access on again.

Use in the function list

SmartClientViewReadOnlyOff (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

See also

Availability for specific devices of system functions (Page 4612)

SmartClientViewReadOnlyOn

Description

Sets read-only access to "On" in the Sm@rtClient view.

This setting allows a distant HMI device to be monitored only. The "SmartClientViewReadOnlyOff" system function is used to switch read-only access off again.

Use in the function list

SmartClientViewReadOnlyOn (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

See also

Availability for specific devices of system functions (Page 4612)

SmartClientViewDisconnect

Description

Executes the "Disconnect" command in the Sm@rtClient view.

This system function is used when the integrated button of the screen object should not be used.

Use in the function list

SmartClientViewDisconnect (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

See also

Availability for specific devices of system functions (Page 4612)

SmartClientViewConnect

Description

Executes the "Connect" command in the Sm@rtClient view.

This system function is used when the integrated button of the screen object should not be used. The Sm@rtClient view establishes a connection with the configured HMI device.

Use in the function list

SmartClientViewConnect (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

See also

Availability for specific devices of system functions (Page 4612)

SmartClientViewLeave

Description

Exits the Sm@rtClient view and returns to operation of the HMI device.

The connection to the HMI device configured in the smart client view is retained.

Use in the function list

SmartClientViewLeave (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

See also

Availability for specific devices of system functions (Page 4612)

SaveDataRecord

Description

Saves the current values of the recipe tags as data record to the memory medium of the HMI device.

This system function is used, for example, to save a recipe data record in the recipe screen.

Use in the function list

SaveDataRecord (Recipe number/name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

SaveDataRecord (Recipe_number/name, Data_record_number/name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Recipe number/name

Number or name of the recipe to which a recipe data record is saved.

Data record number/name

Number or name of the recipe data record to be saved. A new data record will be created if no record of this name or number was found in the recipe, independent of the value at the "Overwrite" parameter.

Overwrite

Specifies whether an existing data record is overwritten:

0 (hmiOverwriteForbidden) = No: The recipe data record is not overwritten, the data record is not saved.

1 (hmiOverwriteAlways) = Yes: The recipe data record is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithConfirmation) = With confirmation: The recipe data record is overwritten only with confirmation by the user.

Output status message

Determines whether a status message is output after saving:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

StartLogging

Description

Starts the logging of data or alarms in the specified log. The function can also be applied to audit trails.

You can interrupt logging at runtime using the "StopLogging" system function.

Use in the function list

StartLogging (Log type, Log)

Use in user-defined functions

StartLogging (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail

Log

Name of the log which is started.

See also

Managing logging behavior when Runtime starts (Page 4254)

Availability for specific devices of system functions (Page 4612)

StartNextLog

Description

Stops the logging of data or alarms for the given log.

Logging is continued in the next log of the segmented circular log you configured for the specified log.

If you did not configure a segmented circular log for the specified log, the system function has no effect.

Use in the function list

StartNextLog (Log type, Log)

Use in user-defined functions

StartNextLog (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

Log

Name of the log for which the logging is stopped and continued in the next log.

See also

Availability for specific devices of system functions (Page 4612)

StartProgram

Description

Starts the specified program on the HMI device.

The runtime software continues running in the background. Alarms continue to be output and data continues to be updated.

When the given application is exited, the screen which was active during the performance of the system function is displayed on the HMI device.

This system function is used, for example, to edit recipe data records in MS Excel on the HMI device.

Note

If Windows CE is installed on the HMI device, during the configuration it must be checked whether the desired application can be started with this system function.

This system function allows all applications to be started which can be started in the "Execute" dialog of Windows CE.

The application to be started must be installed on the HMI device.

Use in the function list

StartProgram (Program name, Program parameters, Layout, Wait for program to end)

Use in user-defined functions

StartProgram (Program_name, Program_parameters, Display_mode, Wait_for_program_to_end)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Program name

Name and path of the program which is started. Upper and lower case are taken into account in this parameter.

Note

If the path contains spaces, the program can only be started correctly if the path is specified in inverted commas, e.g. "C:\Program Files\START\start.exe".

Program parameters

The parameters you transfer at the start of the program, for example a file that is opened after the start of the program.

The description of the necessary parameters is found in the documentation of the program to be started.

Layout

Determines how the program window is displayed on the HMI device:

0 (hmiShowNormal) = Normal

1 (hmiShowMinimized) = Minimized

2 (hmiShowMaximized) = Maximized

3 (ShowMinimizedAndInactive) = Minimized and inactive

Wait for program to end

Determines whether there is a change back to the project after the called up program has ended:

0 (hmiNo) = No: No change to project

1 (hmiYes) = Yes: Change to project

Note

The "Wait for program to end" parameter is only available for Runtime Advanced and Panels.

See also

Availability for specific devices of system functions (Page 4612)

StatusForceGetValues

Description

Starts or stops the updating of values in the Status/Force display. The values are read from the PLC connected to the HMI device until the update is stopped.

Note

As long as the values are updated, no entries are possible in the input fields of the Status/Force display.

Use in the function list

StatusForceGetValues (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Status/Force display to which data from the PLC is written.

See also

Availability for specific devices of system functions (Page 4612)

StatusForceSetValues

Description

Writes the values from the Status/Force display to the PLC to which the HMI device is connected.

Use in the function list

StatusForceSetValues (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Status/Force display from which data is written to the PLC.

See also

Availability for specific devices of system functions (Page 4612)

ControlSmartServer

Description

Starts or stops the Sm@rtServer.

Use in the function list

ControlSmartServer (Mode)

Use in user-defined functions

ControlSmartServer (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Specifies whether the Sm@rtServer is started or stopped.

-1 (hmiToggle) = Toggle: Toggles between the two modes

0 (hmiStop) = Stop: Sm@rtServer is stopped

1 (hmiStart) = Start: Sm@rtServer is started

See also

Availability for specific devices of system functions (Page 4612)

ControlWebServer

Description

Starts or stops the Web server.

Use in the function list

ControlWebServer (Mode)

Use in user-defined functions

ControlWebServer (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Specifies whether the Web server is started or stopped.

-1 (hmiToggle) = Toggle: Toggles between the two modes

0 (hmiStop) = Stop: The Web server is stopped.

1 (hmiStart) = Start: The Web server is started.

See also

Availability for specific devices of system functions (Page 4612)

StopLogging

Description

Stops the logging of process values or alarms in the specified log. The function can also be applied to audit trails.

The "StartLogging" system function is used to resume logging at runtime.

Note

When logging is stopped, a connection between WinCC and the log files or log database still exists. Use the "CloseAllLogs" system function to disconnect this connection.

Use in the function list

StopLogging (Log type, Log)

Use in user-defined functions

StopLogging (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail

Log

Name of the log that is stopped.

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "StopLogging" and "CloseAllLogs" system functions on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When the button "Close Archive" is pressed, the specified log is stopped and all open logs are closed. The data medium can be changed. The "Open Archive" button opens all logs and continues logging in the specified log.

See also

Availability for specific devices of system functions (Page 4612)

StopRuntime

Description

Exits the runtime software and thereby the project running on the HMI device.

Use in the function list

StopRuntime (Mode)

Use in user-defined functions

StopRuntime (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Mode

Determines whether the operating system is shut down after exiting runtime.

0 (hmiStopRuntime) = Runtime: Operating system is not shut down

1 (hmiStopRuntimeAndOperatingSystem) = Runtime and operating system: The operating system is shut down (not possible with WinCE)

Example

The following program code shuts down Runtime and the operating system.

```
{

//Stop runtime and shutdown
StopRuntime (hmiStopRuntimeAndOperationSystem);

}
```

The saved return value can be processed in the following code.

See also

Availability for specific devices of system functions (Page 4612)

ActivateSystemDiagnosticsView

Description

Activates the system diagnostics view. The system diagnostics view shows the detail view of the device concerned.

Use in the function list

ActivateSystemDiagnosticsView (Screen name, Object name)

Use in user-defined functions

ActivateSystemDiagnosticsView (Screen_name, Object_name)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Screen name

Name of the screen contained in the system diagnostics view.

Object name

Object name of the system diagnostics view.

See also

Availability for specific devices of system functions (Page 4612)

SystemDiagnosticsViewRefreshPLCBuffer

Description

Refreshes the data from the diagnostics buffer of the PLC in the system diagnostics view.

Use in the function list

SystemDiagnosticsViewRefreshPLCBuffer (Object name)

Use in user-defined functions

-

Parameters

Object name

Name of the system diagnostics view.

SystemDiagnosticsViewDetailView

Description

The detail view returns detailed information about the selected connection and errors pending.

Use in the function list

SystemDiagnosticsViewDetailView (Object name)

Use in user-defined functions

-

Parameters

Object name

Name of the system diagnostics view that is switched to the detail view.

SystemDiagnosticsViewDiagnosticsBuffer

Description

Opens a view in a system diagnostics for displaying the current data of the diagnostics buffer.

Use in the function list

SystemDiagnosticsViewDiagnosticsBuffer (Object name)

Use in user-defined functions

-

Parameters

Object name

Name of the system diagnostics view that is switched to the detail view.

SystemDiagnosticsViewDeviceView

Description

Opens a view in a system diagnostics for displaying all available connections.

Use in the function list

SystemDiagnosticsViewDeviceView (Object name)

Use in user-defined functions

-

Parameters

Object name

Name of the system diagnostics view.

SystemDiagnosticsViewBack

Description

Changes in a system diagnostics to the higher-level view.

- The function opens the device view if the diagnostics buffer view is set in the system diagnostics view.
- The function opens the diagnostics buffer view if the detail view is displayed in the system diagnostics view.

Use in the function list

SystemDiagnosticsViewBack (Object name)

Use in user-defined functions

-

Parameters

Object name

Name of the system diagnostics view.

LookupText

Description

Identifies an entry from a text list. The result depends on the value and on the selected runtime language. The result is saved to a tag of data type "String".

Use in the function list

LookupText (Output text, Value, Language, Text list)

Use in user-defined functions

LookupText (Output_text, Index, Language, Text_list)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Output text

The tag to which the result is written.

Value

The tag that defines the value of the list entry.

Language

Defines the runtime language in which the list entry is identified.

- Runtime language
  Language code as per VBScript reference, for example, "de-DE" for German (Germany) or "en-US" for English (United States of America). The selection depends on the activated runtime languages.
- Tag
  The tag that contains the language. Enter the runtime language as decimal value of the country ID, for example, 1031 for German - Standard, 1033 for English - USA. Details are available in the VBScript basics, "Locale identifier (LCID) diagram".
- Integer
  The number that corresponds to the order of runtime languages for language switching. The selection depends on the activated runtime languages, for example, "0" for the language that appears when you first start runtime. You can find more information under the topic of "Languages in Runtime".

Text list

Defines the text list. The list entry is read from the text list.

See also

Availability for specific devices of system functions (Page 4612)

ResetTagToHandWheel

Description

Separates the tag that is connected with the operating element handwheel, and reconnects the handwheel with the global tags.

Use in the function list

ResetTagToHandWheel

Use in user-defined functions

-

Parameters

--

See also

Availability for specific devices of system functions (Page 4612)

SetTagToHandWheel

Description

Connect the tag with the operating device handwheel If the handwheel is operated, the tag value changes. The connection can be reset using the "ResetTagToHandWheel" system function.

Use in the function list

SetTagToHandWheel (Value)

Use in user-defined functions

-

Parameters

Value

Tag names which are connected to the operating device handwheel.

See also

Availability for specific devices of system functions (Page 4612)

TraceUserChange

Description

Outputs a system event that shows which user is currently logged in on the HMI device.

This system function can only be used in the Scheduler.

Use in the function list

TraceUserChange

Use in user-defined functions

-

Parameters

--

See also

Availability for specific devices of system functions (Page 4612)

DecreaseFocusedValue

Description

Subtracts the specified value from the value of the tag which is connected to the screen object and currently has the focus.

This system function can only be used for function keys.

Use in the function list

DecreaseFocusedValue (Value)

Use in user-defined functions

-

Parameters

Value

The value which is subtracted from the tag value.

See also

Availability for specific devices of system functions (Page 4612)

DecreaseTag

Description

Subtracts the given value from the tag value.

X = X - a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. The auxiliary tags are assigned to the tag value with the "SetTag" system function.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

DecreaseTag (Tag, Value)

Use in user-defined functions

DecreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag from which the given value is subtracted.

Value

The value which is subtracted.

See also

Availability for specific devices of system functions (Page 4612)

ChangeConnection

Description

Disconnects the connection to the currently used PLC in use and establishes a connection to a PLC with another address. The newly connected PLC must belong to the same device class (S7-1200, S7-300, ...etc). With S7-1200, the use of the function is also only permitted for absolute addressing.

Note

When changing to another address, ensure that the address is not already being used by another HMI device.

The follows address types are supported:

- IP address
- MPI address

The follows PLC types are supported:

- SIMATIC S7 300/400
- SIMATIC S7-NC
- SIMOTION

Use in the function list

ChangeConnection (Connection, Address, Slot, Rack)

Use in user-defined functions

ChangeConnection (Connection, Address, Slot, Rack)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Connection

Name of the connection that is disconnected. The name is set during configuration, for example, in the "Connections" editor.

Address

MPI/PROFIBUS or IP address of the PLC to which the connection will be established.

Note

Set the address by means of a tag. The objects list displays the tags of all data types. Select only tags of the following data types:

- Ethernet connection: "String" data type
- MPI connection: "Int" data type

Slot

Slot of the PLC to which the connection will be established.

Rack

Rack of the PLC to which the connection will be established.

Application example

You want to operate one HMI device on several machines. Configure the necessary PLCs in the project, to which you want to change by pressing a key. When changing the PLC, the connection to the PLC in use is disconnected. Then the connection to the new PLC with other address parameters is reestablished. To access the values of the new PLC, configure the same tags for the PLC used.

The PLC which you have indicated when creating the project will be used as default.

1. Enter the name and address of the PLC in the "Connections" editor.
2. Configure a button in the process screen.
3. Configure the "ChangeConnection" system function on the "Press" event.
4. Enter the name of the connection and address of the PLC as parameters.

See also

Availability for specific devices of system functions (Page 4612)

ShowLogonDialog

Description

Opens a dialog on the HMI device with which the user can log on to the HMI device.

Use in the function list

ShowLogonDialog

Use in user-defined functions

-

Parameters

--

See also

Availability for specific devices of system functions (Page 4612)

ShowOperatorNotes

Application

Displays the tooltip configured for the selected object.

If the system function is configured on a function key, the tooltip for the screen object that currently has the focus is displayed. If a tooltip is configured for the screen itself, you can switch to this text by pressing <Enter> or by double-clicking on the help window.

If the system function is configured on a button, only the tooltip for the current screen is displayed. If a tooltip is configured on the button itself, initially only the tooltip for the button is displayed. You can press <Enter> or double-click on the help window to switch to the tooltip for the current screen.

Note

No other screen object can be used while the help window is open. To use the screen object, close the help window.

Closing the help window

You can close the help window in the following ways:

Using the keys:

- By pressing the <HELP> key again
- By pressing the <ESC> key

Using the touch screen:

- By pressing the button

Use in the function list

ShowOperatorNotes (Layout)

Use in user-defined functions

ShowOperatorNotes (Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines whether the configured tooltip is hidden or shown:

0 (hmiOff) = Off: Configured tooltip is hidden

1 (hmiOn) = On: Configured tooltip is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

See also

Availability for specific devices of system functions (Page 4612)

ShowAlarmWindow

Description

Hides or shows the alarm window on the HMI device.

Use in the function list

ShowAlarmWindow (Object name, Layout)

Use in user-defined functions

ShowAlarmWindow (Object_name, Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Object name

Name of the alarm view which is hidden or shown.

Layout

Determines whether the alarm window is hidden or shown:

0 (hmiOff) = Off: Alarm view is hidden

1 (hmiOn) = On: Alarm view is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

See also

Availability for specific devices of system functions (Page 4612)

ShowPopupScreen

Description

Opens the pop-up-screen, for example, when a button is pressed.

You can enter a constant value or assign a tag as coordinates.
If the configured pop-up screen is not visible or only partially visible, the coordinates are set to 0.0.

Use in the function list

ShowPopupScreen (Screen object)

Use in user-defined functions

ShowPopupScreen (Name_of_the_screen, Coordinate_X, Coordinate_Y, Layout)

Parameters

Screen name

Specifies the name of the screen that appears in Runtime when a button is pressed.

X coordinate

Position of the screen in the current screen on the X axis

Y coordinate

Position of the screen in the current screen on the Y axis

Layout

Specifies the mode for the slide-in screen:

Switching

Off

On

See also

Availability for specific devices of system functions (Page 4612)

System functions for Runtime Advanced (Page 4651)

ShowSlideInScreen

Description

Calls the slide-in screen, for example, when operating a button.

Use in the function list

ShowSlideinScreen (screen name, mode)

Use in user-defined functions

ShowSlideInScreen (SlideInScreen_name, Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Specifies the slide-in screen that appears in Runtime when a button is pressed:

Slide-in screen top

Slide-in screen bottom

Slide-in screen left

Slide-in screen right

Mode

Specifies the mode for the slide-in screen:

Switching

Off

On

See also

Availability for specific devices of system functions (Page 4612)

System functions for Comfort Panels (Page 4636)

System functions for Mobile Panels (Page 4643)

System functions for Runtime Advanced (Page 4651)

ShowSoftwareVersion

Description

Hides or shows the version number of the runtime software.

Use this system function if during servicing, for example, you required the version of the runtime software used.

Use in the function list

ShowSoftwareVersion (Layout)

Use in user-defined functions

ShowSoftwareVersion (Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Layout

Determines whether the version number is shown:

0 (hmiOff) = Off: Version number is not shown

1 (hmiOn) = On: Version number is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

See also

Availability for specific devices of system functions (Page 4612)

ShowSystemDiagnosticsWindow

Description

Hides or shows the system diagnostic window on the HMI device. The system diagnostic view is only available in the global screen for Comfort Panels and for WinCC Runtime Advanced.

Use in the function list

ShowSystemDiagnosticsWindow (Screen object)

Use in user-defined functions

ShowSystemDiagnosticsWindow (Target_Object_name)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen object

Name of the system diagnostic window which is hidden or shown.

See also

Availability for specific devices of system functions (Page 4612)

ShowSystemAlarm

Description

Displays the value of the parameter passed as a system event to the HMI device.

Use in the function list

ShowSystemAlarm (Text/value)

Use in user-defined functions

ShowSystemAlarm (Text/value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Text/Value

The text or the value, which was output as a system alarm.

See also

Availability for specific devices of system functions (Page 4612)

Functions for WinAC MP

WinACMPUpdateControllerForStartAtBoot

Description

Reads whether WinAC MP is automatically started after startup of the HMI device. The LED display of the "StartAtBoot" button is updated.

Use in the function list

WinACMPUpdateControllerForStartAtBoot(Start at boot, Action)

Use in user-defined functions

-

Parameters

Start at boot

The tag that contains the value.

Action

Determines whether the display is updated:

0 (SwitchOff) = Off: LED display is gray. WinAC MP is not started on startup of the HMI device.

1 (SwitchOn) = On: LED display has the color "cyan". WinAC MP is started automatically on startup of the HMI device.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateKeySwitchSetting

Description

Updates the status display of the position of the mode selector switch.

Use in the function list

WinACMPUpdateKeySwitchSetting (Key switch, Action)

Use in user-defined functions

-

Parameters

Key switch

The tag that contains the value of the key switch:

0 (MRES) = MRES: PLC memory reset

1 (STOP) = STOP: STOP mode

3 (RUN) = RUN: RUN mode

Action

Determines whether the status display will be updated:

0 (SwitchOff) = Off: Status display is not updated.

1 (SwitchOn) = On: Status display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateBUSF1LED

Description

Updates the status display of the BUSF1 LED tag.

Use in the function list

WinACMPUpdateBUSF1LED(BUSF1, Action)

Use in user-defined functions

-

Parameters

BUSF1

The tag that contains the value of the status display.

0 = Off

1 = Slow flashing

2 = Quick flashing

3 = On

Action

Defines whether the status display of the BUSF1 LED tag is updated:

0 (SwitchOff) = Off: Status display is not updated.

1 (SwitchOn) = On: Status display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateBUSF2LED

Description

Updates the status display of the BUSF2 LED tag.

Use in the function list

WinACMPUpdateBUSF2LED(BUSF2, Action)

Use in user-defined functions

-

Parameters

BUSF2

The tag that contains the value of the status display.

0 = Off

1 = Slow flashing

2 = Quick flashing

3 = On

Action

Defines whether the status display of the BUSF2 LED tag is updated:

0 (SwitchOff) = Off: Status display is not updated.

1 (SwitchOn) = On: Status display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".

- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateAverageExecTime

Description

Updates the display of the average OB 1 execution time. The OB 1 execution time will be displayed in milliseconds.

Use in the function list

WinACMPUpdateAverageExecTime(Cycle time, Action)

Use in user-defined functions

-

Parameters

Cycle time

Tag that contains the value of the average OB 1 execution time.

Action

Determines whether the display is updated:

0 (SwitchOff) = Off: Display is not updated.

1 (SwitchOn) = On: Display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".

- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateAverageCycleTime

Description

Updates the display of the average cycle time. The cycle time is displayed in milliseconds.

Use in the function list

WinACMPUpdateAverageCycleTime(Cycle time, Action)

Use in user-defined functions

-

Parameters

Cycle time

The tag that contains the value of the average cycle time.

Action

Determines whether the display is updated:

0 (SwitchOff) = Off: Display is not updated.

1 (SwitchOn) = On: Display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".

- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateEXTFLED

Description

Updates the status display of the EXTF LED tag.

Use in the function list

WinACMPUpdateEXTFLED(EXTF, Action)

Use in user-defined functions

-

Parameters

EXTF

The tag that contains the value of the status display.

0 = Off

1 = Slow flashing

2 = Quick flashing

3 = On

Action

Defines whether the status display of the EXTF LED tag is updated:

0 (SwitchOff) = Off: Status display is not updated.

1 (SwitchOn) = On: Status display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateHMIEnableTime

Description

Updates the value of the HMI Enable Time from WinAC MP (in ms).

Use in the function list

WinACMPUpdateHMIEnableTime(Enable Time, Action)

Use in user-defined functions

-

Parameters

HMI Enable time

The tag that contains the value of the HMI Enable Time.

Action

Determines whether the value is updated:

0 (SwitchOff) = Off: Value is not updated.

1 (SwitchOn) = On: Value is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateINTFLED

Description

Updates the status display of the INTF LED tag.

Use in the function list

WinACMPUpdateINTFLED(INTF, Action)

Use in user-defined functions

-

Parameters

INTF

The tag that contains the value of the status display.

0 = Off

1 = Slow flashing

2 = Quick flashing

3 = On

Action

Defines whether the status display of the INTF LED tag is updated:

0 (SwitchOff) = Off: Status display is not updated.

1 (SwitchOn) = On: Status display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateLastCycleTime

Description

Updates the display of the last cycle time. The cycle time is displayed in milliseconds.

Use in the function list

WinACMPUpdateLastCycleTime (Cycle time , Action)

Use in user-defined functions

-

Parameters

Cycle time

The tag that contains the value of the last cycle time.

Action

Determines whether the display is updated:

0 (SwitchOff) = Off: Display is not updated.

1 (SwitchOn) = On: Display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateMaximumCycleTime

Description

Updates the display of the longest cycle time. The cycle time is displayed in milliseconds.

Use in the function list

WinACMPUpdateMaximumCycleTime(Cycle time, Action)

Use in user-defined functions

-

Parameters

Cycle time

The tag that contains the value of the longest cycle time.

Action

Determines whether the display is updated:

0 (SwitchOff) = Off: Display is not updated.

1 (SwitchOn) = On: Display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

WinACMPUpdateMinimumCycleTime

Description

Updates the display of the shortest cycle time. The cycle time is displayed in milliseconds.

Use in the function list

WinACMPUpdateMinimumCycleTime (Cycle Time, Action)

Use in user-defined functions

-

Parameters

Cycle time

The tag that contains the value of the shortest cycle time.

Action

Determines whether the display is updated:

0 (SwitchOff) = Off: The display is not updated.

1 (SwitchOn) = On: The display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".

- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdatePowerLED

Description

Updates the status display ON/OFF of the Power LED tag.

Use in the function list

WinACMPUpdatePowerLED (Power, Action)

Use in user-defined functions

-

Parameters

Power

The tag that contains the value of the status display.

0 = Off

1 = Slow flashing

2 = Quick flashing

3 = On

Action

Determines whether the status display ON/OFF is updated:

0 (SwitchOff) = Off: Status display is not updated.

1 (SwitchOn) = On: Status display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateSleepTime

Description

Updates the value of the HMI Enable Time from WinAC MP (in ms).

Use in the function list

WinACMPUpdateSleepTime(SleepTime, Action)

Use in user-defined functions

-

Parameters

SleepTime

The tag that contains the value.

Action

Determines whether the value is updated:

0 (SwitchOff) = Off: Value is not updated.

1 (SwitchOn) = On: Value is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateRUNLED

Description

Updates the status display of the RUN LED tag.

Use in the function list

WinACMPUpdateRUNLED(RUN, Action)

Use in user-defined functions

-

Parameters

RUN

The tag that contains the value of the status display.

0 = Off

1 = Slow flashing

2 = Quick flashing

3 = On

Action

Defines whether the status display of the RUN LED tag is updated:

0 (SwitchOff) = Off: Status display is not updated.

1 (SwitchOn) = On: Status display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPUpdateSTOPLED

Description

Updates the status display of the STOP LED tag.

Use in the function list

WinACMPUpdateSTOPLED (STOP, Action)

Use in user-defined functions

-

Parameters

STOP

The tag that contains the value of the status display.

0 = Off

1 = Slow flashing

2 = Quick flashing

3 = On

Action

Defines whether the status display of the STOP LED tag is updated:

0 (SwitchOff) = Off: Status display is not updated.

1 (SwitchOn) = On: Status display is updated.

Notes on configuring

You configure this system function on the "Loaded" event or on the "Cleared" event of a screen.

Note the following during configuration:

- System function on the "Loaded" event
  Set the "Update" parameter to "On".
- System function on the "Cleared" event
  Set the "Update" parameter to "Off".
- If you call the system function repeatedly in a screen, always use the same tag. Otherwise, only the last activated tag will be updated.
- If you want to configure all system functions with the prefix "WinACMPUpdate" in a screen, each system function is only permitted once per screen.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPArchive

Description

Saves the current STEP 7 user program, the current system configuration and the current values of the DBs in a log file.

Use in the function list

WinACMPArchive (File name)

Use in user-defined functions

-

Parameters

File name

Name of the log file in which the data is saved. Enter the file location and the file extension (*.wld), for example, "\flash\AddOn\WinACMP\Default.wld ".

See also

Availability for specific devices of system functions (Page 4612)

WinACMPGetStartMode

Description

Reads in the 'desired' operating state after startup of the HMI device of WinAC MP.

Use in the function list

WinACMPGetStartMode (Operating state, Update)

Use in user-defined functions

-

Parameters

Start mode

The tag that defines the value of the operating mode.

Action

Determines whether the operating mode is read out:

0 (SwitchOff) = Off: Operating mode is not read out.

1 (SwitchOn) = On: Operating mode is read out.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPGetVersion

Description

Reads out the value of the version number of WinAC MP.

Use in the function list

WinACMPGetVersion (Version, Action)

Use in user-defined functions

WinACMPGetVersion (Version, Action)

Can be used if the configured device supports user-defined functions. For more information, refer to Auto-Hotspot.

Parameters

Version

The tag that contains the value.

Action

Determines whether the version number is read out:

0 (SwitchOff) = Off: Version number is not read out.

1 (SwitchOn) = On: Version number is read out.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPClearCycleTimeBuffer

Description

Clears the cycle time data of the histogram.

Use in the function list

WinACMPClearCycleTimeBuffer

Use in user-defined functions

-

Parameters

--

See also

Availability for specific devices of system functions (Page 4612)

WinACMPSetStartAtBoot

Description

Determines whether or not WinAC MP is started automatically after startup of the HMI device.

Use in the function list

WinACMPSetStart at boot(Start at boot)

Use in user-defined functions

WinACMPSetStartAtBoot (Start at boot)

Can be used if the configured device supports user-defined functions. For more information, refer to Auto-Hotspot.

Parameters

StartAtBoot

Defines whether WinAC MP is started automatically.

0 (StartAtBootOff) = Off: WinAC MP is not started on startup of the HMI device.

1 (StartAtBootOn) = On: WinAC MP is started automatically on startup of the HMI device.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPSetKeySwitch

Description

Sets the mode selector switch to RUN or STOP and is also used for memory reset.

Use in the function list

WinACMPSetKeySwitch(Key switch)

Use in user-defined functions

-

Parameters

Key switch

Defines the position of the key switch:

0 (MRES) = MRES: PLC memory reset

1 (STOP) = STOP: STOP mode

3 (RUN) = RUN: RUN mode

See also

Availability for specific devices of system functions (Page 4612)

WinACMPSetHMIEnableTime

Description

Sets the value of the HMI Enable Time.

Use in the function list

WinACMPSetHMIEnableTime(Execution Time)

Use in user-defined functions

-

Parameters

HMI Enable time

The tag that contains the value.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPSetResetMethod

Description

Sets the restart method, either cold restart (CRST) or warm restart (WRST).

Use in the function list

WinACMPSetRestartMethod (Restart behavior)

Use in user-defined functions

-

Parameters

Restart characteristics

Defines the restart action:

0 (WarmRestart) = Warm restart: A warm restart is performed.

1 (ColdRestart) = Cold restart: A cold restart is performed.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPSetSleepTime

Description

Sets the value of the HMI Execution Time of WinAC MP.

Use in the function list

WinACMPSetSleeptime (SleepTime)

Use in user-defined functions

-

Parameters

Sleep time

The tag that contains the value.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPSetStartMode

Description

Sets the operating mode of WinAC MP after startup of the HMI device.

Use in the function list

WinACSetStartMode (Autostart)

Use in user-defined functions

WinACSetStartMode (Autostart)

Can be used if the configured device supports user-defined functions. For more information, refer to Auto-Hotspot.

Parameters

Action

Defines whether the Autostart function of WinAC MP is activated.

0 (AutoStartOff) = Off: After startup, WinAC MP remains in the STOP operating mode.

1 (AutoStartOn) = On: After startup, WinAC MP changes to the operating mode it was in before it was closed.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPStartHistogram

Description

Starts the cyclic sending of histogram values and updates the display. The histogram shows the percentual distribution of the measured cycle times.

Use in the function list

WinACMPStartHistogram (Update, Percentage, Cycle time, Y axis, ID)

Use in user-defined functions

-

Parameters

Action

Defines whether WinAC returns histogram values:

0 (SwitchOff) = Off: The cyclic sending of the histogram values is not started. The display is not updated.

1 (SwitchOn) = On: The cyclic sending of the histogram values is started. The display is updated.

Percent

Percentage of the cycles which are performed during the recorded cycle times.

Cycle time

Histogram of cycle times.

Y-axis bounds

Maximum value of Y axis

ID

Histogram ID

See also

Availability for specific devices of system functions (Page 4612)

WinACMPControl

Description

Starts or stops WinAC MP.

Use in the function list

WinACMPControl (WinAC)

Use in user-defined functions

-

Parameters

WinAC

Determines whether WinAC MP is started or stopped.

0 (ShutdownWinACMP) = Stop: WinAC MP is stopped.

1 (StartWinACMP) = Start: WinAC MP is started.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPStopHistogram

Description

Stops the cyclic sending of histogram data and deletes the history data.

Use in the function list

WinACMPStopHistogram (ID)

Use in user-defined functions

-

Parameters

ID

Histogram ID

See also

Availability for specific devices of system functions (Page 4612)

WinACMPRestore

Description

Loads the STEP 7 user program, the system configuration and the DBs from a log file.

Use in the function list

WinACMPRestore (File name)

Use in user-defined functions

-

Parameters

File name

Name of the log file in which the data is saved. Enter the file location and the log file, for example, ""\flash\AddOn\WinACMP\Default.wld ".

See also

Availability for specific devices of system functions (Page 4612)

12.8.7.2 Events

Overview

Editors

Introduction

The following table shows which events occur in which editor.

Technical data subject to change.

Icon	Editor
	Screens
	HMI alarms
	HMI tags
	Scheduler

1	Cleared (Page 4847)	X	--	--	--
2	Activate (Page 4847)	--	--	--	--
3	Change (Page 4847)	--	--	--	--
4	Loaded (Page 4848)	X	--	--	--
5	Execute (Page 4848)	--	--	--	X
6	Selection changed (Page 4848)	--	--	--	--
7	On exceeding (Page 4848)	--	--	X	--
8	On falling below (Page 4849)	--	--	X	--
9	When dialog is opened (Page 4849)	--	--	--	X
10	When dialog is closed (Page 4849)	--	--	--	X
11	User change (Page 4849)	--	--	--	X
12	Screen change (Page 4850)	--	--	--	X
13	Deactivate (Page 4850)	--	--	--	X
14	Double-click (Page 4850)	--	--	--	--
15	Press (Page 4851)	--	--	--	--
16	On Finish Input (Page 4851)	--	--	--	--
17	Press ESC twice (Page 4851)	--	--	--	--
18	Outgoing (Page 4852)	--	X	--	--
19	Incoming (Page 4852)	--	X	--	--
20	Click (Page 4852)	--	--	--	--
21	Click when flashing (Page 4853)	--	--	--	--
22	Loop-in alarm (Page 4853)	--	X	--	--
23	Release (Page 4853)	--	--	--	--
24	Alarm buffer overflow (Page 4854)	--	--	--	X

25	Acknowledge (Page 4854)	--	X	--	--
26	Reach margin (Page 4854)	--	--	--	--
27	Runtime Stop (Page 4855)	--	--	--	X
28	Press key (Page 4855)	--	--	--	--
29	Release key (Page 4855)	--	--	--	--
30	Overflow (Page 4856)	--	--	--	--
31	Switch OFF (Page 4856)	--	--	--	--
32	Switch ON (Page 4856)	--	--	--	--
33	Low free storage space (Page 4857)	--	--	--	--
34	Free space critically low (Page 4857)	--	--	--	--
35	Value change (Page 4857)	--	--	X	--
36	Time expired (Page 4857)	--	--	--	--

Basic objects

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	Line
	Ellipse
	Circle
	Rectangle
A	Text field
	Graphic view

1	Cleared (Page 4847)	--	--	--	--	--	--
2	Activate (Page 4847)	--	--	--	--	--	--
3	Change (Page 4847)	--	--	--	--	--	--
4	Loaded (Page 4848)	--	--	--	--	--	--
5	Execute (Page 4848)	--	--	--	--	--	--
6	Selection changed (Page 4848)	--	--	--	--	--	--
7	On exceeding (Page 4848)	--	--	--	--	--	--
8	On falling below (Page 4849)	--	--	--	--	--	--

9	When dialog is opened (Page 4849)	--	--	--	--	--	--
10	When dialog is closed (Page 4849)	--	--	--	--	--	--
11	User change (Page 4849)	--	--	--	--	--	--
12	Screen change (Page 4850)	--	--	--	--	--	--
13	Deactivate (Page 4850)	--	--	--	--	--	--
14	Double-click (Page 4850)	--	--	--	--	--	--
15	Press (Page 4851)	--	--	--	--	--	--
16	On Finish Input (Page 4851)	--	--	--	--	--	--
17	Press ESC twice (Page 4851)	--	--	--	--	--	--
18	Outgoing (Page 4852)	--	--	--	--	--	--
19	Incoming (Page 4852)	--	--	--	--	--	--
20	Click (Page 4852)	--	--	--	--	--	--
21	Click when flashing (Page 4853)	--	--	--	--	--	--
22	Loop-in alarm (Page 4853)	--	--	--	--	--	--
23	Release (Page 4853)	--	--	--	--	--	--
24	Alarm buffer overflow (Page 4854)	--	--	--	--	--	--
25	Acknowledge (Page 4854)	--	--	--	--	--	--
26	Reach margin (Page 4854)	--	--	--	--	--	--
27	Runtime Stop (Page 4855)	--	--	--	--	--	--
28	Press key (Page 4855)	--	--	--	--	--	--
29	Release key (Page 4855)	--	--	--	--	--	--
30	Overflow (Page 4856)	--	--	--	--	--	--
31	Switch OFF (Page 4856)	--	--	--	--	--	--
32	Switch ON (Page 4856)	--	--	--	--	--	--
33	Low free storage space (Page 4857)	--	--	--	--	--	--
34	Free space critically low (Page 4857)	--	--	--	--	--	--
35	Value change (Page 4857)	--	--	--	--	--	--
36	Time expired (Page 4857)	--	--	--	--	--	--

Elements

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	IO field
	Button
	Symbolic IO field
	Graphic IO field
	Date/time field

	Object
	Bar
	Switch

1	Cleared (Page 4847)	--	--	--	--	--	--	--
2	Activate (Page 4847)	X	X	X	X	--	--	X
3	Change (Page 4847)	--	X	X	--	--	--	X
4	Loaded (Page 4848)	--	--	--	--	--	--	--
5	Execute (Page 4848)	--	--	--	--	--	--	--
6	Selection changed (Page 4848)	--	--	--	--	--	--	--
7	On exceeding (Page 4848)	--	--	--	--	--	--	--
8	On falling below (Page 4849)	--	--	--	--	--	--	--
9	When dialog is opened (Page 4849)	--	--	--	--	--	--	--
10	When dialog is closed (Page 4849)	--	--	--	--	--	--	--
11	User change (Page 4849)	--	--	--	--	--	--	--
12	Screen change (Page 4850)	--	--	--	--	--	--	--
13	Deactivate (Page 4850)	X	X	X	X	--	--	X
14	Double-click (Page 4850)	--	--	--	--	--	--	--
15	Press (Page 4851)	--	X	--	--	--	--	--
16	On Finish Input (Page 4851)	--	--	--	--	--	--	--
17	Press ESC twice (Page 4851)	--	--	--	--	--	--	--
18	Outgoing (Page 4852)	--	--	--	--	--	--	--
19	Incoming (Page 4852)	--	--	--	--	--	--	--
20	Click (Page 4852)	--	X	--	--	--	--	--
21	Click when flashing (Page 4853)	--	--	--	--	--	--	--
22	Loop-in alarm (Page 4853)	--	--	--	--	--	--	--
23	Release (Page 4853)	--	X	--	--	--	--	--
24	Alarm buffer overflow (Page 4854)	--	--	--	--	--	--	--
25	Acknowledge (Page 4854)	--	--	--	--	--	--	--
26	Reach margin (Page 4854)	--	--	--	--	--	--	--
27	Runtime Stop (Page 4855)	--	--	--	--	--	--	--
28	Press key (Page 4855)	--	--	--	--	--	--	--
29	Release key (Page 4855)	--	--	--	--	--	--	--
30	Overflow (Page 4856)	--	--	--	--	--	--	--
31	Switch OFF (Page 4856)	--	--	--	--	--	--	X
32	Switch ON (Page 4856)	--	--	--	--	--	--	X
33	Low free storage space (Page 4857)	--	--	--	--	--	--	--
34	Free space critically low (Page 4857)	--	--	--	--	--	--	--
35	Value change (Page 4857)	--	--	--	--	--	--	--
36	Time expired (Page 4857)	--	--	--	--	--	--	--

Controls

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

	Object
	Alarm view/alarm window
	Alarm indicator
	Trend view
	User view
	Recipe view
	Auxiliary indicator

1	Cleared (Page 4847)	--	--	--	--	--	--
2	Activate (Page 4847)	X	--	X	X	--	--
3	Change (Page 4847)	--	--	--	--	--	--
4	Loaded (Page 4848)	--	--	--	--	--	--
5	Execute (Page 4848)	--	--	--	--	--	--
6	Selection changed (Page 4848)	--	--	--	--	--	--
7	On exceeding (Page 4848)	--	--	--	--	--	--
8	On falling below (Page 4849)	--	--	--	--	--	--
9	When dialog is opened (Page 4849)	--	--	--	--	--	--
10	When dialog is closed (Page 4849)	--	--	--	--	--	--
11	User change (Page 4849)	--	--	--	--	--	--
12	Screen change (Page 4850)	--	--	--	--	--	--
13	Deactivate (Page 4850)	X	--	X	X	--	--
14	Double-click (Page 4850)	--	--	--	--	--	--
15	Press (Page 4851)	--	--	--	--	--	--
16	On Finish Input (Page 4851)	--	--	--	--	--	--
17	Press ESC twice (Page 4851)	--	--	--	--	--	--
18	Outgoing (Page 4852)	--	--	--	--	--	--
19	Incoming (Page 4852)						
20	Click (Page 4852)	--	X	--	--	--	--
21	Click when flashing (Page 4853)	--	X	--	--	--	--
22	Loop-in alarm (Page 4853)	--	--	--	--	--	--
23	Release (Page 4853)	--	--	--	--	--	--
24	Alarm buffer overflow (Page 4854)	--	--	--	--	--	--
25	Acknowledge (Page 4854)	--	--	--	--	--	--
26	Reach margin (Page 4854)	--	--	--	--	--	--
27	Runtime Stop (Page 4855)	--	--	--	--	--	--

28	Press key (Page 4855)	--	--	--	--	--	--
29	Release key (Page 4855)	--	--	--	--	--	--
30	Overflow (Page 4856)	--	--	--	--	--	--
31	Switch OFF (Page 4856)	--	--	--	--	--	--
32	Switch ON (Page 4856)	--	--	--	--	--	--
33	Low free storage space (Page 4857)	--	--	--	--	--	--
34	Free space critically low (Page 4857)	--	--	--	--	--	--
35	Value change (Page 4857)	--	--	--	--	--	--
36	Time expired (Page 4857)	--	--	--	--	--	--

Overview

Editors

Introduction

The following table shows which events occur in which editor.

Technical data subject to change.

Icon	Editor
	Screens
	HMI alarms
	HMI tags
	Logs
	Scheduler
	Audit Trail

1	Cleared (Page 4847)	X	--	--	--	--	--
2	Activate (Page 4847)	--	--	--	--	--	--
3	Change (Page 4847)	--	--	--	--	--	--
4	Loaded (Page 4848)	X	--	--	--	--	--
5	Execute (Page 4848)	--	--	--	--	X	--
6	Selection changed (Page 4848)	--	--	--	--	--	--
7	On exceeding (Page 4848)	--	--	X	--	--	--
8	On falling below (Page 4849)	--	--	X	--	--	--
9	When dialog is opened (Page 4849)	--	--	--	--	X	--
10	When dialog is closed (Page 4849)	--	--	--	--	X	--
11	User change (Page 4849)	--	--	--	--	X	--
12	Screen change (Page 4850)	--	--	--	--	X	--
13	Deactivate (Page 4850)	--	--	--	--	--	--

14	Double-click (Page 4850)	--	--	--	--	--	--
15	Press (Page 4851)	--	--	--	--	--	--
16	On Finish Input (Page 4851)	--	--	--	--	--	--
17	Press ESC twice (Page 4851)	--	--	--	--	--	--
18	Incoming (Page 4852)	--	X	--	--	--	--
19	Outgoing (Page 4852)	--	X	--	--	--	--
20	Click (Page 4852)	--	--	--	--	--	--
21	Click when flashing (Page 4853)	--	--	--	--	--	--
22	Loop-in alarm (Page 4853)	--	X	--	--	--	--
23	Release (Page 4853)	--	--	--	--	--	--
24	Alarm buffer overflow (Page 4854)	--	--	--	--	X	--
25	Acknowledge (Page 4854)	--	X	--	--	--	--
26	Reach margin (Page 4854)	--	--	--	--	--	--
27	Runtime Stop (Page 4855)	--	--	--	--	X	--
28	Press key (Page 4855)	--	--	--	--	--	--
29	Release key (Page 4855)	--	--	--	--	--	--
30	Overflow (Page 4856)	--	--	--	X	--	--
31	Switching (Page 4856)	--	--	--	--	--	--
32	Switch OFF (Page 4856)	--	--	--	--	--	--
33	Switch ON (Page 4856)	--	--	--	--	--	--
34	Low free storage space (Page 4857)	--	--	--	X	--	X
35	Free space critically low (Page 4857)	--	--	--	X	--	X
36	Value change (Page 4857)	--	--	X	--	--	--
37	Time expired (Page 4857)	--	--	--	--	--	--

Basic objects

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	Line
	Polyline
	Polygon
	Ellipse
	Circle
	Rectangle
A	Text field
	Graphic view

1	Cleared (Page 4847)	--	--	--	--	--	--	--	--
2	Activate (Page 4847)	--	--	--	--	--	--	--	--
3	Change (Page 4847)	--	--	--	--	--	--	--	--
4	Loaded (Page 4848)	--	--	--	--	--	--	--	--
5	Execute (Page 4848)	--	--	--	--	--	--	--	--
6	Selection changed (Page 4848)	--	--	--	--	--	--	--	--
7	On exceeding (Page 4848)	--	--	--	--	--	--	--	--
8	On falling below (Page 4849)	--	--	--	--	--	--	--	--
9	When dialog is opened (Page 4849)	--	--	--	--	--	--	--	--
10	When dialog is closed (Page 4849)	--	--	--	--	--	--	--	--
11	User change (Page 4849)	--	--	--	--	--	--	--	--
12	Screen change (Page 4850)	--	--	--	--	--	--	--	--
13	Deactivate (Page 4850)	--	--	--	--	--	--	--	--
14	Double-click (Page 4850)	--	--	--	--	--	--	--	--
15	Press (Page 4851)	--	--	--	--	--	--	--	--
16	On Finish Input (Page 4851)	--	--	--	--	--	--	--	--
17	Press ESC twice (Page 4851)	--	--	--	--	--	--	--	--
18	Outgoing (Page 4852)	--	--	--	--	--	--	--	--
19	Incoming (Page 4852)	--	--	--	--	--	--	--	--
20	Click (Page 4852)	--	--	--	--	--	--	--	--
21	Click when flashing (Page 4853)	--	--	--	--	--	--	--	--
22	Loop-in alarm (Page 4853)	--	--	--	--	--	--	--	--
23	Release (Page 4853)	--	--	--	--	--	--	--	--
24	Alarm buffer overflow (Page 4854)	--	--	--	--	--	--	--	--
25	Acknowledge (Page 4854)	--	--	--	--	--	--	--	--
26	Reach margin (Page 4854)	--	--	--	--	--	--	--	--
27	Runtime Stop (Page 4855)	--	--	--	--	--	--	--	--
28	Press key (Page 4855)	--	--	--	--	--	--	--	--
29	Release key (Page 4855)	--	--	--	--	--	--	--	--
30	Overflow (Page 4856)	--	--	--	--	--	--	--	--
31	Switching (Page 4856)	--	--	--	--	--	--	--	--
32	Switch OFF (Page 4856)	--	--	--	--	--	--	--	--
33	Switch ON (Page 4856)	--	--	--	--	--	--	--	--
34	Low free storage space (Page 4857)	--	--	--	--	--	--	--	--
35	Free space critically low (Page 4857)	--	--	--	--	--	--	--	--
36	Value change (Page 4857)	--	--	--	--	--	--	--	--
37	Time expired (Page 4857)	--	--	--	--	--	--	--	--

Elements

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	IO field
	Button
	Symbolic IO field
	Graphic IO field
	Date/time field
	Bar
	Switch
	Symbol library
	Slider
	Gauge
	Clock

1	Cleared (Page 4847)	--	--	--	--	--	--	--	--	--	--	--
2	Activate (Page 4847)	X	X	X	X	--	--	X	X	X	--	--
3	Change (Page 4847)	--	X	X	--	--	--	X	--	X	X	--
4	Loaded (Page 4848)	--	--	--	--	--	--	--	--	--	--	--
5	Execute (Page 4848)	--	--	--	--	--	--	--	--	--	--	--
6	Selection changed (Page 4848)	--	--	--	--	--	--	--	--	--	--	--
7	On exceeding (Page 4848)	--	--	--	--	--	--	--	--	--	--	--
8	On falling below (Page 4849)	--	--	--	--	--	--	--	--	--	--	--
9	When dialog is opened (Page 4849)	--	--	--	--	--	--	--	--	--	--	--
10	When dialog is closed (Page 4849)	--	--	--	--	--	--	--	--	--	--	--
11	User change (Page 4849)	--	--	--	--	--	--	--	--	--	--	--
12	Screen change (Page 4850)	--	--	--	--	--	--	--	--	--	--	--
13	Deactivate (Page 4850)	X	X	X	X	--	--	X	X	X	--	--
14	Double-click (Page 4850)	--	--	--	--	--	--	--	X	--	--	--
15	Press (Page 4851)	--	X	--	--	--	--	--	X	--	--	--
16	On Finish Input (Page 4851)	X		X	X	--	--	--	--	--	--	--
17	Press ESC twice (Page 4851)	--	--	--	--	--	--	--	--	--	--	--
18	Outgoing (Page 4852)	--	--	--	--	--	--	--	--	--	--	--
19	Incoming (Page 4852)	--	--	--	--	--	--	--	--	--	--	--
20	Click (Page 4852)	--	X	--	--	--	--	--	X	--	--	--
21	Click when flashing (Page 4853)	--	--	--	--	--	--	--	--	--	--	--
22	Loop-in alarm (Page 4853)	--	--	--	--	--	--	--	--	--	--	--

23	Release (Page 4853)	--	X	--	--	--	--	--	X	--	--	--
24	Alarm buffer overflow (Page 4854)	--	--	--	--	--	--	--	--	--	--	--
25	Acknowledge (Page 4854)	--	--	--	--	--	--	--	--	--	--	--
26	Reach margin (Page 4854)	--	--	--	--	--	--	--	--	--	--	--
27	Runtime Stop (Page 4855)	--	--	--	--	--	--	--	--	--	--	--
28	Press key (Page 4855)	--	--	--	--	--	--	--	--	--	--	--
29	Release key (Page 4855)	--	--	--	--	--	--	--	--	--	--	--
30	Overflow (Page 4856)	--	--	--	--	--	--	--	--	--	--	--
31	Switching (Page 4856)	--	--	--	--	--	--	--	--	--	--	--
32	Switch OFF (Page 4856)	--	--	--	--	--	--	X	--	--	--	--
33	Switch ON (Page 4856)	--	--	--	--	--	--	X	--	--	--	--
34	Low free storage space (Page 4857)	--	--	--	--	--	--	--	--	--	--	X
35	Free space critically low (Page 4857)	--	--	--	--	--	--	--	--	--	--	X
36	Value change (Page 4857)	--	--	--	--	--	--	--	--	--	--	--
37	Time expired (Page 4857)	--	--	--	--	--	--	--	--	--	--	--

Controls

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

	Object
	Alarm view/alarm window
	Alarm indicator
	Trend view
	User view
	HTML browser
	Status/Force
	Sm@rtClient view
	Recipe view
	f(x) trend view
	System diagnostics view / system diagnostics window
	Function key

1	Cleared (Page 4847)	--	--	--	--	--	--	--	--	--	--	--
2	Activate (Page 4847)	X	--	X	X	X	X	X	X	--	X	--
3	Change (Page 4847)	--	--	--	--	--	--	--	--	--	--	--
4	Loaded (Page 4848)	--	--	--	--	--	--	--	--	--	--	--

5	Execute (Page 4848)	--	--	--	--	--	--	--	--	--	--	--
6	Selection changed (Page 4848)	X	--	--	--	--	--	--	--	--	--	--
7	On exceeding (Page 4848)	--	--	--	--	--	--	--	--	--	--	--
8	On falling below (Page 4849)	--	--	--	--	--	--	--	--	--	--	--
9	When dialog is opened (Page 4849)	--	--	--	--	--	--	--	--	--	--	--
10	When dialog is closed (Page 4849)	--	--	--	--	--	--	--	--	--	--	--
11	User change (Page 4849)	--	--	--	--	--	--	--	--	--	--	--
12	Screen change (Page 4850)	--	--	--	--	--	--	--	--	--	--	--
13	Deactivate (Page 4850)	X	--	X	X	X	X	X	X	--	--	--
14	Double-click (Page 4850)	--	--	--	--	--	--	--	--	--	--	--
15	Press (Page 4851)	--	--	--	--	--	--	--	--	--	--	--
16	On Finish Input (Page 4851)	--	--	--	--	--	--	--	--		--	--
17	Press ESC twice (Page 4851)	--	--	--	--	--	--	--	--	--	--	--
18	Outgoing (Page 4852)	--	--	--	--	--	--	--	--	--	--	--
19	Incoming (Page 4852)	--	--	--	--	--	--	--	--	--	--	--
20	Click (Page 4852)	--	X	--	--	--	--	--	--	--	--	--
21	Click when flashing (Page 4853)	--	X	--	--	--	--	--	--	--	--	--
22	Loop-in alarm (Page 4853)	--	--	--	--	--	--	--	--	--	--	--
23	Release (Page 4853)	--	--	--	--	--	--	--	--	--	--	--
24	Alarm buffer overflow (Page 4854)	--	--	--	--	--	--	--	--	--	--	--
25	Acknowledge (Page 4854)	--	--	--	--	--	--	--	--	--	--	--
26	Reach margin (Page 4854)	--	--	--	--	--	--	--	--	--	--	--
27	Runtime Stop (Page 4855)	--	--	--	--	--	--	--	--	--	--	--
28	Press key (Page 4855)	--	--	--	--	--	--	--	--	--	--	X
29	Release key (Page 4855)	--	--	--	--	--	--	--	--	--	--	X
30	Overflow (Page 4856)	--	--	--	--	--	--	--	--	--	--	--
31	Switching (Page 4856)	--	--	--	--	--	--	--	--	--	--	--
32	Switch OFF (Page 4856)	--	--	--	--	--	--	--	--	--	--	--
33	Switch ON (Page 4856)	--	--	--	--	--	--	--	--	--	--	--
34	Low free storage space (Page 4857)	--	--	--	--	--	--	--	--	--	--	--
35	Free space critically low (Page 4857)	--	--	--	--	--	--	--	--	--	--	--
36	Value change (Page 4857)	--	--	--	--	--	--	--	--	--	--	--
37	Time expired (Page 4857)	--	--	--	--	--	--	--	--	--	--	--

Events

Cleared

Description

Occurs when the active screen on the HMI device is cleared.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Activate

Description

Occurs when the user selects a display or operating object using the configured tab sequence.

Note

Note that the availability of the event depends on the HMI device and object type.

Note

If the user e.g. clicks a button with the mouse, the "Click" event is triggered. Users wishing to trigger the "Enable" event must select the button using the tab key.

The "Enable" event is only used to detect whether an object was selected. The event does not trigger a password prompt.

For this reason, do not use the "Enable" event if you want to configure access protection on the function call of the object.

Change

Description

Occurs if the status of a display and operator control object changes.

The status of an object changes if, for example, the user presses the key.

Note

Note that the availability of the event depends on the HMI device and object type.

Loaded

Description

Occurs when all configured display and operating objects are loaded in the active screen after a screen change.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

Enable a screen change to ensure that the connection with the control is established after switch-on.

Execute

Description

Occurs when the scheduled task has been executed.

Selection changed

Description

Occurs when the user changes the selection.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

On exceeding

Description

Occurs when the high limit of a tag is exceeded.

Note

Note that the availability of the event depends on the HMI device and object type.

On falling below

Description

Occurs when the low limit of a tag is undershot.

Note

Note that the availability of the event depends on the HMI device and object type.

When dialog is opened

Description

The event is triggered when a modal dialog opens.

Note

Please note that the availability of the event depends on the HMI device and object type.

When dialog is closed

Description

The event is triggered when a modal dialog closes.

Note

Please note that the availability of the event depends on the HMI device and object type.

User change

Description

Occurs when a user logs off at an HMI device or another user logs on at the HMI device.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Screen change

Description

Occurs when all configured display and operating objects are loaded in the screen after a screen change.

Use the "Loaded" event if you want to perform other system functions during a screen change to a certain screen.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Deactivate

Description

Occurs when the user takes the focus from a display and operating object.

A screen object can be disabled using the configured tab order or by performing another action with the mouse.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

System functions or user-defined functions on the "Deactivate" event of a screen are not executed when a screen is being closed.

The "Deactivate" event is only used to detect whether an object was deselected. The event does not trigger a password prompt.

For this reason, do not use the "Deactivate" event if you want to configure access protection on the function call of the object.

Double-click

Description

Occurs when the user double-clicks on an object from the symbol library.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Press

Description

Occurs when the user clicks on a button with the left mouse button, presses <RETURN> or <SPACE>.

Also occurs when the user right-clicks on an object of the symbol library.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

On Finish Input

Description

Triggered when you confirm input at an I/O field by pressing ENTER, or by mouse click, or by touch screen operation.

The "On Finish Input" event is also started if the value of a tag does not change, for example, if a value is exceeded, or if a user cancels the dialog to acknowledge a tag (Audit option package) that has to be acknowledged.

The event is not triggered, on the other hand, by user logon or by input fields configured with an authorization.

Note

Note that the availability of the event depends on the HMI device and object type.

Press ESC twice

Description

Occurs when the user presses the <ESC> key twice at the HMI device.

Note

Note that the availability of the event depends on the HMI device and object type.

Outgoing

Description

Occurs when an alarm is deactivated.

Note

Please note that the availability of the event depends on the HMI device and object type.

Incoming

Description

Occurs when an alarm is triggered and displayed in the alarm view.

Note

Please note that the availability of the event depends on the HMI device and object type.

Click

Description

Occurs if the user clicks a display and operating object with the mouse or touches the touch display with a finger.

In case you click the incorrect object, prevent processing of configured function list as follows:

- Move the mouse pointer away from the object while keeping the mouse button pressed. Release the mouse button as soon as the mouse pointer leaves the object. The function list will then not be processed.
- On touch displays, the display must be touched with the finger until a reaction occurs, e.g., a screen change.

  Note

  Please note that the availability of the event is dependent upon the HMI device and object type.

Click when flashing

Description

Occurs when the user clicks a flashing alarm indicator with the mouse or touches it with a finger.

Note

Note that the availability of the event depends on the HMI device and object type.

Loop-in alarm

Description

Occurs as soon as the user selects an alarm in the alarm view and then clicks on the "Loop-In-Alarm" button or double clicks on the alarm.

For the "Loop-In-Alarm" event, you configure system functions, such as a change to the screen in which the alarm occurred.

You cannot configure local scripts for the "Loop-In-Alarm" event in Runtime Professional.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Release

Description

Occurs when the user releases a button.

This even does not occur, as long as the button remains pressed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Alarm buffer overflow

Description

Occurs when the configured size of the alarm buffer is reached.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Acknowledge

Description

Occurs when the user acknowledges an alarm.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Reach margin

Description

Occurs when the user reaches the beginning or the end of a scrollable area.

Note

Note that the availability of the event depends on the HMI device and object type.

Note

A user-defined function must not be configured for the "Boundary reached" event.

Configurable objects

The event can only be configured on the <Up> and <Down> keys, or on the keys on which you have configured the "ScreenObjectPageUp" or "ScreenObjectPageDown" system functions.

Runtime Stop

Description

Occurs when the user exits the Runtime software on the HMI device.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

A user-defined function must not be configured for the "Runtime stop" event.

Press key

Description

Occurs when the user presses a function key.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Release key

Description

Occurs when the user releases a function key.

Note

Note that the availability of the event depends on the HMI device and object type.

Overflow

Description

Occurs when the configured size of the log is reached. You use the log type "Trigger event".

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Switching

Description

Occurs when the user toggles a display and operating object, e.g. a switch from "ON" to "OFF".

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Switch OFF

Description

Occurs when the user moves the display and operating object "Switch" to the OFF position.

Note

Note that the availability of the event depends on the HMI device and object type.

Switch ON

Description

Occurs when the user moves the display and operating object "Switch" to the ON position.

Note

Note that the availability of the event depends on the HMI device and object type.

Low free storage space

Description

This event is triggered if the storage space available on the medium to which the Audit Trail is less than the configured minimum.

Free space critically low

Description

This event is triggered if the storage medium to which an Audit Trail is saved provides insufficient storage space due to hardware restrictions.

Value change

Description

Occurs when the value of a tag or the value of an array element changes.

The value change of a tag is triggered by the PLC or by the user, e.g. when a new value is entered. No event is output if a system function changes the value.

Note

Please note that the availability of the event depends on the HMI device and object type.

Time expired

Description

Occurs when the time configured in the scheduler expires.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

12.8.7.3 VB scripts

System functions

AcknowledgeAlarm

Description

Acknowledges all selected alarms.

This system function is used when the HMI device does not have an ACK key or when the integrated key of the alarm view should not be used.

This system function can only be used for function keys.

Use in the function list

AcknowledgeAlarm

Use in user-defined functions

AcknowledgeAlarm

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

ActivatePreviousScreen

Description

Performs a screen change to the screen which was activated before the current screen. The screen change is not performed if no screen was activated beforehand.

The last 10 screens that were called up are saved. A system alarm is output when you change to a screen which is no longer saved.

Note

If you want to use the system function, the screen to which you change has to be used in the navigation structure.

Use in the function list

ActivatePreviousScreen

Use in user-defined functions

ActivatePreviousScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

ActivateScreen

Description

Performs a screen change to the given screen.

Use the "ActivateScreenByNumber" system function to change from the root screen to the permanent window or vice versa.

Use in the function list

ActivateScreen (Screen name, Object number)

Use in user-defined functions

ActivateScreen (Screen_name, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Name of the screen to which you change.

Object number

The operator control element which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent window when the system function is called up, the permanent window maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

Note

If the "Reach margin" event is assigned to the "ActivateScreen" system function, only the value "0" is valid for the "Object number" parameter. The active object is not defined by the object number, but rather by the X position it had prior to the screen change.

Example

The ActivateScreen function is used in the following program code to activate the "Screen_2" screen when you click a button.

```
{

// User defined code
// i.e. when pressing a button
ActivateScreen ("Screen_2", 0);
...
}
```

See also

Availability for specific devices of system functions (Page 4612)

ActivateScreenByNumber

Description

Performs a screen change to a screen depending on a tag value.

The screen is identified by its screen number.

Use in the function list

ActivateScreenByNumber (Screen number, Object number)

Use in user-defined functions

ActivateScreenByNumber (Screen_number, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Screen number

Tag which contains the screen number of the destination screen.

When a change from the root screen to the permanent window is desired, "0" or "-1" is specified:

0 = Change from root screen to permanent window

-1 = Change from permanent window to root screen

Object number

The number of the screen object which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent window when the system function is called up, the permanent window maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

See also

Availability for specific devices of system functions (Page 4612)

ActivateSystemDiagnosticsView

Description

Activates the system diagnostics view. The system diagnostics view shows the detail view of the device concerned.

Use in the function list

ActivateSystemDiagnosticsView (Screen name, Object name)

Use in user-defined functions

ActivateSystemDiagnosticsView (Screen_name, Object_name)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Screen name

Name of the screen contained in the system diagnostics view.

Object name

Object name of the system diagnostics view.

See also

Availability for specific devices of system functions (Page 4612)

ArchiveLogFile

Description

This system function moves or copies a log to another storage location for long-term logging.

With Audit Trails, always use the "Move" (hmiMove)" mode, otherwise you will be violating the FDA guideline by duplicating the data stored.

Prior to "ArchiveLogFile" always run the system function "CloseAllLogs".

After this system function, run the "OpenAllLogs" system function.

In "Copy log" mode, the logs are not reopened until the log has been successfully copied, or unless a timeout occurs during the copy process. In "Move log" mode, the logs to be moved are renamed and the new logs are opened immediately. To move the renamed logs, a job is submitted which attempts to move them every 300 seconds if the server cannot be reached. The job is also retained after Runtime is restarted until it is executed.

Use in the function list

ArchiveLogFile (Log type, Log, Directory name, Mode)

Use in user-defined functions

ArchiveLogFile (Log_type, Log, Directory_name, Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (miAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail log available for GMP compliant projects. Additional information is available under "Activating GMP-compliant configuration".

Log

Name of the log being archived.

Directory name

Path for saving the log.

Mode

0 (hmiCopy) = Copy log

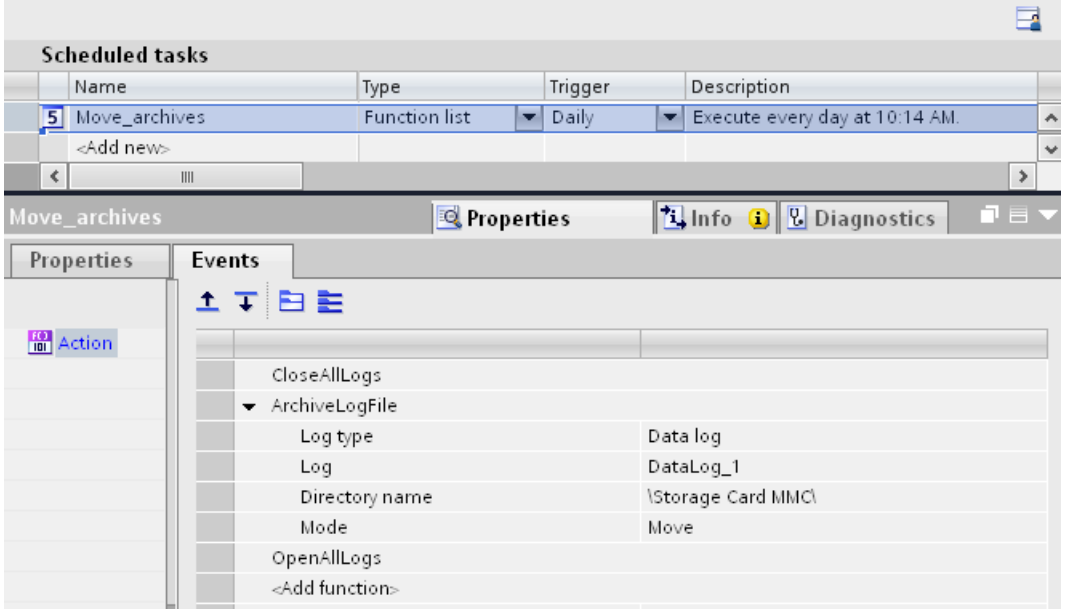
1 (hmiMove) = Move log

Application example

You want to move a log file from a local storage medium to the server in order to generate a backup copy of this file at cyclic intervals.

Notes on configuring

Set up a task in the scheduler which is executed at a specific time on a daily basis. Configure the following function list for the task:



Procedure on HMI device

- All log files will be closed.
- The log file you specified is moved to the server.
- All closed log files will be opened again.

See also

Availability for specific devices of system functions (Page 4612)

Back up RAM file system

Description

Backs up the RAM file system in the memory medium of the HMI device.

After restarting the HMI device, the data is automatically reloaded in the RAM file system.

Applications such as the Internet Explorer save data (e.g. the last web sites called up) temporarily to the DRAM file system of the HMI device.

Use in the function list

BackupRAMFileSystem

Use in user-defined functions

BackupRAMFileSystem

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

CalibrateTouchScreen

Description

Calls a program for calibrating the touch screen.

During the calibration process, there is a prompt to touch five positions on the screen display. Touch the screen display within 30 seconds, to confirm the calibration process. If the calibration

is not completed within this time span, the calibration settings are discarded. The user prompt is in English.

Use this system function the first time you start the HMI device.

Use in the function list

CalibrateTouchScreen

Use in user-defined functions

CalibrateTouchScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Note

The CalibrateTouchScreen system function cannot be simulated.

See also

Availability for specific devices of system functions (Page 4612)

ChangeConnection

Description

Disconnects the connection to the currently used PLC in use and establishes a connection to a PLC with another address. The newly connected PLC must belong to the same device class (S7-1200, S7-300, ...etc). With S7-1200, the use of the function is also only permitted for absolute addressing.

Note

When changing to another address, ensure that the address is not already being used by another HMI device.

The follows address types are supported:

- IP address
- MPI address

The follows PLC types are supported:

- SIMATIC S7 300/400
- SIMATIC S7-NC
- SIMOTION

Use in the function list

ChangeConnection (Connection, Address, Slot, Rack)

Use in user-defined functions

ChangeConnection (Connection, Address, Slot, Rack)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Connection

Name of the connection that is disconnected. The name is set during configuration, for example, in the "Connections" editor.

Address

MPI/PROFIBUS or IP address of the PLC to which the connection will be established.

Note

Set the address by means of a tag. The objects list displays the tags of all data types. Select only tags of the following data types:

- Ethernet connection: "String" data type
- MPI connection: "Int" data type

Slot

Slot of the PLC to which the connection will be established.

Rack

Rack of the PLC to which the connection will be established.

Application example

You want to operate one HMI device on several machines. Configure the necessary PLCs in the project, to which you want to change by pressing a key. When changing the PLC, the connection to the PLC in use is disconnected. Then the connection to the new PLC with other address parameters is reestablished. To access the values of the new PLC, configure the same tags for the PLC used.

The PLC which you have indicated when creating the project will be used as default.

1. Enter the name and address of the PLC in the "Connections" editor.
2. Configure a button in the process screen.
3. Configure the "ChangeConnection" system function on the "Press" event.
4. Enter the name of the connection and address of the PLC as parameters.

See also

Availability for specific devices of system functions (Page 4612)

ClearAlarmBuffer

Description

Deletes alarms from the alarm buffer on the HMI device.

Note

Alarms which have not yet been acknowledged are also deleted.

Use in the function list

ClearAlarmBuffer (Alarm class number)

Use in user-defined functions

ClearAlarmBuffer (Alarm_class_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Alarm class number

Determines which alarms are to be deleted from the alarm buffer:

0 (hmiAll) = All alarms/events

1 (hmiAlarms) = Alarms of alarm class "Errors"

2 (hmiEvents) = Alarms of alarm class "Warnings"

3 (hmiSystem) = Alarms of alarm class "System"

4 (hmiS7Diagnosis) = Alarms of alarm class "Diagnosis Events"

Note

Device dependency

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

See also

Availability for specific devices of system functions (Page 4612)

ClearAlarmBufferProtool

Description

The system function exists to ensure compatibility.

It has the same functionality as the "ClearAlarmBuffer" system function, but uses the old ProTool numbering.

Use in the function list

ClearAlarmBufferProTool (Alarm class number)

Use in user-defined functions

ClearAlarmBufferProtoolLegacy (Alarm_class_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Alarm class number

Alarm class number whose messages are to be deleted:

-1 (hmiAllProtoolLegacy) = All alarms/events

0 (hmiAlarmsProtoolLegacy) = Alarms of alarm class "Errors"

1 (hmiEventsProtoolLegacy) = Alarms of alarm class "Warnings"

2 (hmiSystemProtoolLegacy) = Alarms of alarm class "System"

3 (hmiS7DiagnosisProtoolLegacy) = Alarms of alarm class "Diagnosis Events"

Note
Device dependency

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

See also

Availability for specific devices of system functions (Page 4612)

ClearDataRecord

Description

Deletes a recipe data record.

Several data records can be deleted from one or more recipes.

Use in the function list

DeleteDataRecord (Recipe number/name, Data record number/name, Confirmation, Output status message, Processing status)

Use in user-defined functions

DeleteDataRecord

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data records are deleted. "0" is specified if you want to delete recipe data records from all available recipes.

Data record number/name

Number or name of the recipe data record to be deleted. "0" is specified if you want to delete all recipe data records.

Confirmation

Determines whether the operator should confirm the deletion:

0 (hmiOff) = Off: Deletion is begun without confirmation.

1 (hmiOn) = On: The starting of the deletion process must be confirmed.

Output status message

Determines whether a status message is output after the deletion:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

ClearDataRecordMemory

Description

Deletes all recipes data records from the specified storage medium.

Use in the function list

ClearDataRecordMemory (Storage location, Confirmation, Output status message, Processing status)

Use in user-defined functions

ClearDataRecordMemory (Storage_location, Confirmation, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Storage location

Determines the storage location:

0 (hmiFlashMemory) = Flash memory: Internal flash memory of the HMI device

1 (hmiStorageCard) = Storage card

2 (hmiStorageCard2) = Storage card 2

3 (hmiStorageCard3) = Storage card MultiMediaCard

4 (hmiStorageCard4) = Storage card USB

Confirmation

Determines whether the operator should confirm the deletion:

0 (hmiOff) = Off: Deletion is begun without confirmation.

1 (hmiOn) = On: The starting of the deletion process must be confirmed.

Output status message

Determines whether a status message is output after the deletion:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

ClearLog

Description

Deletes all data records in the given log.

Use in the function list

ClearLog (Log type, Log)

Use in user-defined functions

ClearLog (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail log Available for GMP-compliant projects Additional information is available under "Activating GMP-compliant configuration".

Log

Name of the log from which all entries are deleted.

See also

Availability for specific devices of system functions (Page 4612)

CloseAllLogs

Description

Disconnects the connection between WinCC and all logs.

Note

Before you close a log, the logging function must first be stopped in the log. Use the "StopLogging" system function.

Use in the function list

CloseAllLogs

Use in user-defined functions

CloseAllLogs

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "StopLogging" and "CloseAllLogs" system functions on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When the button "Close Archive" is pressed, the specified log is stopped and all open logs are closed. The data medium can be changed. You open all logs with the "Open Archive" button. The logging process will continue in the specified log.

See also

Availability for specific devices of system functions (Page 4612)

ControlSmartServer

Description

Starts or stops the Sm@rtServer.

Use in the function list

ControlSmartServer (Mode)

Use in user-defined functions

ControlSmartServer (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Specifies whether the Sm@rtServer is started or stopped.

-1 (hmiToggle) = Toggle: Toggles between the two modes

0 (hmiStop) = Stop: Sm@rtServer is stopped

1 (hmiStart) = Start: Sm@rtServer is started

See also

Availability for specific devices of system functions (Page 4612)

ControlWebServer

Description

Starts or stops the Web server.

Use in the function list

ControlWebServer (Mode)

Use in user-defined functions

ControlWebServer (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Specifies whether the Web server is started or stopped.

-1 (hmiToggle) = Toggle: Toggles between the two modes

0 (hmiStop) = Stop: The Web server is stopped.

1 (hmiStart) = Start: The Web server is started.

See also

Availability for specific devices of system functions (Page 4612)

CopyLog

Description

Copies the contents of a log to another log. Tag values can only be copied to other data logs and alarms only to other alarm logs, however.

Note

If you copy a log using the "CopyLog" system function, it is possible that external applications will be unable to read certain country-specific special characters in the logged message text of the log copy. This does not apply to WinCC Runtime. WinCC Runtime can read the copied log files without errors.

Note

80% of the log entries are copied when copying the circular logs. 20% of entries are not copied because the space is reserved for buffer overflow by default.

Use in the function list

CopyLog (Log type, Destination log, Source log, Mode, Delete source log)

Use in user-defined functions

CopyLog (Log_type, Destination log, Source_log, Mode, Delete_source_log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

Destination log

Name of the log into which the entries are copied (Destination log).

Source log

Name of the log from which the entries are copied (Source log).

Mode

Determines what is done with copied entries in the destination log:

0 (hmiOverwrite) = Overwrite: Existing entries are overwritten.

2 (hmiAppend) = Append: The entries are inserted at the end of the destination log. When the configured size of the log has been reached, the destination log is treated like a circular log.

Delete source log

Determines whether the source log is deleted after copying.

0 (hmiNo) = No: Is not deleted.

1 (hmiYes) = Yes: Is deleted.

See also

Availability for specific devices of system functions (Page 4612)

DecreaseTag

Description

Subtracts the given value from the tag value.

X = X - a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. The auxiliary tags are assigned to the tag value with the "SetTag" system function.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

DecreaseTag (Tag, Value)

Use in user-defined functions

DecreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag from which the given value is subtracted.

Value

The value which is subtracted.

See also

Availability for specific devices of system functions (Page 4612)

EditAlarm

Description

Triggers the "Edit" event for all selected alarms.

If the alarms to be edited have not yet been acknowledged, the acknowledgment takes place automatically when this system function is called up.

This system function can only be used for function keys.

Use in the function list

EditAlarm

Use in user-defined functions

EditAlarm

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

Encode

Description

Adapts the "String" data type of a tag for transfer to the automation system (AS). The tag of WinCC data type "String" is converted into the AS data type "Array of byte". The result is written to a tag.

Use in the function list

Encode (Byte array, String, Coding)

Use in user-defined functions

Encode (Byte_array, String, Encoding)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Byte array

The tag that contains the converted value.

Note

The Byte array must be twice the size of the string length + 2. The Byte array must contain 242 array elements if the string has a length of 120 characters.

Characters are either truncated or not converted if the size is insufficient.

String

The tag of data type "String" that is converted.

Encode

0 (hmiEncodeUTF16LE) - String is encoded in UTF16LE (Unicode 16 Little Endian).

See also

Availability for specific devices of system functions (Page 4612)

EncodeEx

Description

Adapts the "String" data type of a tag for transfer to the automation system (AS). The tag of WinCC data type "String" is converted into the AS data type "Array of byte". The result is written to a tag.

By contrast to the Encode system function, you can define the Line break parameter. Using the Line break parameter you can delete line breaks or replace these with predefined characters.

Use in the function list

EncodeEx (Byte Array, String, Encode, Line break)

Use in user-defined functions

EncodeEx (Byte_array, String, Encoding, Line_break)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Byte array

The tag that contains the converted value.

Note

The byte array must be twice the size of the string length +2. If the string has a length of 120 characters, the byte array must contain 242 array elements.

Characters are either truncated or not converted if the size is insufficient.

String

The tag of data type "String" that is converted.

Encoding

0 (hmiEncodeUTF16LE) - String is encoded in UTF16LE (Unicode 16 Little Endian).

Line break

All line breaks are either deleted or replaced with predefined characters. Do not replace line breaks if set as the default value.

0 (replace with "\r\n' (0x000D, 0x000A)) - the line break is replaced with "\r\n".

1 (replace with "\n' (0x000A)) - the line break is replaced with "\n".

2 (do not replace) - the line break is not replaced.

3 (remove line breaks) - the line breaks are removed.

See also

Availability for specific devices of system functions (Page 4612)

EstablishPROFIsafe

Description

Establishes the PROFIsafe connection for fail-safe operation between a KTP Mobile Panel and the PLC if the connection was previously disconnected by means of TerminatePROFIsafe.

Note

A connection for the PROFIsafe communication can only be established if ONLINE mode has been set on the HMI device. Use the "SetDeviceMode" system function for this purpose.

Use in the function list

EstablishPROFIsafe

Use in user-defined functions

EstablishPROFIsafe

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

--

ExportDataRecords

Description

Exports one or all data records of a recipe to a CSV file.

One file is created per recipe.

Use in the function list

ExportDataRecords (Recipe number/name, Data record number/name, File name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ExportDataRecords (Recipe_number/name,Data_record number/name, File_name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Recipe number/name

Number or name of the recipe from which the data records are exported. Specify "0" if recipe data records are to be exported from all available recipes.

File name

Name of the CSV file to which the recipe data records are exported. Enter the storage location and the file extension (*.csv), for example, "C:\TEMP\Orange.csv", for Basic Panels "\USB_X61.1\Orange.csv".

Note

Storage of the CSV file

Does not apply for Basic Panels:

- If a storage card is used as storage location, specify the storage location as follows: "\StorageCard\<File name>".
- If you only enter one file name and no path, the file will be saved in a system directory, for example, "C:\Documents and Settings\[User]".
- When only one path for the export is specified, the file name is automatically created from the respective recipe name. It is a requirement that the folder "D:\Temp\", for example, has been created. If the folder "D:\Temp" has not been created, the folder name will be used as the prefix of the file name, Temp_Recipe names.csv.

Data record number/name

Number or name of the recipe data record to be exported. Specify "0" if all recipe data records are to be exported.

Overwrite

Determines whether an existing CSV file with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: CSV file will not be overwritten. The export process will not be carried out.

1 (hmiOverwriteAlways) = Yes: CSV file is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithPrompting) = With confirmation: CSV file is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the export:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Export format of the recipe data records

If ".csv" is chosen as a file extension for the export file, then only valid characters from the ANSI character set will be supported. This also applies to separators in decimal numbers and list elements. The separators used are defined in the country/regional settings of the operating system of the exporting computer.

Does not apply for Basic Panels:

You may also set "Unicode text" (".txt") format for the export file. This file format supports the WinCC and WinCC Runtime character set. Again, the separators used are specified in the country/regional settings of the operating system of the exporting computer. This file format always uses tab separators in list elements.

The corresponding file import function also supports the ".csv" and ".txt" (Unicode) file formats.

Application example

You want to export all data records using a key.

Notes on configuring

Configure the "ExportDataRecords" system function on the "Press" event for the desired key. Transfer the following parameters:

- Recipe number/name = 1
- Data record number/name = 0
- File name = c:\temp\orange.csv, for Basic Panels "\USB_X61.1\orange.csv"
- Overwrite = 1
- Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data records are exported.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated. All data records of Recipe 1 are exported to the orange.csv file. If the file already exists, it will be overwritten.

After exporting the data records, a system alarm is output.

See also

Availability for specific devices of system functions (Page 4612)

ExportDataRecordsWithChecksum

Description

Exports one or all data records of a recipe to a CSV file and generates a checksum for each line in the file.

One file is created per recipe.

Use in the function list

ExportDataRecordsWithChecksum (Recipe number/name, Data record number/name, File name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ExportDataRecordsWithChecksum (Recipe_number/name, Data_record number/name, File_name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which the data records are exported. Specify "0" if recipe data records are to be exported from all available recipes.

Data record number/name

Number or name of the recipe data record to be exported. Specify "0" if all recipe data records are to be exported.

File name

Name of the CSV file to which the recipe data records are exported. Enter the path and the file extension e.g. "C:\TEMP\Orange.CSV".

If a storage card is used as storage location, specify the storage location as follows: "\StorageCard\<File name>".

If you define only a file name without specifying a path, the file is saved to the directory from which Runtime was started. If write access to this directory is not enabled in the Windows 7 operating system, the file is saved to the "VirtualStore" folder of the user directory.

When only one path for the export is specified, the file name is automatically created from the respective recipe name. This requires, for example, that the directory "D:\Temp\" has been created. If the directory "D:\Temp" does not exist, the directory name is used as the prefix for the file name, Temp_Recipe names.csv.

Overwrite

Determines whether an existing CSV file with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: CSV file will not be overwritten. The export process will not be carried out.

1 (hmiOverwriteAlways) = Yes: CSV file is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithConfirmation) = With confirmation: CSV file is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the export:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Export format of the recipe data records

If ".csv" is chosen as a file extension for the export file, then only valid characters from the ANSI character set will be supported. This also applies to separators in decimal numbers and list elements. The separators used are defined in the country/regional settings of the operating system of the exporting computer.

You may also set "Unicode text" (".txt") format for the export file. This file format supports the WinCC and WinCC Runtime character set. Again, the separators used are specified in the country/regional settings of the operating system of the exporting computer. This file format always uses tab separators in list elements.

The corresponding file import function also supports the ".csv" and ".txt" (Unicode) file formats.

Application example

Using a key, you want to export all data records and assign a checksum.

Notes on configuring

Configure the "ExportDataRecordsWithChecksum" system function on the "Press" event for the desired key. Transfer the following parameters:

- Recipe number/name = 1
- Data record number/name = 0
- File name = c:\temp\orange.csv
- Overwrite = 1
- Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data records are exported.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated. All data records of Recipe 1 are exported to the orange.csv file and assigned checksums. If the file already exists, it will be overwritten.

After exporting the data records, a system message is output.

See also

Availability for specific devices of system functions (Page 4612)

ExportImportUserAdministration

Description

Exports all users of the user administration of the currently active project to the given file or imports the users from the given file into the currently active project.

Users, their passwords and rights are saved in the user administration.

All users are overwritten when importing. The imported users are valid immediately.

Use in the function list

ExportImportUserAdministration (File name, Direction)

Use in user-defined functions

ExportImportUserAdministration (File_name, Direction)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

File name

Name of the file which contains the passwords or to which the passwords are written. Enter the file location and the file extension (*.txt), for example, "C:\TEMP\Passwords.txt".

Note

If a storage card is used as file location, specify the file location as follows: "\StorageCard\<File name>".

Direction

Specifies whether passwords are exported or imported:

0 (hmiExport) = Export: Passwords are exported.

1 (hmiImport) = Import: Passwords are imported.

See also

Availability for specific devices of system functions (Page 4612)

GetBrightness

Description

Reads the value of the brightness.

Use in the function list

GetBrightness (Brightness)

Use in user-defined functions

GetBrightness (Brightness)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Brightness

The tag to which the value is written.

See also

Availability for specific devices of system functions (Page 4612)

GetDataRecordFromPLC

Description

Transfers the selected recipe data record from the PLC to the storage medium of the HMI device.

Use in the function list

GetDataRecordFromPLC (Recipe number/name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

GetDataRecordFromPLC (Recipe_number/name, Data_record_number/name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data records are transferred.

Data record number/name

Number or name of the recipe data record which is transferred from the PLC to the data medium of the HMI device.

Overwrite

Determines whether an existing recipe data record with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data record is not overwritten. The transfer process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data record is overwritten without prompting.

2 (hmiOverwriteWithPrompting) = With confirmation: Recipe data record is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the transfer:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Application example

You want to transfer a data record from the PLC to the data medium of the HMI device using a key.

Notes on configuring

Configure the "GetDataRecordFromPLC" system function on the "Press" event for the desired key. Transfer the following parameters:

Recipe number/name = 1

Data record number/name = 1

Overwrite = 1

Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data record is transferred from the PLC.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated and the first data record of Recipe 1 is transferred from the PLC to the data medium of the HMI device. If the recipe data record already exists, it will be overwritten.

A system message is output after the transfer.

See also

Availability for specific devices of system functions (Page 4612)

GetDataRecordName

Description

Writes the name of the given recipe and recipe data record to the given tags.

Note

If the recipe or the recipe data record do not exist, wildcards ("###") are written to the tags.

Use in the function list

GetDataRecordName (Recipe number, Data record number, Recipe name, Data record name, Processing status)

Use in user-defined functions

GetDataRecordName (Recipe_number, Data_record_number, Recipe_name, Data_record_name, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number

Number of the recipe whose name is written to the given tag.

Data record number

Number of the recipe data record whose name is written to the given tag.

Recipe name

The tag to which the recipe name is written. The tag must be of the STRING type.

Data record name

The tag to which the name of the recipe data record is written. The tag must be of the STRING type.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Application example

You want to output the names of the displayed recipes and the name of the displayed recipe data record on the HMI device.

Configure the following tags:

- "RecNumber" of type INTEGER
- "RecDataNumber" of type INTEGER
- "RecName" of type STRING
- "RecDataName" of type STRING

Configure a recipe view with the tags "RecNumber" for the recipe number and "RecDataNumber" for the data record number.

Configure the "GetDataRecordName" system function on the "Press" event for a button and pass the following parameters:

- Recipe number: RecNumber
- Data record number: RecDataNumber
- Recipe name: RecName
- Data record name: RecDataName

Configure two output fields and connect these to the "RecName" and "RecDataName" tags.

Select a recipe and a data record number from the recipe view. As soon as the button is activated, the system function is triggered and the name of the recipe and the recipe data record are displayed in both output fields.

See also

Availability for specific devices of system functions (Page 4612)

GetDataRecordTagsFromPLC

Description

Transfers the values of the recipe data record loaded in the PLC to the corresponding recipe tags.

This system function is used, for example, during teach-in mode on a machine.

Use in the function list

GetDataRecordTagsFromPLC (Recipe number/name, Processing status)

Use in user-defined functions

GetDataRecordTagsFromPLC (Recipe_number/name, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe data record whose values are written from the PLC to the tags.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

GetGroupNumber

Description

Reads the number of the group to which the user logged on to the HMI device belongs, and writes it to the given tag.

Use in the function list

GetGroupNumber (Tag)

Use in user-defined functions

GetGroupNumber (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the number of the group is written.

See also

Availability for specific devices of system functions (Page 4612)

GetPassword

Description

Writes the password of the user currently logged on to the HMI device in the given tag.

Note

Make sure that the value of the given tag is not displayed in another place in the project.

Note

The passwords of SIMATIC Logon users cannot be read.

Use in the function list

GetPassword (Tag)

Use in user-defined functions

GetPassword (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the password is written.

See also

Availability for specific devices of system functions (Page 4612)

GetUserName

Description

Writes the user name of the user currently logged on to the HMI device in the given tag.

If the given tag has a control connection, the user name is also available in the PLC connection. This system function makes it possible, for example, to implement a user-dependent release of certain functionalities.

Use in the function list

GetUserName (Tag)

Use in user-defined functions

GetUserName (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the user name is written.

See also

Availability for specific devices of system functions (Page 4612)

GoToEnd

Description

Executes the key function <End> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToEnd

Use in user-defined functions

GoToEnd

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

GoToHome

Description

Executes the key function <Home> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToHome

Use in user-defined functions

GoToHome

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

ImportDataRecords

Description

Imports one or all data records of a recipe from a CSV file.

When a path is specified, all records of the given directory are imported.

Use in the function list

ImportDataRecords (File name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ImportDataRecords (File_name, Data_record number/name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

File name

Name of the CSV file from which the recipe data records are imported. Enter the storage location and the file extension (*.csv), for example, "C:\TEMP\Orange.csv", for Basic Panels "\USB_X61.1\Orange.csv".

Note

Does not apply for Basic Panels: If a storage card is used as file location, specify the file location as follows: "\StorageCard\<File name>".

Data record number/name

Number or name of the recipe data record to be imported. Specify "0" if all recipe data records are to be imported.

Overwrite

Determines whether existing recipe data records are to be overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data records are not overwritten. The import process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data records are overwritten without prompting.

2 (hmiOverwriteWithPrompting) = With confirmation: Recipe data records are not overwritten until confirmed.

Output status message

Determines whether a status message is output after the import:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Configurable objects

Object	Event
Tag	Value change Upper limit exceeded Lower limit violated
Function key (global)	Release Press
Function key (local)	Release Press
Screen	Loaded Cleared
Screen object	Press Release Click Change (or toggle for switch) Switch on Switch off Enable Disable
Scheduler	Time expired

See also

Availability for specific devices of system functions (Page 4612)

ImportDataRecordsWithChecksum

Description

Imports one or all data records of a recipe from a CSV file with a checksum and verifies the checksum.

Use in the function list

ImportDataRecordsWithChecksum (File name, Data record number/name, Overwrite, Output status message, Processing status, verify checksum)

Use in user-defined functions

ImportDataRecordsWithChecksum (File_name, Data_record number/name, Overwrite, Output_status_message, Processing_status, Verify_Checksum)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

File name

Name of the CSV file from which the recipe data records are imported. Enter the path and the file extension, for example, "C:\TEMP\Orange.CSV".

If a storage card is used as storage medium, specify the storage location as follows: "\StorageCard\<File name>".

If you specify a directory instead of an individual CSV file, all files in the specified directory will be imported.

Data record number/name

Number or name of the recipe data record to be imported. Specify "0" if all recipe data records are to be imported.

Overwrite

Determines whether existing recipe data records are to be overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data records are not overwritten. The import process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data records are overwritten without prompting.

2 (hmiOverwriteWithConfirmation) = With confirmation: Recipe data records are not overwritten until confirmed.

Output status message

Determines whether a status message is output after the import:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example to delay execution of other system functions, until this system function has been successfully completed.

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Verify checksum

Determines if the checksum should be verified during import:

0 (hmiFalse) = No: Checksum is not verified.

1 (hmiTrue) = Yes: Checksum is verified.

See also

Availability for specific devices of system functions (Page 4612)

IncreaseTag

Description

Adds the given value to the value of the tags.

X = X + a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, help tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

IncreaseTag (Tag, Value)

Use in user-defined functions

IncreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the given value is added.

Value

The value that is added.

See also

Availability for specific devices of system functions (Page 4612)

InverseLinearScaling

Description

Assigns a value to the tag X, which is calculated from the value of the given tag Y using the linear function X = (Y - b) / a.

The tags X and Y must not be identical. This system function is the inverse of the "LinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

InvertLinearScaling (X, Y, b, a)

Use in user-defined functions

InvertLinearScaling (X, Y, b, a)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

X

The tag which is assigned the value calculated from the linear equation.

Y

The tag that contains the value used for calculation.

b

The value which is subtracted.

a

The value through which is divided.

Example

The following program code assigns a value to the varX tag by means of the InverseLinearScaling function.

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
 InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n, varX);
...
}
```

The saved return value can be processed in the following code.

See also

Availability for specific devices of system functions (Page 4612)

InvertBit

Description

Inverts the value of the given tag of the "Bool" type:

- If the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

Use in the function list

InvertBit (Tag)

Use in user-defined functions

InvertBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag whose bit is set.

Example

The following program code inverts the value of the boolean tag b_value and outputs the result along with the original b_saved value.

```
{
BOOL b_value = 0;
BOOL b_saved = b_value;

//Invert variable
invertBit(b_value);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",b_value, b_saved);
  ...
}
```

See also

Availability for specific devices of system functions (Page 4612)

InvertBitInTag

Description

Inverts a bit in the given tag:

- If the bit in the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the bit in the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "InvertBit" system function instead.

Use in the function list

InvertBitInTag (Tag, Bit)

Use in user-defined functions

InvertBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which the given bit is set.

Bit

The number of the bit that is set.

When this system function is used in a user-defined function, the bits in a tag are counted from right to left. The counting begins with 0.

Example

The following program code inverts a bit at the specified bitposition in the bvalue tag and outputs the result along with the original bsaved value.

```
{
BYTE bvalue;
BYTE bsaved = bvalue;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bvalue, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bvalue, bsaved);
  ...
}
```

See also

Availability for specific devices of system functions (Page 4612)

LinearScaling

Description

Assigns a value to the tag Y, which is calculated from the value of the given tag X using the linear function Y= (a *X) + b.

The inverse of this function is the "InvertLinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

LinearScaling (Y, a, X, b)

Use in user-defined functions

LinearScaling (Y, a, X, b)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Y

The tag which is assigned the value calculated from the linear equation.

a

The value with which is multiplied.

X

The tag that contains the value used for calculation.

b

The value that is added.

Example

The following program code uses the LinearScaling function to assign a value to the Yvar tag.

```
{
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
 LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n, Yvar);
...
}
```

The saved return value can be processed in the following code.

See also

Availability for specific devices of system functions (Page 4612)

LoadDataRecord

Description

Loads the given recipe data record from the memory medium of the HMI device in the recipe tags. This system function is used, for example, to display a recipe data record in the recipe screen.

Activate the "Synchronize recipe view and recipe tags" option in the synchronization settings of the recipe. If this option is deactivated, the system function has no effect.

Use in the function list

LoadDataRecord (Recipe number/name, Data record number/name, Processing status)

Use in user-defined functions

LoadDataRecord (Recipe_number/name, Data_record_number/name, Confirmation, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which a recipe data record is loaded.

Data record number/name

Number or name of the recipe data record to be loaded.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

LogOff

Description

Logs off the current user on the HMI device.

Use in the function list

Logoff

Use in user-defined functions

Logoff

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

Logon

Description

Logs on the current user on the HMI device.

Use in the function list

Logon (Password, User name)

Use in user-defined functions

Logon (Password, User_name)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Password

The tag from which the password for the user logging on is read.

If the user is logged on, the password in the tag is deleted.

User name

The tag from which the user name for the user logging on is read.

See also

Availability for specific devices of system functions (Page 4612)

LookupText

Description

Identifies an entry from a text list. The result depends on the value and on the selected runtime language. The result is saved to a tag of data type "String".

Use in the function list

LookupText (Output text, Value, Language, Text list)

Use in user-defined functions

LookupText (Output_text, Index, Language, Text_list)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Output text

The tag to which the result is written.

Value

The tag that defines the value of the list entry.

Language

Defines the runtime language in which the list entry is identified.

- Runtime language
  Language code as per VBScript reference, for example, "de-DE" for German (Germany) or "en-US" for English (United States of America). The selection depends on the activated runtime languages.
- Tag
  The tag that contains the language. Enter the runtime language as decimal value of the country ID, for example, 1031 for German - Standard, 1033 for English - USA. Details are available in the VBScript basics, "Locale identifier (LCID) diagram".
- Integer
  The number that corresponds to the order of runtime languages for language switching. The selection depends on the activated runtime languages, for example, "0" for the language that appears when you first start runtime. You can find more information under the topic of "Languages in Runtime".

Text list

Defines the text list. The list entry is read from the text list.

See also

Availability for specific devices of system functions (Page 4612)

NotifyUserAction

Description

This system function is used to log user actions that are not automatically logged in the audit trail. You can also use this system function to require the user to enter an acknowledgment or an electronic signature and a comment for the operator action. Requirement for the use of the system function is that the GMP-compliant configuration is activated under "Runtime settings".

If you use the "NotifyUserAction" system function in a function, the debugger may open if you cancel your input by clicking "Cancel". To compensate for this reaction, you can use the "On Error Resume Next" statement in a function. With this instruction, the next instruction is executed after a runtime error. If you use the "On Error Resume Next" statement, output of system events is also suppressed.

Use in the function list

NotifyUserAction (Confirmation type, Mandatory commenting, Category, Object name, Description)

Use in user-defined functions

NotifyUserAction (Confirmation_type, Mandatory_commenting, Category, Object_name, Description)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Confirmation type

Establishes how the action must be confirmed

0 = (None): No confirmation required, an entry is created in the audit trail

1 = (Acknowledgement): Acknowledgment, the user must acknowledge the action; an entry is created in the audit trail

2 = (Digital Signature): Electronic signature, a dialog window opens in which the user must enter his electronic signature - an entry is created in the audit trail

Mandatory commenting

Establishes if the user has to enter a comment. The comment is logged in the audit trail.

0 = (True): True; a dialog window opens in which the user must enter a comment

1 = (False): False; no mandatory commenting

Category

Category or class name of the modified object

Object name

Name of the modified object

Description

Text which describes the archiving user action.

See also

Availability for specific devices of system functions (Page 4612)

OpenAllLogs

Description

Reestablishes the connection between WinCC and the logs. Logging can be continued.

Note

Run the "StartLogging" system function in order to restart logging.

Use in the function list

OpenAllLogs

Use in user-defined functions

OpenAllLogs

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "StopLogging" and "CloseAllLogs" system functions on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When the button "Close Archive" is pressed, the specified log is stopped and all open logs are closed. The data medium can be changed. You open all logs with the "Open Archive" button. Logging will be continued in the specified log.

See also

Availability for specific devices of system functions (Page 4612)

OpenCommandPrompt

Description

Opens a Windows system prompt.

This function is used, e.g., to copy files or to call up another application.

Use in the function list

OpenCommandPrompt

Use in user-defined functions

OpenCommandPrompt

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

OpenControlPanel

Description

Opens a dialog that you can use to edit selected Control Panel settings.

This system function lets you set the following on the HMI device:

- Properties and value of the IP address
- User identification on the network
- WinCC Internet settings

Note

Project security

The "OpenControlPanelDialog" system function is used to bypass the SecureMode on the HMI device. Take the corresponding precautions to ensure the security of your project.

Use in the function list

OpenControlPanelDialog (Dialog)

Use in user-defined functions

-

Parameters

Dialog

Sets the Control Panel dialog to be opened.

- PROFINET_X1: Setting of the IP address and Ethernet parameters
- PROFINET_X3: Setting of the IP address and Ethernet parameters; only for Comfort Panel KP 1500, TP 1500; TP1900, TP2200
- WinCC Internet settings: Setting for Web Server, e-mail notification, provided the HMI device supports these functions
- Network ID: Setting for identification on the network, provided the HMI device supports these functions

See also

Availability for specific devices of system functions (Page 4612)

OpenInternetExplorer

Description

Opens the Internet Explorer on the HMI device.

If the Internet Explorer is already open, it will be closed and reopened when you call the system function.

Note

The Internet Explorer saves data temporarily in the DRAM file system of the HMI device, e.g. the last web sites that were be called up.

This data can be saved using the "BackupRAMFileSystem" system function so that it is still available when the HMI device is restarted.

Use in the function list

OpenInternetExplorer (Start page)

Use in user-defined functions

OpenInternetExplorer (Start_page)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Start page

The page which is loaded when Internet Explorer is started, e.g. "www.siemens.com".

See also

Availability for specific devices of system functions (Page 4612)

OpenScreenKeyboard

Description

Hides or shows the screen keyboard.

The screen keyboard remains open until the screen keyboard is explicitly closed. In this way, the screen keyboard can also be used in other applications.

Use in the function list

OpenScreenKeyboard (Layout)

Use in user-defined functions

OpenScreenKeyboard (Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Layout

Specifies whether the window is opened minimized or maximized with the screen keyboard:

0 (hmiScreenKeyboardMinimized) = Minimized

1 (hmiScreenKeyboardMaximized) = Maximized

See also

Availability for specific devices of system functions (Page 4612)

OpenTaskManager

Description

Shows the task manager.

The task manager allows changing to other open applications on the HMI device.

Note

The appearance of the task manager depends on the operating system installed.

Use in the function list

OpenTaskManager

Use in user-defined functions

OpenTaskManager

Can be used if the configured device supports user-defined scripts. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

PageDown

Description

Executes the key function <Pagedown> on the HMI device.

This system function can only be used for function keys.

Use in the function list

PageDown

Use in user-defined functions

PageDown

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

-

See also

Availability for specific devices of system functions (Page 4612)

PageUp

Description

Executes the key function <PageUp> on the HMI device.

This system function can only be used for function keys and tasks with a time trigger.

Use in the function list

PageUp

Use in user-defined functions

PageUp

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

-

See also

Availability for specific devices of system functions (Page 4612)

PrintReport

Description

Prints the given report from the printer which is connected to the HMI device. The report is printed in the language which is set on the HMI device.

Note

If runtime is closed whilst log data are being printed using the system function, the report will cease to be supplied with data as soon as runtime stops.

Use in the function list

PrintReport (Report)

Use in user-defined functions

PrintReport (Report)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Report

Name of the report to be printed.

Note

If you have set up via the "Function list" dialog box a new report for the "PrintReport" function, when compiling, the following warning appears: "The report "Report_1" has no printed pages."

In order to remedy the warning, open the "Report_1" via the project view and recompile the project.

See also

Availability for specific devices of system functions (Page 4612)

PrintScreen

Description

Prints the screen currently being displayed on the HMI device from the printer which is connected to the HMI device.

Opened windows are also printed.

Note

The printer settings are taken over from the current settings in the Windows operation system.

Use in the function list

PrintScreen

Use in user-defined functions

PrintScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Availability for specific devices of system functions (Page 4612)

ResetBit

Description

Sets the value of a "Bool" type tag to 0 (FALSE).

Use in the function list

ResetBit (Tag)

Use in user-defined functions

ResetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 0 (FALSE).

Example

The following program code resets the value of the boolean tag b_value to 0 by means of the ResetBit function and outputs the result along with the original b_saved value.

```
{
BOOL b_value = 1;
BOOL b_saved = b_value;

//Reset bit
  ResetBit (b_value);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",b_value, b_saved);
...
}
```

See also

Availability for specific devices of system functions (Page 4612)

ResetBitInTag

Description

Sets a bit in the specified tag to 0 (FALSE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "ResetBit" system function instead.

Use in the function list

ResetBitInTag (Tag, Bit)

Use in user-defined functions

ResetBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 0 (FALSE).

Bit

The number of the bit that is set to 0 (FALSE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Example

The following program code sets a bit at the specified bitposition in the bvalue tag to 0 and outputs the result along with the original bsaved value.

```
{
BYTE bvalue;
BYTE bsaved = bvalue;
BYTE bitposition = 2;

//Reset bit in bitposition
ResetBitInTag (bvalue, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bvalue, bsaved);
  ...
}
```

See also

Availability for specific devices of system functions (Page 4612)

SafelyRemoveHardware

Description

Checks whether there is read or write access to the external storage medium. If there is no access, the external storage medium can be removed without data loss.

Use in the function list

SafelyRemoveHardware(Path, Result)

Use in user-defined functions

SafelyRemoveHardware(Path, Result)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Path

Path of storage medium , e.g. \Storage Card USB\

Result

The tag in which the result is entered.

TRUE : The storage medium can be removed safely. A corresponding system alarm will be output.

FALSE : The storage medium cannot be removed safely. A corresponding system alarm will be output.

See also

Availability for specific devices of system functions (Page 4612)

SaveDataRecord

Description

Saves the current values of the recipe tags as data record to the memory medium of the HMI device.

This system function is used, for example, to save a recipe data record in the recipe screen.

Use in the function list

SaveDataRecord (Recipe number/name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

SaveDataRecord (Recipe_number/name, Data_record_number/name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Recipe number/name

Number or name of the recipe to which a recipe data record is saved.

Data record number/name

Number or name of the recipe data record to be saved. A new data record will be created if no record of this name or number was found in the recipe, independent of the value at the "Overwrite" parameter.

Overwrite

Specifies whether an existing data record is overwritten:

0 (hmiOverwriteForbidden) = No: The recipe data record is not overwritten, the data record is not saved.

1 (hmiOverwriteAlways) = Yes: The recipe data record is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithConfirmation) = With confirmation: The recipe data record is overwritten only with confirmation by the user.

Output status message

Determines whether a status message is output after saving:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

SendEMail

Description

Sends an e-mail from the HMI device to the given addressee.

This system function is used, for example, when, in the case of service, the alarm is to be passed on directly to the service technician.

Note

To send alarms as e-mails, the HMI system must have an e-mail client at its disposal.

Use in the function list

SendEMail (Address, Subject, Text, Return address)

Use in user-defined functions

SendEMail (Address, Subject, Text, Return_address)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Address

The e-mail address of the addressee.

Subject

The subject line of the e-mail.

Text

The text sent in the e-mail.

Return address

The e-mail address to which the addressee of this e-mail should send the reply.

See also

Availability for specific devices of system functions (Page 4612)

SetAcousticSignal

Description

Configures the acoustic feedback of touch screen operation on the HMI device.

Note

The configuration that was set at Switch off is reestablished when restarting the HMI device.

Use in the function list

SetAcousticSignal (Volume)

Use in user-defined functions

SetAcousticSignal (Volume)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Volume

Determines whether and how loud an acoustic signal is emitted:

-1 (hmiToggle) = Toggle: Toggles the emission of the acoustic signal as follows: Muted > Quiet > Loud.

0 (hmiMuted) = Mute: no acoustic signal

1 (hmiQuiet) = Quiet: quiet acoustic signal

2 (hmiLoud) = Loud: loud acoustic signal

See also

Availability for specific devices of system functions (Page 4612)

SetAlarmReportMode

Description

Switches the automatic reporting of alarms on the printer on or off.

Use in the function list

SetAlarmReportMode (Mode)

Use in user-defined functions

SetAlarmReportMode (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether alarms are reported automatically on the printer:

0 (hmiDisablePrinting) = Reporting off: Alarms are not printed automatically.

1 (hmiEnablePrinting) = Reporting On: Alarms are printed automatically.

-1 (hmiToggle) = Toggle: Toggles between the two modes.

See also

Availability for specific devices of system functions (Page 4612)

SetAndGetBrightness

Description

Determines the brightness of the MP 377 Touch daylight readable display. The brightness value can be interpreted as absolute or relative to the current value.

Note

The configuration that is set in the Control Panel will be reestablished when you restart the HMI device.

Use in the function list

SetAndGetBrightness (Brightness, Mode, Current value)

Use in user-defined functions

SetAndGetBrightness (Brightness, Mode, Actual brightness)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Brightness

New value for the brightness.

Mode

Specifies if the new brightness value is set as absolute or relative to the current value.

Current value

Tag in which the current brightness value is stored.

SetBit

Description

Sets the value of a "Bool" type tag to 1 (TRUE).

Use in the function list

SetBit (Tag)

Use in user-defined functions

SetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 1 (TRUE).

Example

The following program code sets the value of the boolean tag b_value to 1 by means of the SetBit function and outputs the result along with the original b_saved value.

```
{
BOOL b_value = 0;
BOOL b_saved = b_value;

//Set bit
  SetBit (b_value);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",b_value, b_saved);
...
}
```

See also

Availability for specific devices of system functions (Page 4612)

SetBitInTag

Description

Sets a bit in the given tag to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitInTag (Tag, Bit)

Use in user-defined functions

SetBitInTag(Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 1 (TRUE).

Bit

The number of the bit that is set to 1 (TRUE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an I/O field or assign the system function to a screen object, such as a button.

If you have configured a short event such as the activation of an alarm for the system function you can only access the actual process values by setting the tag for continuous reading.

Example

The following program code sets a bit to 1 at the specified bitposition in the bvalue tag and outputs the result along with the original bsaved value.

```
{
BYTE bvalue;
BYTE bsaved = bvalue;
BYTE bitposition = 2;

//Reset bit in bitposition
SetBitInTag (bvalue, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bvalue, bsaved);
  ...
}
```

See also

Availability for specific devices of system functions (Page 4612)

SetBrightness

Description

Determines the brightness of the display.

Note

The configuration that is set in the Control Panel / Start Center will be reestablished when you restart the HMI device.

For Basic Panels 2nd Generation, Mobile Panels and Comfort Panels:

The value for the system function "SetBrightness" can be set between 0% and 100%. The set value is transferred to the HMI device. The brightness settings on the HMI device can be viewed and edited in "Start Center > Settings > Display". The HMI devices support a brightness setting between 10% and 100%.

If the system function "SetBrightness" is assigned a value of 0%, the display of the HMI device is switched off by default in Runtime. If the operator touches the display, the display switches to the previous brightness setting.

If the system function "SetBrightness" is assigned a value between 1% and 10% and the operator opens the display settings in the Start Center, brightness is reset to 10%.

Use in the function list

SetBrightness (value)

Use in user-defined functions

SetBrightness (Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Value

New value for the brightness.

See also

Availability for specific devices of system functions (Page 4612)

SetConnectionMode

Description

Connects or disconnects the given connection.

Note

A connection to the PLC cannot be established until the operating mode ONLINE has been set on the HMI device. Use the "SetDeviceMode" system function for this purpose.

Use in the function list

SetConnectionMode (Mode, Connection)

Use in user-defined functions

SetConnectionMode (Mode, Connection)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether a connection to the PLC is established or disconnected:

0 (hmiOnline) = Online: Connection is established.

1 (hmiOffline) = Offline: Connection is disconnected.

Connection

The PLC to which the HMI device is connected. You specify the name of the PLC in the connection editor.

Multiple use of the system function in a user-defined function

If you use the "SetConnectionMode" system function for different connections, it may be possible that not all system functions are executed correctly. Proceed as follows to prevent this situation:

1. Create a "BOOL" type tag with the start value "0".
2. Configure the "SetConnectionMode" system function on the "Value change" event of the HMI tags. If you want to disconnect three connections, for example, you must configure the system function three times.
3. In the user-defined function, apply the "InvertBit" system function to the HMI tag.

Application example

Two typical application examples for this system function are as follows:

- Test
  As long as no PLC is connected to the HMI device, no error messages will be output during the test on the HMI device. If the HMI device is connected to a PLC, the connection to the PLC can be established by pressing a key.
- Commissioning
  Several PLCs are to be configured for a system. At first, all PLCs except one are configured "Offline". After commissioning of the first PLC, the connection to each of the other PLCs is established by pressing a key. In this way, the other PLCs are started up one after another.

See also

Availability for specific devices of system functions (Page 4612)

SetDataRecordTagsToPLC

Description

Transfers the values of the recipe tags to the PLC. The recipe tags contain the values of the data record which is displayed on the HMI device.

Use in the function list

SetDataRecordTagsToPLC (Recipe number/name, Processing status)

Use in user-defined functions

SetDataRecordTagsToPLC (Recipe_ number/name, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data record is transferred to the PLC.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

SetDataRecordToPLC

Description

Transfers the given recipe data record directly from the data medium of the HMI device to the PLC with which the HMI device is connected.

Note

The values of the recipe data record don't need to be displayed on the HMI device.

Use in the function list

SetDataRecordToPLC (Recipe number/name, Data record number/name, Output status message, Processing status)

Use in user-defined functions

SetDataRecordToPLC (Recipe_number/name, Data_record_number/name, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data record is transferred to the PLC.

Data record number/name

Number or name of the recipe data record to be transferred to the PLC.

Output status message

Determines whether a status message is output after the transfer:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

SetDaylightSavingTime

Description

The system function "SetDaylightSavingTime" changes the setting of the HMI device from daylight saving to standard time and vice versa.

Time settings will take place immediately following system function.

Note

The "SetDaylightSavingTime" system function does not support time zones without daylight saving time.

Note

Windows 7

The system function "SetDaylightSavingTime" is not supported for PC-based HMI devices under Windows 7.

Use in the function list

SetDaylightSavingTime(DaylightSavingTime)

Use in user-defined functions

SetDaylightSavingTime (Daylight_saving_time)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Daylight Saving Time

Determines whether Daylight Saving Time is set on the HMI device:

0 = Daylight Saving Time is not activated

1 = Daylight Saving Time is activated

See also

Availability for specific devices of system functions (Page 4612)

SetDeviceMode

Description

Toggles the operating mode on the HMI device. The following types of operation are possible: "Online", "Offline" and "Loading".

Use in the function list

SetDeviceMode (Operating mode)

Use in user-defined functions

SetDeviceMode (Operating_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Operating mode

Determines the operating mode of the HMI device:

0 (hmiOnline) = Online: The connection to the PLC is established. The configured connection status is always set in this process. The states that were last used in Runtime are not considered.

1 (hmiOffline) = Offline: The connection to the PLC is disconnected.

2 (hmiTransfer) = load: A project can be transferred from the configuration computer to the HMI device.

Note

If you use a PC as an HMI device, the runtime software will be exited when you switch operating mode after "Load".

See also

Availability for specific devices of system functions (Page 4612)

SetDisplayMode

Description

Changes the settings of the screen in which the runtime software runs.

The runtime software runs in full-screen mode as default. The Windows task switch is disabled.

Use in the function list

SetDisplayMode (Layout)

Use in user-defined functions

SetDisplayMode (Display mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines the settings for the screen in which the runtime software runs.

1 (hmiScreenFull): Full-screen: Title bar of the screen is not visible

2 (hmiScreenMaximized): Maximized

3 (hmiScreenRestore): Restore: The last used screen setting is used. This layout can only be used when the window is displayed minimized or maximized.

4 (hmiScreenMinimized): Minimized

5 (hmiScreenAutoAdjust): Automatic: The size of the window is set so that all screen objects in it will be visible.

6 (hmiScreenOnTop): Foreground; either the window appears in the foreground or the program icon associated with the window flashes on the taskbar depending on the Windows setting. The setting can be changed in the Windows configuration and applies to all Windows applications.

See also

Availability for specific devices of system functions (Page 4612)

SetLanguage

Description

Toggles the language on the HMI device. All configured text and system events are displayed on the HMI device in the newly set language.

Use in the function list

SetLanguage (Language)

Use in user-defined functions

SetLanguage (Language)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Language

Determines which language is set on the HMI device. The following specifications are possible:

- -1 (hmiToggle) = Toggle: Changes to the next language. The sequence is determined during configuration in the "Project languages" editor.
- Number you have defined under "Languages and fonts" in the "Runtime Settings" editor. Changes to the language with the given number.
- Language you have defined under "Languages and fonts" in the "Runtime Settings" editor.
- Language abbreviation in accordance with the VBScript5 reference: This changes to the language corresponding to the specified language code, e.g. "de-DE" for German (Germany) or "en-US" for English (United States).
  An overview of the language abbreviations is available in the basic information of VBScript under the topic "Area diagram-ID (LCID) Diagram".

See also

Availability for specific devices of system functions (Page 4612)

SetPLCDateTime

Description

Changes the data and the time of the linked PLC

The system function "SetPLCDateTime" can only be configured for the following PLCs:

- SIMATIC S7 1200
- SIMATIC S7 1500

Use in the function list

SetPLCDateTime (connection, time)

Use in user-defined functions

SetPLCDateTime (Connection, Time)

Parameters

Connection

Connection of PLC and HMI device.

Time

Transfers the date and the time of the HMI device to the PLC. The PLC applies the date and the time of the HMI device.

SetRecipeTags

Description

Changes the status of the recipe tags from "Online" to "Offline" and vice versa.

This system function is used, for example, when recipe data record values are fine tuned when starting up a machine.

Use in the function list

SetRecipeTags (Recipe number/name, Status, Output status message, Processing status)

Use in user-defined functions

SetRecipeTags (Recipe_number/name, Status, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe in which the recipe data record is saved.

Status

Determines the status of the recipe tags:

0 (hmiOnline) = Online: Value changes of the recipe tags are transferred immediately to the PLC connected to the HMI device.

1 (hmiOffline) = Offline: Value changes to the recipe tags are only transferred to the PLC connected to the HMI device when, for example, the "SetDataRecordTagsToPLC" system function is executed.

Output status message

Determines whether a status message is output after saving:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

See also

Availability for specific devices of system functions (Page 4612)

SetScreenKeyboardMode

Description

Enables or disables the automatic display of the screen keyboard on the HMI device.

This system function is also used to prevent the display of the screen keyboard, e.g. because an external keyboard is connected to the HMI device.

Note

To enable the "SetScreenKeyboardMode" ("SetScreenKeyboardMode") system function on an HMI other than a Touch Panel device, set the "Use on-screen keyboard" check box in the "Runtime settings" dialog of the device settings.

Use in the function list

SetScreenKeyboardMode (Mode)

Use in user-defined functions

SetScreenKeyboardMode (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether the screen keyboard is hidden or shown:

0 (hmiOff) = Off: Screen keyboard is hidden

1 (hmiOn) = On: Screen keyboard is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes.

See also

Availability for specific devices of system functions (Page 4612)

SetTag

Description

Assigns a new value to the given tag.

Note

This system function can be used to assign strings and numbers, depending on the type of tag.

Use in the function list

SetTag (Tag, Value)

Use in user-defined functions

SetTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag to which the given value is assigned.

Value

The value which the given tag is assigned.

Note

The "SetTag" system function is only executed after a connection has been established.

Example

The following program code uses the SetTag function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag.

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
bvalue = GetTagBit("gs_tag_bit");
  printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

The saved return value can be processed in the following code.

See also

Availability for specific devices of system functions (Page 4612)

ShiftAndMask

Description

This system function converts the input bit pattern of the source tags into an output bit pattern of the target tags. This involves bit shifting and masking.

Note

If the source and target tag have a different number of bits, using the system function in the target tag can result in a violation of the value range.

Use in the function list

ShiftAndMask (Source tag, Target tag, Bits to shift, Bits to mask)

Use in user-defined functions

ShiftAndMask (Source_tag, Target_tag, Bits_to_shift, Bits_to_mask)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Source tag

The tag includes the input bit pattern. Integer-type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The actual value 72 is set at the 16-bit integer source tag: 0000000001001000.

Target tag

The output bit pattern is saved in the tag. Integer type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The shifted input bit pattern is multiplied by the bit mask, with bit-by-bit logical AND operation: 0000000000001001. The resultant decimal value "8" is saved to the target tag.

Please note the following:

- The source and target tags have the same number of bits.
- The number of bits to shift is less than the number of bits in the source tag and target tag.
- Bits to mask does not have more bits than the source tag and the target tag.

Bits to shift

Number of bits by which the input bit pattern is shifted right. A negative value shifts the input bit pattern to the left.

Example: "Bits to shift" has the value "+3". The input bit pattern is shifted right by three bits when the system function is called: 0000000000001001.
Bits to the left are padded with "0". Three bits are truncated on the right. The new decimal value is "9".

Note

The left bit is "1" in a source tag of the data type with negative signed integer. This sign bit is padded with "0" when the bits are shifted right. The sign changes to "+".

Bits to mask

An integer serves as bit mask. The bit pattern is used to multiply the shifted input bit pattern.
Example: Integer "2478" with the bit pattern "0000100110101110".

You can enter the bit mask in three different ways:

- Hexadecimal: First enter the prefix "0h" or "0H", followed by an optional space for better readability. Then group the bit pattern in blocks of four (0000)(1001)(1010)(1110) and set each block in hexadecimal code: (0)(9)(A)(E). Only the characters 0-9, A-F, a-f are allowed: "0h 09AE".
- Binary: First enter the prefix "0b" or "0B", followed by an optional space for better readability. Then group the binary bit pattern into blocks of four 0000 1001 1010 1110 with spaces in between as a check. Only the characters "0" or "1" are allowed: "0b 0000 1001 1010 1110".
- Decimal: Enter the value "2478" directly, without a prefix.

See also

Availability for specific devices of system functions (Page 4612)

ShowAlarmWindow

Description

Hides or shows the alarm window on the HMI device.

Use in the function list

ShowAlarmWindow (Object name, Layout)

Use in user-defined functions

ShowAlarmWindow (Object_name, Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Object name

Name of the alarm view which is hidden or shown.

Layout

Determines whether the alarm window is hidden or shown:

0 (hmiOff) = Off: Alarm view is hidden

1 (hmiOn) = On: Alarm view is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

See also

Availability for specific devices of system functions (Page 4612)

ShowOperatorNotes

Application

Displays the tooltip configured for the selected object.

If the system function is configured on a function key, the tooltip for the screen object that currently has the focus is displayed. If a tooltip is configured for the screen itself, you can switch to this text by pressing <Enter> or by double-clicking on the help window.

If the system function is configured on a button, only the tooltip for the current screen is displayed. If a tooltip is configured on the button itself, initially only the tooltip for the button is displayed. You can press <Enter> or double-click on the help window to switch to the tooltip for the current screen.

Note

No other screen object can be used while the help window is open. To use the screen object, close the help window.

Closing the help window

You can close the help window in the following ways:

Using the keys:

- By pressing the <HELP> key again
- By pressing the <ESC> key

Using the touch screen:

- By pressing the ❌ button

Use in the function list

ShowOperatorNotes (Layout)

Use in user-defined functions

ShowOperatorNotes (Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines whether the configured tooltip is hidden or shown:

0 (hmiOff) = Off: Configured tooltip is hidden

1 (hmiOn) = On: Configured tooltip is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

See also

Availability for specific devices of system functions (Page 4612)

ShowPopUpScreen

Description

Opens the pop-up-screen, for example, when a button is pressed.

You can enter a constant value or assign a tag as coordinates.
If the configured pop-up screen is not visible or only partially visible, the coordinates are set to 0.0.

Use in the function list

ShowPopupScreen (Screen object)

Use in user-defined functions

ShowPopupScreen (Name_of_the_screen, Coordinate_X, Coordinate_Y, Layout)

Parameters

Screen name

Specifies the name of the screen that appears in Runtime when a button is pressed.

X coordinate

Position of the screen in the current screen on the X axis

Y coordinate

Position of the screen in the current screen on the Y axis

Layout

Specifies the mode for the slide-in screen:

Switching

Off

On

ShowSlideInScreen

Description

Calls the slide-in screen, for example, when operating a button.

Use in the function list

ShowSlideinScreen (screen name, mode)

Use in user-defined functions

ShowSlideInScreen (SlideInScreen_name, Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Specifies the slide-in screen that appears in Runtime when a button is pressed:

Slide-in screen top

Slide-in screen bottom

Slide-in screen left

Slide-in screen right

Mode

Specifies the mode for the slide-in screen:

Switching

Off

On

ShowSoftwareVersion

Description

Hides or shows the version number of the runtime software.

Use this system function if during servicing, for example, you required the version of the runtime software used.

Use in the function list

ShowSoftwareVersion (Layout)

Use in user-defined functions

ShowSoftwareVersion (Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Layout

Determines whether the version number is shown:

0 (hmiOff) = Off: Version number is not shown

1 (hmiOn) = On: Version number is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

See also

Availability for specific devices of system functions (Page 4612)

ShowSystemAlarm

Description

Displays the value of the parameter passed as a system event to the HMI device.

Use in the function list

ShowSystemAlarm (Text/value)

Use in user-defined functions

ShowSystemAlarm (Text/value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Text/Value

The text or the value, which was output as a system alarm.

See also

Availability for specific devices of system functions (Page 4612)

ShowSystemDiagnosticsWindow

Description

Hides or shows the system diagnostic window on the HMI device. The system diagnostic view is only available in the global screen for Comfort Panels and for WinCC Runtime Advanced.

Use in the function list

ShowSystemDiagnosticsWindow (Screen object)

Use in user-defined functions

ShowSystemDiagnosticsWindow (Target_Object_name)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen object

Name of the system diagnostic window which is hidden or shown.

See also

Availability for specific devices of system functions (Page 4612)

StartLogging

Description

Starts the logging of data or alarms in the specified log. The function can also be applied to audit trails.

You can interrupt logging at runtime using the "StopLogging" system function.

Use in the function list

StartLogging (Log type, Log)

Use in user-defined functions

StartLogging (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail

Log

Name of the log which is started.

See also

Availability for specific devices of system functions (Page 4612)

StartNextLog

Description

Stops the logging of data or alarms for the given log.

Logging is continued in the next log of the segmented circular log you configured for the specified log.

If you did not configure a segmented circular log for the specified log, the system function has no effect.

Use in the function list

StartNextLog (Log type, Log)

Use in user-defined functions

StartNextLog (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

Log

Name of the log for which the logging is stopped and continued in the next log.

See also

Availability for specific devices of system functions (Page 4612)

StartProgram

Description

Starts the specified program on the HMI device.

The runtime software continues running in the background. Alarms continue to be output and data continues to be updated.

When the given application is exited, the screen which was active during the performance of the system function is displayed on the HMI device.

This system function is used, for example, to edit recipe data records in MS Excel on the HMI device.

Note

If Windows CE is installed on the HMI device, during the configuration it must be checked whether the desired application can be started with this system function.

This system function allows all applications to be started which can be started in the "Execute" dialog of Windows CE.

The application to be started must be installed on the HMI device.

Use in the function list

StartProgram (Program name, Program parameters, Layout, Wait for program to end)

Use in user-defined functions

StartProgram (Program_name, Program_parameters, Display_mode, Wait_for_program_to_end)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Program name

Name and path of the program which is started. Upper and lower case are taken into account in this parameter.

Note

If the path contains spaces, the program can only be started correctly if the path is specified in inverted commas, e.g. "C:\Program Files\START\start.exe".

Program parameters

The parameters you transfer at the start of the program, for example a file that is opened after the start of the program.

The description of the necessary parameters is found in the documentation of the program to be started.

Layout

Determines how the program window is displayed on the HMI device:

0 (hmiShowNormal) = Normal

1 (hmiShowMinimized) = Minimized

2 (hmiShowMaximized) = Maximized

3 (ShowMinimizedAndInactive) = Minimized and inactive

Wait for program to end

Determines whether there is a change back to the project after the called up program has ended:

0 (hmiNo) = No: No change to project

1 (hmiYes) = Yes: Change to project

Note

The "Wait for program to end" parameter is only available for Runtime Advanced and Panels.

See also

Availability for specific devices of system functions (Page 4612)

StopLogging

Description

Stops the logging of process values or alarms in the specified log. The function can also be applied to audit trails.

The "StartLogging" system function is used to resume logging at runtime.

Note

When logging is stopped, a connection between WinCC and the log files or log database still exists. Use the "CloseAllLogs" system function to disconnect this connection.

Use in the function list

StopLogging (Log type, Log)

Use in user-defined functions

StopLogging (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail

Log

Name of the log that is stopped.

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "StopLogging" and "CloseAllLogs" system functions on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When the button "Close Archive" is pressed, the specified log is stopped and all open logs are closed. The data medium can be changed. The "Open Archive" button opens all logs and continues logging in the specified log.

See also

Availability for specific devices of system functions (Page 4612)

StopRuntime

Description

Exits the runtime software and thereby the project running on the HMI device.

Use in the function list

StopRuntime (Mode)

Use in user-defined functions

StopRuntime (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Mode

Determines whether the operating system is shut down after exiting runtime.

0 (hmiStopRuntime) = Runtime: Operating system is not shut down

1 (hmiStopRuntimeAndOperatingSystem) = Runtime and operating system: The operating system is shut down (not possible with WinCE)

Example

The following program code shuts down Runtime and the operating system.

```
{

//Stop runtime and shutdown
StopRuntime (hmiStopRuntimeAndOperationSystem);

}
```

The saved return value can be processed in the following code.

See also

Availability for specific devices of system functions (Page 4612)

TerminatePROFIsafe

Description

Disconnects the PROFIsafe connection for fail-safe operation between a KTP Mobile Panel and the PLC.

After execution of the system function "TerminatePROFIsafe", the connector of the KTP Mobile Panel can be removed from the PLC without the system signaling an error.

Use in the function list

TerminatePROFIsafe

Use in user-defined functions

TerminatePROFIsafe

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

--

WinACMPGetVersion

Description

Reads out the value of the version number of WinAC MP.

Use in the function list

WinACMPGetVersion (Version, Action)

Use in user-defined functions

WinACMPGetVersion (Version, Action)

Can be used if the configured device supports user-defined functions. For more information, refer to Auto-Hotspot.

Parameters

Version

The tag that contains the value.

Action

Determines whether the version number is read out:

0 (SwitchOff) = Off: Version number is not read out.

1 (SwitchOn) = On: Version number is read out.

See also

Availability for specific devices of system functions (Page 4612)

WinACMPSetStartAtBoot

Description

Determines whether or not WinAC MP is started automatically after startup of the HMI device.

Use in the function list

WinACMPSetStart at boot(Start at boot)

Use in user-defined functions

WinACMPSetStartAtBoot (Start at boot)

Can be used if the configured device supports user-defined functions. For more information, refer to Auto-Hotspot.

Parameters

StartAtBoot

Defines whether WinAC MP is started automatically.

0 (StartAtBootOff) = Off: WinAC MP is not started on startup of the HMI device.

1 (StartAtBootOn) = On: WinAC MP is started automatically on startup of the HMI device.

See also

Availability for specific devices of system functions (Page 4612)

WinACSetStartMode

Description

Sets the operating mode of WinAC MP after startup of the HMI device.

Use in the function list

WinACSetStartMode (Autostart)

Use in user-defined functions

WinACSetStartMode (Autostart)

Can be used if the configured device supports user-defined functions. For more information, refer to Auto-Hotspot.

Parameters

Action

Defines whether the Autostart function of WinAC MP is activated.

0 (AutoStartOff) = Off: After startup, WinAC MP remains in the STOP operating mode.

1 (AutoStartOn) = On: After startup, WinAC MP changes to the operating mode it was in before it was closed.

See also

Availability for specific devices of system functions (Page 4612)

VBScript for Windows

VBScript for Windows

VBScript

If you have worked with Visual Basic or Visual Basic for applications, then VBScript will seem familiar to you. If you do not know Visual Basic and are getting familiar with it, then you will learn all Visual Basic programming languages at once. The step-by-step guides from Microsoft Press are a good introduction to programming.

You will find fundamental information on VBScript language elements on the Microsoft homepage:

http://msdn.microsoft.com/en-us/library/t0aew7h6.aspx

Local ID (LCID)

An overview of all language codes can be found on the Microsoft homepage:

http://msdn.microsoft.com/en-us/goglobal/bb964664

VBScript for Windows CE

VBScript for Windows CE

Attr

Function

This property of the File control returns a number indicating the file mode that was used to open the file.

Syntax

file.Attr

Parameters

File

Reference to a File control.

Return values

The return values listed in the following table indicate the file access mode. If the return value is 0, the file is closed.

Constant	Value
None	0
fsModeInput	1
fsModeOutput	2
fsModeRandom	4
fsModeAppend	8
fsModeBinary	32

Remarks

The Attr property is read-only. Use the Open method of the File control to set the file mode.

Close

Function

This method closes an open File control.

Syntax

file.Close

Parameters

File

Name of a File control.

Return Values

None.

Remarks

Use the Open method to open a file.

CreateObject

Function

This function creates a reference to an Automation object.

Syntax

CreateObject (Object)

Parameters

Object

A string containing the ProgID of the object to create.

Return values

Returns a reference to an Automation object.

Remarks

Use CreateObject to create non-visible ActiveX controls at runtime. You cannot use CreateObject to create graphical objects such as a TreeView control or a ListView control. CreateObject produces objects that cannot respond to events. To produce objects that can respond to events, use the CreateObjectWithEvents function. The following table lists the ProgIDs for the ActiveX controls without events.

Control	ProgID
Microsoft CE File control 6.0	.file
Microsoft CE FileSystem control 6.0	.filesystem
Microsoft CE ImageList control 6.0	CEimageList.imagelistctrl

Example

```
Dim f, fwModeAppend
Set f = CreateObject("FileCtl.File")
fwModeAppend=8
f.Open "\Storage Card\testfile.txt", fwModeAppend
f.Close
```

Dir

Function

This method returns the name of a file, directory, or folder that matches a specified pattern or file attribute.

Syntax

File.Dir (Pathname,[Attributes])

Parameters

File

Reference to a FileSystem control.

Pathname

Optional. String expression that specifies a file name or path.

Attributes

Optional. Numeric expression whose sum specifies file attributes. If omitted, all files that match pathname are returned.

The following table describes the parameter settings of attributes.

Constant	Value	Description
fsAttrNormal	0	Normal
fsAttrReadOnly	1	Read only
fsAttrHidden	2	Hidden
fsAttrSystem	4	System file
fsAttrVolume	8	Volume label. If specified, all other attributes are ignored.
fsAttrDirectory	16	Directory or folder
fsAttrArchive	32	Logs

Return values

String. File name that matches pathname and attributes. Dir returns a zero-length string ("") if pathname is not found.

Remarks

Dir supports the use of multiple-character (*) and single-character (?) wildcards to specify multiple files. You must specify pathname the first time you call the Dir method. In addition, if you specify file attributes you must include pathname.
The Dir method returns the first file name that matches pathname. To get any additional file names that match pathname, call Dir again with no parameters. When no more file names match, Dir returns a zero-length string (" "). Once a zero-length string is returned, you must specify pathname in subsequent calls.

EOF

Function

This property returns True when the end of a file opened for random or sequential input is reached.

Syntax

File.EOF

Parameters

File

Reference to a File control.

Remarks

Use the EOF property to avoid the error generated by attempting to read past the end of a file.

The EOF property returns False until the end of the file has been reached. For files opened with a fsModeRandom or fsModeBinary file mode, EOF returns False until the last executed Get statement is unable to read an entire record.

For files opened with a fsModeBinary file mode, an attempt to read through the file using the Input function until EOF returns True generates an error. Use the LOF and LOC properties instead of EOF when reading binary files with Input, or use Get when using the EOF property. For files opened with a fsModeOutput file mode, EOF always returns True.

FileCopy

Function

This method copies an existing file to a new file.

Syntax

Filesystem.FileCopy PathName, NewPathName

Parameters

Filesystem

Reference to a FileSystem object.

PathName

String that contains the path and file name.

NewPathName

String that contains the file name and path of the new file.

Return Values

None.

Remarks

FileCopy returns an error if the new file does not exist.

FileLen

Function

This method returns a value specifying the length, in bytes, of a file.

Syntax

Filesystem.FileLen(pathname)

Parameters

Filesystem

Reference to a FileSystem control.

Pathname

Required. String expression that specifies a file. The pathname can include a directory or folder.

Return Values

Returns the number of bytes in a file.

Remarks

If the specified file is open when the FileLen method is called, the value returned represents the size of the file immediately before it was opened.

FileDateTime

Function

This method returns a variant (Date) that indicates the date and time when a file was created or last modified.

Syntax

filesystem.FileDateTime(pathname)

Parameters

Filesystem

Reference to a FileSystem control.

Pathname

Required. String expression that specifies a file name. The pathname can include a directory or folder.

Return Values

Returns the date the file was last modified.

Remarks

FileDateTime returns an error if the new file does not exist.

Get

Function

This method reads data from an open disk file into a variable.

Syntax

file.Get Data, [Recnumber]

Parameters

File

Reference to a File control.

Data

Required. Variant variable into which data is read.

Recnumber

Optional. Variant. Record number at which reading begins. For files opened in binary mode, Recnumber specifies the byte position.

Return Values

None

Remarks

Data read with the Get method usually is written to a file with the Put method. The first record or byte in a file is at position 1, the second record or byte is at position 2, and so on. If you omit Recnumber, the next record or byte following the last Get or Put method (or pointed to by the last Seek function) is read.

For files opened in Random mode, the following rules apply:

- If the length of the data being read is less than the length specified in the Len clause of the Open method, Get reads subsequent records on record-length boundaries. The space between the end of one record and the beginning of the next record is padded with the existing contents of the file buffer. Because the amount of padding data cannot be determined with any certainty, it is generally advisable to match the record length with the length of the data being read.
- If Data is a Variant of numeric type, Get reads 2 bytes identifying the VarType of the Variant and then reads the data that goes into the variable. For example, when reading a Variant of VarType 3, Get reads 6 bytes: 2 bytes identifying the Variant as VarType 3 (Long) and 4 bytes containing the Long data. The record length specified by the Len clause in the Open method must be at least 2 bytes greater than the actual number of bytes required to store the variable.
- You can use the Get method to read a Variant array from a disk, but you cannot use Get to read a scalar Variant containing an array. You also cannot use Get to read objects from a disk.
- If the variable being read into is a Variant of VarType 8 (String), Get reads 2 bytes identifying the VarType and 2 bytes indicating the length of the string. Then it reads the string data. The record length specified by the Len clause in the Open method must be at least 4 bytes greater than the actual length of the string.
- If the variable being read into is a dynamic array, Get reads a descriptor whose length equals 2 plus 8 times the number of dimensions, that is, 2 + 8 * NumberOfDimensions. The record length specified by the Len clause in the Open method must be greater than or equal to the sum of all the bytes required to read the array data and the array descriptor.

For files opened in Binary mode, the Len clause in the Open method has no effect. Get reads all variables from a disk contiguously; that is, with no padding between records.

GetAttr

Function

This method returns a number representing the attributes of a file, directory, or folder.

Syntax

filesystem.GetAttr(pathname)

Parameters

File system

Reference to a FileSystem control.

Pathname

Required. String expression that specifies a file name or directory or a folder name. The pathname can include the directory or folder.

Return values

Sum of attribute values. The following table shows the sums that can be returned.

Constant	Value	Description
vbNormal	0	Normal
VbReadOnly	1	Read only
VbHidden	2	Hidden
VbSystem	4	System
VbDirectory	16	Directory or folder
VbArchive	32	File has changed since last backup

Remarks

To determine which attributes are set, use the And operator to perform a bitwise comparison of the value returned by the GetAttr method and the value of the individual file attribute you want. If the result is not zero, that attribute is set for the named file.

Input

Function

This method returns a string containing characters from a file opened in Input or Binary mode.

Syntax

file.Input(number)

Parameters

File

Reference to a File control.

Number

Any valid numeric expression that specifies the number of characters to return.

Return Values

String containing characters read from file.

Remarks

Data read with the Input method usually is written to a file with the LinePrint or Put functions. Use this method only with files opened in Input or Binary mode.

Unlike the LineInputString method, the Input method returns all the characters it reads, including commas, carriage returns, line feeds, quotation marks, and leading spaces.

With files opened for Binary access, an attempt to read through the file using the Input method until the EOF function returns True generates an error. To avoid an error, use the LOF and Loc functions instead of EOF when reading binary files with the Input method or use Get when using the EOF function.

InputFields

Function

This method reads data from an open sequential file and returns a single dimension Variant array.

Syntax

file.InputFields(number)

Parameters

File

Reference to a File control.

Number

Number of comma-delimited fields to read from the file.

Return values

Array containing the fields read from the file.

Remarks

Data read with the InputFields method usually is written to a file with WriteFields. Use this method only with files opened in Input or Binary mode.

InputFields reads standard string or numeric data without modification. The following table shows how InputFields reads other input data.

Data	Value Assigned to Variable
Delimiting comma or blank line	Empty
#ZERO#	Zero
#TRUE# or #FALSE#	True or False
#yyyy-mm-dd hh:mm:ss#	The date and/or time represented by the expression

Double quotation marks ("") within input data are discarded.
If you reach the end of the file while you are inputting a data item, the input is terminated and an error occurs.

To correctly read data from a file into variables using InputFields, use the WriteFields method instead of the LinePrint method to write the data to the files. Using WriteFields ensures each separate data field is properly delimited.

InputB

Function

This method returns bytes from a file opened in Input or Binary mode.

Syntax

file.InputB(number)

Parameters

File

Reference to a File control.

Number

Any valid numeric expression that specifies the number of bytes to return.

Return Values

Array containing bytes read from file.

Remarks

Data read with the InputB method usually is written to a file with the LinePrint or Put functions. Use this method only with files opened in Input or Binary mode.

Kill

Function

This method deletes files from a disk.

Syntax

filesystem.Kill pathname

Parameters

Filesystem

Reference to a FileSystem control.

Pathname

Required. String expression that specifies one or more file names to be deleted. The pathname can include the directory or folder.

Return Values

None.

Remarks

The Kill method supports the use of multiple-character (*) and single-character (?) wildcards to specify multiple files.
An error occurs if you try to use Kill to delete an open file.

LineInputString

Function

This method reads a single line from an open sequential file and assigns it to a string variable.

Syntax

file.LineInputString

Parameters

File

Reference to a File control.

Return Values

None.

Remarks

Data read with LineInputString usually is written from a file with LinePrint.
The LineInputString method reads from a file one character at a time until it encounters a carriage return (Chr(13)) or carriage return/line feed (Chr(13) + Chr(10)) sequence. Carriage return/line feed sequences are skipped rather than appended to the character string.

LinePrint

Function

This method writes a single line to an open sequential file.

Syntax

file.LinePrint output

Parameters

File

Reference to a File control.

Output

String expression to write to a file.

Return Values

None.

Remarks

Data written with LinePrint is usually read from a file with LineInputString.
A carriage return/line feed (Chr(13) + Chr(10)) sequence is appended to the end of the string.

Loc

Function

This property returns a number specifying the current read/write position.

Syntax

file.Loc

Parameters

File

Reference to a File control.

Remarks

For files opened with the fsModeRandom file mode, Loc returns the number of the last record read or written. For files opened with all other modes, Loc returns the position of the last byte read or written.

LOF

Function

This property returns a number representing the size, in bytes, of a file.

Syntax

file.LOF

Parameters

File

Reference to a File control.

Remarks

The LOF property can be used with the Loc property to guarantee that a read operation does not continue past the end of a file.

MkDir

Function

This method creates a new directory.

Syntax

filesystem.MkDir PathName

Parameters

Filesystem

Reference to a FileSystem control.

Pathname

String expression that contains the directory name.

Return Values

None.

Remarks

MkDir generates an error if the directory already exists.

MoveFile

Function

This method renames an existing file or a directory, including all its subdirectories.

Syntax

filesystem.MoveFile PathName, NewPathName

Parameters

Filesystem

Reference to a FileSystem control.

PathName

String that contains the file name.

NewPathName

String that contains the file name to copy to.

Return Values

None.

Open

Function

This method opens a file in either the Input (1), Output (2), Random (4), Append (8), or Binary mode (32).

Syntax

file.Open pathname, mode, [access], [lock], [reclength]

Parameters

File

Reference to a File control.

Pathname

String expression that specifies a file name.

Mode

Specifies the file mode: Input (1), Output (2), Random (4) , Append (8), or Binary (32).

Access

Operation permitted on the open file: Read, Write, or ReadWrite [Default]. (1, 2, 3)

Lock

Operations permitted on the open file by other processes: Shared, LockRead, LockWrite [Default], and LockReadWrite. (1, 2, 3, 0)

Reclength

Number, in bytes, that is less than 32,767. For files opened for random access, this value is the record length. For sequential files, this value is the number of characters buffered.

Return Values

None.

Remarks

The reclength parameter is ignored if the mode is Binary. When opening a file in Random mode, you must specifiy a record size of greater than zero or an error will occur.

Put

Function

This method writes data from a variable to a disk file.

Syntax

file.Put data, [recnumber]

Parameters

Data

Required. Variant variable that contains data to be written to disk.

Recnumber

Optional. Variant (Long). Record number (Random mode files) or byte number (Binary mode files) at which writing begins.

Return Values

None.

Remarks

Data written with Put usually is read from a file with Get.
The first record or byte in a file is at position 1, the second record or byte is at position 2, and so on. If you omit recnumber, the next record or byte after the last Get or Put method or pointed to by the last Seek function is written.

For files opened in Random mode, the following rules apply:

- If the length of the data being written is less than the length specified in the Len clause of the Open method, Put writes subsequent records on record-length boundaries. The space between the end of one record and the beginning of the next record is padded with the existing contents of the file buffer. Because the amount of padding data cannot be determined with any certainty, it generally is a good idea to have the record length match the length of the data being written. If the length of the data being written is greater than the length specified in the Len clause of the Open method, an error occurs.
- If the variable being written is a Variant of a numeric type, Put writes 2 bytes identifying the VarType of the Variant and then writes the variable. For example, when writing a Variant of VarType 3, Put writes 6 bytes: 2 bytes identifying the Variant as VarType 3 (Long) and 4 bytes containing the Long data. The record length specified by the Len clause in the Open method must be at least 2 bytes greater than the actual number of bytes required to store the variable.

You can use the Put method to write a Variant array to disk, but you cannot use Put to write a scalar Variant containing an array to disk. You also cannot use Put to write objects to disk. If the variable being written is a Variant of VarType 8 (String), Put writes 2 bytes identifying the VarType and 2 bytes indicating the length of the string. It then writes the string data. The record length specified by the Len clause in the Open method must be at least 4 bytes greater than the actual length of the string.

If the variable being written is a dynamic array, Put writes a descriptor whose length equals 2 plus 8 times the number of dimensions, that is, 2 + 8 * NumberOfDimensions. The record length specified by the Len clause in the Open method must be greater than or equal to the sum of all the bytes required to write the array data and the array descriptor. For example, the following array declaration requires 118 bytes when the array is written to disk.

For files opened in Binary mode, the Len clause in the Open method has no effect. Put writes all variables to disk contiguously; that is, with no padding between records.

RmDir

Function

This method deletes an existing empty directory.

Syntax

filesystem.RmDir PathName

Parameters

Filesystem

Reference to a FileSystem control.

PathName

String that contains the directory name.

Return Values

None.

Remarks

The directory must be empty before it can be removed. You must specify a complete file path.

Seek

Function

This property returns and sets the next position in a file that will be read or written.

Syntax

file.Seek [= position]

Parameters

File

Reference to a File control.

Position

Numeric expression that specifies a position within a file.

Remarks

The Seek property specifies the next file position, whereas the Loc property specifies the current position. Seek always will be one more than Loc, except when a file is first opened and Seek and Loc are both 1.

Negative Seek or 0 causes an error.

SetAttr

Function

This method sets attribute data for a file.

Syntax

filesystem.SetAttr pathname, attributes

Parameters

File system

Reference to a FileSystem control.

Pathname

Required. String expression that specifies a file name.

Attributes

Required. Numeric expression whose sum specifies file attributes. The following table shows the parameter settings of attributes.

Constant	Value	Description
vbNormal	0	Normal (default)
vbReadOnly	1	Read only
vbHidden	2	Hidden
VbSystem	4	System file
VbArchive	32	File has changed since last backup

Return values

None.

Remarks

A run-time error occurs if you try to set the attributes of an open file.

WriteFields

Function

This method writes data to a sequential file.

Syntax

file.WriteFields [data]

Parameters

File

Reference to a File control.

Data

Variant or Variant array of numeric or string expressions to write to a file.

Prinzip

Return Values

None.

Remarks

Data written with WriteFields is usually read from a file with InputFields.
If you omit d ata, a blank line is printed to the file.

- Numeric data is always written using the period as the decimal separator.
- For Boolean data, either #TRUE# or #FALSE# is printed. The True and False keywords are not translated, regardless of locale.
- Date data is written to the file using the universal date format. When either the date or the time component is missing or is zero, only the component provided gets written to the file.
- Nothing is written to the file if Data is Empty. However, for Null data, #NULL# is written.
- If d ata is Null, #NULL# is written to the file.

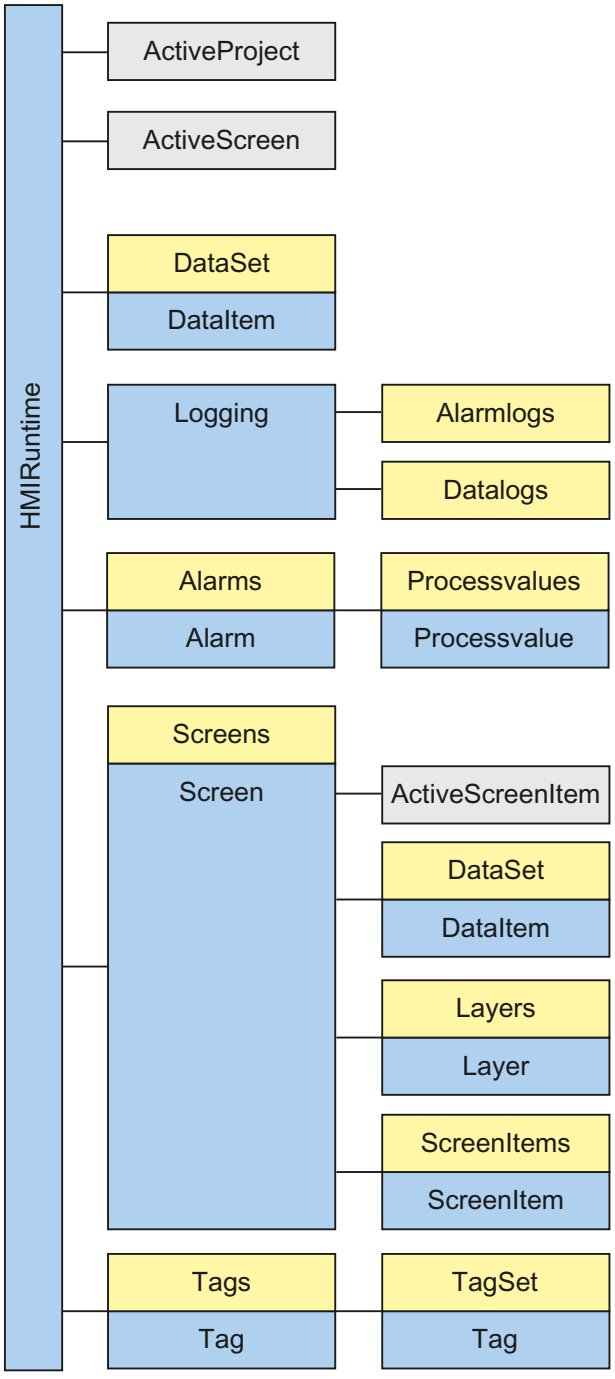
The WriteFields method inserts commas between items and quotation marks around strings as they are written to the file. You do not have to put explicit delimiters in the list. WriteFields inserts a newline character—that is, a carriage return/line feed (Chr(13) + Chr(10))—after it has written the final character in data to the file.

VBS object model

VBS object model

VBS object model in WinCC

The following screen shows the VBS object model in WinCC:



Use the WinCC object model of the graphic Runtime system to access objects and tags in Runtime.

Objects

Objects and lists are provided for access to all the objects in the graphic Runtime systems:

- Display and operating objects
- Screens
- Layers
- Tags

Properties

You can specifically change the display and operating elements and tags in Runtime via the properties of the individual objects. For example, you can enable a button with a click or trigger a color change by changing a tag value.

Methods

Methods which are applied to individual objects can be used to read out tag values for further processing or displaying alarms in Runtime.

See also

ActiveProject (Page 5226)

ActiveScreen (Page 5228)

ActiveScreenItem (Page 5229)

AlarmLogs (list) (Page 4987)

DataItem (Page 4988)

DataLogs (list) (Page 4990)

DataSet (list) (Page 4991)

Logging (Page 4999)

Screen (Page 5001)

Layer (Page 4996)

Layers (list) (Page 4998)

ScreenItems (list) (Page 5006)

ScreenItem (Page 5004)

Tag (Page 5013)

Tags (list) (Page 5016)

TagSet (list) (Page 5017)

HMIRuntime (Page 4994)

Alarm (Page 4984)

Alarms (list) (Page 4985)

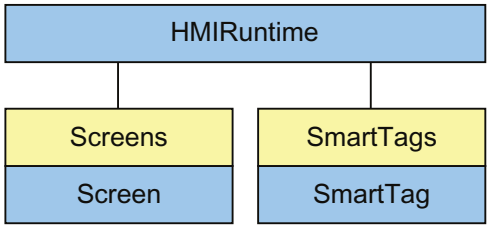
ProcessValue (Page 5456)

Screen object (list) (Page 5008)

Objects

HMIRuntime

Description



Portrays the graphic runtime system.

The HMIRuntime object contains properties and methods which return the objects to the main layer, e.g. returns the ActiveScreen property of a screen object.

Application

You use the "HMIRuntime" object, e.g. as follows:

- Read or set the current Runtime language ("Language" property).
- Read name of current base screen or trigger a base screen change by setting a new screen name (property "BaseScreenName")
- Access tags (List "SmartTags")
- End runtime (Method "Stop")
- Output information on sequence tracing output (Method "Trace")
- Access the screens displayed during runtime (List "Screens")

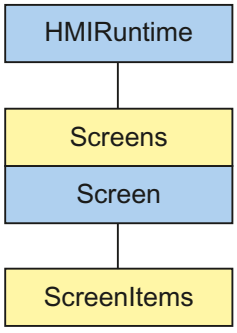
See also

ActiveScreen (Page 5227)

Screen (Page 4977)

Screen object (list)

Description



A list of screen objects.

The list contains the following two elements:

- The first element with the index 0 represents the permanent window.
- The second element with the index 1 represents the root screen.

Alternatively, the two elements can be addressed with their their names as follows:

- Permanent window: "Overview"
- Root screen: Name of the screen displayed in the root screen

If the named screen is not displayed, an error occurs during access.

Permanent window "Overview" is displayed in the objects list and in Auto complete.

Note

The alarm window and the alarm indicator are not contained in the screens list, even if they have the focus in Runtime.

Application

Use the screen property to return the screen list. In the following example, the background color is changed from black to green:

Use the object name as an index.

```
'VBS_Example_BackColor
HMIRuntime.Screens("Rootscreen").BackColor = vbGreen
```

Note

If you perform a screen change, the open references to the screen that is no longer available will become invalid. As a result, it is no longer possible to work with these references.

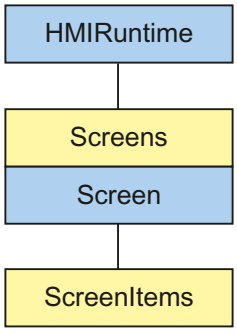
See also

Screen (Page 4977)

ScreenItems (Page 4980)

Screen

Description



Represents the process screen which is being displayed on the HMI device at the moment or the permanent window in runtime. The screen object is returned to you as a result of accessing the screen list.

The screen object also contains a list of all graphic objects in the screen that can be addressed through the list "ScreenItems".

Application

You can also use the screen object to do the following:

- Read the height and width of a screen (properties "Height" and "Width").
- Change the background color (property "BackColor").

Use the object name as an index.

In the following example, the background color is changed from black to green:

```
'VBS example background color
HMIRuntime.Screens("Rootscreen").BackColor = vbGreen
```

See also

ScreenItem (Page 4978)

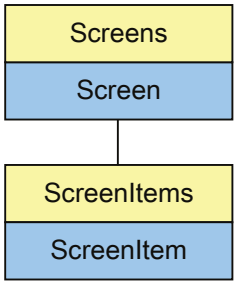
ScreenItems (Page 4980)

HMIRuntime (Page 4974)

Screen object (list) (Page 4975)

ScreenItem

Description



Represents an object in the specified screen. The ScreenItem object is an element of the ScreenItems list.

Application

You can use the ScreenItem object to access the properties of graphic objects within a screen, depending on certain events.

You use the "ScreenItem" object as follows, for example:

- "Visible" property
  Switch the visibility of an object on or off.
- "Height" and "Width" properties
  Query the width and height of an object.
- "Top" and "Left" properties
  Change the position of an object.
- "ObjectName" property
  Read the name of a graphic object
- "Parent" property
  Set a reference to the parent screen

Use the ScreenItems property to return an object to the screen. Use the object name as an index.

Example

In the following example, the background color of the "RootScreen" circle in the "myCircle" screen is set to green.

```
'VBS_Example_ScreenItems

Dim objCircle
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")
objCircle.BackColor = vbGreen
```

Note

To save memory space in the HMI device, no object names are loaded during transfer of the project. If you still want to transfer object names, call up the Runtime settings for the respective HMI device in WinCC. You can change the setting under "General". The object name is required when the object should be accessed via the object name or for debugging a project.

The "ScreenItem" object has different properties depending on its features. The following properties are provided for every "ScreenItem" object:

- Enabled
- Height
- Left

- ObjectName
- Parent
- Top
- Type
- Visible
- Width

If a specific object type is addressed, further properties are added to the standard properties. The additional properties are provided in the descriptions of the individual object types.

See also

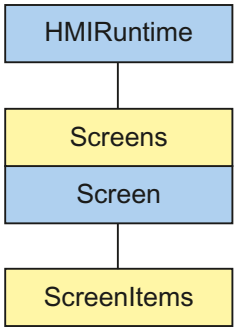
Screen (Page 4976)

Screen object (list) (Page 4975)

ScreenItems (Page 4980)

ScreenItems

Description



A list of screen item objects with all screen objects of the given process screen. The list has a parent property. This property provides a reference to the process screen in which the screen object is located.

Application

By means of the "ScreenItems" list you are able

- To edit or output all objects within the list (that is, all objects within a screen)
- To count the objects of a screen (property "Count").
- To work on a particular object in the list (method "Item").

Use the screen items property to return an object from the process screen. Use the object name as an index.

In the following example, the background color of the "RootScreen" circle in the "myCircle" screen is set to green.

```
Dim objCircle
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")
objCircle.BackColor = vbGreen
```

See also

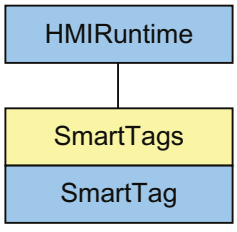
Screen (Page 4976)

HMIRuntime (Page 4974)

Screen object (list) (Page 4975)

SmartTags

Description



A list of SmartTag objects which represent all of the tags in WinCC Runtime.

Note

The SmartTags list has a limited range of functions. You can only use the tag names to access a SmartTag object. Access via the index or by using "For each" instruction is not supported.

Note

In order to access a tag, which has still not been created in the project, using the SmartTags list, no value is returned. Assignment to a non-existing tag is not executed:

```
Dim intVar
intVar = SmartTags("FillLevel")
```

"intVar" remains empty when the "FillLevel" tag has not been created.

Application

Use the SmartTags list to return a SmartTag object. Use the tag name to reference the SmartTag object.

```
'VBS_Example_SmartTags
'Writes tag value to local tag and returns a user-defined text through the
operating system channel for debug alarms.
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

In Runtime Advanced and Panels you address the tag directly using its name. If the tag name corresponds to the VBS name conventions, you do not need to use the SmartTags list. Follow the example below:

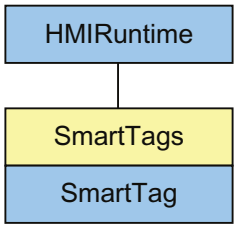
```
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(AirPressure)
HMIRuntime.Trace strAirPressure
```

See also

Access to HMI tags (Page 4580)

SmartTag

Description



Represents the value of the specified process tag. The SmartTag object is an element of the SmartTag list.

Application

The SmartTag object provides read and write access to the value of the specified process tag. The SmartTag object does not return an object reference. Use the SmartTags list to return the value of a process tag. Use the tag name as an index.

Note

With the "SmartTags reads values from the cache" setting, values are read from the process image (cache) instead of directly from the controller.

The SmartTag object can also read values directly from the controller. However, you can then expect a substantially higher communication load between the HMI device and the controller.

Example

```
'VBS_Example_SmartTags
'Writes tag value to local tag and returns a user-defined text through the
operating system channel for debug alarms.
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

Note

In order to access a tag, which has still not been created in the project, using the SmartTags list, no value is returned. Assignment to a non-existing tag is not executed:

```
Dim intVar
intVar = SmartTags("FillLevel")
```

"intVar" remains empty when the "FillLevel" tag has not been created.

Note

If you want to return the "TypeName" of a SmartTag object data type with the VBS function "TypeName", use the following syntax:

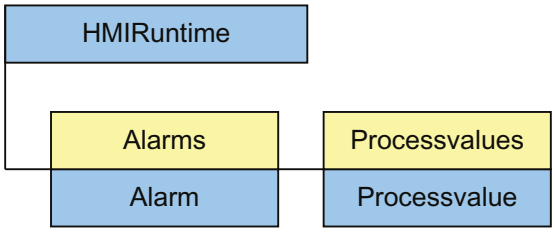
```
TypeName(SmartTags("FillLevel").value)
```

Use "SmartTags("<tag>")(index)" to access the value of an array element. Set the number of the desired array element for "index", for example, "SmartTags("AirPressure")(2)".

Objects

Alarm

Description



The alarm object is used to access the Alarms object list.

Note

The properties of the alarm object are not automatically updated when the values of the properties change.

See also

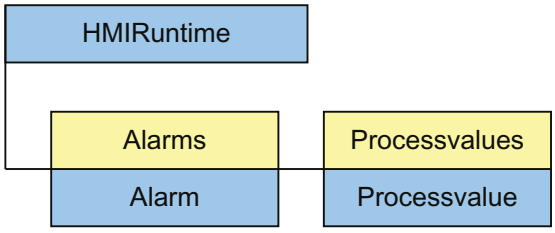
Alarms (list) (Page 4985)

HMIRuntime (Page 4994)

ProcessValue (Page 5456)

Alarms (list)

Description



Use the alarm object to trigger existing messages.

Usage

Using the "Alarms" list you can:

- Access a message in the list (Item method)
- Create a new alarm object (Create method)
- Read the alarm ID of the message (AlarmID attribute)
- Read the status of a message (State property)

- Read the time stamp of the message (Timestamp property)
- Generate an instance of the alarm object (Instance property)
- Read the name of the computer on which the message came (ComputerName property)
- Read or set the name of the user who triggered the message (UserName property)
- Read or set the name of the process value blocks (ProcessValues property)
- Read or set the message commentary (Comment property)
- Read or set the message server prefix (Context property)

Example

In the following example the alarm with the alarm number "1" configured in the "HMI alarms" editor is activated.

```
'VBS360
Dim MyAlarm
Set MyAlarm = HMIRuntime.Alarms(1)
MyAlarm.State = 5 'hmiAlarmStateCome + hmiAlarmStateComment
MyAlarm.Comment = "MyComment"
MyAlarm.UserName = "Hans-Peter"
MyAlarm.ProcessValues(1) = "Process Value 1"
MyAlarm.ProcessValues(4) = "Process Value 4"
MyAlarm.Create "MyApplication"
```

See also

AlarmID (Page 5235)

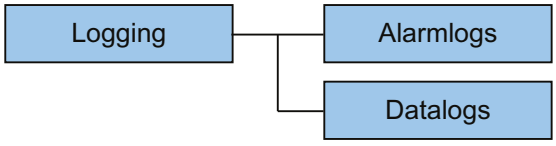
ProcessValue (Page 5456)

Alarm (Page 4983)

HMIRuntime (Page 4994)

AlarmLogs (list)

Description



You can use the object to reconnect swapped-out log segments of the alarm log to Runtime or to remove previously swapped-in log segments of the alarm log. The log segments to be swapped in are copied to the "Common logging" folder of the WinCC project. The previously swapped-in log segments are removed from the "Common logging" folder.

You use parameters to control the location from which log segments are to be swapped in. You specify the time period over which the log segments are to be swapped in or removed.

If an error occurs during the operation with log segments, the applied method returns an error alarm.

Usage

- "Restore" method
  Previously swapped-out log segments of the alarm log are connected to Runtime.
- "Remove" method
  Previously swapped-in log segments of the alarm log are removed from the Runtime project.

Example

In the following example, log segments from the alarm log are swapped in and the return value is output as a trace.

```
'VBS187
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.AlarmLogs.Restore("D:
\Folder","2004-09-14","2004-09-20",-1) & vbNewLine
```

See also

Restore (Page 5915)

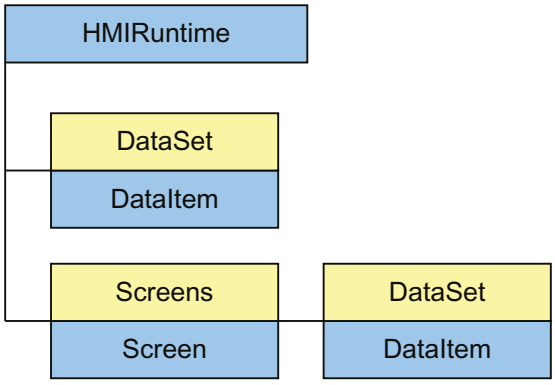
Remove (Page 5910)

DataLogs (list) (Page 4990)

Logging (Page 4999)

DataItem

Description



You can use the DataItem object to access the contents of the DataSet list. Values or object references are stored in the list as DataItem.

Access uses the name under which the value was added to the list. Single access using an index is not recommended since the index changes during adding or deleting of values. You can use the index to output the complete contents of the list. The output is in alphabetical order.

Note

For object references, ensure that the objects are capable of multi-threading.

Example

The example shows how the value of 'Motor1' is output as a trace.

```
'VBS163
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
```

The following example enumerates all DataItem objects of the DataSet list. Name and value are output as a trace.

```
'VBS164
Dim data
For Each data In HMIRuntime.DataSet
HMIRuntime.Trace data.Name & ": " & data.Value & vbNewLine
Next
```

Note

The value may not be output directly for objects.

See also

Screen (Page 5001)

HMIRuntime (Page 4994)

DataSet (list) (Page 4991)

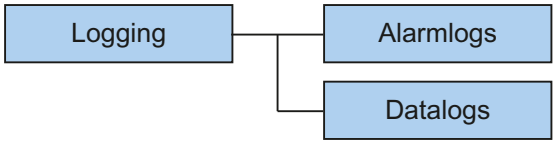
Value (Page 5640)

Name (Page 5426)

Screen object (list) (Page 5008)

DataLogs (list)

Description



You can use the object to reconnect swapped-out log segments of the data log to Runtime or to delete previously swapped-in log segments of the data log. The log segments to be swapped in are copied to the "Common logging" folder of the WinCC project. The previously swapped-in log segments are removed from the "Common logging" folder.

You use parameters to control the location from which log segments are to be swapped in. You specify the time period over which the log segments are to be swapped in or removed. You also set the type of log ("Fast data log", "Slow data log", "Fast data log and Slow data log").

If an error occurs during the operation with log segments, the applied method returns an error alarm.

Usage

- "Restore" method
  Previously swapped out log segments of the data log are connected to Runtime.
- "Remove" method
  Previously swapped-in log segments of the data log are removed from the Runtime project.

Example

In the following example, log segments from the Fast data log are swapped in and the return value is output as a trace.

```
'VBS188
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.DataLogs.Restore("D:
\Folder","2004-09-14","2004-09-20",-1,1) & vbNewLine
```

See also

Restore (Page 5915)

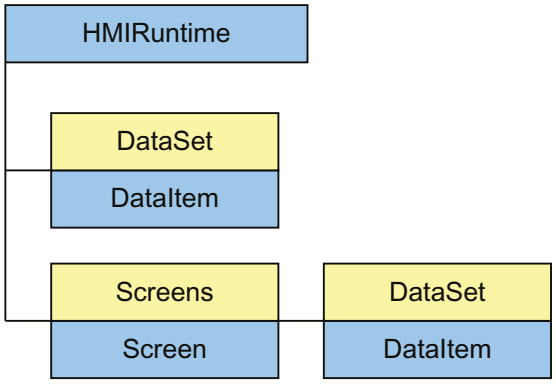
Remove (Page 5910)

AlarmLogs (list) (Page 4986)

Logging (Page 4999)

DataSet (list)

Description



Using the DataSet object, you can exchange data throughout several actions.

A DataSet object is global and defined by the screen object. You can can access the data from any VBS action.

You address the screen object according to the screen hierarchy . The DataSet object persists as long as the screen is displayed. The global object persists over the entire runtime time period.

Access uses the DataItem object.

Note

You cannot include objects with the types Screen, Screens, ScreenItem, ScreenItems, Tag and TagSet in the DataSet list.

The DataSet object does not support any classes.

Usage

Use the "DataSet" list as follows:

- Enumeration
  Output or process all objects in the list
- "Count" property
  Output the number of elements contained
- "Item" method
  Work on a particular object in the list
- "Add" method
  Add an object to the list
- "Remove" method
  Remove a particular object from the list
- "RemoveAll" method
  Remove all objects from the list

Access to list elements is made as follows:

```
HMIRuntime.DataSet("Itemname")
```

For a picture-specific list, access is performed as follows:

```
HMIRuntime.Screens("Screenname").DataSet("Itemname")
```

You access the DataSet object of the screen as follows:

```
DataSet("Itemname")
```

If upon access the stated name does not exist in the list, VT_Empty is returned and an Exception is triggered.

Example

The example shows how a value can be entered in the list, read and removed from the list throughout various actions.

```
'VBS162
HMIRuntime.DataSet.Add "motor1", 23
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet.Remove("motor1")
```

See also

RemoveAll (Page 5914)

Remove (Page 5910)

Item (Page 5891)

Add (Page 5830)

DataItem (Page 4987)

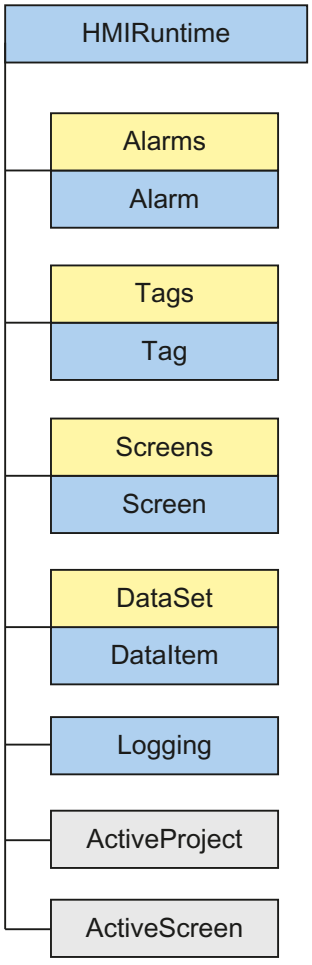
HMIRuntime (Page 4994)

Screen (Page 5001)

Screen object (list) (Page 5008)

HMIRuntime

Description



The HMIRuntime object represents the graphic Runtime environment.

Usage

You can use the "HMIRuntime" object as follows:

- "Language" property
  Read or set the current Runtime language
- "BaseScreenName" property
  Read or set the name of the current root screen
- "ActiveProject" property
  Read the path of the active Runtime project

- "Tags" property
  Accessing tags
- "DataSet" property
  Accessing tags in a list
- "Stop" method
  Stop Runtime
- "Trace" method
  Display messages in a diagnostics window

Example

The following command closes WinCC Runtime:

```
'VBS3
HMIRuntime.Stop
```

See also

Trace (Page 5934)

Stop (Page 5934)

Logging (Page 4999)

DataSet (list) (Page 4990)

Language (Page 5380)

Tags (Page 5532)

Logging (Page 5402)

DataSet (Page 5314)

CurrentContext (Page 5309)

ActiveProject (Page 5226)

MenuToolBarConfig (Page 5406)

Alarm (Page 4983)

Alarms (list) (Page 4984)

Tag (Page 5013)

Tags (list) (Page 5016)

DataItem (Page 4987)

Screen (Page 5001)

Screen object (list) (Page 5008)

ActiveScreen (Page 5228)

Item

Description

The Item object provides a reference to the current object.

Usage

Use the Item object, for example, to address the properties of the object currently selected in the screen.

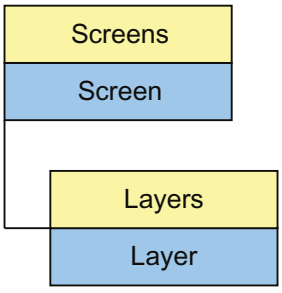
Example

In the following example, you set the background color of the object selected in the screen to red:

```
'VBS195
Item.BackColor = RGB(255,0,0)
```

Layer

Description



The Layer object returns the result of access to the layers list.

Parent object

Screen in which the screen layer is located

Usage

Depending on certain events, you can use the Layer object to obtain access to the properties of a complete layer to hide or display a layer with operating elements according to the operator authorization.

You can use the "Layer" object as follows:

- "Visible" property
  Activate or deactivate visibility of a layer
- "Name" property
  Read the name of a layer

Note

The layer property specifies the layer in which the object is located. The layer "0" is output as layer "0".

When accessed, the layers are counted up from 1 in VBS. Therefore, address level "1" with "Layers(2)".

Example

In the following example, Layer 1 is set to "invisible":

```
'VBS4
Layers(2).Visible = vbFalse
```

See also

Name (Page 5426)

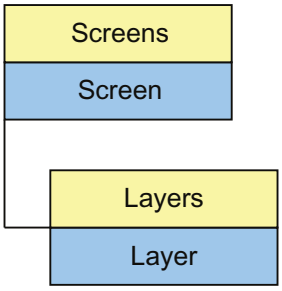
Layers (list) (Page 4998)

Screen (Page 5001)

Screen object (list) (Page 5008)

Layers (list)

Description



Use the layers list to access all 32 layers of the graphical Runtime system.

Parent object

Screen in which the screen layer is located

Usage

You use the "Layers" list as follows:

- "_NewEnum" property
  Process all layers in the list
- "Count" property
  Count all layers contained in the list
- "Item" method
  Process a layer from the list

The properties represent default properties and methods of a list and are not described in detail in the WinCC documentation.

See also

Item (Page 5891)

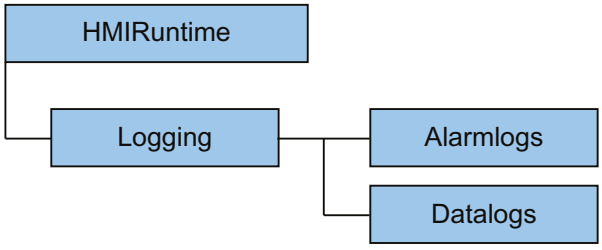
Layer (Page 4995)

Screen (Page 5001)

Screen object (list) (Page 5008)

Logging

Description



You can use the object to reconnect swapped-out log segments to Runtime or to remove previously swapped-in log segments. The log segments to be swapped in are copied to the "Common logging" folder of the WinCC project. The previously swapped-in log segments are removed from the "Common logging" folder.

You use parameters to control the location from which log segments are to be swapped in. You specify the time period over which the log segments are to be swapped in or removed.

If an error occurs during the operation with log segments, the applied method returns an error alarm.

Usage

- "Restore" method
  Previously swapped out log segments of the alarm log and the data log are connected to Runtime.
- "Remove" method
  Previously swapped-in log segments of the alarm log and data log are removed from the Runtime project.

Example

In the following example, log segments from the alarm log and data log are swapped in and the return value is output as a trace.

```
'VBS189
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Restore("D:
\Folder","2004-09-14","2004-09-20",-1) & vbNewLine
```

See also

Restore (Page 5915)

Remove (Page 5910)

DataLogs (list) (Page 4989)

AlarmLogs (list) (Page 4986)

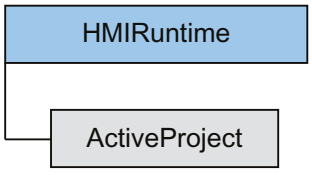
DataLogs (Page 5313)

AlarmLogs (Page 5236)

HMIRuntime (Page 4993)

Project

Description



Using the object, you can query information from the current Runtime project.

The project object is returned as the result of ActiveProject.

Usage

You can read the following using the "Project" object:

- The path of the current Runtime project ("Path" property)
- The name of the current Runtime project, without path or file extension ("Name" property)

Example

The following example returns name and path of the current Runtime project as a trace:

```
'VBS159
HMIRuntime.Trace "Name: " & HMIRuntime.ActiveProject.Name & vbNewLine
HMIRuntime.Trace "Path: " & HMIRuntime.ActiveProject.Path & vbNewLine
```

See also

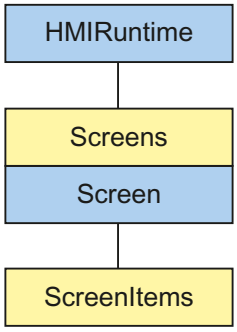
ActiveProject (Page 5226)

Name (Page 5426)

Path (Page 5450)

Screen

Description



Represents the screen which is being displayed on the HMI device at the moment or the permanent window in Runtime. The screen object is returned as a result of accessing the screen list.

The screen object also contains the following lists:

- You can address all the graphic objects in the addressed screen using the "ScreenItems" list.
- You can address all the layers in the addressed screen using the "Layers" object.

Application

You can use the screen object for the following actions, for example:

- "Width" and "Height" properties
  Reading the width and height of a screen
- "BackColor" property
  Changing the background color

Use the object name as an index.

Example

In the following example, the background color is changed from black to green:

```
'VBS_Example_BackColor
HMIRuntime.Screens("Rootscreen").BackColor = vbGreen
```

Parent object

Screen window in which the screen object is embedded.

If the screen object is the root screen, the parent object is not defined and set to zero.

Note

If you perform a screen change, the open references to screens that are no longer available will become invalid. As a result, it is no longer possible to work with these references.

Example

In the following example, the width of the first screen is increased by 20 pixels in Runtime:

```
'VBS7
Dim objScreen
Set objScreen = HMIRuntime.Screens(1)
MsgBox "Screen width before changing: " & objScreen.Width
objScreen.Width = objScreen.Width + 20
MsgBox "Screen width after changing: " & objScreen.Width
```

Notes on Cross References

All the screens you address with the standard formulation are automatically compiled by the CrossReference of WinCC and then listed in the screen properties.

```
HMIRuntime.BaseScreenName = "Screenname"
```

If you call screens with different formulations in the code, you make them known by the following section of the CrossReference:

```
' ' WINCC:SCREENNAME_SECTION_START
Const ScreenNameInAction = "ScreenName"
' WINCC:SCREENNAME_SECTION_END
The section can be inserted in VBS actions as often as required.
```

See also

Refresh (Page 5909)

Activate (Page 5825)

ObjectSizeDeclutteringEnable (Page 5432)

ObjectSizeDeclutteringMax (Page 5433)

ObjectSizeDeclutteringMin (Page 5434)

LayerDeclutteringEnable (Page 5387)

Layers (Page 5388)

DataSet (Page 5314)

ExtendedZoomingEnable (Page 5334)

AccessPath (Page 5226)

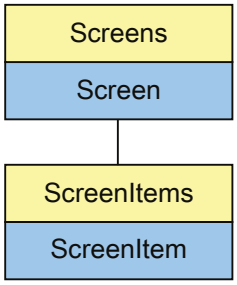
HMIRuntime (Page 4993)

ScreenItems (list) (Page 5006)

Screen object (list) (Page 5008)

ScreenItem

Description



Represents an object in the specified screen. The ScreenItem object is an element of the ScreenItems list.

Application

You can use the ScreenItem object to access the properties of graphic objects within a screen, depending on certain events.

You use the "ScreenItem" object as follows, for example:

- "Visible" property
  Switch the visibility of an object on or off.
- "Height" and "Width" properties
  Query the width and height of an object.
- "Top" and "Left" properties
  Change the position of an object.
- "ObjectName" property
  Read the name of a graphic object
- "Parent" property
  Set a reference to the parent screen

Use the ScreenItems property to return an object to the screen. Use the object name as an index.

Example

In the following example, the background color of the "RootScreen" circle in the "myCircle" screen is set to green.

```
'VBS_Example_ScreenItems

Dim objCircle
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")
objCircle.BackColor = vbGreen
```

Note

To save memory space in the HMI device, no object names are loaded during transfer of the project. If you still want to transfer object names, call up the Runtime settings for the respective HMI device in WinCC. You can change the setting under "General". The object name is required when the object should be accessed via the object name or for debugging a project.

The "ScreenItem" object has different properties depending on its features. The following properties are provided for every "ScreenItem" object:

- Enabled
- Height
- Left
- ObjectName
- Parent
- Top
- Type
- Visible
- Width

If a specific object type is addressed, further properties are added to the standard properties. The additional properties are provided in the descriptions of the individual object types.

See also

Activate (Page 5825)

Layer (Page 5384)

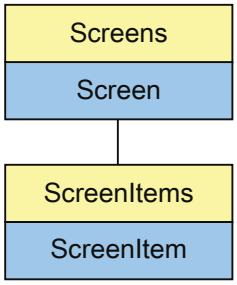
Screen (Page 5000)

ScreenItems (list) (Page 5006)

Screen object (list) (Page 5008)

ScreenItems (list)

Description



A list of ScreenItem objects with all screen objects of the specified screen. The list has a "Parent" property. This "Parent" property provides a reference to the screen in which the screen object is located.

Usage

You use the "ScreenItems" list as follows:

- To edit or output all objects within the list (that is, all objects within a screen)
- "Count" property
  Count the objects of a screen
- "Item" method
  Work on a particular object in the list

Use the screen items property to return an object from the screen. Use the object name as an index.

Example

In the following example, the background color of the "RootScreen" circle in the "myCircle" screen is set to green.

```
Dim objCircle
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")
objCircle.BackColor = vbGreen
```

Peculiarities of the ScreenItem object

If you embed an external control (ActiveX control or OLE object) in WinCC, the properties of the embedded control may have the same names as the general properties of the ScreenItem object. In this case, the ScreenItem properties take priority.

You can address the properties of the embedded controls, however, via the "object" property. The "object" property is only available in ActiveX controls and OLE objects.

Example:

```
'Control1 is an embedded ActiveX-Control with property "type"
'VBS196
Dim Control
Set Control=ScreenItems("Control1")
Control.object.type

'Control1 is a WinCC-Control
'VBS197
Dim Control
Set Control=ScreenItems("Control1")
Control.type
```

Example

In the following example, you output the name of the objects in the current screen in a message box:

```
Sub OnClick(ByVal Item)
'VBS6
Dim lngAnswer
Dim lngIndex
lngIndex = 1
For lngIndex = 1 To ScreenItems.Count
lngAnswer = MsgBox(ScreenItems(lngIndex).Objectname, vbOKCancel)
If vbCancel = lngAnswer Then Exit For
Next
End Sub
```

See also

Item (Page 5891)

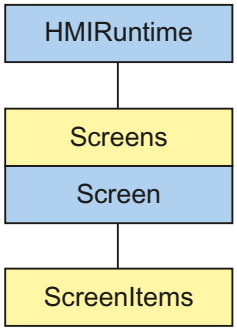
ScreenItem (Page 5003)

Screen (Page 5000)

Screen object (list) (Page 5008)

Screen object (list)

Description



Several screens can be opened simultaeously in WinCC Runtime by means of the screen window technique, whereby there is only one main screen. The "Screens" list allows access to all open screens in Runtime using the screen name. The screen list contains all hidden screens.

The access key required in the VBS environment in the HMIRuntime.Screens(<access key>) instruction must conform to the following syntax description:

```
[<Root screen name>.]<Screen window name>[:<Screen name>] ...
.<Screen window name>[:<Screen name>]
```

- The access key represents the screen hierarchy.
- You can omit the screen name at all locations in the key.
- The "AccessPath" property of the "Screen" object corresponds to the full access key.
- The root screen can be addressed via the "" access key.

Examples

The screens are addressed by specifying the hierarchy in the list. You can address the screens with or without using the screen names. In the following example, a "BaseScreenName" root screen is configured with a "ScreenWindow". The screen window contains a "ScreenName" screen.

Addressing using the screen name

```
'VBS8
Set objScreen = HMIRuntime.Screens("BaseScreenName.ScreenWindow:ScreenName")
```

Addressing without using the screen name

```
'VBS9
Set objScreen = HMIRuntime.Screens("ScreenWindow")
```

Referencing the root screen in various ways

```
'VBS10
Set objScreen = HMIRuntime.Screens(1)

'VBS11
Set objScreen = HMIRuntime.Screens("")

'VBS12
Set objScreen = HMIRuntime.Screens("BaseScreenName")
```

See also

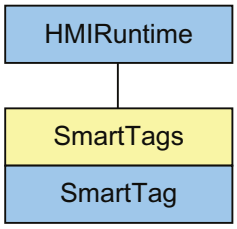
HMIRuntime (Page 4993)

Screen (Page 5000)

ScreenItem (Page 5003)

SmartTag

Description



Represents the value of the specified process tag. The SmartTag object is an element of the SmartTag list.

Application

The SmartTag object provides read and write access to the value of the specified process tag. The SmartTag object does not return an object reference. Use the SmartTags list to return the value of a process tag. Use the tag name as an index.

Note

With the "SmartTags reads values from the cache" setting, values are read from the process image (cache) instead of directly from the controller.

The SmartTag object can also read values directly from the controller. However, you can then expect a substantially higher communication load between the HMI device and the controller.

Example

```
'VBS_Example_SmartTags
'Writes tag value to local tag and returns a user-defined text through the
operating system channel for debug alarms.
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

Note

In order to access a tag, which has still not been created in the project, using the SmartTags list, no value is returned. Assignment to a non-existing tag is not executed:

```
Dim intVar
intVar = SmartTags("FillLevel")
```

"intVar" remains empty when the "FillLevel" tag has not been created.

Note

If you want to return the "TypeName" of a SmartTag object data type with the VBS function "TypeName", use the following syntax:

```
TypeName(SmartTags("FillLevel").value)
```

Use "SmartTags("<tag>")(index)" to access the value of an array element. Set the number of the desired array element for "index", for example, "SmartTags("AirPressure")(2)".

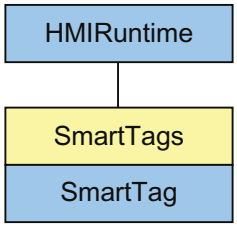
See also

HMIRuntime (Page 4993)

SmartTags (Page 5011)

SmartTags

Description



A list of SmartTag objects which represent all of the tags in WinCC Runtime.

Note

The SmartTags list has a limited range of functions. You can only use the tag names to access a SmartTag object. Access via the index or by using "For each" instruction is not supported.

Note

In order to access a tag, which has still not been created in the project, using the SmartTags list, no value is returned. Assignment to a non-existing tag is not executed:

```
Dim intVar
intVar = SmartTags("FillLevel")
```

"intVar" remains empty when the "FillLevel" tag has not been created.

Application

Use the SmartTags list to return a SmartTag object. Use the tag name to reference the SmartTag object.

```
'VBS_Example_SmartTags
'Writes tag value to local tag and returns a user-defined text through the
operating system channel for debug alarms.
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

In Runtime Advanced and Panels you address the tag directly using its name. If the tag name corresponds to the VBS name conventions, you do not need to use the SmartTags list. Follow the example below:

```
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(AirPressure)
HMIRuntime.Trace strAirPressure
```

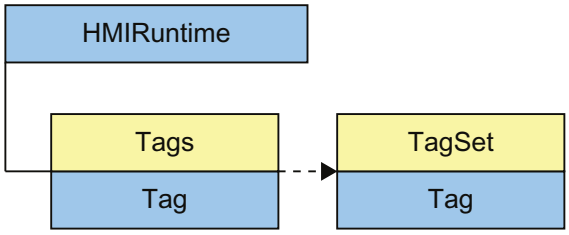
See also

HMIRuntime (Page 4993)

SmartTag (Page 5009)

Tag

Description



A tag object is returned via the "Tags" list. A tag object can be used to address all the properties and methods of a tag.

When creating a tag object, all the properties are initialized with the following values:

- Value = VT_EMPTY
- Name = Tag name
- QualityCode = BAD NON-SPECIFIC
- TimeStamp = 0
- LastError = 0
- ErrorDescription = " "

Note

A summary of possible Quality Codes can be found in WinCC Information System under key word "Communication" > "Diagnostics" or "Communication" > "Quality Codes".

Usage

Use the "Tag" object as follows:

- "Name", "QualityCode", "TimeStamp", "LastError" and "ErrorDescription" properties
  Read information on the tag
- "Write" method, "Value" property
  Set a value for a tag
- "Read" method, "Value" property
  Read a value for a tag

Example

The following example reads the value in the "Tag1" tag:

```
'VBS13
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read()
MsgBox objTag.Value
```

Declaration of tags in WinCC

Always define internal tags in VB script using the "Dim" instruction in order to prevent writing tags wrongly.

When creating a new action, the "Option explicit" instruction is automatically entered in the declaration area and cannot be deleted.

Do not use the "Option explicit" statement in your code as it can lead to Runtime errors.

Example

The following example reads the declaration value in the "lngVar" VB Script tag:

```
'VBS14
Dim lngVar
lngVar = 5
MsgBox lngVar
```

Note

Tag names must not contain any special characters.

When creating a tag, ensure it does not contain a value (Value = VT_EMPTY). Initialize the tags after declaration with the corresponding value.

Notes on Cross References

All the pictures which are addressed with the standard formulation

```
HMIRuntime.Tags("Tagname")
```

are automatically compiled by the CrossReference of WinCC and then listed in the picture properties.

If you address tags with different formulations in the code, you can make them known by the following section of the CrossReference:

```
' ' WINCC:TAGNAME_SECTION_START
Const TagNameInAction = "TagName"
' WINCC:TAGNAME_SECTION_END
```

The section can be inserted in VBS actions as often as required.

Note

Composed tag names may not be recognized by the CrossReference.

See also

Name (Page 5426)

Value (Page 5640)

ErrorDescription (Page 5327)

LastError (Page 5383)

QualityCode (Page 5457)

Timestamp (Page 5556)

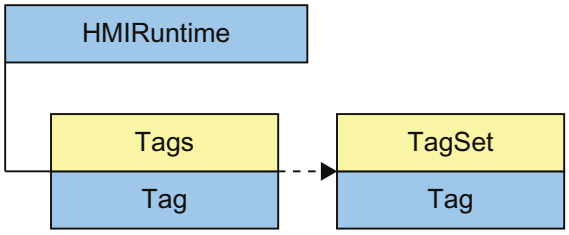
HMIRuntime (Page 4993)

Tags (list) (Page 5016)

TagSet (list) (Page 5017)

Tags (list)

Description



The "Tags" list enables access to tags in WinCC Runtime. The result of access to the "Tags" list is returned by an object of the type "Tag". The Tag object can be used to access all the tag properties and methods.

Note

"Tags" is a list with a restricted functional scope. The tags in the list cannot be accessed via the index but only by using the tag names. The standard methods get_Count and get_NewEnum cannot be used in the Tags list.

Application

Tags in the list are accessed via:

```
HMIRuntime.Tags("Tagname")
```

The Tags list is used to declare tags (tag objects) for read and write access. The appropriate HMI tags must exist in order for the write and read acccess to be executed without error.

You can address HMI tags directly in VBScript and set and read values. If you wish to inquire about additional tag properties, such as quality code or time stamp, or wish to execute error processing, you must address the tag through tags list. The tag object returned enables access to all tag properties and methods.

Using the "CreateTagSet" method, you may generate a TagSet object which allows simultaneous access to several tags.

Example

You use tag names when you set tags.

```
'VBS16
Dim objTag
Set objTag = HMIRuntime.Tags("Tagname")
If you only use the tag name, the "TagPrefix" property is assigned the values from the
current context (the current screen window).
```

See also

HMIRuntime (Page 4993)

Tag (Page 5012)

TagSet (list) (Page 5017)

TagSet (list)

Description

The "TagSet" object enables simultaneous access to several tags in one call. Simultaneous access demonstrates better performance and lower communication load than single access to multiple tags.

Usage

You can use the TagSet object as follows:

- "Add" method
  Add tags to the list
- "Item" method
  Access tag objects contained in the list and their properties
- "Write" method
  Write all tags of the list
- "Read" method
  Read all tags of the list
- "Remove" method
  Remove single tags from the list
- "RemoveAll" method
  Remove all tags from the list

Tags in the list are accessed via:

```
'VBS169
Dim myTags
myTags = HMIRuntime.Tags.CreateTagSet
myTags("Tagname")
```

In order to have error-free read/write access to tags (tag objects) of the list, the respective tags must exist in WinCC.

If a read/write access error has occurred, the method used will return an error message using the "LastError" and "ErrorDescription" properties.

Example

The following example shows how to generate a TagSet object, how to add tags, and how to write values.

```
'VBS168
'Build a Reference to the TagSet Object
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
'Add Tags to the Collection
group.Add "Motor1"
group.Add "Motor2"
'Set the Values of the Tags
group("Motor1").Value = 3
group("Motor2").Value = 9
'Write the Values to the DataManager
group.Write
```

See also

ErrorDescription (Page 5327)

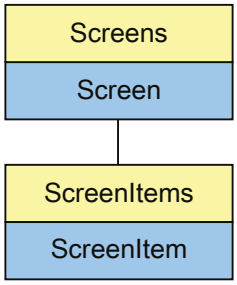
LastError (Page 5383)

Object types

Objects A-I

AlarmControl

Description



Represents the "Alarm view" object. The AlarmControl object is an element of the ScreenItems list.

Type identifier in VBS

HMIAlarmControl

Example

In the following example, the object with the name "Control1" is moved 10 pixels to the right:

```
'VBS54
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 10
```

Abbreviation	Validity
Pa	Panels
A	RT Advanced
P	RT Professional

Table 12-19 Properties

Properties	Read	Write	Description
Activate (Page 5739)	P	P	Specifies whether the data to be visualized is requested by the alarm server.
AlarmListType	P	P	Specifies the alarm classes to be reported in runtime.
AllServer	P	P	Sets the display of the alarms of all available servers.
ApplyProjectSettings (Page 5243)	P	P	Specifies whether the project settings from the "HMI alarms" editor are applied.
ApplyProjectSettingsForDesign-Mode	P	P	Specifies that the project settings are used for the design.
AutoCompleteColumns (Page 5247)	P	P	Specifies whether empty columns are shown if the control is wider than the configured columns.
AutoCompleteRows (Page 5247)	P	P	Specifies whether empty rows are shown if the control is longer than the number of configured rows.
AutoSelectionColors (Page 5248)	P	P	Specifies whether the colors defined by the system are used as the selection colors for cells and rows.
AutoSelectionRectColor (Page 5249)	P	P	Specifies whether the selection frame is shown with the color defined by the system.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
Blocks	P	P	Sets the alarm blocks.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
Caption (Page 5287)	P	P	Specifies the text that is displayed in the header of the selected object.
CellCut (Page 5289)	P	P	Specifies whether the contents of the cells are abbreviated if the cells are too narrow.
CellSpaceBottom (Page 5290)	P	P	Specifies the bottom margin of the table cells.
CellSpaceLeft (Page 5290)	P	P	Sets the left indent for the table cells.
CellSpaceRight (Page 5291)	P	P	Specifies the right indent of the table cells.
CellSpaceTop (Page 5291)	P	P	Specifies the top margin of the table cells.
Closeable (Page 5295)	P	P	Specifies that the control can be closed in runtime.
ColumnResize (Page 5299)	P	P	Enables changes to the width of columns.
ColumnScrollbar (Page 5300)	P	P	Specifies the scroll bar type.
ColumnTitleAlignment (Page 5303)	P	P	Specifies the type of column title alignment.
ColumnTitles (Page 5303)	P	P	Specifies whether the column title is displayed.
ControlDesignMode (Page 5507)	P	P	Specifies the design.
DefaultMsgFilterSQL	P	P	Specifies an SQL statement as default for alarm filters.

Properties	Read	Write	Description
DefaultSort	P	P	Specifies the sorting order.
DefaultSort2	P	P	Specifies the sorting order.
DefaultSort2Column	P	P	Specifies the sorting order.
DisplayOptions (Page 5318)	P	P	Specifies the alarms to be displayed.
DoubleClickAction (Page 5318)	P	P	Specifies the action to be executed in runtime by double-clicking an alarm row.
ExportDirectoryChangeable (Page 5328)	P	P	Specifies whether the data export directory can be changed in runtime.
ExportDirectoryname (Page 5329)	P	P	Specifies the target directory for the exported Runtime data.
ExportFileExtension (Page 5330)	P	P	Specifies the file extension of the export file. Only the "csv" file extension is currently supported.
ExportFilename (Page 5330)	P	P	Specifies the name of the target file for the exported Runtime data.
ExportFilenameChangeable (Page 5331)	P	P	Specifies whether the export file name can be changed in runtime.
ExportFormatGuid (Page 5331)	P	P	Default assignment of the ID number and export provider.
ExportFormatName (Page 5332)	P	P	Specifies the export file format. Only the "csv" file format is currently available for the export.
ExportParameters (Page 5332)	P	P	Specifies the parameters of the selected format by means of the "Properties" dialog.
ExportSelection (Page 5333)	P	P	Specifies the file extension of the export file. Only the "csv" file extension is currently supported.
ExportShowDialog (Page 5334)	P	P	Specifies whether the data export dialog is shown in runtime.
Filters	P	P	Specifies database criteria in SQL syntax.
Font (Page 5348)	P	P	Specifies or returns the font.
GridLineColor (Page 5356)	P	P	Specifies the color for the grid lines.
GridLineWidth (Page 5357)	P	P	Specifies the width of the separation lines in pixels.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HitlistColumnAdd (Page 5364) *	P	P	Transfers the selected alarm block from the list of available alarm blocks to the list of selected alarm blocks.
HitlistColumnCount (Page 5365) *	P	P	Sets the number of selected alarm blocks to be displayed in the hitlist in runtime.
HitlistColumnIndex (Page 5365)	P	P	References an alarm block that is selected for the hitlist.
HitlistColumnName (Page 5366) *	P	P	Displays the name of the alarm block of the hitlist that is referenced by means of the "HitlistColumnIndex" attribute.
HitlistColumnRemove (Page 5366) *	P	P	Cuts the marked alarm block from the list of selected alarm blocks and pastes it into the list of existing alarm blocks.
HitlistColumnRepos (Page 5367) *	P	P	Changes the sorting order of the alarm blocks. The "Up" and "Down" commands move the selected alarm block accordingly in the list.
HitlistColumnSort (Page 5367) *	P	P	Specifies the sorting order in the hitlist for the alarm block that is referenced in "MessageColumnIndex".
HitlistColumnSortIndex (Page 5368) *	P	P	Specifies the sorting order in the hitlist for the alarm block that is referenced in "HitlistColumnIndex".

Properties	Read	Write	Description
HitlistColumnVisible (Page 5368) *	P	P	The list displays the selected alarm blocks from the alarm list or hitlist that are to be used in the control in runtime.
HitListDefaultSort (Page 5369)	P	P	Sets the default sorting order in the table columns of the hitlist.
HitListMaxSourceItems (Page 5369)	P	P	Sets the maximum number of data records from the alarm log that are to be used to create the hit list.
HitListMaxSourceItemsWarn (Page 5370)	P	P	Sets the output of a warning on reaching the maximum number of data records that was specified in "HitlistMaxSourceItems".
HitListRelTime (Page 5371)	P	P	Sets a time range for the statistics.
HitListRelTimeFactor	P	P	Sets the time factor that is used in combination with the "HitlistRelativeTimeFactorType" to calculate the period for creating the hitlist statistics.
HitListRelTimeFactorType	P	P	Sets the time type that is used in combination with the "HitlistRelativeTimeFactor" to calculate the period for creating the hitlist statistics.
HorizontalGridLines (Page 5373)	P	P	Specifies whether horizontal separation lines are displayed.
IconSpace (Page 5375)	P	P	Specifies the spacing between icons and text in the table cells.
Layer (Page 5384)	P	P	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineColor (Page 5398)	P	P	Sets the color of the window separation lines.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
LongTermArchiveConsistency (Page 5403)	P	P	If "LongTimeArchiveConsistency" is set to "No", 1000 alarms will be displayed in the historical alarm list (long-term) on the single station, on the server, or on the client for each server or redundant server pair.
MessageBlockAlign (Page 5407) *	P	P	Sets the mode of alignment for the table contents of the selected alarm block.
MessageBlockAutoPrecisions (Page 5407) *	P	P	Enables automatic setting of the number of decimal places.
MessageBlockCaption (Page 5408) *	P	P	Specifies the caption of the column header in the alarm view for the selected alarm block.
MessageBlockCount (Page 5408)	P	P	Specifies the number of alarm blocks that are available for the alarm list and hitlist.
MessageBlockDateFormat (Page 5409) *	P	P	Specifies the date format for displaying the alarms.
MessageBlockExponentialFormat (Page 5409) *	P	P	Specifies the exponential notation for visualization of the values of a selected alarm block.
MessageBlockFlashOn (Page 5410) *	P	P	Enables flashing of the selected alarm block in runtime when an alarm appears.
MessageBlockHideText (Page 5410) *	P	P	Enables the textual display of the content of the selected alarm block.
MessageBlockHideTitleText (Page 5411) *	P	P	Specifies textual display of the caption of the selected alarm block.
MessageBlockId (Page 5411)	P	P	Specified assignment of ID number and alarm block in the alarm view.
MessageBlockIndex (Page 5411)	P	P	References an existing alarm block.
MessageBlockLeadingZeros (Page 5412) *	P	P	Enables the display of the selected alarm block with leading zeroes.

Properties	Read	Write	Description
MessageBlockLength (Page 5412) *	P	P	Specifies the length of the selected alarm block based on the number of characters.
MessageBlockName (Page 5413)	P	P	Displays the name of the selected alarm block.
MessageBlockPrecision (Page 5413) *	P	P	Specifies the number of decimal places for the values of the selected alarm block.
MessageBlockSelected (Page 5414) *	P	P	Existing alarm blocks are blocks that are available for use in the control in runtime for the alarm list or hitlist.
MessageBlockShowDate (Page 5414) *	P	P	Specifies whether to display both the date and time in the "Time" alarm block.
MessageBlockShowIcon (Page 5415) *	P	P	Enables the display of the content of the selected alarm block as icon.
MessageBlockShowTitleIcon (Page 5415) *	P	P	Specifies textual display of the caption of the selected alarm block.
MessageBlockTextId (Page 5416)	P	P	Specifies naming of the selected alarm block by means of a text ID that was taken from the text library.
MessageBlockTimeFormat (Page 5416) *	P	P	Specifies the format for the time or period used to display the alarms.
MessageColumnAdd (Page 5417) *	P	P	Adds the selected alarm block from the list of available alarm blocks to the list of selected alarm blocks.
MessageColumnCount (Page 5418) *	P	P	Specifies the number of alarm blocks to be displayed in the alarm list in runtime.
MessageColumnIndex (Page 5418)	P	P	References an alarm block selected for the alarm list.
MessageColumnName (Page 5419) *	P	P	Shows the name of the alarm block from the alarm list, which is referenced by means of the "MessageColumnIndex" property.
MessageColumnRemove (Page 5419) *	P	P	Cuts the marked alarm block from the list of selected alarm blocks and pastes it into the list of existing alarm blocks.
MessageColumnRepos (Page 5420) *	P	P	Changes the sorting order of the alarm blocks. The "Up" and "Down" commands move the selected alarm block accordingly in the list.
MessageColumnSort (Page 5420) *	P	P	Specifies the sorting order for the alarm block that is referenced in "MessageColumnIndex" .
MessageColumnSortIndex (Page 5421) *	P	P	Specifies the sorting order for the alarm block that is referenced in "MessageColumnIndex".
MessageColumnVisible (Page 5421) *	P	P	The list shows the selected alarm blocks of the alarm list or hitlist that are are used in the alarm view in runtime.
Moveable (Page 5425)	P	P	Specifies whether the window can be moved in runtime.
MsgFilterSQL (Page 5425)	P	P	Specifies one or several SQL statements for user-defined selection of the alarms.
Name (Page 5426)	P	P	Returns the object name as STRING.
OperatorAlarms	P	P	Specifies the operator message settings.
OperatorMessageId (Page 5435) *	P	P	Default assignment of the ID number and trigger event in the alarm view.
OperatorMessageIndex (Page 5436) *	P	P	References the operator message event.
OperatorMessageName (Page 5437) *	P	P	On alarm events, this displays the name that is referenced by means of "OperatorMessageIndex" for operator alarms.

Properties	Read	Write	Description
OperatorMessageNumber (Page 5437) *	P	P	Specify an alarm number for the operator message of the selected alarm event if you do not use the operator message of WinCC.
OperatorMessageSelected (Page 5438) *	P	P	Activate the alarm events of the list that trigger operator messages.
OperatorMessageSource1 (Page 5438) *	P	P	Specify the alarm block of the operated alarm to be added to "Process value block 1" of the operator message configured in Source.
OperatorMessageSource10 (Page 5443) *	P	P	Specify the alarm block of the operated alarm to be added to "Process value block 10" of the operator message configured in Source.
OperatorMessageSource2 (Page 5439) *	P	P	Specify the alarm block of the operated alarm to be added to "Process value block 2" of the operator message configured in Source.
OperatorMessageSource3 (Page 5439) *	P	P	Specify the alarm block of the operated alarm to be added to "Process value block 3" of the operator message configured in Source.
OperatorMessageSource4 (Page 5440) *	P	P	Specify the alarm block of the operated alarm to be added to "Process value block 4" of the operator message configured in Source.
OperatorMessageSource5 (Page 5440) *	P	P	Specify the alarm block of the operated alarm to be added to "Process value block 5" of the operator message configured in Source.
OperatorMessageSource6 (Page 5441) *	P	P	Specify the alarm block of the operated alarm to be added to "Process value block 6" of the operator message configured in Source.
OperatorMessageSource7 (Page 5441) *	P	P	Specify the alarm block of the operated alarm to be added to "Process value block 7" of the operator message configured in Source.
OperatorMessageSource8 (Page 5442) *	P	P	Specify the alarm block of the operated alarm to be added to "Process value block 8" of the operator message configured in Source.
OperatorMessageSource9 (Page 5442) *	P	P	Specify the alarm block of the operated alarm to be added to "Process value block 9" of the operator message configured in Source.
OperatorMessageSourceType1 (Page 5444) *	P	P	Specifies the format of the source content to be transferred.
OperatorMessageSourceType10 (Page 5448) *	P	P	Specifies the format of the source content to be transferred.
OperatorMessageSourceType2 (Page 5444) *	P	P	Specifies the format of the source content to be transferred.
OperatorMessageSourceType3 (Page 5445) *	P	P	Specifies the format of the source content to be transferred.
OperatorMessageSourceType4 (Page 5445) *	P	P	Specifies the format of the source content to be transferred.
OperatorMessageSourceType5 (Page 5446) *	P	P	Specifies the format of the source content to be transferred.
OperatorMessageSourceType6 (Page 5446) *	P	P	Specifies the format of the source content to be transferred.

Properties	Read	Write	Description
OperatorMessageSourceType7 (Page 5447) *	P	P	Specifies the format of the source content to be transferred.
OperatorMessageSourceType8 (Page 5447) *	P	P	Specifies the format of the source content to be transferred.
OperatorMessageSourceType9 (Page 5448) *	P	P	Specifies the format of the source content to be transferred.
PageMode (Page 5449)	P	P	Sets the page orientation (portrait / landscape).
PageModeMessageNumber (Page 5449)	P	P	Defines the number of alarm shown per page when paging the historical alarm list (long-term).
PrintJob	P	P	Specifies a print job
RowScrollbar (Page 5465)	P	P	Specifies whether row scroll bars are displayed.
RowTitleAlignment	P	P	Specifies the type of row title alignment.
RowTitles	P	P	Specifies that row headers will be displayed.
RTPersistence (Page 5466)	P	P	Specifies how online configurations of WinCC are retained.
RTPersistenceAuthorization (Page 5716)	P	P	Specifies the authorization required for online configuration in runtime.
RTPersistenceType (Page 5467)	-	-	Specifies how online configurations of WinCC are retained.
SelectedCellColor (Page 5478)	P	P	Specifies the background color of the selected cell.
SelectedCellForeColor (Page 5479)	P	P	Specifies the font color of the selected cell.
SelectedRowColor (Page 5480)	P	P	Specifies the background color of the selected row.
SelectedRowForeColor (Page 5481)	P	P	Specifies the font color of the selected row.
SelectedTitleColor (Page 5482)	P	P	Specifies the background color of a selected table header.
SelectedTitleForeColor (Page 5482)	P	P	Specifies the font color of the selected table header.
SelectionColoring (Page 5485)	P	P	Specifies whether selection colors are used for cells or rows.
SelectionRect (Page 5486)	P	P	Specifies whether a selection frame is used for the selected cells or rows.
SelectionRectColor (Page 5486)	P	P	Specifies the color of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionRectWidth (Page 5487)	P	P	Specifies the line width of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionType (Page 5488)	P	P	Specifies the number of lines you can mark.
ServerNames (Page 5491)	P	P	Specifies the servers of a distributed system that form the data source for the alarm view.
ShowSortButton (Page 5500)	P	P	Specifies whether the sorting button is shown above the vertical scroll bar.
ShowSortIcon (Page 5501)	P	P	Specifies whether the sorting icon is displayed.
ShowSortIndex (Page 5501)	P	P	Specifies whether the sorting icon is displayed.
ShowTitle (Page 5504)	P	P	Specifies the representation of the control's window title.
Sizeable (Page 5506)	P	P	Specifies that the size of an object can be changed in runtime.
SortSequence (Page 5508)	P	P	Specifies how the sorting order can be changed by mouse click.
StatusbarBackColor (Page 5511)	P	P	Specifies the background color of the status bar.
StatusbarElementAdd (Page 5512) *	P	P	Creates a new, user-defined status bar element.

Properties	Read	Write	Description
StatusbarElementAutoSize (Page 5512) *	P	P	Specifies whether the width of the selected status bar element is set automatically.
StatusbarElementCount (Page 5513) *	P	P	Specifies the number of configurable status bar elements.
StatusbarElementIconId (Page 5514) *	P	P	Default assignment of the ID number and icon of a status bar element.
StatusbarElementId (Page 5514) *	P	P	Unique ID of the selected status bar element.
StatusbarElementIndex (Page 5515)	P	P	References a status bar element. You can use this property to assign the values of other properties to a specific element of the status bar.
StatusbarElementName (Page 5516) *	P	P	Displays the object name of the selected status bar element.
StatusbarElementRemove (Page 5516) *	P	P	Removes the selected status bar element.
StatusbarElementRename (Page 5517) *	P	P	Renames the user-defined status bar element that is referenced by means of the "StatusbarElementIndex" property.
StatusbarElementRepos *	P	P	Specifies the index that is assigned to the element.
StatusbarElements (Page 5517)	-	-	Specifies the elements that are to be shown in the status bar.
StatusbarElementText *	P	P	Specifies the text of the currently selected status bar element.
StatusbarElementTooltipText (Page 5518) *	P	P	Specifies the tooltip text for the user-defined status bar element.
StatusbarElementUserDefined (Page 5518) *	P	P	Indicates whether the project engineer has added the toolbar button as a new user-defined element.
StatusbarElementVisible (Page 5519) *	P	P	In the list, activate the status bar elements that you wish to display in runtime.
StatusbarElementWidth (Page 5519) *	P	P	Specifies the width of the selected status bar element in pixels.
StatusbarFont	-	-	Status bar font
StatusbarFontColor (Page 5520)	P	P	Specifies the color of the text in the status bar.
StatusbarShowTooltips (Page 5521)	P	P	Specifies whether tooltips are displayed for the elements of the status bar in runtime.
StatusbarText (Page 5521)	P	P	Default text in the status bar.
StatusbarUseBackColor (Page 5522)	P	P	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 5523)	P	P	Specifies whether the status bar of the control is displayed.
TableColor	P	P	Specifies the background color of the rows. Use this to open the color selection dialog.
TableColor2	P	P	Specifies the background color of "Row color 2".
TableForeColor	P	P	Specifies the font color of "Row color".
TableForeColor2	P	P	Specifies the font color of "Row color 2".
TimeBase (Page 5540)	P	P	Sets the timebase.
TitleColor	P	P	Specifies the color of the header.
TitleCut	P	P	Specifies whether the contents of the fields of a header should be abbreviated if the column is too narrow.
TitleDarkShadowColor	P	P	Specifies the color of the dark side of shading.
TitleForeColor	P	P	Specifies the font color of the table header for the selected status.

Properties	Read	Write	Description
TitleGridLineColor	P	P	Specifies the color of separation lines in the table header.
TitleLightShadowColor	P	P	Specifies the color of the bright side of shading.
TitleSort	P	P	Specifies how sorting by column title is triggered.
TitleStyle	P	P	Specifies whether a shading color is used for the table header.
ToolbarAlignment	P	P	Specifies or returns the position of the toolbar.
ToolbarBackColor	P	P	Specifies the background color of the toolbar.
ToolbarButtonActive *	P	P	Specifies whether the function linked to the button is activated in runtime.
ToolbarButtonAdd *	P	P	Creates a new, user-defined button function.
ToolbarButtonBeginGroup *	P	P	Specifies whether a separator is inserted before the selected button function.
ToolbarButtonClick	P	P	Clicks a toolbar button.
ToolbarButtonCount *	P	P	Specifies the number of configurable button functions.
ToolbarButtonEnabled *	P	P	Specifies whether a user-defined toolbar button can be operated.
ToolbarButtonHotKey *	P	P	Shows the hotkey for the selected button function.
ToolbarButtonId *	P	P	Unique ID number for the selected button function.
ToolbarButtonIndex	P	P	References a button function.
ToolbarButtonLocked *	P	P	Specifies whether the clicked, pressed in state is displayed for a user-defined toolbar button.
ToolbarButtonName *	P	P	Shows the name of the selected button function.
ToolbarButtonPasswordLevel *	P	P	Shows the authorization for the selected button function.
ToolbarButtonRemove *	P	P	Removes the selected button function from the list.
ToolbarButtonRename *	P	P	Renames the user-defined toolbar element that is referenced by means of the "ToolbarButtonIndex" property.
ToolbarButtonRepos *	P	P	Changes the sequence of button functions.
ToolbarButtons	-	-	Specifies which buttons are available for the toolbar.
ToolbarButtonTooltipText *	P	P	Specifies the tooltip text for the button.
ToolbarButtonUserDefined *	P	P	Indicates whether the project engineer has added the toolbar button as a new user-defined button.
ToolbarButtonVisible *	P	P	Specifies that the button is visible.
ToolbarShowTooltips	P	P	Specifies whether the tooltips for the button functions are displayed in runtime.
ToolbarUseBackColor	P	P	Specifies whether the background color for the toolbar is shown.
ToolbarUseHotKeys	P	P	Specifies whether the hotkeys for the button functions are activated in runtime.
ToolbarVisible	P	P	Specifies whether the toolbar of the control is displayed.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
UseMessageColor (Page 5630)	P	P	Specifies whether scrolling is enabled.
UseSelectedTitleColor (Page 5630)	P	P	Specifies whether a selection color is used for the headers of selected table cells.
UseTableColor2 (Page 5631)	P	P	Specifies whether a second row color is used for the representation of the table.
VerticalGridLines (Page 5644)	P	P	Specifies whether vertical separation lines are displayed.

Properties	Read	Write	Description
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-20 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
AttachDB (Page 5832)	P, A	Executes the "Connect backup" button function of the control.
CopyRows (Page 5833)	P	Executes the "Copy rows" button function of the control.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.
DetachDB (Page 5837)	P, A	Executes the "Disconnect backup" button function of the control.
Export (Page 5838)	P, A	Executes the "Export log" or "Export data" button function of the control.
GetHistlistColumnCollection (Page 5841)	P	Returns the list of all column objects of the alarm view hitlist as "ICCAxCollection" type.
GetHitlistColumn (Page 5842)	P	Returns the column object of the alarm view hitlist designated by name or index as "ICCAxMessageColumn" type.
GetMessageBlock (Page 5843)	P	Returns the alarm block object of the alarm view designated by name or index as "ICCAxMessageBlock" type.
GetMessageBlockCollection (Page 5844)	P	Returns the list of all alarm block objects of the alarm view as "ICCAxCollection" type.
GetMessageColumn (Page 5846)	P	Returns the column object of the alarm view designated by name or index as "ICCAxMessageColumn" type.
GetOperatorMessage (Page 5847)	P	Returns the operator message object of the alarm view designated by name or index as "ICCAxOperatorMessage" type.
GetMessageColumnCollection (Page 5848)	P	Returns the list of all column objects of the message view as "ICCAxCollection" type.
GetOperatorMessageCollection (Page 5849)	P	Returns the list of all operator message objects of the alarm view as "ICCAxCollection" type.
GetRow (Page 5850)	P	Returns the row object defined by its row number in the table-based controls as "ICCAxDataRow" type.
GetRowCollection (Page 5851)	P	Returns the list of all row objects of the table-based controls as "ICCAxDataRowCollection" type.
GetSelectedRow (Page 5858)	P	Returns the selected row object of a table-based control as "ICCAxDataRow" type.
GetSelectedRows (Page 5859)	P	Returns the selected row objects of a table-based control as type "ICCAxDataRow" for multiple selection.
GetStatusbarElement (Page 5865)	P, A	Returns the element of the control's status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 5866)	P, A	Returns the list of all status bar elements of the control as "ICCAxCollection" type.

Methods	Valid	Description
GetToolbarButton (Page 5873)	P, A	Returns the button function designated by name or index on the control toolbar as "ICCAxToolbarButton" type.
GetToolbarButtonCollection (Page 5874)	P, A	Returns the list of all button functions of the control toolbar as "ICCAx-Collection" type.
HideAlarm (Page 5890)	P	Executes the "Hide alarm" button function of the alarm view.
LockAlarm (Page 5893)	P	Executes the "Disable alarm" button function of the alarm view.
LoopInAlarm (Page 5893)	P	Executes the "Loop in alarm" button function of the alarm view.
MoveToFirstLine (Page 5895)	P	Executes the "First alarm" button function of the alarm view.
MoveToFirstPage (Page 5895)	P	Executes the "First page" button function of the alarm view.
MoveToLastLine (Page 5896)	P	Executes the "Last alarm" button function of the alarm view.
MoveToLastPage (Page 5897)	P	Executes the "Last page" button function of the alarm view.
MoveToNextLine (Page 5898)	P	Executes the "Next alarm" button function of the alarm view.
MoveToNextPage (Page 5898)	P	Executes the "Next page" button function of the alarm view.
MoveToPreviousLine (Page 5899)	P	Executes the "Previous alarm" button function of the alarm view.
MoveToPreviousPage (Page 5900)	P	Executes the "Previous page" button function of the alarm view.
Print (Page 5903)	P, A	Executes the "Print" button function of the control.
QuitHorn (Page 5904)	P	Executes the "Acknowledge central alarm generator" button function of the alarm view.
QuitSelected (Page 5904)	P	Executes the "Single acknowledgment" button function of the alarm view.
QuitVisible (Page 5905)	P	Executes the "Group acknowledgment" button function of the alarm view.
SelectAll (Page 5918)	P	Selects all rows in a table-based control.
SelectRow (Page 5919)	P	Selects a specific row in a table-based control.
ShowComment (Page 5921)	P	Executes the "Comment dialog" button function of the recipe view.
ShowDisplayOptionsDialog (Page 5922)	P	Executes the "Display options dialog" button function of the alarm view.
ShowEmergencyQuitDialog (Page 5922)	P	Executes the "Single acknowledgment" button function of the alarm view.
ShowHelp (Page 5923)	P, A	Executes the "Help" button function of the control.
ShowHideList (Page 5923)	P	Executes the "List of alarm to hide" button function of the alarm view.
ShowHitList (Page 5924)	P	Executes the "Hitlist" button function of the alarm view.
ShowInfoText (Page 5924)	P	Executes the "About dialog" button function of the alarm view.
ShowLockDialog (Page 5925)	P	Executes the "Lock dialog" button function of the alarm view.
ShowLockList (Page 5925)	P	Executes the "Lock list" button function of the alarm view.
ShowLongTermArchiveList (Page 5926)	P	Executes the "Historical alarm list (long-term)" button function of the alarm view.
ShowMessageList (Page 5926)	P	Executes the "Alarm list" button function of the alarm view.
ShowPropertyDialog (Page 5927)	P, A	Executes the "Configuration dialog" button function of the control.
ShowSelectionDialog (Page 5929)	P	Executes the "Selection dialog" button function of the alarm view.
ShowShortTermArchiveList (Page 5930)	P	Executes the "Historical alarm list (short-term)" button function of the alarm view.
ShowSortDialog (Page 5931)	P	Executes the "Sorting dialog" button function of the alarm view.
ShowTimebaseDialog (Page 5932)	P	Executes the "Timebase dialog" button function of the alarm view.
UnhideAlarm (Page 5935)	P	Executes the "Unhide alarm" button function of the alarm view.
UnlockAlarm (Page 5935)	P	Executes the "Unlock alarm" button function of the alarm view.

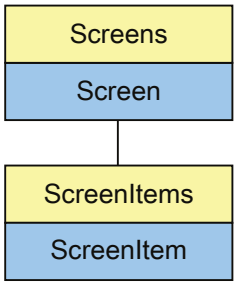
Methods	Valid	Description
UnselectAll (Page 5936)	P	Removes all selections from the cells of a table-based control.
UnselectRow (Page 5936)	P	Removes the selections from a specific cell of a table-based control.

See also

Screen object (list) (Page 5007)

AlarmView

Description



Represents the "Alarm view" object. The MessageView object is an element of the ScreenItems list.

If you change the settings for this object with a user-defined function, the changed settings are retained even after the screen is called again.

Note

The object "Simple AlarmView" cannot be dynamized with a user-defined function.

Abbreviation	Validity
Pa	Panels
A	RT Advanced
P	RT Professional

Table 12-21 Properties

Properties	Read	Write	Description
AlarmAreaHeight **			
AlarmAreaWidth **			
AlarmClasses	-	-	Specifies the alarm classes to be reported in runtime.
AlarmLog	-	-	Specifies the alarm types to be displayed in the alarm view.
AlarmSource	-	-	Specifies that alarms should be displayed.
Authorization (Page 5245)	-	-	Specifies the operating rights of the selected object in runtime.
BackColor	A	A	Specifies the background color of the selected object.
ButtonBarElements **			Specifies that the buttons will be displayed.
ButtonBarStyle	-	-	
Columns	-	-	Specifies the columns to be displayed.
ColumnsMoveable	-	-	Specifies that the columns can be moved.
ColumnTextAckGroup	-	-	Specifies the header for the "AckGroup" column.
ColumnTextAlarmState	-	-	Specifies the header for the "State" column.
ColumnTextAlarmText	-	-	Specifies the header for the "Text" column.
ColumnTextClassName	-	-	Specifies the header for the "Class Name" column.
ColumnTextDate	-	-	Specifies the header for the "Date" column.
ColumnTextDevice	-	-	Specifies the header for the "Device" column.
ColumnTextDiagnosable	-	-	Specifies the header for the "Diagnosable" column.
ColumnTextNumber	-	-	Specifies the header for the "Number" column.
ColumnTextTime	-	-	Specifies the header of the "Time" column.
CountOfLinesPerAlarms	-	-	
CountOfVisibleAlarms	-	-	
Enabled	A	A	Specifies whether the selected object can be operated in runtime.
FilterTag **			Specifies a tag of the string type for filtering alarm texts.
FilterText **			Specifies the text for filtering the alarm text.
FitToSize	-	-	Specifies that the size of the object should be automatically adapted to the content.
Flashing	-	-	Specifies that the text block flashes.
FocusColor	A	A	Specifies the color for the focus frame of the selected object when it is in focus.
FocusWidth	A	A	Specifies the border width of the specified object when the object is in focus.
ForeColor	-	-	Specifies the font color of the text for the selected object.
GridlineColor	A	A	Specifies the color for the grid lines.

Properties	Read	Write	Description
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HorizontalScrollingEnabled	-	-	
IsRunningUnderCE *	-	-	
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
LineAlarmView **			
Name (Page 5426)	A	A	Returns the object name as STRING.
PreferredUseOnAck	-	-	
S7Device	-	-	
SecurityForSimpleViewEnabled **			
SelectionBackColor (Page 5484)	A	A	Specifies the background color of the selected cells.
SelectionForeColor (Page 5485)	A	A	Specifies the foreground color of the selected cells.
SeparateLineForAlarmText	-	-	
ShowAcknowledgeButton	-	-	
ShowAlarmsFromDate (Page 5492)	A	A	Specifies that only those message events are displayed that are saved in this tag.
ShowAlarmsToAcknowledge	-	-	Specifies that unacknowledged alarms should be displayed.
ShowColumnHeaders	-	-	
ShowHelpButton	-	-	
ShowHorizontalGridlines	-	-	
ShowLoopInAlarmButton	-	-	Specifies that the "Loop-In-Alarm" button will be displayed.
ShowMilliseconds	-	-	Display time in milliseconds.
ShowPendingAlarms	-	-	
SortByTimeDirection (Page 5508)	-	-	Specifies whether the last incoming alarm is displayed at the top of the "AlarmControl" object.
SortByTimeEnabled (Page 5722)	A	A	Specifies whether the sorting order of alarms based on the time in the "AlarmControl" object can be changed.
TableBackColor (Page 5525)	A	A	Specifies the background color of the table cells for the selected object.
TableFont	-	-	Specifies the font in the table.
TableForeColor	A	A	Specifies the font color of "Row color".
TableHeaderBackColor (Page 5529)	A	A	Specifies the background color in the header of the table of the selected object.
TableHeaderFont	-	-	Specifies the text color of the header.
TableHeaderForeColor (Page 5530)	A	A	Specifies the text color in the header of the table of the selected object.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
VerticalScrollbarEnabled	-	-	
VerticalScrollingEnabled	-	-	
ViewType	-	-	Specifies the alarm view type.
ViewTypeForSaveStream **			Specifies the display type for the Save Stream.

Properties	Read	Write	Description
Visible (Page 5645) *	A	A	Specifies whether the selected object is visible.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES, ** no relevant information available

Table 12-22 Methods

Methods	Valid	Description
Activate (Page 5825)	???	Activates the permanent window or the root screen.
???		

See also

Screen object (list) (Page 4975)

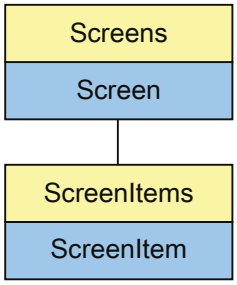
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

Bar

Description



Represents the "Bar" object. The Bar object is an element of the ScreenItems list.

Type identifier in VBS

HMIBar

Abbreviation	Validity
Pa	Panels
A	RT Advanced
P	RT Professional

Table 12-23 Properties

Properties	Read	Write	Description
AlarmLowerLimit (Page 5236)	P	P	Specifies the low limit at which the interrupt is triggered.
AlarmLowerLimitColor (Page 5237)	P	P	Specifies the bar color of the "AlarmLowerLimit" limit.
AlarmLowerLimitEnabled (Page 5237)	P	P	Specifies whether the "AlarmLowerLimit" limit is monitored.
AlarmLowerLimitRelative (Page 5238)	P	P	Specifies whether the low limit at which the interrupt is triggered is evaluated as a percentage or an absolute value.
AlarmUpperLimit (Page 5238)	P	P	Specifies the high limit that triggers the interrupt.
AlarmUpperLimitColor (Page 5239)	P	P	Specifies the bar color of the "AlarmUpperLimit" limit.

Properties	Read	Write	Description
AlarmUpperLimitEnabled (Page 5240)	P	P	Specifies whether the "AlarmUpperLimit" limit is monitored.
AlarmUpperLimitRelative (Page 5240)	P	P	Specifies whether the high limit at which the interrupt is triggered is evaluated as a percentage or an absolute value.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
AverageLast15Values	P	P	Specifies that the average of the last 15 values is shown.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackFlashingColorOff (Page 5256) *	P	P	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 5257) *	P	P	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258) *	P	P	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258) *	P	P	Specifies the flash rate of the background for the selected object.
BarBackColor (Page 5259)	P, A	P, A	Specifies the color of the bar background for the selected object.
BarBackFillStyle (Page 5260)	P	P	Specifies the fill style for the bar.
BarBackFlashingColorOff	-	-	Specifies the border background color of the object for the "OFF" state.
BarBackFlashingColorOn	-	-	Specifies the border background color of the object for the "ON" state.
BarBackFlashingEnabled	-	-	
BarBackFlashingRate	-	-	Specifies the flash rate of the background.
BarEdgeStyle	-	-	Specifies the style of the border line.
BarOrientation	P	P	Specifies the alignment of the bar.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderStyle3D	-	-	Specifies whether the object has a 3D border shading.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
ColorChangeHysteresis (Page 5297)	P	P	Specifies the hysteresis as a percentage of the display value.
ColorChangeHysteresisEnabled (Page 5297)	P	P	Specifies whether the object is displayed with hysteresis.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
CountDivisions (Page 5308)	P	P	Specifies the number of segments into which the bar will be split by means of the large scale tick marks.

Properties	Read	Write	Description
CountSubDivisions (Page 5308)	A	A	Specifies the number of scale tick marks between two main tick marks of the "Bar" object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	P	P	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340) *	-	-	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342) *	-	-	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343) *	-	-	Specifies whether the border line of the selected object flashes in runtime.
FlashingOnLimitViolation	-	-	Specifies that flashing is activated when a limit is violated.
FlashingRate (Page 5344) *	-	-	Specifies the flash rate of the border line for the selected object.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	P	P	Specifies whether the text of the selected object is shown in bold.
FontName (Page 5351)	P	P	Specifies the font of the selected object.
FontSize (Page 5351)	P	P	Specifies the font size of the text for the selected object.
ForeColor (Page 5353)	P, A	P, A	Specifies the font color of the text for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
IntegerDigits (Page 5378)	P	P, A	Specifies the number of digits to the left of the decimal point (0 to 20).
LargeTickLabelingStep	-	-	Specifies which scale sections are labeled.
LargeTicksBold (Page 5381)	P	P	Specifies whether the long tick marks of a scale are shown in bold.
LargeTicksSize (Page 5381)	P	P	Specifies the length of the long tick marks of a scale.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
Limit4LowerLimit (Page 5389)	P	P	Specifies the low limit for "Reserve4".
Limit4LowerLimitColor (Page 5390)	P	P	Specifies the color for the "Reserve4" low limit.
Limit4LowerLimitEnabled (Page 5390)	P	P	Specifies whether the "Reserve4" low limit is monitored.
Limit4LowerLimitRelative (Page 5391)	P	P	Specifies whether the "Reserve4" low limit is evaluated as a percentage or an absolute value.
Limit4UpperLimit (Page 5392)	P	P	Specifies the high limit for "Reserve4".
Limit4UpperLimitColor (Page 5392)	P	P	Specifies the color for the "Reserve4" high limit.
Limit4UpperLimitEnabled (Page 5393)	P	P	Specifies whether the "Reserve4" high limit is monitored.
Limit4UpperLimitRelative (Page 5393)	P	P	Specifies whether the "Reserve4" high limit is evaluated as a percentage or an absolute value.
Limit5LowerLimit (Page 5394)	P	P	Specifies the low limit for "Reserve5".

Properties	Read	Write	Description
Limit5LowerLimitColor (Page 5394)	P	P	Sets the color for the "Reserve5" low limit.
Limit5LowerLimitEnabled (Page 5395)	P	P	Specifies whether the "Reserve5" low limit is monitored.
Limit5LowerLimitRelative (Page 5396)	P	P	Specifies whether the "Reserve5" low limit is evaluated as a percentage or an absolute value.
Limit5UpperLimit (Page 5396)	P	P	Specifies the high limit for "Reserve5".
Limit5UpperLimitColor (Page 5397)	P	P	Specifies the color for the "Reserve5" high limit.
Limit5UpperLimitEnabled (Page 5397)	P	P	Specifies whether the "Reserve5" high limit is monitored.
Limit5UpperLimitRelative (Page 5398)	P	P	Specifies whether the "Reserve5" high limit is evaluated as a percentage or an absolute value.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line ends.
MaximumValue (Page 5406)	P, A	P, A	Specifies the maximum value of the scale in the selected object.
MinimumValue (Page 5422)	P, A	P, A	Specifies the minimum value of the scale in the selected object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
Precision (Page 5455)	P	P	Specifies the number of decimal places (0 to 20).
ProcessValue (Page 5456)	P, A	P, A	Specifies the default for the value to be displayed.
ScaleColor (Page 5469)	P, A	P, A	Specifies the color of the scale of the selected object.
ScaleGradation (Page 5469)	P, A	P, A	Specifies the distance between two large tick marks of the scale.
ScaleLabelFieldLength *	-	-	Distance to the position of the axis label.
ScaleLabelingDoubleLined	-	-	Specifies that the horizontal bar orientation of the scale has two lines.
ScalePosition (Page 5470)	P	P	Specifies the position of the scale of the selected object.
ScaleStart	-	-	Specifies the minimum of the scale.
ScalingType (Page 5474)	P	P	Specifies the type of bar scaling.
SegmentColoring (Page 5477)	P, A	P, A	Specifies the type of color change to be displayed in the "Bar" object if limits are exceeded.
ShowBadTagState (Page 5493)	P	P	Specifies whether an object is grayed out if its associated tag has a bad quality code or tag state.
ShowLargeTicksOnly (Page 5496)	P	P	Specifies or returns the length of axis sections in pixels.
ShowLimitLines	-	-	Specifies the display of a line at the limits.
ShowLimitMarkers (Page 5496)	P	P	Specifies whether the limit values are shown as a scale value.
ShowScale (Page 5499)	P	P	Specifies whether the values are also shown in a scale.
ShowSignForPositiveLabel	-	-	Specifies that a "+" is displayed for positive labels.
ShowTickLabels (Page 5503)	-	-	Specifies whether the label is shown in the scale.
ShowTrendIndicator (Page 5506)	P	P	Specifies whether the trend (rising or falling) of the measured value to be monitored is indicated by means of a small arrow.
StartValue (Page 5510)	P	P	Specifies the zero point value for the scale display.
ToleranceLowerLimit (Page 5563)	P	P	Specifies whether the "ToleranceLowerLimit" limit is monitored.
ToleranceLowerLimitColor (Page 5564)	P	P	Specifies the color for the "ToleranceLowerLimit" low limit.

Properties	Read	Write	Description
ToleranceLowerLimitEnabled (Page 5565)	P	P	Specifies whether the "ToleranceLowerLimit" limit is monitored.
ToleranceUpperLimit (Page 5566)	P	P	Specifies whether the "ToleranceUpperLimit" limit is monitored.
ToleranceUpperLimitColor (Page 5566)	P	P	Specifies the color for the "ToleranceUpperLimit" high limit.
ToleranceUpperLimitEnabled (Page 5567)	P	P	Specifies whether the "ToleranceUpperLimit" limit is monitored.
ToleranceUpperLimitRelative (Page 5567)	P	P	Specifies whether the "ToleranceHigh" low limit is evaluated as a percentage or an absolute value.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
TrendIndicatorColor (Page 5595)	P	P	Specifies the color for the trend view.
Unit (Page 5622)	P, A	P, A	Specifies the unit of measurement in the "IOField" object.
UseAutoScaling	-	-	Specifies auto-scaling.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
UseExponentialFormat (Page 5629)	P	P	Specifies whether the numbers are displayed with exponentials (e.g. "1.00e+000").
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
WarningLowerLimit (Page 5648)	P	P	Specifies the "WarningLowerLimit" low limit.
WarningLowerLimitColor (Page 5649)	P	P	Specifies the color for the "WarningLowerLimit" low limit.
WarningLowerLimitEnabled (Page 5650)	P	P	Specifies whether the "WarningLowerLimit" limit is monitored.
WarningLowerLimitRelative (Page 5650)	P	P	Specifies whether the "WarningLowerLimit" low limit is evaluated as a percentage or an absolute value.
WarningUpperLimit (Page 5653)	P	P	Specifies the "WarningUpperLimit" high limit.
WarningUpperLimitColor (Page 5653)	P	P	Specifies the color for the "WarningUpperLimit" high limit.
WarningUpperLimitEnabled (Page 5654)	P	P	Specifies whether the "WarningUpperLimit" limit is monitored.
WarningUpperLimitRelative (Page 5654)	P	P	Specifies whether the "WarningUpperLimit" high limit is evaluated as a percentage or an absolute value.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.
ZeroPoint (Page 5678)	P	P	Specifies the position of the zero point as a percentage of the bar height.

* not visible in the ES

Table 12-24 Methods

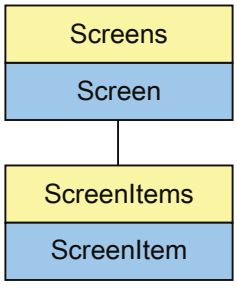
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

BatteryView

Description



Represents the "Charge condition" object. The BatteryView object is an element of the ScreenItems list.

Abbreviation	Validity
Pa	Panels
A	RT Advanced
P	RT Professional

Table 12-25 Properties

Properties	Read	Write	Description
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	A	-	Returns the object name as STRING.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-26 Methods

Methods	Valid	Description
		Not found

See also

Screen object (list) (Page 4975)

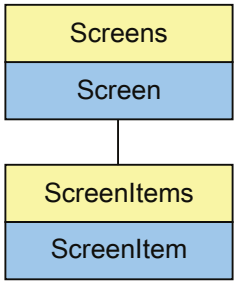
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

Button

Description



Represents the "Button" object. The Button object is an element of the ScreenItems list.

The availability of the following object properties depends on the selected mode of the "button":

Property	"Text"	"Text list"	"Graphic"
TextOff	x	--	--
TextOn	x	--	--

Type identifier in VBS

HMIButton

Abbreviation	Validity
Pa	Panels
A	RT Advanced
P	RT Professional

Table 12-27 Properties

Properties	Read	Write	Description
AdaptBorder (Page 5232)	P	P	Specifies whether the border of the specified object is dynamically adapted to the text size.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.

Properties	Read	Write	Description
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackFlashingColorOff (Page 5256)	P	P	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 5257)	P	P	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258)	P	P	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258)	P	P	Specifies the flash rate of the background for the selected object.
BitNumber (Page 5264)	-	-	Specifies the bit whose status must change to trigger a value change.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderBrightColor3D (Page 5268)	P	P	Specifies the border color of the following border parts in the 3D display of the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderShadeColor3D (Page 5277)	P	P	Specifies the border color of the following border parts in the 3D display of the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderStyle3D	A	A	Specifies whether the object has a 3D border shading.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
BorderWidth3D (Page 5281)	P	P	Specifies the width of the border for 3D display of the selected object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	P	P	Specifies the line style of the selected object.
Enabled (Page 5323)	P, A	P, A	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
FitToLargest	-	-	Specifies that the size of the object depends on the text length in all languages.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.

Properties	Read	Write	Description
FocusColor (Page 5346)	A	A	Specifies the color for the focus frame of the selected object when it is in focus.
FocusWidth (Page 5347)	A	A	Specifies the border width of the specified object when the object is in focus.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	P	P	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 5350)	P	P	Specifies whether the text of the selected object is shown in italics.
FontName (Page 5351)	P	P	Specifies the font of the selected object.
FontSize (Page 5351)	P	P	Specifies the font size of the text for the selected object.
FontUnderline (Page 5352)	P	P	Specifies whether the text of the selected object is underlined.
ForeColor (Page 5353)	P, A	P, A	Specifies the font color of the text for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HelpText	A	-	Returns the tooltip that is shown in runtime as user help for the specified object.
HorizontalAlignment (Page 5372)	P, A	P, A	Specifies the horizontal alignment of the text within the selected object.
HotKey (Page 5374)	-	-	Specifies the hotkey.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line ends.
Mode (Page 5424)	P	P	Specifies the field type of the selected object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
PictureAlignment (Page 5451)	P	P	Specifies or returns the display type of the background image in the process picture.
PictureList	-	-	Specifies the graphic list that supplies the object with values.
PictureOff (Page 5452)	P	P	Specifies the screen to be displayed in the "Off" state.
PictureOn (Page 5453)	P	P	Specifies the screen to be displayed in the "On" state.
Pressed (Page 5455)	P	P	Specifies whether the selected object is displayed in a pressed state.
ProcessValue	-	-	Specifies the default for the value to be displayed.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
StyleSettings (Page 5524)	P	P	Specifies the display style for the object:
TextList (Page 5534)	-	-	A list that contains the assignments between the output value and the output text to be actually output.
TextOff (Page 5534)	P, A	P, A	Specifies the text to be displayed in the "Off" state of the selected object.
TextOn	A	A	Specifies the text to be displayed in the "On" state of the selected object.
TextOrientation (Page 5536)	P	P	Specifies the text orientation of the selected object.
Toggle (Page 5562)	P	P	Specifies whether the selected object engages after it has been operated in runtime.
ToolTipText (Page 5581)	P	P, A	Specifies the tooltip text.

Properties	Read	Write	Description
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
TransparentColorPictureOff (Page 5588)	P	P	Specifies the color to be set to "transparent" for the "Off" state of the assigned bitmap object.
TransparentColorPictureOn (Page 5589)	P	P	Specifies the color to be set to "transparent" for the "On" state of the assigned bitmap object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
UseTransparentColorPicture-Off (Page 5633)	P	P	Specifies whether the transparent color defined in the "TransparentColorPictureOff" property is used for the "Off" state.
UseTransparentColorPictur-eOn (Page 5633)	P	P	Specifies whether the transparent color defined in the "TransparentColorPictureOn" property is used for the "On" state.
VerticalAlignment (Page 5643)	P, A	P, A	Specifies the vertical alignment of the text for the selected object.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.
WindowsStyle (Page 5659)	P	P	Specifies whether the object is displayed in the general Windows style.

Table 12-28 Methods

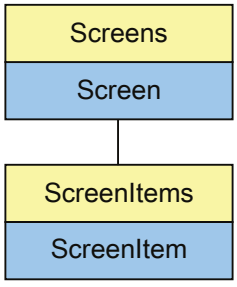
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

ChannelDiagnose

Description



Represents the "ChannelDiagnose" object. The ChannelDiagnose object is an element of the ScreenItems list.

Type identifier in VBS

HMIChannelDiagnose

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-29 Properties

Properties	Read	Write	Description
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
Flashing *	-	-	Specifies that the text block flashes.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
Name (Page 5426)	P	-	Returns the object name as STRING.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.

Properties	Read	Write	Description
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-30 Methods

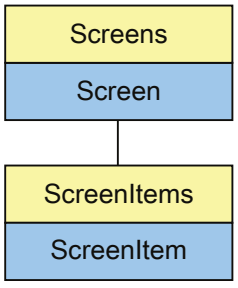
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

CheckBox

Description



Represents the "Check box" object. The CheckBox object is an element of the ScreenItems list.

Type identifier in VBS

HMICheckBox

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-31 Properties

Properties	Read	Write	Description
AdaptBorder (Page 5232)	P	P	Specifies whether the border of the specified object is dynamically adapted to the text size.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackFlashingColorOff (Page 5256)	P	P	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 5257)	P	P	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258)	P	P	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258)	P	P	Specifies the flash rate of the background for the selected object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
CheckmarkAlignment (Page 5294)	P	P	Specifies whether the fields are right aligned.
CheckmarkCount (Page 5294)	P	P	Specifies the number of fields.
CornerStyle (Page 5306)	-	-	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.

Properties	Read	Write	Description
EdgeStyle (Page 5320)	-	-	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	P	P	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 5350)	P	P	Specifies whether the text of the selected object is shown in italics.
FontName (Page 5351)	P	P	Specifies the font of the selected object.
FontSize (Page 5351)	P	P	Specifies the font size of the text for the selected object.
FontUnderline (Page 5352)	P	P	Specifies whether the text of the selected object is underlined.
ForeColor (Page 5353)	P	P	Specifies the font color of the text for the selected object.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HorizontalAlignment (Page 5372)	P	P	Specifies the horizontal alignment of the text within the selected object.
Index (Page 5375)	P	P	Specifies the background for grid control elements.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	-	-	Specifies the shape of the line ends.
LogOperation (Page 5403)	P	P	Specifies whether, after operating this object, an alarm is output to the alarm system.
Name (Page 5426)	P	P	Returns the object name as STRING.
ProcessValue (Page 5456)	P	P	Specifies the default for the value to be displayed.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
ShowBadTagState (Page 5493)	P	P	Specifies whether an object is grayed out if its associated tag has a bad quality code or tag state.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
Text (Page 5533)	P	P	Specifies the label for the text field.
TextOrientation (Page 5536)	P	P	Specifies the text orientation of the selected object.
Texts	-	-	Specifies the texts for the check boxes.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.

Properties	Read	Write	Description
VerticalAlignment (Page 5643)	P	P	Specifies the vertical alignment of the text for the selected object.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-32 Methods

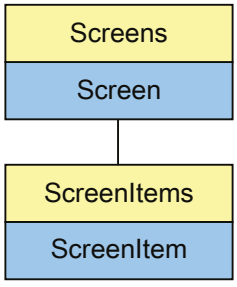
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

Circle

Description



Represents the "Circle" object. The Circle object is an element of the ScreenItems list.

Type identifier in VBS

HMICircle

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-33 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P, A	P, A	Defines the fill pattern of the specified object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P, A	P, A	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P, A	P, A	Specifies the line thickness of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	A	A	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.

Properties	Read	Write	Description
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
Radius	P	P, A	Specifies the radius of the selected object "Circle".
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-34 Methods

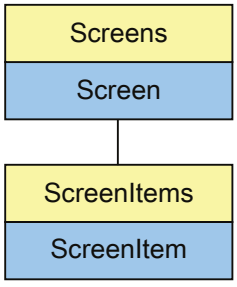
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

CircleSegment

Description



Represents the "CircleSegment" object. The CircleSegment object is an element of the ScreenItems list.

Type identifier in VBS

HMICircleSegment

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-35 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.

Properties	Read	Write	Description
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle	P	P	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
EndAngle (Page 5325)	P	P	Specifies the angle at which the end point of the selected object deviates from the zero position (0°).
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
Name (Page 5426)	P	-	Returns the object name as STRING.
Radius	P	P	Specifies the radius of the selected object "Circle".
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
StartAngle (Page 5509)	P	P	Specifies the SQL statement.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.

Properties	Read	Write	Description
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-36 Methods

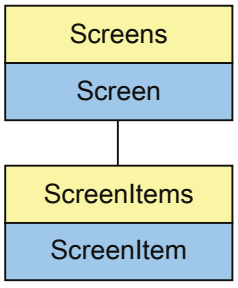
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

CircularArc

Description



Represents the "CircularArc" object. The CircularArc object is an element of the ScreenItems list.

Type identifier in VBS

HMICircularArc

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-37 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
Color (Page 5296)	P	P	Specifies the line color of the selected object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
EndAngle (Page 5325)	P	P	Specifies the angle at which the end point of the selected object deviates from the zero position (0°).
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
Name (Page 5426)	P	-	Returns the object name as STRING.
Radius	P	P	Specifies the radius of the selected object "Circle".
StartAngle (Page 5509)	P	P	Specifies the SQL statement.

Properties	Read	Write	Description
Style (Page 5523)	P	P	Specifies the line style of the selected object.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-38 Methods

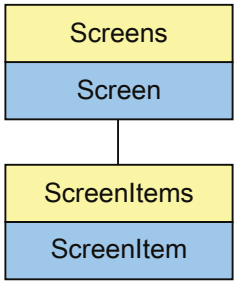
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

Clock

Description



Represents the "Clock" object. The Clock object is an element of the ScreenItems list.

Type identifier in VBS

Clock

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-39 Properties

Properties	Read	Write	Description
Analog (Page 5241)	P	P	Specifies whether the clock is displayed as an analog clock.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
DialColor (Page 5314)	P, A	P, A	Specifies the color of the dial in the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
Flashing	-	-	Specifies that the text block flashes.
Font (Page 5348)	-	-	Specifies or returns the font.

Properties	Read	Write	Description
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HourNeedleHeight (Page 5374)	P, A	P, A	Specifies the length of the hour hand in the "Clock" object.
HourNeedleWidth (Page 5374)	P, A	P, A	Specifies the width of the hour hand in the "Clock" object.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LockSquaredExtent (Page 5402)	P	P	Specifies whether the tag values are downloaded from the logs for the time range to be displayed when you call a screen.
MinuteNeedleHeight (Page 5423)	P, A	P, A	Specifies the length of the minute hand in the "Clock" object.
MinuteNeedleWidth (Page 5423)	P, A	P, A	Specifies the width of the minute hand in the "Clock" object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
NeedleBorderColor (Page 5428)	P, A	P, A	Specifies the name of the data source.
NeedleColor (Page 5429)	P, A	P, A	Specifies the hand color in the "Clock" object.
NeedleFillStyle (Page 5430)	P	P	Specifies the fill style for the clock hand.
Picture (Page 5451)	-	-	Specifies the screen to be displayed in the graphics object in runtime.
SecondNeedleHeight (Page 5476)	P, A	P, A	Specifies the length of the seconds hand in the "Clock" object.
SecondNeedleWidth (Page 5477)	P, A	P, A	Specifies the width of the seconds hand in the "Clock" object.
ShowFocusRectangle (Page 5496)	P	P	Specifies whether the button receives a selection border when it is activated in runtime.
ShowTicks (Page 5504)	P, A	P, A	Specifies whether the tick marks are displayed in the scale of the object.
TicksColor (Page 5537)	P, A	P, A	Specifies the scale display.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-40 Methods

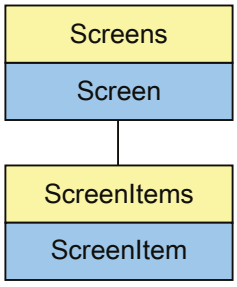
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

Connector

Description



Represents the "Connector" object. The Connector object is an element of the ScreenItems list.

Type identifier in VBS

HMIConnector

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-41 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BorderEndStyle (Page 5270)	P	P	Specifies the type of line ends for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
Color (Page 5296)	P	P	Specifies the line color of the selected object.
ConnectionType (Page 5304)	P	P	Specifies the type of connector. You can select one of two connection types.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
EndStyle (Page 5326)	-	-	Specifies how the line end of the selected object is displayed.
FirstConnectedObjectIndex (Page 5339)	P	P	Specifies the index number of the upper connector point.
FirstConnectedObjectName (Page 5339)	P	P	Specifies the name of the object that is docked to the upper connector point.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
LastConnectedObjectIndex (Page 5382)	P	P	Specifies the index number of the connection point to the last connected object.
LastConnectedObjectName (Page 5382)	P	P	Specifies the name of the object that is docked to the lower connector point.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
Name (Page 5426)	P	P	Returns the object name as STRING.
Points *	-	-	Specifies the corner points.
StartStyle	-	-	Specifies how the line start of the selected object is displayed.

Properties	Read	Write	Description
Style (Page 5523)	P	P	Specifies the line style of the selected object.
SwapFirstWithLastConnection (Page 5524)	P	P	Specifies whether the text in the object is shown horizontally.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies that the font size from the data source should be used.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	-	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-42 Methods

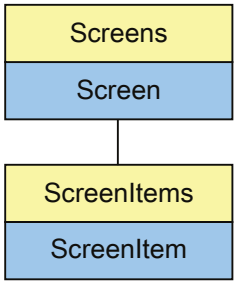
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

DateTimeField

Description



Represents the "Date/time field" object. The DateTimeField object is an element of the ScreenItems list.

Type identifier in VBS

HMIDateTimeField

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-43 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	-	-	Specifies the operating rights of the selected object in runtime.
BackColor	A	A	Specifies the background color of the selected object.
BackFillStyle	-	-	Defines the fill pattern of the specified object.
BorderBackColor (Page 5266) *	-	-	Specifies the background color of the broken border line for the selected object.
BorderColor	A	A	Specifies the line color of the object.

Properties	Read	Write	Description
BorderStyle3D (Page 5279)	A	A	Specifies whether the object has a 3D border shading.
DisplaySystemTime	-	-	Specifies that the system time is displayed.
EdgeStyle	-	-	Specifies the line style of the selected object.
Enabled	A	A	Specifies whether the selected object can be operated in runtime.
FitToLargest	-	-	Specifies that the size of the object depends on the text length in all languages.
Flashing	-	-	Specifies that the text block flashes.
Font (Page 5348) *	-	-	Specifies or returns the font.
ForeColor	A	A	Specifies the font color of the text for the selected object.
Format	-	-	Specifies that the format of the text block is displayed.
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HelpText (Page 5362)	A	-	Returns the tooltip that is shown in runtime as user help for the specified object.
HorizontalAlignment	A	A	Specifies the horizontal alignment of the text within the selected object.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
LineWrap *	-	-	Specifies the line break within the object.
LongDateTimeFormat	-	-	Specifies that the date and time are displayed in long format.
Mode (Page 5424)	-	-	Specifies the field type of the selected object.
Name (Page 5426)	A	-	Returns the object name as STRING.
ProcessValue (Page 5456)	A	A	Specifies the default for the value to be displayed.
ShowDate	-	-	Specifies that the date is to be displayed.
ShowTime	-	-	Specifies that only the time is displayed.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
VerticalAlignment (Page 5643)	A	A	Specifies the vertical alignment of the text for the selected object.
Visible (Page 5645) *	A	A	Specifies whether the selected object is visible.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-44 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.

See also

Screen (Page 5000)

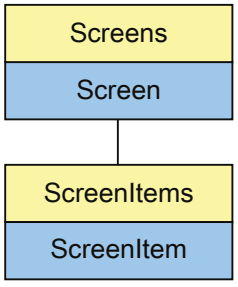
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

DiskSpaceView

Description



Represents the "Disk space view" object. The DiskSpaceView object is an element of the ScreenItems list.

Type identifier in VBS

IXDiskSpaceView

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-45 Properties

Properties	Read	Write	Description
Alarm (Page 5234)	P	P	Specifies the limit for the memory space view from which an alarm is output.
AlarmColor (Page 5235)	P	P	Specifies the alarm types to be displayed in the alarm view.
Drive (Page 5320)	P	P	Specifies the characters of the drive that is to be monitored.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
Free (Page 5354)	P	P	Returns the size of the free memory space.
FreePercent (Page 5355)	P	P	Returns the measured values for the free disk space as a percentage.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Interval (Page 5378)	P	P	Specifies the time interval for updating the displayed measured values.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
Name (Page 5426)	P	-	Returns the object name as STRING.
NormalColor	P	P	Specifies the color of the object in the normal range.
Tolerance (Page 5562)	P	P	Specifies the limit for the storage space display as of which a deviation will be reported.
ToleranceColor (Page 5563)	P	P	Specifies the colors in which the bars of the memory space display are shown as soon as the tolerance range is exceeded.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Total (Page 5585)	P	P	Returns the storage capacity.
Used (Page 5625)	P	P	Specifies whether scrolling is enabled.
UsedPercent (Page 5628)	P	P	Returns the measured values for the occupied memory space as a percentage.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Warning (Page 5647)	P	P	Specifies the limit for the memory space view from which a warning is output.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-46 Methods

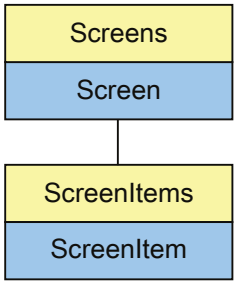
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

Ellipse

Description



Represents the "Ellipse" object. The Ellipse object is an element of the ScreenItems list.

Type identifier in VBS

HMIEllipse

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-47 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.

Properties	Read	Write	Description
BackFillStyle (Page 5254)	P, A	P, A	Defines the fill pattern of the specified object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P, A	P, A	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P, A	P, A	Specifies the line thickness of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle	A	A	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
RadiusHeight	P	P	Specifies the minor axis of the "Ellipsis" object.
RadiusWidth	P	P	Specifies the major axis of the "Ellipsis" object.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.

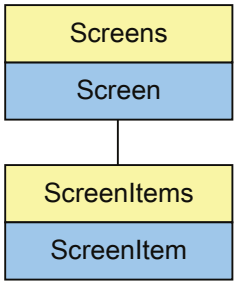
Properties	Read	Write	Description
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-48 Method

Method	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

EllipseSegment

Description



Represents the "EllipseSegment" object. The EllipseSegment object is an element of the ScreenItems list.

Type identifier in VBS

HMIEllipseSegment

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-49 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	P	P	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
EndAngle (Page 5325)	P	P	Specifies the angle at which the end point of the selected object deviates from the zero position (0°).
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Properties	Read	Write	Description
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
Name (Page 5426)	P	-	Returns the object name as STRING.
RadiusHeight	P	P	Specifies the minor axis of the "Ellipsis" object.
RadiusWidth	P	P	Specifies the major axis of the "Ellipsis" object.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
StartAngle (Page 5509)	P	P	Specifies the SQL statement.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-50 Methods

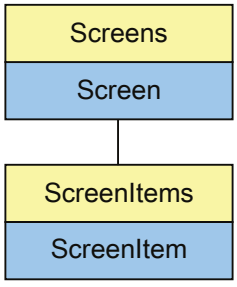
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

EllipticalArc

Description



Represents the "EllipticalArc" object. The EllipticalArc object is an element of the ScreenItems list.

Type identifier in VBS

HMIEllipticalArc

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-51 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
Color (Page 5296)	P	P	Specifies the line color of the selected object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.

Properties	Read	Write	Description
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
EndAngle (Page 5325)	P	P	Specifies the angle at which the end point of the selected object deviates from the zero position (0°).
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	-	-	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	-	-	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	-	-	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	-	-	Specifies the flash rate of the border line for the selected object.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
Name (Page 5426)	P	-	Returns the object name as STRING.
RadiusHeight	P	P	Specifies the minor axis of the "Ellipsis" object.
RadiusWidth	P	P	Specifies the major axis of the "Ellipsis" object.
StartAngle (Page 5509)	P	P	Specifies the SQL statement.
Style (Page 5523)	P	P	Specifies the line style of the selected object.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-52 Methods

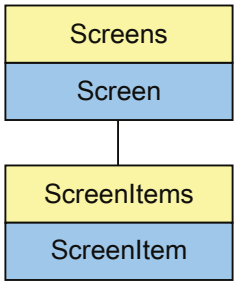
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

ForeignControl

Description



Object Type of ScreenItem Object. Represents the graphic object "Control"

The control object type always assumes the properties of the control type selected. In the case of controls provided by WinCC, the properties are indicated under the description of the corresponding control.

In the case of controls from external suppliers, the control properties are supplied and thus not a part of this description. However, the control properties can be queried using the "Item" property.

Type Identifier in VBS

Special WinCC type descriptions or version-independent ProgID

Use

In the following example, the object with the name "Control1" is moved 10 pixels to the right:

```
'VBS36
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 10
```

Special feature

The controls provided by WinCC return a special ID as the type. It can be found under the topic "Type Identification in VBS" in the individual descriptions of the WinCC controls.

Access to specific properties instead of event parameters of the third-party controls

The parameter list of an event with third-party control is not read in WinCC. To have access to custom parameters of control events, you need to implement properties in the control instead of parameters.

The following code example shows the source code of the "UserControlTestModify" control, in which the "Title" and "Description" properties are defined instead of parameters.

```
public partial class UserControlTestModify : UserControl
    {
        public event MyEventHandler MyEvent = null;

        public string Title
        {
            get;
            private set;
        }

        public string Description
        {
            get;
            set;
        }

        public UserControlTestModify()
        {
            InitializeComponent();
        }
        private void OnMyEvent(Exception e)
        {
            if (this.MyEvent!= null)
            {
               //  Place place data in Properties before invoking event.
                Title = "title";
                Description = "description"; ";
                this.MyEvent.Invoke();
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.OnMyEvent (null);
        }
    }
```

The specific properties of a third-party control are not displayed in WinCC with IntelliSense.

The following example shows how to access the specific "Title" property of the third-party control "FControl" in WinCC :

```
HMIRuntime.Screens("StartScreen").ScreenItems("FControl").Title = "MyForeignControl"
```

ProgID and UserfriendlyName of third-party controls

The version-independent ProgID is returned as the type if non-WinCC controls are used.

It is possible to determine the version-dependent ProgID or "UserfriendlyName" from the ProgID: In the following example, "Control1" is a control embedded in the picture which already returns the version-independent ProgID as a result of the Type property.

Note

Since not every control has a version-dependent ProgID, an error handling measure should be integrated to query the version-dependent ProgID or UserFriendlyName. If no error handling is used, the code is terminated immediately without any result when no ProgID is found.

Determine the version-dependent ProgID as follows:

```
'VBS37
Dim objControl
Dim strCurrentVersion
Set objControl = ScreenItems("Control1")
strCurrentVersion = CreateObject("WScript.Shell").RegRead("HKCR\" & objControl.Type &
"\CurVer\")
MsgBox strCurrentVersion
```

Note

In order that example above works, a multimedia control should be inserted in the picture.

Determine the UserFriendlyName as follows:

```
'VBS38
Dim objControl
Dim strFriendlyName
Set objControl = ScreenItems("Control1")
strFriendlyName = CreateObject("WScript.Shell").RegRead("HKCR\" & objControl.Type & "\")
MsgBox strFriendlyName
```

Note

In order that example above works, a multimedia control should be inserted in the picture.

If a non-WinCC control is used, it is possible that the properties provided by the control have the same names as the general ScreenItem properties. In such cases, the ScreenItem properties have priority. The "hidden" properties of an external control supplier can be accessed using the additional "object" property. Address the properties of an external control supplier as follows:

```
Control.object.type
```

The properties of the ScreenItem object are used in the case of identical names, if you use the following form:

```
Control.type
```

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-53 Properties

Properties	Read	Write	Description
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
Name (Page 5426)	P	-	Returns the object name as STRING.
OCXState (Page 5434) *	-	-	Sets the OCX status.
ProgID (Page 5457) *	-	-	The version-independent ProgID is returned as the type if non-WinCC controls are used.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-54 Methods

Methods	Valid	Description

See also

Screen (Page 5000)

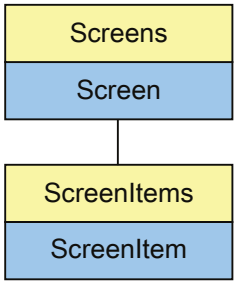
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

FunctionTrendControl

Description



Represents the "f(x)FunctionTrendView" object. The FunctionTrendControl object is an element of the ScreenItems list.

Type identifier in VBS

HMIFunctionTrendControl

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-55 Properties

Properties	Read	Write	Description
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
Height (Page 5358)	P	-	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
ObjectName (Page 5706)	P	-	Returns the object name as STRING.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.

Properties	Read	Write	Description
Type (Page 5809)	P	-	Returns the type of the specified object as STRING.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	-	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-56 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
AttachDB (Page 5832)	P, A	Executes the "Connect backup" button function of the control.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.
DetachDB (Page 5837)	P, A	Executes the "Disconnect backup" button function of the control.
Export (Page 5838)	P, A	Executes the "Export log" or "Export data" button function of the control.
GetStatusbarElement (Page 5865)	P, A	Returns the element of the control's status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 5866)	P, A	Returns the list of all status bar elements of the control as "ICCAxCollection" type.
GetToolbarButton (Page 5873)	P, A	Returns the button function designated by name or index on the control toolbar as "ICCAxToolbarButton" type.
GetToolbarButtonCollection (Page 5874)	P, A	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.
GetTrend (Page 5876)	P, A	Returns the f(t) or f(x) trend view designated by name or index as "ICCAxTrend" or "ICCAxFunctionTrend" type.
GetTrendCollection (Page 5877)	P, A	Returns the list of all trends of the f(t) or f(x) trend view as "ICCAxCollection" type.
GetTrendWindow (Page 5878)	P, A	Returns the trend window object of the f(t) or f(x) trend view designated by name or index as "ICCAxTrendWindow" type.
GetTrendWindowCollection (Page 5879)	P, A	Returns the list of all trend window objects of the f(t) or f(x) trend views as "ICCAxCollection" type.
GetXAxis (Page 5885)	P, A	Returns the X axis object of the f(x) trend view designated by name or index as "ICCAxValueAxis" type.
GetXAxisCollection (Page 5886)	P, A	Returns the list of all X axis objects of the f(x) trend view as "ICCAxCollection" type.
GetYAxis (Page 5888)	P, A	Returns the Y axis object of the f(x) trend view designated by name or index as "ICCAxValueAxis" type.
GetYAxisCollection (Page 5889)	P, A	Returns the list of all Y axis objects of the f(x) trend view as "ICCAxCollection" type.
MoveAxis (Page 5894)	P, A	Executes the "Move axes area" button function of the f(t) and f(x) trend views.
NextTrend (Page 5901)	P, A	Executes the "Next trend" button function of the f(t) and f(x) trend views.
OneToOneView (Page 5901)	P, A	Executes the "Original view" button function of the f(t) and f(x) trend views.

Methods	Valid	Description
PreviousTrend (Page 5903)	P, A	Executes the "Previous trend" button function of the f(t) and f(x) trend views.
Print (Page 5903)	P, A	Executes the "Print" button function of the control.
ShowHelp (Page 5923)	P, A	Executes the "Help" button function of the control.
ShowPropertyDialog (Page 5927)	P, A	Executes the "Configuration dialog" button function of the control.
ShowTagSelection (Page 5931)	P, A	Executes the "Select data connection" button function of the control.
ShowTimeSelection (Page 5932)	P, A	Executes the "Select time range" button function of the control.
ShowTrendSelection (Page 5933)	P, A	Executes the "Select trends" button function of the f(t) and f(x) trend views.
StartStopUpdate (Page 5933)	P, A	Executes the "Start" or "Stop" button function of the control.
ZoomArea (Page 5940)	P, A	Executes the "Zoom area" button function of the f(t) and f(x) trend views.
ZoomInOut (Page 5941)	P, A	Executes the "Zoom +/-" button function of the f(t) and f(x) trend views.
ZoomInOutX (Page 5942)	P, A	Executes the "Zoom X axis +/-" button function of the f(x) trend view.
ZoomInOutY (Page 5943)	P, A	Executes the "Zoom Y axis +/-" button function of the f(x) trend view.
ZoomMove (Page 5943)	P, A	Executes the "Move trend area" button function of the f(t) and f(x) trend views.

See also

BackColor (Page 5251)

Caption (Page 5287)

LoadDataImmediately (Page 5401)

Online (Page 5434)

TimeBase (Page 5540)

ShowRuler (Page 5498)

Screen object (list) (Page 5007)

ApplyProjectSettings (Page 5243)

BorderColor (Page 5268)

BorderWidth (Page 5280)

Closeable (Page 5295)

ConnectTrendWindows (Page 5304)

ExportDirectoryChangeable (Page 5328)

ExportDirectoryname (Page 5329)

ExportFileExtension (Page 5330)

ExportFilename (Page 5330)

ExportFilenameChangeable (Page 5331)

ExportFormatGuid (Page 5331)

ExportFormatName (Page 5332)

ExportParameters (Page 5332)

ExportSelection (Page 5333)
ExportShowDialog (Page 5334)
Font (Page 5348)
GraphDirection (Page 5356)
Layer (Page 5384)
LineColor (Page 5398)
LineWidth (Page 5400)
Moveable (Page 5425)
Name (Page 5426)
RTPersistence (Page 5466)
RTPersistenceType (Page 5467)
UseTrendNameAsLabel (Page 5634)
ToolbarAlignment (Page 5568)
ToolbarBackColor (Page 5568)
ToolbarButtonActive (Page 5569)
ToolbarButtonAdd (Page 5570)
ToolbarButtonBeginGroup (Page 5570)
ToolbarButtonCount (Page 5571)
ToolbarButtonEnabled (Page 5572)
ToolbarButtonHotKey (Page 5572)
ToolbarButtonID (Page 5573)
ToolbarButtonIndex (Page 5573)
ToolbarButtonLocked (Page 5574)
ToolbarButtonName (Page 5575)
ToolbarButtonAuthorization (Page 5575)
ToolbarButtonRemove (Page 5576)
ToolbarButtonRename (Page 5576)
ToolbarButtonRepos (Page 5577)
ToolbarButtonTooltipText (Page 5577)
ToolbarButtonUserDefined (Page 5578)
ToolbarShowTooltips (Page 5579)
ToolbarUseBackColor (Page 5579)
ToolbarUseHotKeys (Page 5580)
ToolbarVisible (Page 5580)
TrendActualize (Page 5589)

TrendAdd (Page 5590)
TrendBeginTime (Page 5590)
TrendColor (Page 5591)
TrendCount (Page 5591)
TrendEndTime (Page 5592)
TrendExtendedColorSet (Page 5592)
TrendFill (Page 5593)
TrendFillColor (Page 5593)
TrendIndex (Page 5594)
TrendLabel (Page 5595)
TrendLineStyle (Page 5596)
TrendLineType (Page 5596)
TrendLineWidth (Page 5597)
TrendLowerLimit (Page 5597)
TrendLowerLimitColor (Page 5598)
TrendLowerLimitColoring (Page 5598)
TrendMeasurePoints (Page 5599)
TrendName (Page 5599)
TrendPointColor (Page 5600)
TrendPointStyle (Page 5600)
TrendPointWidth (Page 5601)
TrendProvider (Page 5601)
TrendRangeType (Page 5602)
TrendRemove (Page 5603)
TrendRename (Page 5603)
TrendRepos (Page 5604)
TrendSelectTagNameX (Page 5604)
TrendSelectTagNameY (Page 5605)
TrendTagNameX (Page 5605)
TrendTagNameY (Page 5606)
TrendTimeRangeBase (Page 5606)
TrendTimeRangeFactor (Page 5607)
TrendTrendWindow (Page 5607)
TrendUncertainColor (Page 5608)
TrendUncertainColoring (Page 5608)

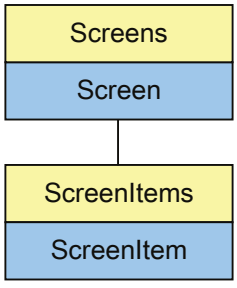
TrendUpperLimit (Page 5609)
TrendUpperLimitColor (Page 5609)
TrendUpperLimitColoring (Page 5610)
TrendVisible (Page 5610)
TrendWindowAdd (Page 5611)
TrendWindowCoarseGrid (Page 5611)
TrendWindowCoarseGridColor (Page 5612)
TrendWindowFineGrid (Page 5613)
TrendWindowFineGridColor (Page 5613)
TrendWindowForegroundTrendGrid (Page 5614)
TrendWindowGridInTrendColor (Page 5614)
TrendWindowHorizontalGrid (Page 5615)
TrendWindowIndex (Page 5615)
TrendWindowName (Page 5616)
TrendWindowRemove (Page 5616)
TrendWindowRename (Page 5616)
TrendWindowRepos (Page 5617)
TrendWindowRulerColor (Page 5617)
TrendWindowRulerLayer (Page 5618)
TrendWindowRulerStyle (Page 5619)
TrendWindowRulerWidth (Page 5619)
TrendWindowSpacePortion (Page 5620)
TrendWindowVerticalGrid (Page 5620)
TrendWindowVisible (Page 5621)
TrendXAxis (Page 5621)
TrendYAxis (Page 5622)
ShowRulerInAxis (Page 5499)
ShowScrollbars (Page 5500)
ShowTitle (Page 5504)
ShowTrendIcon (Page 5505)
StatusbarBackColor (Page 5511)
StatusbarElementAdd (Page 5512)
StatusbarElementAutoSize (Page 5512)
StatusbarElementCount (Page 5513)
StatusbarElementIconId (Page 5514)

StatusbarElementID (Page 5514)
StatusbarElementIndex (Page 5515)
StatusbarElementName (Page 5516)
StatusbarElementRemove (Page 5516)
StatusbarElementRename (Page 5517)
StatusbarElements (Page 5517)
StatusbarElementTooltipText (Page 5518)
StatusbarElementUserDefined (Page 5518)
StatusbarElementVisible (Page 5519)
StatusbarElementWidth (Page 5519)
StatusbarFontColor (Page 5520)
StatusbarShowTooltips (Page 5521)
StatusbarText (Page 5521)
StatusbarUseBackColor (Page 5522)
StatusbarVisible (Page 5523)
XAxisAdd (Page 5660)
XAxisAutoPrecisions (Page 5661)
XAxisAutoRange(i) (Page 5661)
XAxisBeginValue (Page 5662)
XAxisColor (Page 5662)
XAxisCount (Page 5663)
XAxisEndValue (Page 5663)
XAxisExponentialFormat (Page 5664)
XAxisIndex (Page 5664)
XAxisInTrendColor (Page 5665)
XAxisName (Page 5666)
XAxisPrecisions (Page 5666)
XAxisRemove (Page 5666)
XAxisRepos (Page 5667)
XAxisTrendWindow (Page 5668)
XAxisVisible (Page 5668)
YAxisAdd (Page 5669)
YAxisAutoPrecisions (Page 5670)
YAxisAutoRange (Page 5670)
YAxisBeginValue (Page 5671)

YAxisColor (Page 5671)
YAxisCount (Page 5672)
YAxisEndValue (Page 5672)
YAxisExponentialFormat (Page 5672)
YAxisIndex (Page 5673)
YAxisInTrendColor (Page 5673)
YAxisLabel (Page 5674)
YAxisName (Page 5674)
YAxisPrecisions (Page 5675)
YAxisRemove (Page 5675)
YAxisRepos (Page 5676)
YAxisScalingType (Page 5677)
YAxisTrendWindow (Page 5677)
YAxisVisible (Page 5678)
ControlDesignMode (Page 5507)
RTPersistencePasswordLevel (Page 5716)

Gauge

Description



Represents the "Gauge" object. The Gauge object is an element of the ScreenItems list.

Type identifier in VBS

HMIGauge

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-57 Properties

Properties	Read	Write	Description
AngleMax (Page 5242)	P, A	P, A	Specifies the angle for the scale end of the "Gauge" object.
AngleMin (Page 5242)	P, A	P, A	Specifies the angle for the scale start of the "Gauge" object.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackPicture	-	-	Specifies a graphic for the background.
BorderInnerStyle3D (Page 5275)	P	P	Specifies the representation of the inner part of the object border.
BorderOuterStyle3D (Page 5276)	P	P	Specifies the representation of the outer part of the object border.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
BorderWidth3D (Page 5281)	P	P	Specifies the width of the border for 3D display of the selected object.
CaptionColor (Page 5288)	P, A	P, A	Specifies the color of the text that is displayed in the header of the selected object.
CaptionFont	-	-	Specifies the character set for the caption.
CaptionText (Page 5288)	P, A	P, A	Specifies the text that is displayed in the header of the selected object.
CaptionTop (Page 5289)	P	P	Specifies the distance of the instrument label from the top edge of the selected object.
CenterColor (Page 5292)	P, A	P, A	Sets the color for the center point of the "Gauge" object.
CenterSize (Page 5293)	P	P	Sets the diameter of the round scale center point.
DangerRangeColor (Page 5311)	P, A	P, A	Sets the color of the danger range on the scale of the "Gauge" object.
DangerRangeStart (Page 5311)	P, A	P, A	Specifies the scale value at which the danger range of the "Gauge" object begins.
DangerRangeVisible (Page 5312)	P, A	P, A	Specifies whether the danger range is displayed in the scale of the "Gauge" object. {
DialColor (Page 5314)	P, A	P, A	Specifies the color of the dial in the selected object.
DialFillStyle (Page 5315)	P	P	Specifies the type of background for the selected object.
DialPicture	-	-	Specifies a graphic for the dial.
DialSize (Page 5315)	P	P	Specifies the diameter of the scale disc based on the smaller value of the "Width" and "Height" geometry attributes.

Properties	Read	Write	Description
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in run-time.
Flashing *	-	-	Specifies that the text block flashes.
Gradation (Page 5355)	P, A	P, A	Specifies the value difference between two main tick marks of the "Gauge" object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LockSquaredExtent (Page 5402)	P	P	Specifies whether the tag values are downloaded from the logs for the time range to be displayed when you call a screen.
MaximumValue (Page 5406)	P, A	P, A	Specifies the maximum value of the scale in the selected object.
MinimumValue (Page 5422)	P, A	P, A	Specifies the minimum value of the scale in the selected object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
NormalRangeColor (Page 5430)	P, A	P, A	Specifies the color of the normal range on the scale of the "Gauge" object. "
NormalRangeVisible (Page 5431)	P, A	P, A	Specifies whether the normal range in the scale of the "Gauge" object will be displayed.
PointerColor (Page 5454)	P, A	P, A	Specifies the pointer color of the "Gauge" object.
ProcessValue (Page 5456)	P, A	P, A	Specifies the default for the value to be displayed.
ScaleLabelColor (Page 5470)	P, A	P, A	Specifies the color of the label for the scale tick marks of the "Gauge" object.
ScaleLabelFont	-	-	Specifies the font of the scale label.
ScaleTickColor (Page 5471)	P, A	P, A	Specifies the color of the scale tick marks of the "Gauge" object.
ScaleTickLabelPosition (Page 5472)	P	P	Specifies the diameter of the imaginary circle on which the scale tick label is location.
ScaleTickLength (Page 5472)	P	P	Specifies the length of the main scale ticks.
ScaleTickPosition (Page 5473)	P	P	Specifies the diameter of the assumed circle on which the scale divisions are located.
ShowDecimalPoint (Page 5494)	P	P	Specifies whether the scale is labeled with decimal figures (decimal point and one decimal place) or with whole integers.
ShowPeakValuePointer (Page 5497)	P, A	P, A	Specifies whether a slave pointer will be used for the selected object.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UnitColor (Page 5623)	P, A	P, A	Specifies the text color for the label of the measurement unit in the "Gauge" object.
UnitFont	-	-	Specifies the font for the unit.
UnitText (Page 5623)	P, A	P, A	Specifies the text for the measurement unit of the selected object.
UnitTop (Page 5624)	P	P	Specifies the distance of the measurement unit from the upper edge of the selected object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.

Properties	Read	Write	Description
WarningRangeColor (Page 5651)	P, A	P, A	Specifies the color of the warning range on the scale of the "Gauge" object.
WarningRangeStart (Page 5651)	P, A	P, A	Specifies the scale value from which the warning range of the "Gauge" object begins.
WarningRangeVisible (Page 5652)	P, A	P, A	Specifies whether the warning range is displayed in the scale of the "Gauge" object.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-58 Methods

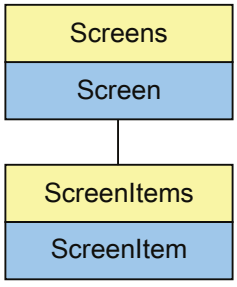
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

GraphicIOField

Description



Represents the "Graphic I/O field" object. The GraphicIOField object is an element of the ScreenItems list.

The availability of the following object properties depends on the selected mode:

Property	"Input"	"Output"	"Input/Output"	"Two states"
BorderColor	--	--	--	x
BorderStyle3D	x	x	x	--
Enabled	x	--	x	--
FocusColor	x	--	x	--
FocusWidth	x	--	x	--
HelpText	x	--	x	--
TransparentColor	x	x	x	--
UseTransparent-Color	x	x	x	--

Type identifier in VBS

HMIGraphicIOField

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-59 Properties

Properties	Read	Write	Description
AboveUpperLimitColor (Page 5224)	-	-	Sets the color of the specified object for the "High limit violated" event.
AdaptPicture	-	-	Specifies that the size of the object is automatically adjusted in runtime.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
AutoSizing	-	-	Specifies that the size of the object should be automatically adapted to the content.
BackColor (Page 5251)	A	A	Specifies the background color of the selected object.
BelowLowerLimitColor (Page 5263)	-	-	Sets the color of the specified object for the "Low limit violated" event.
BitNumber (Page 5264)	-	-	Specifies the bit whose status must change to trigger a value change.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderStyle3D	A	A	Specifies whether the object has a 3D border shading.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	P	P	Specifies the line style of the selected object.
Enabled (Page 5323)	P, A	P, A	Specifies whether the selected object can be operated in run-time.
FitToLargest	-	-	Specifies that the size of the object depends on the text length in all languages.
Flashing	-	-	Specifies that the text block flashes.

Properties	Read	Write	Description
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingOnLimitViolation	-	-	Specifies that flashing is activated when a limit is violated.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
FlashTransparentColor (Page 5345)	P	P	Specifies the color of the bitmap object of a flashing graphic that is set to "transparent".
FocusColor (Page 5346)	A	A	Specifies the color for the focus frame of the selected object when it is in focus.
FocusWidth (Page 5347)	A	A	Specifies the border width of the specified object when the object is in focus.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HelpText	A	-	Returns the tooltip that is shown in runtime as user help for the specified object.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line ends.
Mode (Page 5424)	-	-	Specifies the field type of the selected object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
OnValue	-	-	Specifies the value for the "ON" state in "two-state" mode.
PictureList	-	-	Specifies the graphic list that supplies the object with values.
PictureOff (Page 5452)	-	-	Specifies the screen to be displayed in the "Off" state.
PictureOn (Page 5453)	-	-	Specifies the screen to be displayed in the "On" state.
ProcessValue (Page 5456)	P	P	Specifies the default for the value to be displayed.
ScrollBarOrientation	-	-	Specifies the orientation of the scroll bar.
ShowScrollBar	-	-	Specifies the scroll bar type.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
TransparentColor (Page 5587)	P, A	P, A	Specifies which color of the allocated graphic (*.bmp, *dib) of the specified object is set to "transparent".
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
UseFlashTransparentColor (Page 5629)	P	P	Specifies whether the color of the bitmap object of a flashing graphic is set to "transparent".
UseTransparentColor (Page 5632)	P, A	P, A	Specifies whether the color defined with the property "TransparentColor" is shown as transparent for the selected object.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-60 Methods

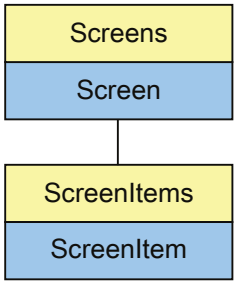
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

GraphicView

Description



Represents the "Graphic view" object. The GraphicView object is an element of the ScreenItems list.

Type identifier in VBS

HMIGraphicView

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-61 Properties

Properties	Read	Write	Description
AdaptPicture	A	-	Specifies that the size of the object is automatically adjusted in runtime.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
AutoSizing	-	-	Specifies that the size of the object should be automatically adapted to the content.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackFlashingColorOff (Page 5257)	P	P	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 5256)	P	P	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258)	P	P	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258)	P	P	Specifies the flash rate of the background for the selected object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderStyle3D	A	A	Specifies whether the object has a 3D border shading.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	P	P	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
FitToLargest	-	-	Specifies that the size of the object depends on the text length in all languages.

Properties	Read	Write	Description
Flashing	-	-	Specifies that the text block flashes.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line ends.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
Picture (Page 5451)	P	P	Specifies the screen to be displayed in the graphics object in runtime.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
TransparentColor (Page 5587)	P, A	P, A	Specifies which color of the allocated graphic (*.bmp, *dib) of the specified object is set to "transparent".
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
UseTransparentColor (Page 5632)	P, A	P, A	Specifies whether the color defined with the property "TransparentColor" is shown as transparent for the selected object.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-62 Methods

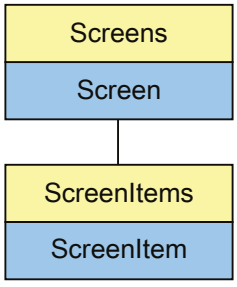
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

Group

Description



Represents the "Group" object. The Group object is an element of the ScreenItems list.

Type identifier in VBS

HMIGroup

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-63 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
ForeColor (Page 5353)	P	P	Specifies the font color of the text for the selected object.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Properties	Read	Write	Description
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
Name (Page 5426)	-	-	Returns the object name as STRING.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-64 Methods

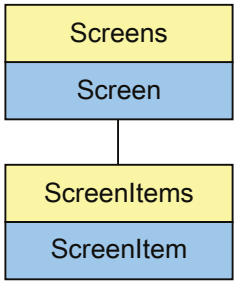
Methods	Valid	Description

See also

Screen object (list) (Page 5007)

HTMLBrowser

Description



Represents the object "HTML Browser". The HTMLBrowser-object is an element of the ScreenItems-list.

Type identifier in VBS

HMIHTMLBrowser

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-65 Properties

Properties	Read	Write	Description
Address (Page 5233)	P, A	P, A	Defines the Web address which is opened in the HTML browser.
Enabled (Page 5323)	P, A	P, A	Specifies whether the selected object can be operated in runtime.
Flashing *	-	-	Specifies that the text block flashes.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.

Properties	Read	Write	Description
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
ShowStatusBar (Page 5502)	P	P	Specifies whether the status bar is shown.
ShowToolBar (Page 5505) *	-	-	Specifies whether the toolbar is shown.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-66 Methods

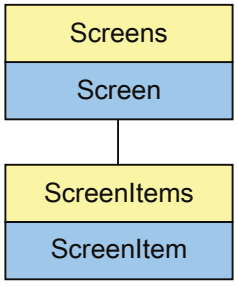
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.
SetHtml	??	

See also

Screen object (list) (Page 5007)

IOField

Description



Represents the "I/O field" object. The IOField object is an element of the ScreenItems list.

Type identifier in VBS

HMIIOField

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-67 Properties

Properties	Read	Write	Description
AboveUpperLimitColor (Page 5224)	-	-	Sets the color of the specified object for the "High limit violated" event.
AcceptOnExit (Page 5225)	P	P	Specifies whether the input field is confirmed automatically when it is left.
AcceptOnFull (Page 5225)	P	P	Specifies whether the input field will be left and confirmed automatically when the defined number of values has been entered.

Properties	Read	Write	Description
AdaptBorder (Page 5232)	P	P	Specifies whether the border of the specified object is dynamically adapted to the text size.
AskOperationMotive (Page 5244)	P	P	Specifies whether the reason for operating this object is logged.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackFlashingColorOff (Page 5256)	P	P	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 5257)	P	P	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258)	P	P	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258)	P	P	Specifies the flash rate of the background for the selected object.
BelowLowerLimitColor (Page 5263)	-	-	Sets the color of the specified object for the "Low limit violated" event.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P, A	P, A	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderStyle3D	A	A	Specifies whether the object has a 3D border shading.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
BottomMargin	-	-	Sets the margin between the text and the bottom edge of the object.
ClearOnError (Page 5295)	P	P	Specifies whether an invalid input in this object is deleted automatically.
ClearOnFocus (Page 5295)	P	P	Specifies whether the field entry is deleted as soon as the I/O field is activated.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
CursorControl (Page 5310)	P	P	Specifies whether the mouse cursor jumps to the next field of the TAB sequence after leaving a field.
DataFormat (Page 5312)	P	P	Returns the data type of the IOField object.
EdgeStyle (Page 5320)	A	A	Specifies the line style of the selected object.
EditOnFocus (Page 5322)	P	P	Specifies whether data input is immediately possible if the input field is selected using the "Tab" key.
Enabled (Page 5323)	P, A	P, A	Specifies whether the selected object can be operated in runtime.
FieldLength (Page 5335)	–	-	Specifies that the "Field length string" field is read-only.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
FitToLargest	-	-	Specifies that the size of the object depends on the text length in all languages.

Properties	Read	Write	Description
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingOnLimitViolation	-	-	Specifies that flashing is activated when a limit is violated.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	P	P	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 5350)	P	P	Specifies whether the text of the selected object is shown in italics.
FontName (Page 5351)	P	P	Specifies the font of the selected object.
FontSize (Page 5351)	P	P	Specifies the font size of the text for the selected object.
FontUnderline (Page 5352)	P	P	Specifies whether the text of the selected object is underlined.
ForeColor (Page 5353)	P, A	P, A	Specifies the font color of the text for the selected object.
FormatPattern (Page 5354)	P	P	Specifies the format of the output value.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HelpText	A	-	Returns the tooltip that is shown in runtime as user help for the specified object.
HiddenInput (Page 5363)	P	P	Specifies whether the input value or a * for each character is shown during the input.
HorizontalAlignment (Page 5372)	P, A	P, A	Specifies the horizontal alignment of the text within the selected object.
InputValue (Page 5377)	P	P	Defines the value entered by the user in the IO field.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LeftMargin	-	A	Sets the margin between the text and the left edge of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
LogOperation (Page 5403)	P	P	Specifies whether, after operating this object, an alarm is output to the alarm system.
LowerLimit (Page 5404)	P	P	Specifies the low limit for input values.
Mode (Page 5424)	P	P	Specifies the field type of the selected object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
ProcessValue (Page 5456)	P, A	P, A	Specifies the default for the value to be displayed.
RightMargin	-	-	Sets the margin between the text and the right edge of the object.
ShiftDecimalPoint (Page 5492)	-	-	Specifies that the "Shift decimal point" field is read-only.
ShowBadTagState (Page 5493)	P	P	Specifies whether an object is grayed out if its associated tag has a bad quality code or tag state.
ShowLeadingZeros	-	-	Specifies that leading zeros should be displayed.
TextOrientation (Page 5536)	P	P	Specifies the text orientation of the selected object.

Properties	Read	Write	Description
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
TopMargin	-	-	Sets the margin between the text and the top edge of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
Unit (Page 5622)	P	P	Specifies the unit of measurement in the "IOField" object.
UpperLimit (Page 5624)	P	P	Specifies the high limit for input values.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
UseTagLimitColors	P	P	Specifies whether the configured colors are used when limits are violated.
VerticalAlignment (Page 5643)	P, A	P, A	Specifies the vertical alignment of the text for the selected object.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-68 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

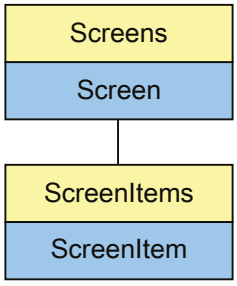
See also

Screen object (list) (Page 5007)

Objects K-Z

Line

Description



Represents the "Line" object. The Line object is an element of the ScreenItems list.

Type identifier in VBS

HMILine

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-69 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BorderEndStyle (Page 5270)	P, A	P, A	Specifies the type of line ends for the selected object.
BorderStyle (Page 5278)	P, A	P, A	Specifies the type of line ends for the selected object.

Properties	Read	Write	Description
Color (Page 5296)	P, A	P, A	Specifies the line color of the selected object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
EndLeft	-	-	Specifies the X position of the end point.
EndStyle (Page 5326)	P, A	P, A	Specifies how the line end of the selected object is displayed.
EndTop	-	-	Sets the Y position of the end point.
FillStyle	A	A	Defines the fill pattern of the specified object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
LineWidth (Page 5400)	P, A	P, A	Specifies the line thickness of the selected object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
RotationAngle (Page 5463)	P	P	Specifies the angle of rotation in degrees.
RotationCenterLeft (Page 5463)	P	P	Specifies the X coordinate of the pivot point for the object in runtime.
RotationCenterTop (Page 5464)	P	P	Specifies the Y coordinate of the pivot point for the object in runtime.
StartLeft	-	-	Specifies the X position of the starting point.
StartStyle	A	A	Specifies how the line start of the selected object is displayed.
StartTop (Page 5510)	-	-	Specifies the vertical distance in pixels between the start point and the top edge of the screen.
Style (Page 5523)	P, A	P, A	Specifies the line style of the selected object.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-70 Methods

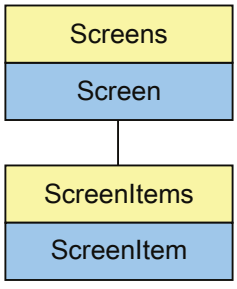
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

Listbox

Description



Represents the "Listbox" object. The Listbox object is an element of the ScreenItems list.

Type identifier in VBS

HMIListBox

Application

In the following example, the object with the name "ListBox1" is moved 10 pixels to the right:

```
'VBS21
Dim objListBox
Set objListBox = ScreenItems("ListBox1")
objListBox.Left = objListBox.Left + 10
```

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-71 Properties

Properties	Read	Write	Description
AskOperationMotive (Page 5244)	P	P	Specifies whether the reason for operating this object is logged.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
CountVisibleItems (Page 5309)	P	P	Specifies the number of lines contained in the selection list.
EdgeStyle (Page 5320)	-	-	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
FitToLargest	-	-	Specifies that the size of the object depends on the text length in all languages.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	P	P	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 5350)	P	P	Specifies whether the text of the selected object is shown in italics.
FontName (Page 5351)	P	P	Specifies the font of the selected object.
FontSize (Page 5351)	P	P	Specifies the font size of the text for the selected object.
FontUnderline (Page 5352)	P	P	Specifies whether the text of the selected object is underlined.
ForeColor (Page 5353)	P	P	Specifies the font color of the text for the selected object.

Properties	Read	Write	Description
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HorizontalAlignment (Page 5372)	P	P	Specifies the horizontal alignment of the text within the selected object.
Index (Page 5375)	P	P	Specifies the background for grid control elements.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
LogOperation (Page 5403)	P	P	Specifies whether, after operating this object, an alarm is output to the alarm system.
Name (Page 5426)	P	-	Returns the object name as STRING.
SelectedIndex (Page 5480)	P	P	Specifies and returns the index whose associated text is highlighted in the combo or list box.
SelectedText (Page 5478)	P	P	Displays the text that is defined with the "Selected field" (SelIndex) attribute and highlighted in the combo box or list box.
Text (Page 5533)	P	P	Specifies the label for the text field.
Texts	-	-	Specifies the texts for the check boxes.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-72 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen (Page 5000)

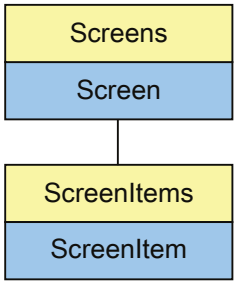
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

MediaPlayer

Description



Represents the "Media Player" object. The MediaControl object is an element of the ScreenItems list.

Type Identifier in VBS

HMIMediaControl

Application

In the following example, the object with the name "Control1" is moved 16 pixels to the right:

```
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 16
```

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-73 Properties

Properties	Read	Write	Description
AspectRatio	P	P	Specifies whether the aspect ratio is maintained when the size is changed.
AutoStart	P	P	Specifies that the Media Player should start automatically.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FileName	P, A	P, A	Specifies the name of the file to be loaded.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
PictureSizeMode	P	P	Specifies that the status bar is displayed.
PlayCount	A	A	Plays the files several times.
PlayEndless	P	P	Specifies that the media file should play endlessly.
PopupMenuEnabled	P	P	Specifies that the shortcut menu can be used.
ShowControls	P, A	P, A	Specifies that a toolbar is displayed.
ShowFeatureBackward	P	P	Specifies that the "Backward" button is displayed in runtime.
ShowFeatureForward	P	P	Specifies that the "Forward" button is displayed in runtime.
ShowFeatureFullScreen	P	P	Specifies that the Media Player can be displayed as a full screen.
ShowFeaturePause	P	P	Specifies that the Media Player can be displayed as a full screen
ShowFeaturePlay	P	P	Specifies that the "Play" button is displayed in runtime.
ShowFeatureStop	P	P	Specifies that the "Stop" button is displayed in runtime.
ShowFeatureVolume	P	P	Sets the volume.
ShowStatusBar (Page 5502)	A	A	Specifies whether the status bar is shown.
ShowTracker	A	A	Specifies that the tracker is displayed.
StepSeconds	P	P	Specifies a time in seconds that is to be skipped when the "Forward" or "Backward" button is pressed.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-74 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen (Page 5000)

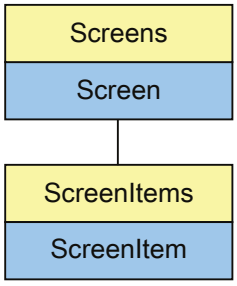
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

MultiLineEdit

Description



Represents the "MultiLine text" object. The MultiLineEdit object is an element of the ScreenItems list.

Type identifier in VBS

HMIMultiLineEdit

Application

In the following example, the object with the name "MultiLineEdit1" is moved 10 pixels to the right:

```
'VBS21
Dim objMultiLineEdit
Set objMultiLineEdit = ScreenItems("MultiLineEdit1")
objMultiLineEdit.Left = objMultiLineEdit.Left + 10
```

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-75 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P, A	Specifies the background color of the selected object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P	P	Specifies the background color of the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
EdgeStyle (Page 5320)	-	-	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	P	P	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 5350)	P	P	Specifies whether the text of the selected object is shown in italics.
FontName (Page 5351)	P	P	Specifies the font of the selected object.
FontSize (Page 5351)	P	P	Specifies the font size of the text for the selected object.
FontUnderline (Page 5352)	P	P	Specifies whether the text of the selected object is underlined.
ForeColor (Page 5353)	P	P	Specifies the font color of the text for the selected object.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HorizontalAlignment (Page 5372)	P	P	Specifies the horizontal alignment of the text within the selected object.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
Name (Page 5426)	P	-	Returns the object name as STRING.

Properties	Read	Write	Description
Text (Page 5533)	P	P	Specifies the label for the text field.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-76 Methods

Methods	Validity	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.
GetSelectionText	??	
SetSelection	??	
SetSelectionText	??	

See also

Screen (Page 5000)

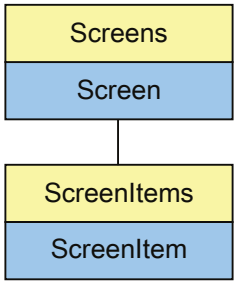
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

OLEView

Description



Represents the "OLE object" object. The OLEView object is an element of the ScreenItems list.

Type identifier in VBS

Version-independent ProgID

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-77 Properties

Properties	Read	Write	Description

Properties	Read	Write	Description

Table 12-78 Methods

Methods	Can be dynamized	Validity	Description

See also

Screen (Page 5000)

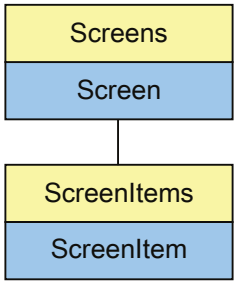
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

OnlineTableControl

Description



Represents the "Table view" object. The OnlineTableControl object is an element of the ScreenItems list.

Type identifier in VBS

HMIOnlineTableControl

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-79 Properties

Properties	Read	Write	Description
ApplyProjectSettingsForDesignMode (Page 5243)	-	-	Specifies that the project settings are used for the design.
AutoCompleteColumns (Page 5247)	P	P	Specifies whether empty columns are shown if the control is wider than the configured columns.
AutoCompleteRows (Page 5247)	P	P	Specifies whether empty rows are shown if the control is longer than the number of configured rows.

Properties	Read	Write	Description
AutoSelectionColors (Page 5248)	P	P	Specifies whether the colors defined by the system are used as the selection colors for cells and rows.
AutoSelectionRectColor (Page 5249)	P	P	Specifies whether the selection frame is shown with the color defined by the system.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
Caption (Page 5287)	P	P	Specifies the text that is displayed in the header of the selected object.
CellCut (Page 5289)	P	P	Specifies whether the contents of the cells are abbreviated if the cells are too narrow.
CellSpaceBottom (Page 5290)	P	P	Specifies the bottom margin of the table cells.
CellSpaceLeft (Page 5290)	P	P	Sets the left indent for the table cells.
CellSpaceRight (Page 5291)	P	P	Specifies the right indent of the table cells.
CellSpaceTop (Page 5291)	P	P	Specifies the top margin of the table cells.
Closeable (Page 5295)	P	P	Specifies that the control can be closed in runtime.
ColumnResize (Page 5299)	P	P	Enables changes to the width of columns.
ColumnScrollbar (Page 5300)	P	P	Specifies the scroll bar type.
ColumnTitleAlignment	P	P	Specifies the type of column title alignment.
ColumnTitles (Page 5303)	P	P	Specifies whether the column header is displayed.
ControlDesignMode (Page 5507)	P	P	Specifies the design.
EnableEdit (Page 5324)	P	P	Specifies whether the data shown can be edited in Runtime.
ExportDirectoryChangeable (Page 5328)	P	P	Specifies whether the data export directory can be changed in runtime.
ExportDirectoryname (Page 5329)	P	P	Specifies the target directory for the exported Runtime data.
ExportFileExtension (Page 5330)	P	P	Specifies the file extension of the export file. Only the "csv" file extension is currently supported.
ExportFilename (Page 5330)	P	P	Specifies the name of the target file for the exported Runtime data.
ExportFilenameChangeable (Page 5331)	P	P	Specifies whether the export file name can be changed in runtime.
ExportFormatGuid (Page 5331)	P	P	Default assignment of the ID number and export provider.
ExportFormatName (Page 5332)	P	P	Specifies the export file format. Only the "csv" file format is currently available for the export.
ExportParameters (Page 5332)	P	P	Specifies the parameters of the selected format by means of the "Properties" dialog.
ExportSelection (Page 5333)	P	P	Specifies the file extension of the export file. Only the "csv" file extension is currently supported.
ExportShowDialog (Page 5334)	P	P	Specifies whether the data export dialog is shown in runtime.
Font (Page 5348)	-	-	Specifies or returns the font.
GridLineColor (Page 5356)	P	P	Specifies the color for the grid lines.
GridLineWidth (Page 5357)	P	P	Specifies the width of the separation lines in pixels.
Height (Page 5358)	P	-	Sets the height of the specified "Ellipse", "Circle" and "Rectangle" objects.
HorizontalGridLines (Page 5373)	P	P	Specifies whether horizontal separation lines are displayed.

Properties	Read	Write	Description
IconSpace (Page 5375)	P	P	Specifies the spacing between icons and text in the table cells.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineColor (Page 5398)	P	P	Sets the color of the window separation lines.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
LoadDataImmediately (Page 5401)	P	P	Specifies whether the tag values are downloaded from the logs for the time range to be displayed when you call a screen.
Moveable (Page 5425)	P	P	Specifies whether the window can be moved in runtime.
Name (Page 5426)	P	-	Returns the object name as STRING.
Online (Page 5434)	P	P	Specifies the name of the data source.
PrintJob	P	P	Specifies a print job
RowScrollbar (Page 5465)	P	P	Specifies whether row scroll bars are displayed.
RowTitleAlignment	P	P	Specifies the type of row title alignment.
RowTitles	P	P	Specifies that row headers will be displayed.
RTPersistence (Page 5466)	P	P	Specifies how online configurations of WinCC are retained.
RTPersistencePasswordLevel (Page 5716)	P	P	Specifies the authorization required for online configuration in runtime.
RTPersistenceType (Page 5467)	-	-	Specifies how online configurations of WinCC are retained.
SelectedCellColor (Page 5478)	P	P	Specifies the background color of the selected cell.
SelectedCellForeColor (Page 5479)	P	P	Specifies the font color of the selected cell.
SelectedRowColor (Page 5480)	P	P	Specifies the background color of the selected row.
SelectedRowForeColor (Page 5481)	P	P	Specifies the font color of the selected row.
SelectedTitleColor (Page 5482)	P	P	Specifies the background color of a selected table header.
SelectedTitleForeColor (Page 5482)	P	P	Specifies the font color of the selected table header.
SelectionColoring (Page 5485)	P	P	Specifies whether selection colors are used for cells or rows.
SelectionRect (Page 5486)	P	P	Specifies whether a selection frame is used for the selected cells or rows.
SelectionRectColor (Page 5486)	P	P	Specifies the color of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionRectWidth (Page 5487)	P	P	Specifies the line width of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionType (Page 5488)	P	P	Specifies the number of lines you can mark.
ShowSortButton (Page 5500)	P	P	Specifies whether the sorting button is shown above the vertical scroll bar.
ShowSortIcon (Page 5501)	P	P	Specifies whether the sorting icon is displayed.
ShowSortIndex (Page 5501)	P	P	Specifies whether the sorting icon is displayed.
ShowTitle (Page 5504)	P	P	Specifies the layout of the control window's header.
Sizeable (Page 5506)	P	P	Specifies that the size of an object can be changed in runtime.
SortSequence (Page 5508)	P	P	Specifies how the sorting order can be changed by mouse click.
StatusbarBackColor (Page 5511)	P	P	Specifies the background color of the status bar.

Properties	Read	Write	Description
StatusbarElementAdd (Page 5512) *	P	P	Creates a new, user-defined status bar element.
StatusbarElementAutoSize (Page 5512) *	P	P	Specifies whether the width of the selected status bar element is set automatically.
StatusbarElementCount (Page 5513) *	P	P	Specifies the number of configurable status bar elements.
StatusbarElementIconId (Page 5514) *	P	P	Default assignment of the ID number and icon of a status bar element.
StatusbarElementId (Page 5514) *	P	P	Unique ID of the selected status bar element.
StatusbarElementIndex (Page 5515)	P	P	References a status bar element. You can use this property to assign the values of other properties to a specific element of the status bar.
StatusbarElementName (Page 5516) *	P	P	Displays the object name of the selected status bar element.
StatusbarElementRemove (Page 5516) *	P	P	Removes the selected status bar element.
StatusbarElementRename (Page 5517) *	P	P	Renames the user-defined status bar element that is referenced by means of the "StatusbarElementIndex" property.
StatusbarElementRepos *	P	P	Specifies the index that is assigned to the element.
StatusbarElements (Page 5517)	-	-	Specifies the elements that are to be shown in the status bar.
StatusbarElementText *	P	P	Specifies the text of the currently selected status bar element.
StatusbarElementTooltipText (Page 5518) *	P	P	Specifies the tooltip text for the user-defined status bar element.
StatusbarElementUserDefined (Page 5518) *	P	P	Indicates whether the configuration engineer has added the status bar element as a new user-defined element.
StatusbarElementVisible (Page 5519) *	P	P	In the list, activate the status bar elements that you wish to display in runtime.
StatusbarElementWidth (Page 5519) *	P	P	Specifies the width of the selected status bar element in pixels.
StatusbarFont	-	-	Status bar font
StatusbarFontColor (Page 5520)	P	P	Specifies the color of the text in the status bar.
StatusbarShowTooltips (Page 5521)	P	P	Specifies whether tooltips are displayed for the elements of the status bar in runtime.
StatusbarText (Page 5521)	P	P	Default text in the status bar.
StatusbarUseBackColor (Page 5522)	P	P	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 5523)	P	P	Specifies whether the status bar of the control is displayed.
TableColor (Page 5526)	P	P	Specifies the background color of the rows. Use this to open the color selection dialog.
TableColor2 (Page 5526)	P	P	Specifies the background color of "Row color 2".
TableForeColor (Page 5527)	P	P	
TableForeColor2 (Page 5528)	P	P	Specifies the font color of "Row color 2".
TimeBase (Page 5540)	P	P	Sets the time zone that forms the basis for the display of time values.
TimeColumnActualize (Page 5541) *	P	P	Specifies whether the values of the selected column are updated.
TimeColumnAdd (Page 5542) *	P	P	Creates a new time column.
TimeColumnAlign (Page 5542) *	P	P	Specifies how the selected time column is aligned.

Properties	Read	Write	Description
TimeColumnBackColor (Page 5543) *	P	P	Specifies the background color of the time column selected.
TimeColumnBeginTime (Page 5544) *	P	P	Specifies the starting time of the time range for a selected time column.
TimeColumnCaption (Page 5544) *	P	P	Specifies the label of the time column.
TimeColumnCount (Page 5545) *	P	P	Specifies the number of configured time columns.
TimeColumnDateFormat (Page 5545) *	P	P	Specifies which date format is used to display the selected time column.
TimeColumnEndTime (Page 5546) *	P	P	Specifies the end point of the time range for the selected time column.
TimeColumnForeColor (Page 5546) *	P	P	Specifies the font color of the time column selected.
TimeColumnHideText (Page 5547) *	P	P	Sets text format for displaying the content of a time column.
TimeColumnHideTitleText (Page 5547) *	P	P	Sets text format for displaying the time column header.
TimeColumnIndex (Page 5548)	P	P	References a configured time column.
TimeColumnLength (Page 5548) *	P	P	Specifies the width of a selected time column.
TimeColumnMeasurePoints (Page 5548) *	P	P	Specifies the number of measuring points to be displayed in the selected time column.
TimeColumnName (Page 5549) *	P	P	Specifies the name of the selected time column.
TimeColumnRangeType (Page 5549) *	P	P	Specifies the time range used for the selected time column.
TimeColumnRemove (Page 5550) *	P	P	Removes the selected time column from the list.
TimeColumnRename (Page 5550) *	P	P	Renames the time column that is referenced by means of the "TimeColumnIndex" property.
TimeColumnRepos (Page 5551) *	P	P	Changes the sorting order of the time columns, including the corresponding value columns.
TimeColumnShowDate (Page 5551) *	P	P	Specifies whether the selected time column is displayed with date and time.
TimeColumnShowIcon (Page 5552) *	P	P	Specifies whether the contents of the time column are shown as an icon.
TimeColumnShowTitleIcon (Page 5552) *	P	P	Specifies whether the values of the selected column are updated.
TimeColumnSort (Page 5553) *	P	P	Specifies how the time column referenced in "TimeColumnIndex" is sorted.
TimeColumnSortIndex (Page 5553) *	P	P	Specifies the sorting order of the time column referenced in "TimeColumnIndex".
TimeColumnTimeFormat (Page 5554) *	P	P	Defines the time format used to display a selected time column.
TimeColumnTimeRangeBase (Page 5554) *	P	P	Specifies the unit of time for calculating the time range.
TimeColumnTimeRangeFactor (Page 5555) *	P	P	Specifies the factor for calculating the time range.
TimeColumnUseValueColumnColors (Page 5555) *	P	P	Specifiess whether the selected time column is displayed in the value column colors.
TimeColumnVisible (Page 5556) *	P	P	The list shows the time columns you have created.
TimeStepBase	P	P	A time range is set via a base and a factor.
TimeStepFactor	P	P	A time range is set via a base and a factor.
TitleColor	P	P	Specifies the color of the header.

Properties	Read	Write	Description
TitleCut (Page 5557)	P	P	Specifies whether the contents of the fields of a header should be abbreviated if the column is too narrow.
TitleDarkShadowColor (Page 5558)	P	P	Specifies the color of the dark side of shading.
TitleForeColor (Page 5559)	P	P	Specifies the font color of the table header for the selected status.
TitleGridLineColor (Page 5559)	P	P	Specifies the color of separation lines in the table header.
TitleLightShadowColor (Page 5560)	P	P	Specifies the color of the bright side of shading.
TitleSort (Page 5561)	P	P	Specifies how sorting by column header is triggered.
TitleStyle (Page 5561)	P	P	Specifies whether a shading color is used for the table header.
ToolbarAlignment (Page 5568)	P	P	Specifies or returns the position of the toolbar.
ToolbarBackColor (Page 5568)	P	P	Specifies the background color of the toolbar.
ToolbarButtonActive (Page 5569) *	P	P	Specifies whether the function linked to the button is activated in runtime.
ToolbarButtonAdd (Page 5570) *	P	P	Creates a new, user-defined button function.
ToolbarButtonBeginGroup (Page 5570) *	P	P	Specifies whether a separator is inserted before the selected button function.
ToolbarButtonClick	P	P	Clicks a toolbar button.
ToolbarButtonCount (Page 5571) *	P	P	Specifies the number of configurable button functions.
ToolbarButtonEnabled (Page 5572) *	P	P	Specifies whether a user-defined toolbar button can be operated.
ToolbarButtonHotKey (Page 5572) *	P	P	Shows the hotkey for the selected button function.
ToolbarButtonId (Page 5573) *	P	P	Unique ID number for the selected button function.
ToolbarButtonIndex (Page 5573)	P	P	References a button function.
ToolbarButtonLocked (Page 5574) *	P	P	Specifies whether the clicked, pressed in state is displayed for a user-defined toolbar button.
ToolbarButtonName (Page 5575) *	P	P	Shows the name of the selected button function.
ToolbarButtonPasswordLevel (Page 5575) *	P	P	Shows the authorization for the selected button function.
ToolbarButtonRemove (Page 5576) *	P	P	Removes the selected button function from the list.
ToolbarButtonRename (Page 5576) *	P	P	Renames the user-defined toolbar element that is referenced by means of the "ToolbarButtonIndex" property.
ToolbarButtonRepos (Page 5577) *	P	P	Changes the sequence of button functions.
ToolbarButtons	-	-	Specifies which buttons are available for the toolbar.
ToolbarButtonTooltipText (Page 5577) *	P	P	Specifies the tooltip text for the button.
ToolbarButtonUserDefined (Page 5578) *	P	P	Indicates whether the configuration engineer has added the toolbar button as a new user-defined button.
ToolbarButtonVisible *	P	P	Specifies that the button is visible.
ToolbarShowTooltips (Page 5579)	P	P	Specifies whether the tooltips for the button functions are displayed in runtime.
ToolbarUseBackColor (Page 5579)	P	P	Specifies whether the background color for the toolbar is shown.
ToolbarUseHotKeys (Page 5580)	P	P	Specifies whether the hotkeys for the button functions are activated in runtime.
ToolbarVisible (Page 5580)	P	P	Specifies whether the toolbar of the control is displayed.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.

Properties	Read	Write	Description
UseColumnBackColor	P	P	Specifies whether scrolling is enabled.
UseColumnForeColor	P	P	Specifies the font used for printing.
UseSelectedTitleColor (Page 5630)	P	P	Specifies whether a selection color is used for the headers of selected table cells.
UseTableColor2 (Page 5631)	P	P	Specifies whether a second row color is used for the representation of the table.
ValueColumnAdd *	P	P	Adds a value column.
ValueColumnAlign *	P	P	Specifies the alignment of a column.
ValueColumnAutoPrecisions *	P	P	Specifies whether the number of decimal places to be displayed is determined automatically.
ValueColumnBackColor *	P	P	Specifies the background color.
ValueColumnCaption *	P	P	Specifies a header.
ValueColumnCount *	P	P	Displays the number of value columns.
ValueColumnExponentialFormat *	P	P	Specifies that exponential notation is used in a column.
ValueColumnForeColor *	P	P	Specifies the foreground color for a value column.
ValueColumnHideText *	P	P	Specifies whether the text is displayed.
ValueColumnHideTitleText *	P	P	Specifies whether the text is displayed.
ValueColumnIndex	P	P	The index specifies on which column other properties, e.g. initial value are based.
ValueColumnLength *	P	P	Specifies the number of displayed characters.
ValueColumnName *	P	P	Specifies the name of a column.
ValueColumnPrecisions *	P	P	Specifies the number of decimal places.
ValueColumnProvider *	P	P	
ValueColumnProviderCLSID *	P	P	Specifies the provider CLSID of the data for a column.
ValueColumnRemove *	P	P	Deletes a value column.
ValueColumnRename *	P	P	Renames a value column.
ValueColumnRepos *	P	P	Specifies that a value column is repositioned.
ValueColumnSelectTagName *	P	P	Specifies a tag.
ValueColumnShowIcon *	P	P	Specifies whether an icon is shown.
ValueColumnShowTitleIcon *	P	P	Specifies whether an icon is shown in the title.
ValueColumnSort *	P	P	Specifies the sorting type of a column.
ValueColumnSortIndex *	P	P	Specifies the sorting order.
ValueColumnTagName *	P	P	Specifies the names of the tags whose values are displayed in a column.
ValueColumnTimeColumn *	P	P	Specifies the corresponding time column.
ValueColumnVisible *	P	P	Specifies the visibility of a value column.
VerticalGridLines (Page 5644)	P	P	Specifies whether vertical separation lines are displayed.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	-	Sets the width of the specified "Ellipse", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-80 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
AttachDB (Page 5832)	P	Executes the "Connect backup" button function of the control.
CalculateStatistic (Page 5832)	P	Executes the "Calculate statistics" button function of the f(t) trend view and table view.
CopyRows (Page 5833)	P	Executes the "Copy rows" button function of the control.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.
DetachDB (Page 5837)	P, A	Executes the "Disconnect backup" button function of the control.
Edit (Page 5838)	P	Executes the "Edit" button function of the table view.
Export (Page 5838)	P, A	Executes the "Export log" or "Export data" button function of the control.
GetRow (Page 5850)	P	Returns the row object defined by its row number in the table-based controls as "ICCAxDataRow" type.
GetRowCollection (Page 5851)	P	Returns the list of all row objects of the table-based controls as "ICCAxDataRowCollection" type.
GetSelectedRow (Page 5858)	P	Returns the selected row object of a table-based control as "ICCAxDataRow" type.
GetSelectedRows (Page 5859)	P	Returns the selected row objects of a table-based control as type "ICCAxDataRow" for multiple selection.
GetStatusbarElement (Page 5865)	P, A	Returns the element of the control's status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 5866)	P, A	Returns the list of all status bar elements of the control as "ICCAxCollection" type.
GetTimeColumn (Page 5871)	P	Returns the time column object of the table view designated by name or index as "ICCAxTime Column" type.
GetTimeColumnCollection (Page 5872)	P	Returns the list of all time column objects of the table view as "ICCAxCollection" type.
GetToolbarButton (Page 5873)	P, A	Returns the button function designated by name or index on the control toolbar as "ICCAxToolbarButton" type.
GetToolbarButtonCollection (Page 5874)	P, A	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.
GetValueColumn (Page 5883)	P	Returns the value column object of the f(t) trend view designated by name or index as "ICCAxValueColumn" type.
GetValueColumnCollection (Page 5884)	P	Returns the list of all value column objects of the f(t) trend view as "ICCAxCollection" type.
MoveToFirst (Page 5894)	P	Executes the "First line" button function of the control.
MoveToLast (Page 5896)	P	Executes the "Last data record" button function of the control.
MoveToNext (Page 5897)	P	Executes the "Next data record" button function of the control.
MoveToPrevious (Page 5899)	P	Executes the "Previous data record" button function of the control.
NextColumn (Page 5900)	P	Executes the "Next column" button function of the table view.
PreviousColumn (Page 5902)	P	Executes the "Previous column" button function of the table view.
Print (Page 5903)	P, A	Executes the "Print" button function of the control.
SelectAll (Page 5918)	P	Selects all rows in a table-based control.
SelectRow (Page 5919)	P	Selects a specific row in a table-based control.

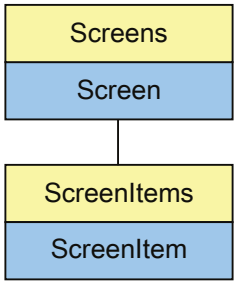
Methods	Valid	Description
SelectStatisticArea (Page 5918)	P	Executes the "Set statistics range" button function of the table view.
ShowColumnSelection (Page 5921)	P	Executes the "Select columns" button function of the table view.
ShowHelp (Page 5923)	P, A	Executes the "Help" button function of the control.
ShowPropertyDialog (Page 5927)	P, A	Executes the "Configuration dialog" button function of the control.
ShowTagSelection (Page 5931)	P, A	Executes the "Select data connection" button function of the control.
ShowTimeSelection (Page 5932)	P, A	Executes the "Select time range" button function of the control.
StartStopUpdate (Page 5933)	P, A	Executes the "Start" or "Stop" button function of the control.
UnselectAll (Page 5936)	P	Removes all selections from the cells of a table-based control.
UnselectRow (Page 5936)	P	Removes the selections from a specific cell of a table-based control.

See also

Screen object (list) (Page 5007)

OnlineTrendControl

Description



Represents the "f(t) TrendView" object. The OnlineTrendControl object is an element of the ScreenItems list.

Type identifier in VBS

HMIOnlineTrendControl

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-81 Properties

Properties	Read	Write	Description
ApplyProjectSettingsForDesignMode (Page 5243)	-	-	Specifies that the project settings are used for the design.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
Caption (Page 5287)	P	P	Specifies the text that is displayed in the header of the selected object.
Closeable (Page 5295)	P	P	Specifies that the control can be closed in runtime.
ConnectTrendWindows (Page 5304)	P	P	Specifies whether the configured trend views are connected.
ControlDesignMode (Page 5507)	P	P	Specifies the design.
ExportDirectoryChangeable (Page 5328)	P	P	Specifies whether the data export directory can be changed in runtime.
ExportDirectoryname (Page 5329)	P	P	Specifies the target directory for the exported Runtime data.
ExportFileExtension (Page 5330)	P	P	Specifies the file extension of the export file. Only the "csv" file extension is currently supported.
ExportFilename (Page 5330)	P	P	Specifies the name of the target file for the exported Runtime data.
ExportFilenameChangeable (Page 5331)	P	P	Specifies whether the export file name can be changed in runtime.
ExportFormatGuid (Page 5331)	P	P	Default assignment of the ID number and export provider.
ExportFormatName (Page 5332)	P	P	Specifies the export file format. Only the "csv" file format is currently available for the export.
ExportParameters (Page 5332)	P	P	Specifies the parameters of the selected format by means of the "Properties" dialog.
ExportSelection (Page 5333)	P	P	Specifies the file extension of the export file. Only the "csv" file extension is currently supported.
ExportShowDialog (Page 5334)	P	P	Specifies whether the data export dialog is shown in runtime.
Font (Page 5348)	-	-	Specifies or returns the font.
GraphDirection (Page 5356)	P	P	Specifies at which border of the trend window the current values are displayed.
Height (Page 5358)	P	-	Sets the height of the specified "Ellipse", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineColor (Page 5398)	P	P	Sets the color of the window separation lines.

Properties	Read	Write	Description
LineWidth	P	P	Specifies the line thickness of the selected object.
LoadDataImmediately (Page 5401)	P	P	Specifies whether the tag values are downloaded from the logs for the time range to be displayed when you call a screen.
Moveable (Page 5425)	P	P	Specifies whether the window can be moved in runtime.
Name (Page 5426)	P	-	Returns the object name as STRING.
Online (Page 5434)	P	P	Specifies the name of the data source.
PercentageAxis	P	P	Specifies that an axis with percentage scaling is shown.
PercentageAxisAlign	P	P	
PercentageAxisColor	P	P	Specifies the color of the percentage axis.
PrintJob	P	P	Specifies a print job
RTPersistence (Page 5466)	P	P	Specifies how online configurations of WinCC are retained.
RTPersistencPasswordLevel (Page 5716)	P	P	Specifies the authorization required for online configuration in runtime.
RTPersistenceType (Page 5467)	-	-	Specifies how online configurations of WinCC are retained.
ShowRuler (Page 5498)	P	P	Specifies whether a scale gradation (auxiliary line) is shown for the axis label of the "OnlineTrendControl" object.
ShowRulerInAxis (Page 5499)	P	P	Specifies whether rulers are also displayed in the time axes.
ShowScrollbars (Page 5500)	P	P	Specifies the scroll bar type.
ShowStatisticRuler	P	P	Specifies the SQL statement.
ShowTitle (Page 5504)	P	P	Specifies the layout of the control window's header.
ShowTrendIcon (Page 5505)	P	P	Specifies whether an icon is displayed below the value axes.
Sizeable (Page 5506)	P	P	Specifies that the size of an object can be changed in runtime.
StatusbarBackColor (Page 5511)	P	P	Specifies the background color of the status bar.
StatusbarElementAdd (Page 5512) *	P	P	Creates a new, user-defined status bar element.
StatusbarElementAutoSize (Page 5512) *	P	P	Specifies whether the width of the selected status bar element is set automatically.
StatusbarElementCount (Page 5513) *	P	P	Specifies the number of configurable status bar elements.
StatusbarElementIconId (Page 5514) *	P	P	Default assignment of the ID number and icon of a status bar element.
StatusbarElementId (Page 5514) *	P	P	Unique ID of the selected status bar element.
StatusbarElementIndex (Page 5515)	P	P	References a status bar element. You can use this property to assign the values of other properties to a specific element of the status bar.
StatusbarElementName (Page 5516) *	P	P	Displays the object name of the selected status bar element.
StatusbarElementRemove (Page 5516) *	P	P	Removes the selected status bar element.
StatusbarElementRename (Page 5517) *	P	P	Renames the user-defined status bar element that is referenced by means of the "StatusbarElementIndex" property.
StatusbarElementRepos *	P	P	Specifies the index that is assigned to the element.
StatusbarElements (Page 5517)	-	-	Specifies the elements that are to be shown in the status bar.
StatusbarElementText *	P	P	Specifies the text of the currently selected status bar element.
StatusbarElementTooltipText (Page 5518) *	P	P	Specifies the tooltip text for the user-defined status bar element.

Properties	Read	Write	Description
StatusbarElementUserDefined (Page 5518) *	P	P	Indicates whether the configuration engineer has added the status bar element as a new user-defined element.
StatusbarElementVisible (Page 5519) *	P	P	In the list, activate the status bar elements that you wish to display in runtime.
StatusbarElementWidth (Page 5519) *	P	P	Specifies the width of the selected status bar element in pixels.
StatusbarFont	-	-	Status bar font
StatusbarFontColor (Page 5520)	P	P	Specifies the color of the text in the status bar.
StatusbarShowTooltips (Page 5521)	P	P	Specifies whether tooltips are displayed for the elements of the status bar in runtime.
StatusbarText (Page 5521)	P	P	Default text in the status bar.
StatusbarUseBackColor (Page 5522)	P	P	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 5523)	P	P	Specifies whether the status bar of the control is displayed.
TimeAxes	-	-	Specifies the settings for the time axes.
TimeAxisActualize *	P	P	
TimeAxisAdd *	P	P	Inserts an axis.
TimeAxisAlignment *	P	P	Specifies the alignment of an axis.
TimeAxisBeginTime (Page 5538) *	P	P	Specifies the starting time for the display of the selected trend.
TimeAxisColor *	P	P	Specifies the color of an axis.
TimeAxisCount *	P	P	Displays the number of time axes.
TimeAxisDateFormat *	P	P	Specifies the date format.
TimeAxisEndTime (Page 5539) *	P	P	Specifies the end time for the display of the selected trend.
TimeAxisIndex	P	P	The index specifies on which axis other properties, e.g. initial value are based.
TimeAxisInTrendColor *	P	P	Specifies that the color of the trend is used for an axis.
TimeAxisLabel (Page 5539) *	P	P	Specifies the header for the time axis. Whether or not the information is evaluated depends on the "ConfigureTimeAxis(i)" property.
TimeAxisMeasurePoints *	P	P	Specifies the measuring points.
TimeAxisName *	P	P	Specifies the name of an axis.
TimeAxisRangeType *	P	P	Specifies the type of time range.
TimeAxisRemove *	P	P	Deletes a time axis
TimeAxisRename *	P	P	Renames a time axis.
TimeAxisRepos *	P	P	Specifies that the time axis is repositioned
TimeAxisShowDate *	P	P	Specifies that the date is to be displayed.
TimeAxisTimeFormat (Page 5540) *	P	P	Specifies the time format.
TimeAxisTimeRangeBase *	P	P	A time range is set via a base and a factor.
TimeAxisTimeRangeFactor *	P	P	A time range is set via a base and a factor.
TimeAxisTrendWindow *	P	P	Specifies the trend view.
TimeAxisVisible *	P	P	Specifies that the time axis is visible.
TimeBase (Page 5540)	P	P	Sets the time zone that forms the basis for the display of time values.
ToolbarAlignment (Page 5568)	P	P	Specifies or returns the position of the toolbar.

Properties	Read	Write	Description
ToolbarBackColor (Page 5568)	P	P	Specifies the background color of the toolbar.
ToolbarButtonActive (Page 5569) *	P	P	Specifies whether the function linked to the button is activated in runtime.
ToolbarButtonAdd (Page 5570) *	P	P	Creates a new, user-defined button function.
ToolbarButtonBeginGroup (Page 5570) *	P	P	Specifies whether a separator is inserted before the selected button function.
ToolbarButtonClick	P	P	Clicks a toolbar button.
ToolbarButtonCount (Page 5571) *	P	P	Specifies the number of configurable button functions.
ToolbarButtonEnabled (Page 5572) *	P	P	Specifies whether a user-defined toolbar button can be operated.
ToolbarButtonHotKey (Page 5572) *	P	P	Shows the hotkey for the selected button function.
ToolbarButtonId (Page 5573) *	P	P	Unique ID number for the selected button function.
ToolbarButtonIndex (Page 5573)	P	P	References a button function.
ToolbarButtonLocked (Page 5574) *	P	P	Specifies whether the clicked, pressed in state is displayed for a user-defined toolbar button.
ToolbarButtonName (Page 5575) *	P	P	Shows the name of the selected button function.
ToolbarButtonPasswordLevel (Page 5575) *	P	P	Shows the authorization for the selected button function.
ToolbarButtonRemove (Page 5576) *	P	P	Removes the selected button function from the list.
ToolbarButtonRename (Page 5576) *	P	P	Renames the user-defined toolbar element that is referenced by means of the "ToolbarButtonIndex" property.
ToolbarButtonRepos (Page 5577) *	P	P	Changes the sequence of button functions.
ToolbarButtons	-	-	Specifies which buttons are available for the toolbar.
ToolbarButtonTooltipText (Page 5577) *	P	P	Specifies the tooltip text for the button.
ToolbarButtonUserDefined (Page 5578) *	P	P	Indicates whether the configuration engineer has added the toolbar button as a new user-defined button.
ToolbarButtonVisible *	P	P	Specifies that the button is visible.
ToolbarShowTooltips (Page 5579)	P	P	Specifies whether the tooltips for the button functions are displayed in runtime.
ToolbarUseBackColor (Page 5579)	P	P	Specifies whether the background color for the toolbar is shown.
ToolbarUseHotKeys (Page 5580)	P	P	Specifies whether the hotkeys for the button functions are activated in runtime.
ToolbarVisible (Page 5580)	P	P	Specifies whether the toolbar of the control is displayed.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
TrendAdd (Page 5590) *	P	P	Creates a new trend.
TrendAutoRangeBeginTagName *	P	P	Specifies a tag that defines the initial value.
TrendAutoRangeBeginValue *	P	P	Specifies the initial value for the automatically adjusted display range.
TrendAutoRangeEndTagName *	P	P	Specifies a tag that defines the end value.
TrendAutoRangeEndValue *	P	P	Specifies a tag that defines the end value.
TrendAutoRangeSource *	P	P	Specifies the source of the display range.
TrendColor *	P	P	Specifies or returns the trend view color.
TrendCount (Page 5591) *	P	P	Specifies the number of configured trends.

Properties	Read	Write	Description
TrendExtendedColorSet (Page 5592) *	P	P	Specifies whether you can configure the point color and the fill color and whether the colors are shown in runtime.
TrendFill (Page 5593) *	P	P	Specifies whether the area below the trend is shown filled.
TrendFillColor (Page 5593) *	P	P	Sets the fill color of the trend.
TrendIndex (Page 5594)	P	P	References a configured trend.
TrendLabel (Page 5595) *	P	P	Specifies the label of the selected trend.
TrendLineStyle (Page 5596) *	P	P	Specifies which line type is used for display of the trend.
TrendLineType (Page 5596) *	P	P	Specifies how the trend is displayed.
TrendLineWidth (Page 5597) *	P	P	Specifies the line thickness of the line displayed.
TrendLowerLimit (Page 5597) *	P	P	Specifies the low limit of a tag.
TrendLowerLimitColor (Page 5598) *	P	P	Specifies the color which indicates tag values below the value of "TrendLowerLimit".
TrendLowerLimitColoring (Page 5598) *	P	P	Specifies whether the selection frame is shown with the color defined by the system.
TrendName (Page 5599) *	P	P	Displays the name of the selected trend.
TrendPointColor (Page 5600) *	P	P	Specifies the color of trend points.
TrendPointStyle (Page 5600) *	P	P	Specifies how the points are displayed on the trend.
TrendPointWidth (Page 5601) *	P	P	Specifies the point width in pixels.
TrendProvider (Page 5601) *	P	P	Specifies the data supply for the selected trend.
TrendProviderCLSID *	P	P	Specifies the provider CLSID of the data for a trend.
TrendRemove (Page 5603) *	P	P	Removes selected trends from the list.
TrendRename (Page 5603) *	P	P	Renames the trend that is referenced by means of the "TrendIndex" property.
TrendRepos (Page 5604) *	P	P	Repositions the selected trend in the sequence in the trend window.
TrendSelectTagName *	P	P	Specifies a tag.
TrendTagName *	P	P	Specifies the name of the tag for the Y direction.
TrendTimeAxis *	P	P	Specifies the time axis.
TrendTrendWindow (Page 5607) *	P	P	Specifies the trend window in which the selected trend is shown.
TrendUncertainColor (Page 5608) *	P	P	Values have an uncertain status if the initial value is unknown after Runtime has been activated, or if a substitute value is used.
TrendUncertainColoring (Page 5608) *	P	P	Values have an uncertain status if the initial value is unknown after Runtime has been activated, or if a substitute value is used.
TrendUpperLimit (Page 5609) *	P	P	Specifies the high limit of a tag. If the tag drops below the value of "TrendLowerLimit", the values are marked with the color set in "TrendUpperLimitColor".
TrendUpperLimitColor (Page 5609) *	P	P	Specifies the color which indicates tag values below the value of "TrendLowerLimit".
TrendUpperLimitColoring (Page 5610) *	P	P	Specifies whether the selection frame is shown with the color defined by the system.
TrendValueAxis *	P	P	Specifies the value axis.
TrendVisible (Page 5610) *	P	P	The list contains all the trends that you have created.
TrendWindowAdd (Page 5611) *	P	P	Creates a new trend view.

Properties	Read	Write	Description
TrendWindowCoarseGrid (Page 5611) *	P	P	Specifies whether grid lines are displayed for the main scaling.
TrendWindowCoarseGridColor (Page 5612) *	P	P	Specifies the color of the grid lines for the main scaling.
TrendWindowCount *	P	P	Displays the number of trend views.
TrendWindowFineGrid (Page 5613) *	P	P	Specifies whether grid lines are displayed for the secondary scaling.
TrendWindowFineGridColor (Page 5613) *	P	P	Specifies the grid color of the main scaling.
TrendWindowForegroundTrendGrid (Page 5614) *	P	P	Specifies whether only the grid lines for the foreground trend are displayed in the selected trend window.
TrendWindowGridInTrendColor (Page 5614) *	P	P	Specifies whether the grid lines for the main scaling are displayed in the trend color.
TrendWindowHorizontalGrid (Page 5615) *	P	P	Specifies whether the horizontal grid lines are shown.
TrendWindowIndex (Page 5615)	P	P	References a configured trend window.
TrendWindowName (Page 5616) *	P	P	Specifies the name of the selected trend window.
TrendWindowRemove (Page 5616) *	P	P	Removes the selected trend view from the list.
TrendWindowRename (Page 5616) *	P	P	Changes the name of the trend window which is referenced by the "TrendWindowIndex" property.
TrendWindowRepos (Page 5617) *	P	P	Changes the order of the trend view.
TrendWindowRulerColor (Page 5617) *	P	P	Specifies the ruler color.
TrendWindowRulerLayer (Page 5618) *	P	P	Specifies the display layer of the ruler in the trend view.
TrendWindowRulerStyle (Page 5619) *	P	P	Specifies the representation of the ruler.
TrendWindowRulerWidth (Page 5619) *	P	P	Specifies the ruler width in pixels.
TrendWindows	-	-	Specifies the settings for trend views.
TrendWindowSpacePortion (Page 5620) *	P	P	Specifies the proportion of the selected trend window for display in the control.
TrendWindowStatisticRulerColor *	P	P	Specifies the color of the statistics ruler.
TrendWindowStatisticRulerStyle *	P	P	Specifies the style of the statistics ruler.
TrendWindowStatisticRulerWidth *	P	P	Specifies the width of the statistics ruler.
TrendWindowVerticalGrid (Page 5620) *	P	P	Specifies whether the vertical grid lines are displayed.
TrendWindowVisible (Page 5621) *	P	P	The list contains the trend views that you have created.
UseTrendNameAsLabel (Page 5634)	P	P	Specifies whether the "Name" or "Label" property is used as a designation for the trend in runtime.
ValueAxes	-	-	Specifies the settings for the value axes.
ValueAxisAdd *	P	P	Inserts an axis.
ValueAxisAlignment *	P	P	Specifies the alignment of an axis.
ValueAxisAutoPrecisions *	P	P	Specifies that the number of decimal places displayed for an axis is adjusted automatically.
ValueAxisAutoRange *	P	P	Specifies that the value range is determined automatically.

Properties	Read	Write	Description
ValueAxisBeginValue *	P	P	Specifies the initial value of an axis.
ValueAxisColor *	P	P	Specifies the color of a value axis.
ValueAxisCount *	P	P	Shows the number of value axes.
ValueAxisEndValue *	P	P	Specifies the end value of an axis.
ValueAxisExponentialFormat *	P	P	Specifies that the label of an axis uses exponential notation.
ValueAxisIndex	P	P	The index specifies on which axis other properties, e.g. initial value are based.
ValueAxisInTrendColor *	P	P	Specifies that the color of the trend is used for an axis.
ValueAxisLabel (Page 5641) *	P	P	Specifies the label of the value axis.
ValueAxisName *	P	P	Specifies the name of an axis.
ValueAxisPrecisions *	P	P	Specifies the number of displayed decimal places.
ValueAxisRemove *	P	P	Renames an axis.
ValueAxisRename *	P	P	Renames an axis.
ValueAxisRepos *	P	P	Specifies that the value axis is repositioned.
ValueAxisScalingType (Page 5642) *	P	P	Specifies the type of axis scaling.
ValueAxisTrendWindow *	P	P	Specifies the trend view.
ValueAxisVisible *	P	P	Specifies whether a value axis is visible.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	-	Sets the width of the specified "Ellipse", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-82 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
AttachDB (Page 5832)	P, A	Executes the "Connect backup" button function of the control.
CalculateStatistic (Page 5832)	P	Executes the "Calculate statistics" button function of the f(t) trend view and table view.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.
DetachDB (Page 5837)	P, A	Executes the "Disconnect backup" button function of the control.
Export (Page 5838)	P, A	Executes the "Export log" or "Export data" button function of the control.
GetRulerData (Page 5857)	P	Returns the value of the called trend at the ruler position.
GetStatusbarElement (Page 5865)	P, A	Returns the element of the control's status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 5872)	P, A	Returns the list of all status bar elements of the control as "ICCAxCollection" type.
GetTimeAxis (Page 5868)	P	Returns the time axis object of the f(t) trend view designated by name or index as "ICCAxTimeAxis" type.
GetTimeAxisCollection (Page 5869)	P	Returns the list of all time axis objects of the f(t) trend view as "ICCAxCollection" type.

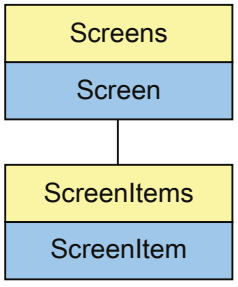
Methods	Valid	Description
GetToolbarButton (Page 5873)	P	Returns the button function designated by name or index on the control toolbar as "ICCAxToolbarButton" type.
GetToolbarButtonCollection (Page 5874)	P	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.
GetTrend (Page 5876)	P, A	Returns the f(t) or f(x) trend view designated by name or index as "ICCAxTrend" or "ICCAxFunctionTrend" type.
GetTrendCollection (Page 5877)	P, A	Returns the list of all trends of the f(t) or f(x) trend view as "ICCAxCollection" type.
GetTrendWindow (Page 5878)	P, A	Returns the trend window object of the f(t) or f(x) trend view designated by name or index as "ICCAxTrendWindow" type.
GetTrendWindowCollection (Page 5879)	P, A	Returns the list of all trend window objects of the f(t) or f(x) trend views as "ICCAxCollection" type.
GetValueAxis (Page 5880)	P	Returns the value axis object of the f(t) trend view designated by name or index as "ICCAxValueAxis" type.
GetValueAxisCollection (Page 5881)	P	Returns the list of all value axis objects of the f(t) trend view as "ICCAxCollection" type.
MoveAxis (Page 5894)	P, A	Executes the "Move axes area" button function of the f(t) and f(x) trend views.
MoveToFirst (Page 5894)	P	Executes the "First line" button function of the control.
MoveToLast (Page 5896)	P	Executes the "Last data record" button function of the control.
MoveToNext (Page 5897)	P	Executes the "Next data record" button function of the control.
MoveToPrevious (Page 5899)	P	Executes the "Previous data record" button function of the control.
NextTrend (Page 5901)	P, A	Executes the "Next trend" button function of the f(t) and f(x) trend views.
OneToOneView (Page 5901)	P, A	Executes the "Original view" button function of the f(t) and f(x) trend views.
PreviousTrend (Page 5903)	P, A	Executes the "Previous trend" button function of the f(t) and f(x) trend views.
Print (Page 5903)	P, A	Executes the "Print" button function of the control.
ShowHelp (Page 5923)	P, A	Executes the "Help" button function of the control.
ShowPercentageAxis (Page 5927)	P	Executes the "Relative axis" button function of the f(t) trend view.
ShowPropertyDialog (Page 5927)	P, A	Executes the "Configuration dialog" button function of the control.
ShowTagSelection (Page 5931)	P, A	Executes the "Select data connection" button function of the control.
ShowTimeSelection (Page 5932)	P, A	Executes the "Select time range" button function of the control.
ShowTrendSelection (Page 5933)	P, A	Executes the "Select trends" button function of the f(t) and f(x) trend views.
StartStopUpdate (Page 5933)	P, A	Executes the "Start" or "Stop" button function of the control.
ZoomArea (Page 5940)	P, A	Executes the "Zoom area" button function of the f(t) and f(x) trend views.
ZoomInOut (Page 5941)	P, A	Executes the "Zoom +/-" button function of the f(t) and f(x) trend views.
ZoomInOutTime (Page 5941)	P	Executes the "Zoom time axis +/-" button function of the f(t) trend view.
ZoomInOutValues (Page 5942)	P	Executes the "Zoom value axis +/-" button function of the f(t) trend view.
ZoomMove (Page 5943)	P, A	Executes the "Move trend area" button function of the f(t) and f(x) trend views.

See also

Screen object (list) (Page 5007)

OptionGroup

Description



Represents the "Option button" object. The OptionGroup object is an element of the ScreenItems list.

Type identifier in VBS

HMIOptionGroup

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-83 Properties

Properties	Read	Write	Description
AdaptBorder (Page 5232)	P	P	Specifies whether the border of the specified object is dynamically adapted to the text size.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackFlashingColorOff (Page 5256)	P	P	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 5257)	P	P	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258)	P	P	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258)	P	P	Specifies the flash rate of the background for the selected object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
CheckmarkAlignment (Page 5294)	P	P	Specifies whether the fields are right aligned.
CheckmarkCount (Page 5294)	P	P	Specifies the number of fields.
CornerStyle (Page 5306)	-	-	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	-	-	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.

Properties	Read	Write	Description
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	P	P	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 5350)	P	P	Specifies whether the text of the selected object is shown in italics.
FontName (Page 5351)	P	P	Specifies the font of the selected object.
FontSize (Page 5351)	P	P	Specifies the font size of the text for the selected object.
FontUnderline (Page 5352)	P	P	Specifies whether the text of the selected object is underlined.
ForeColor (Page 5353)	P	P	Specifies the font color of the text for the selected object.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HorizontalAlignment (Page 5372)	P	P	Specifies the horizontal alignment of the text within the selected object.
Index (Page 5375)	P	P	Specifies the background for grid control elements.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	-	-	Specifies the shape of the line ends.
LogOperation (Page 5403)	P	P	Specifies whether, after operating this object, an alarm is output to the alarm system.
Name (Page 5426)	P	-	Returns the object name as STRING.
ProcessValue (Page 5456)	P	P	Specifies the default for the value to be displayed.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
ShowBadTagState (Page 5493)	P	P	Specifies whether an object is grayed out if its associated tag has a bad quality code or tag state.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
Text (Page 5533)	P	P	Specifies the label for the text field.
TextOrientation (Page 5536)	P	P	Specifies the text orientation of the selected object.
Texts	-	-	Specifies the texts for the check boxes.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
VerticalAlignment (Page 5643)	P	P	Specifies the vertical alignment of the text for the selected object.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-84 Methods

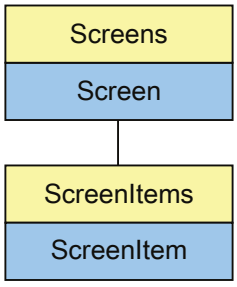
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

Polygon

Description



Represents the "Polygon" object. The Polygon object is an element of the ScreenItems list.

Type identifier in VBS

HMIPolygon

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-85 Properties

Properties	Read	Write	Description
ActualPointIndex (Page 5230)	P	P	Specifies the number of the current corner point of the selected object.
ActualPointLeft (Page 5230)	P	P	Specifies the X coordinate of the current corner point with reference to the screen origin.
ActualPointTop (Page 5231)	P	P	Specifies the Y coordinate of the current corner point with reference to the screen origin.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P, A	P, A	Defines the fill pattern of the specified object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P, A	P, A	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P, A	P, A	Specifies the line thickness of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
EdgeStyle (Page 5320)	A	A	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Properties	Read	Write	Description
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
Points	-	-	Specifies the corner points.
PointsCount (Page 5454)	P	-	Specifies the number of corner points of the polyline or of the polygon.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
RotationAngle (Page 5463)	P	P	Specifies the angle of rotation in degrees.
RotationCenterLeft (Page 5463)	P	P	Specifies the X coordinate of the pivot point for the object in runtime.
RotationCenterTop (Page 5464)	P	P	Specifies the Y coordinate of the pivot point for the object in runtime.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-86 Methods

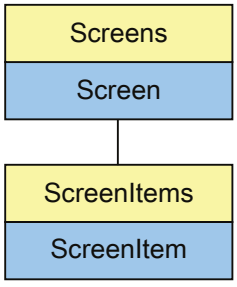
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

Polyline

Description



Represents the "Polyline" object. The Polyline object is an element of the ScreenItems list.

Type identifier in VBS

HMIPolyline

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-87 Properties

Properties	Read	Write	Description
ActualPointIndex (Page 5230)	P	P	Specifies the number of the current corner point of the selected object.
ActualPointLeft (Page 5230)	P	P	Specifies the X coordinate of the current corner point with reference to the screen origin.
ActualPointTop (Page 5231)	P	P	Specifies the Y coordinate of the current corner point with reference to the screen origin.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.

Properties	Read	Write	Description
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BorderEndStyle (Page 5270)	P; A	P	Specifies the type of line ends for the selected object.
BorderStyle (Page 5278)	P, A	P	Specifies the type of border lines for the selected object.
Color (Page 5296)	P, A	P, A	Specifies the line color of the selected object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
EndStyle (Page 5326)	A	A	Specifies how the line end of the selected object is displayed.
FillStyle	A	A	Defines the fill pattern of the specified object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
LineWidth (Page 5400)	P, A	P, A	Specifies the line thickness of the selected object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
Points	-	-	Specifies the corner points.
PointsCount (Page 5454)	P	-	Specifies the number of corner points of the polyline or of the polygon.
RotationAngle (Page 5463)	P	P	Specifies the angle of rotation in degrees.
RotationCenterLeft (Page 5463)	P	P	Specifies the X coordinate of the pivot point for the object in runtime.
RotationCenterTop (Page 5464)	P	P	Specifies the Y coordinate of the pivot point for the object in runtime.
StartStyle	A	A	Specifies how the line start of the selected object is displayed.
Style (Page 5523)	A	A	Specifies the line style of the selected object.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-88 Methods

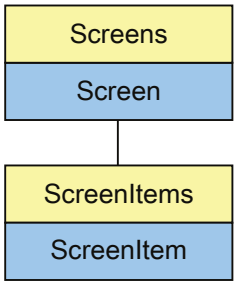
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

ProjectName

Description



Represents the "Project name" object. The ProjectName object is an element of the ScreenItems list.

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-89 Properties

Properties	Read	Write	Description
BackColor	-	-	Specifies the background color of the selected object.
BackFillStyle	-	-	Defines the fill pattern of the specified object.
BorderBackColor (Page 5266)	-	-	Specifies the background color of the broken border line for the selected object.
BorderColor	-	-	Specifies the line color of the object.
BorderWidth	-	-	Specifies the line thickness of the object.
EdgeStyle (Page 5320)		-	Specifies the line style of the selected object.
FillPatternColor (Page 5336)	-	-	Specifies the color of the fill pattern for the selected object.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	-	-	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 5350)	-	-	Specifies whether the text of the selected object is shown in italics.
FontName (Page 5351)	-	-	Specifies the font of the selected object.
FontSize (Page 5351)	-	-	Specifies the font size of the text for the selected object.
FontUnderline (Page 5352)	-	-	Specifies whether the text of the selected object is underlined.
ForeColor	-	-	Specifies the font color of the text for the selected object.
Format	-	-	Specifies that the format of the text block is displayed.
Height (Page 5358)	-	-	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HorizontalAlignment	-	-	Specifies the horizontal alignment of the text within the selected object.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	-	-	Specifies the value of the X coordinate of the object.
LineWrap	-	-	Specifies the line break within the object.
Name (Page 5426)	-	-	Returns the object name as STRING.
Top (Page 5582)	-	-	Specifies the value of the Y coordinate of the object.
VerticalAlignment	-	-	Specifies the vertical alignment of the text for the selected object.
Visible (Page 5645)	-	-	Specifies whether the selected object is visible.
Width (Page 5655)	-	-	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

None of the properties are visible in the ES.

Table 12-90 Methods

Methods	Validity	Description

Methods	Validity	Description

See also

Screen (Page 5000)

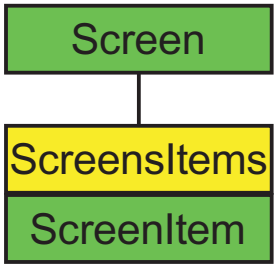
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

ProtectedAreaName

Description



Represents the "EffectiveRangeName" (RFID)" object. The ProtectedAreaName object is an element of the ScreenItems list.

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-91 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	-	-	Specifies the operating rights of the selected object in runtime.
Font (Page 5348)	-	-	Specifies or returns the font.

Properties	Read	Write	Description
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	A	-	Returns the object name as STRING.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-92 Methods

Methods	Valid	Description

See also

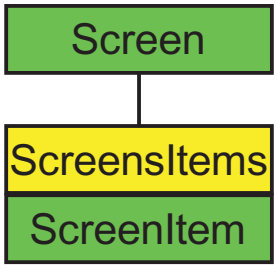
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

RangeLabelView

Description



Represents the "EffectiveRangeName" object. The RangeLabelView object is an element of the ScreenItems list.

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-93 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	-	-	Specifies the operating rights of the selected object in runtime.
Font (Page 5348)	-	-	Specifies or returns the font.
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	A	-	Returns the object name as STRING.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-94 Methods

Methods	Valid	Description

See also

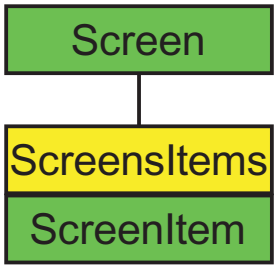
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

RangeQualityView

Description



Represents the "EffectiveRangeSignal" object. The RangeQualityView object is an element of the ScreenItems list.

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-95 Properties

Properties	Read	Write	Description
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	A	-	Returns the object name as STRING.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-96 Methods

Methods	Valid	Description

Methods	Valid	Description

See also

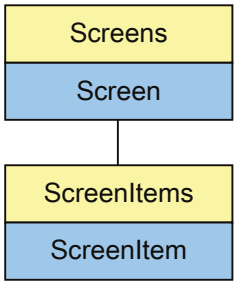
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

RecipeView

Description



Represents the "RecipeView" object. The RecipeView object is an element of the ScreenItems list.

If you change the settings for this object with a user-defined function, the changed settings are retained even after the screen is called again.

Note

The object "Simple RecipeView" cannot be dynamized with a user-defined function.

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-97 Properties

Properties	Read	Write	Description
AllowEdit	-	-	Enables editing.
Authorization (Page 5245)	-	-	Specifies the operating rights of the selected object in runtime.
BackButtonVisible	-	-	Enables display of the "Back" button.
BackColor	A	A	Specifies the background color of the selected object.
ComboBoxFont	-	-	Sets the font type for selection list.
DataRecordNameCaption	-	-	Sets the caption for the name of the data record.
DataRecordNrCaption	-	-	Sets the caption for the number of the data record.
Display3D	-	-	Sets the 3D view.
DisplayButton2Plc	-	-	Enables the display of the "Write to PLC" button.
DisplayButtonComparison	-	-	Enables the display of the "Synchronization" button.
DisplayButtonDelete	-	-	Enables the display of the "Delete data record" button.
DisplayButtonFromPlc	-	-	Enables the display of the "Read from PLC" button.
DisplayButtonHelp	-	-	Enables the display of the "Tooltip" button.
DisplayButtonNew	-	-	Enables the display of the "Add data record" button.
DisplayButtonSave	-	-	Enables the display of the "Save" button.
DisplayButtonSaveAs	-	-	Enables the display of the "Save As" button.
DisplayComboBox	-	-	Enables the display of the selection list.
DisplayGridlines	-	-	Enables the display of grid lines.
DisplayLabeling	-	-	Enables the display labelings.
DisplayNumbers	-	-	Enables the display of numbers.
DisplaySize *	-	-	
DisplayStatusBar	-	-	Specifies that the status bar is displayed.
DisplayTable	-	-	Enablers the display of the table.
Enabled	A	A	Specifies whether the selected object can be operated in runtime.
EntryNameCaption	-	-	Sets the caption for the name of the recipe element.
EntryValueColFirst	-	-	Sets the "Entry value" column to first place.
EntryValueFieldLength	-	-	Sets the field length for the element value.
EntryValuePos	-	-	Sets the position of the first value.
FitToSize	-	-	Specifies that the size of the object should be automatically adapted to the content.
FixedRecipeNumber *	-	-	
Flashing	-	-	Specifies that the text block flashes.
FocusColor	A	A	Specifies the color for the focus frame of the selected object when it is in focus.

Properties	Read	Write	Description
FocusWidth	A	A	Specifies the border width of the specified object when the object is in focus.
Font (Page 5348)	-	-	Specifies or returns the font.
ForeColor	A	A	Specifies the font color of the text for the selected object.
HeaderFont	-	-	Sets the header font.
Height (Page 5358)	A	-	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
KeyboardOnline	-	-	Enables the use of key combinations.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
ListAreaHeight	-	-	
ListAreaWidth	-	-	
MenuButtonVisible	-	-	Enables visibility of the menu buttons.
Name (Page 5426)	A	-	Returns the object name as STRING.
NameColumnWidth	-	-	
NumberOfLines	-	-	Sets the number of rows of the selection list for the text list object, or returns the value.
Recipe	-	-	Sets the recipe.
RecipeNameCaption	-	-	Sets the caption for the name of the recipe.
RecipeNameSelected *	-	-	
RecipeNrCaption	-	-	Specifies that the "Recipe number" column is in first place.
RecipeNrColFirst	-	-	Specifies that the "Recipe number" column is in first place.
RecipeVarSelected *	-	-	
Record *	-	-	
RecordNameSelected *	-	-	
RecordNrColFirst	-	-	Specifies that the "Data record number" column is in first place.
RecordVarSelected *	-	-	
RenameButtonVisible	-	-	Enables the display of the "Rename" button.
SelectionBackColor (Page 5484)	A	A	Specifies the background color of the selected cells.
SelectionForeColor (Page 5485)	A	A	Specifies the foreground color of the selected cells.
StatusFont	-	-	
TableBackColor (Page 5525)	A	A	Specifies the background color of the table cells for the selected object.
TableForeColor	A	A	
TableGridlineColor (Page 5528)	A	A	Specifies the color for the grid lines.
TableHeaderBackColor (Page 5529)	A	A	Specifies the background color in the header of the table of the selected object.
TableHeaderForeColor (Page 5530)	A	A	Specifies the text color in the header of the table of the selected object.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
ValueCaption	-	-	Sets the labeling for values.
ValueColumnWidth	-	-	
VerticalScrolling	-	-	Sets the display of a scroll bar for the selected object.
ViewType	-	-	Specifies the alarm view type.

Properties	Read	Write	Description
ViewTypeForSaveStream *	-	-	Specifies the display type for the Save Stream.
Visible (Page 5645)	A	A	Specifies whether the selected object is visible.
VisibleItems	-	-	Sets the maximum number of visible entries.
Width (Page 5655)	A	-	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-98 Methods

Methods	Valid	Description

See also

Screen object (list) (Page 4975)

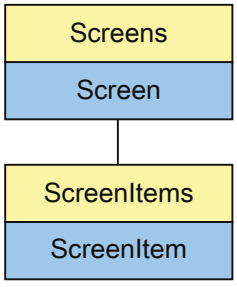
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

Rectangle

Description



Represents the "Rectangle" object. The Rectangle object is an element of the ScreenItems list.

Type identifier in VBS

HMIRectangle

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-99 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P, A	P, A	Defines the fill pattern of the specified object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P, A	P, A	Specifies the line color of the object.

Properties	Read	Write	Description
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P, A	P, A	Specifies the line thickness of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	A	A	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
RoundCornerHeight (Page 5465)	P	P	Sets or returns the corner radius.
RoundCornerWidth (Page 5465)	P	P	Sets or returns the corner radius.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-100 Methods

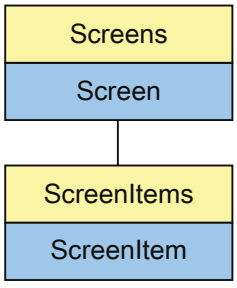
Methods	Validity	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

RoundButton

Description



Represents the "RoundButton" object. The RoundButton object is an element of the ScreenItems list.

Type identifier in VBS

HMIRoundButton

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-101 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackFlashingColorOff (Page 5256)	P	P	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 5257)	P	P	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258)	P	P	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258)	P	P	Specifies the flash rate of the background for the selected object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderBrightColor3D (Page 5268)	P	P	Specifies the border color of the following border parts in the 3D display of the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderShadeColor3D (Page 5277)	P	P	Specifies the border color of the following border parts in the 3D display of the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
BorderWidth3D (Page 5281)	P	P	Specifies the width of the border for 3D display of the selected object.
CornerStyle (Page 5306) *	-	-	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	P	P	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.

Properties	Read	Write	Description
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	P	P	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 5350)	P	P	Specifies whether the text of the selected object is shown in italics.
FontName (Page 5351)	P	P	Specifies the font of the selected object.
FontSize (Page 5351)	P	P	Specifies the font size of the text for the selected object.
FontUnderline (Page 5352)	P	P	Specifies whether the text of the selected object is underlined.
ForeColor (Page 5353)	P	P	Specifies the font color of the text for the selected object.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HorizontalAlignment (Page 5372)	P	P	Specifies the horizontal alignment of the text within the selected object.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399) *	-	-	Specifies the shape of the line ends.
Mode (Page 5424)	P	P	Specifies the field type of the selected object.
Name (Page 5426)	P	-	Returns the object name as STRING.
PictureAlignment (Page 5451)	P	P	Specifies or returns the display type of the background image in the process picture.
PictureDeactivated (Page 5452)	P	P	Specifies the picture that is displayed in the "Disabled" state.
PictureOff (Page 5452)	P	P	Specifies the screen to be displayed in the "Off" state.
PictureOn (Page 5453)	P	P	Specifies the screen to be displayed in the "On" state.
Pressed (Page 5455)	P	P	Specifies whether the selected object is displayed in a pressed state.
Radius	P	P	Sets the radius of the specified "Circle" object.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
StyleSettings (Page 5524)	P	P	Specifies the display style for the object:
Text (Page 5533)	P	P	Specifies the label for the text field.
TextOrientation (Page 5536)	P	P	Specifies the text orientation of the selected object.
Toggle (Page 5562)	P	P	Specifies whether the selected object engages after it has been operated in runtime.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.

Properties	Read	Write	Description
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
TransparentColorDeactivatedPicture (Page 5587)	P	P	Specifies the color to be set to "transparent" for the "Off" state of the assigned bitmap object.
TransparentColorPictureOff (Page 5588)	P	P	Specifies the color to be set to "transparent" for the "Off" state of the assigned bitmap object.
TransparentColorPictureOn (Page 5589)	P	P	Specifies the color to be set to "transparent" for the "On" state of the assigned bitmap object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
UseTransparentColorDeactivated-Picture (Page 5632)	P	P	Enables the use of the transparency color specified with the "TransparentColorDeactivatedPicture" property for the "Disabled" state.
UseTransparentColorPictureOff (Page 5633)	P	P	Specifies whether the transparent color defined in the "TransparentColorPictureOff" property is used for the "Off" state.
UseTransparentColorPictureOn (Page 5633)	P	P	Specifies whether the transparent color defined in the "TransparentColorPictureOn" property is used for the "On" state.
VerticalAlignment (Page 5643)	P	P	Specifies the vertical alignment of the text for the selected object.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-102 Methods

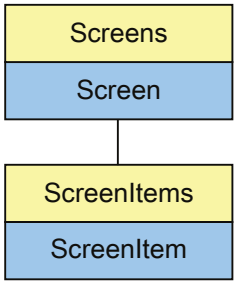
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

ScreenWindow

Description



Represents the "ScreenWindow" object. The ScreenWindow object is an element of the ScreenItems list.

Type identifier in VBS

HMIScreenWindow

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-103 Properties

Properties	Read	Write	Description
AdaptScreenToWindow (Page 5232)	-	-	Specifies whether or not the screen visualized in a screen window is adapted to fit the size of the screen window in runtime.
AdaptWindowToScreen (Page 5233)	-	-	Specifies whether or not the screen window is adapted to fit the screen it visualizes in runtime.
BorderEnabled (Page 5270)	-	-	TRUE, if the window is visualized with a border in runtime.

Properties	Read	Write	Description
CaptionText (Page 5288)	P	P	Specifies the text that is displayed in the header of the selected object.
Flashing *	-	-	Specifies that the text block flashes.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HorizontalScrollbarPosition (Page 5373)	P	P	Sets horizontal positioning of the scroll bar in a screen window with slider, or returns its value.
IndependentWindow	-	-	Specifies whether the screen window is embedded in a screen or displayed as independent screen window.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LeftOffset (Page 5389)	P	P	Sets horizontal shift of the display of a graphic object that is larger than the screen window.
MenuToolBarConfig (Page 5406)	-	-	Loads the specified configuration file with configured menu and toolbars, or returns the name of the configuration file.
MonitorNumber	-	-	Sets the monitor number.
Name (Page 5426)	P	-	Returns the object name as STRING.
ScreenName (Page 5475)	P	P	Specifies the screen to be displayed in the screen window, or returns the screen name.
ShowCaption (Page 5494)	-	-	Hides or shows the caption.
ShowScrollBars (Page 5500)	-	-	Enables the display of scroll bars.
TagPrefix (Page 5530)	P	P	Sets the prefix for all tags that exist in the screen.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
TopOffset (Page 5584)	P	P	Sets or returns the offset of the screen from the top edge of the screen window.
VerticalScrollbarPosition (Page 5644)	P	P	Specifies the vertical positioning of the scroll bar in a screen window with slider, or returns its value.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.
WindowCloseEnabled (Page 5657)	-	-	Enables closing of the window in runtime.
WindowMaximizeEnabled (Page 5657)	-	-	TRUE, if the window can be maximized in runtime.
WindowMovingEnabled (Page 5658)	-	-	TRUE, if the window can be moved in runtime.
WindowOnTop (Page 5658)	-	-	TRUE, if the object is always in the foreground in runtime.
WindowSizingEnabled (Page 5659)	-	-	Enables resizing of the specified object in runtime.
WindowStartupPosition	-	-	??
ZoomFactor (Page 5679)	P	P	Sets or reads the zoom factor of a screen or screen window.

* not visible in the ES

Table 12-104 Methods

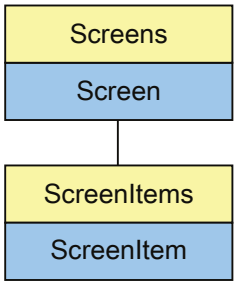
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

ScriptDiagnostics

Description



Represents the "ApplicationWindow" object. The ApplicationWindow object is an element of the ScreenItems list.

Type identifier in VBS

HMIApplicationWindow

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-105 Properties

Properties	Read	Write	Description
BorderEnabled (Page 5270)	-	-	TRUE, if the window is visualized with a border in runtime.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
Name (Page 5426)	P	P	Returns the object name as STRING.
ShowCaption (Page 5494)	-	-	Hides or shows the caption.
Template (Page 5532)	-	-	Returns the templte for displaying the window content of the "Application window" object.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.
WindowCloseEnabled (Page 5657)	-	-	Enables closing of the window in runtime.
WindowMaximizeEnabled (Page 5657)	-	-	TRUE, if the window can be maximized in runtime.
WindowMovingEnabled (Page 5658)	-	-	TRUE, if the window can be moved in runtime.
WindowOnTop (Page 5658)	-	-	TRUE, if the object is always in the foreground in runtime.
WindowsContents (Page 5659)	-	-	Returns the content of the application window
WindowSizingEnabled (Page 5659)	-	-	Enables resizing of the specified object in runtime.

Table 12-106 Methods

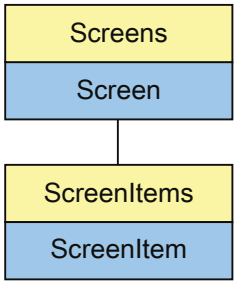
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

Slider

Description



Represents the "Slider" object. The Slider object is an element of the ScreenItems list.

Type identifier in VBS

HMISlider

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-107 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	-	-	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackPicture	-	-	Specifies a graphic for the background.
BarBackColor (Page 5259)	P, A	P, A	Specifies the color of the bar background for the selected object.
BarColor (Page 5262)	P, A	P, A	Sets the color of the slider in the "Slider" object.
BorderBrightColor3D (Page 5268)	P, A	P, A	Specifies the border color of the following border parts in the 3D display of the selected object.
BorderInnerStyle3D (Page 5275)	P	P	Specifies the representation of the inner part of the object border.
BorderInnerWidth3D (Page 5276)	P, A	P, A	Specifies the width of the inner border for 3D presentation of the selected object.
BorderOuterStyle3D (Page 5276)	P	P	Specifies the representation of the outer part of the object border.
BorderOuterWidth3D (Page 5277)	P, A	P, A	Sets the width of the outer border for 3D presentation of the selected object.
BorderShadeColor3D (Page 5277)	P, A	P, A	Specifies the border color of the following border parts in the 3D display of the selected object.
BorderWidth (Page 5280)	P, A	P, A	Specifies the line thickness of the object.
Caption (Page 5287)	P, A	P, A	Specifies the text that is displayed in the header of the selected object.
ContinousChange (Page 5305)	P	P	Specifies whether to transfer the value of the "ProcessValue" when the mouse button is released, or immediately on a change of the slider position in runtime.
Enabled (Page 5323)	P, A	P, A	Specifies whether the selected object can be operated in runtime.
Flashing	-	-	Specifies that the text block flashes.
FocusColor (Page 5346)	P, A	P, A	Specifies the color for the focus frame of the selected object when it is in focus.
FocusWidth (Page 5347)	P, A	P, A	Specifies the border width of the specified object when the object is in focus.
Font	-	-	Specifies or returns the font.
ForeColor (Page 5353)	P, A	P, A	Specifies the font color of the text for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
LabelColor (Page 5380)	P, A	P, A	Sets the color of the scale labeling in the "Slider" object.
Layer (Page 5384)	-	-	Specifies the value of the X coordinate of the object.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
MaximumValue (Page 5406)	P, A	P, A	Specifies the maximum value of the scale in the selected object.

Properties	Read	Write	Description
MinimumValue (Page 5422)	P, A	P, A	Specifies the minimum value of the scale in the selected object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
PositionFont	-	-	Sets the font type for the labeling.
ProcessValue (Page 5456)	P, A	P, A	Specifies the default for the value to be displayed.
ShowBar (Page 5493)	P, A	P, A	Specifies whether to display the process value along with a filled bar in the "Slider" object.
ShowPosition (Page 5497)	P, A	P, A	Adds a numerical display of the value of the slider position.
ShowThumb (Page 5503)	P, A	P, A	Displays the slider of the "Slider" object.
ShowTickLabels (Page 5503)	P, A	P, A	Specifies whether the label is shown in the scale.
ShowTicks (Page 5504)	P, A	P, A	Specifies whether the tick marks are displayed in the scale of the object.
ThumbBackColor (Page 5536)	P, A	P, A	Specifies the background color of the slider in the "Slider" object.
ThumbPicture	-	-	Specifies a graphic object for the slider.
TickStyle (Page 5538)	P	P	Specifies the scale display.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-108 Methods

Methods	Validity	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

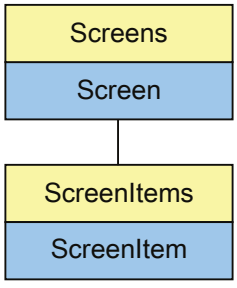
ObjectName (Page 5706)

Screen object (list) (Page 5007)

Font (Page 5348)

SmartClientView

Description



Represents the "Sm@rtClient View" object. The SmartClientView object is an element of the ScreenItems list.

Type identifier in VBS

HMISmartClientView

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-109 Properties

Properties	Read	Write	Description
AllowMenu (Page 5241)	-	-	Specifies whether the shortcut menu is enabled to control the Sm@rtClient view.
ConnectionType (Page 5304)	-	-	Specifies the type of connector. You can select one of two connection types.
ConnectOnStart	-	-	Sets up the connection at startup.
Enabled	A	A	Specifies whether the selected object can be operated in runtime.
Flashing	-	-	Specifies that the text block flashes.
Font (Page 5348)	-	-	Specifies or returns the font.
Height (Page 5358)	A	-	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
LocalCursor (Page 5402)	-	-	Specifies whether the cursor data is transferred separately in order to increase the performance.
MachineName (Page 5405)	A	A	Sets the network ID of the device that is to be monitored.
Name (Page 5426)	A	-	Returns the object name as STRING.
Password (Page 5450)	A	A	Sets the password for remote monitoring.
ScaleDenominator (Page 5473)	-	-	Sets the scaling numerator on the client.
ScaleNumerator (Page 5470)	-	-	Sets the scaling numerator on the client.
Scaling (Page 5474)	-	-	TRUE, if an additional scale is used to visualize the values.
ServerScale (Page 5491)	-	-	Specifies whether the Sm@rtClient view can be zoomed in or out.
Shared (Page 5492)	-	-	Specifies that an HMI device is shared by several Sm@rtClient Views.
ShowControls	-	-	Specifies that a toolbar is displayed.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
UseCursorKeyScroll	-	-	Sets the use of the background color of the object for printing.
ViewOnly (Page 5645)	A	A	Enables the use of the Sm@rtClient view for remote monitoring or remote maintenance.
Visible (Page 5645) *	A	A	Specifies whether the selected object is visible.
Width (Page 5655)	A	-	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-110 Methods

Methods	Valid	Description
Activate (Page 5825)	A	Activates the permanent window or the root screen.

See also

Screen object (list) (Page 4975)

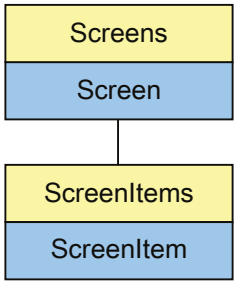
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

StatusForce

Description



Represents the "Status/Force" object. The StatusForce object is an element of the ScreenItems list.

If you change the settings for this object with a user-defined function, the changed settings are retained even after the screen is called again.

Type identifier in VBS

HMIStatusForce

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-111 Properties

Properties	Read	Write	Description
Appearance	-	-	Specifies the layout for this NC subroutine.
BackColor	A	A	Specifies the background color of the selected object.
ColumnsMoveable	-	-	Specifies that the columns can be moved.
CountVisibleItems (Page 5309) *	-	-	Specifies the number of lines contained in the selection list.
Enabled	A	A	Specifies whether the selected object can be operated in runtime.
FitToSize	-	-	Specifies that the size of the object should be automatically adapted to the content.
Flashing	-	-	Specifies that the text block flashes.
FocusColor	A	A	Specifies the color for the focus frame of the selected object when it is in focus.
FocusWidth	A	A	Specifies the border width of the specified object when the object is in focus.
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	A	-	Returns the object name as STRING.
PLCFilter	-	-	Sets the controller type.
SelectionBackColor (Page 5484)	A	A	Specifies the background color of the selected cells.
SelectionForeColor (Page 5485)	A	A	Specifies the foreground color of the selected cells.
SetOfVisibleColumns	-	-	Sets the visible columns.
ShowReadButton	-	-	Enables the display of the "Read" button.
ShowTableGridlines (Page 5502)	A	A	Enables the display of gridlines in the table of the specified object.
ShowWriteButton	-	-	Enables the display of the "Write" button.
TableBackColor (Page 5525)	A	A	Specifies the background color of the table cells for the selected object.
TableFont	-	-	Specifies the font in the table.
TableForeColor	A	A	??
TableHeaderBackColor (Page 5529)	A	A	Specifies the background color in the header of the table of the selected object.
TableHeaderFont	-	-	Specifies the text color of the header.
TableHeaderForeColor (Page 5530)	A	A	Specifies the text color in the header of the table of the selected object.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Visible (Page 5645) *	A	A	Specifies whether the selected object is visible.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-112 Methods

Methods	Valid	Description
Activate (Page 5825)	A	Activates the permanent window or the root screen.

See also

Screen object (list) (Page 4975)

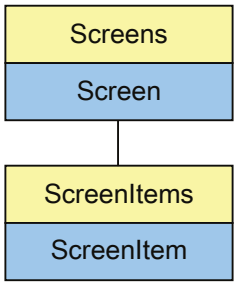
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

Switch

Description



Represents the "Switch" object. The Switch object is an element of the ScreenItems list.

The availability of the following object properties depends on the selected mode:

Property	"Switch with text"	"Switch with graphic"	"Switch"
CaptionColor	--	--	x
CaptionText	--	--	x
HorizontalAlignment	X	--	--
InnerBackColorOff	--	--	x

Property	"Switch with text"	"Switch with graphic"	"Switch"
InnerBackColorOn	--	--	x
TextOn	x	--	x
TextOff	x	--	x
VerticalAlignment	x	--	--

Type identifier in VBS

HMISwitch

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-113 Properties

Properties	Read	Write	Description
AboveUpperLimitColor (Page 5224)	-	-	Sets the color of the specified object for the "High limit violated" event.
Authorization (Page 5245)	-	-	Specifies the operating rights of the selected object in runtime.
AutoSizing	-	-	Specifies that the size of the object should be automatically adapted to the content.
BackColor (Page 5251)	A	A	Specifies the background color of the selected object.
BackFlashingColorOff (Page 5256) *	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 5257) *	-	-	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258) *	-	-	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258) *	-	-	Specifies the flash rate of the background for the selected object.
BelowLowerLimitColor (Page 5263)	-	-	Sets the color of the specified object for the "Low limit violated" event.
BorderColor (Page 5268) *	-	-	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271) *	-	-	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272) *	-	-	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273) *	-	-	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274) *	-	-	Specifies the flash rate of the border line for the selected object.
BorderStyle3D	A	A	Specifies whether the object has a 3D border shading.
BorderWidth (Page 5280) *	-	-	Specifies the line thickness of the object.

Properties	Read	Write	Description
BorderWidth3D (Page 5281) *	-	-	Specifies the width of the border for 3D display of the selected object.
CaptionColor (Page 5288)	A	A	Specifies the color of the text that is displayed in the header of the selected object.
CaptionFont	-	-	Specifies the character set for the caption.
CaptionText (Page 5288)	A	A	Specifies the text that is displayed in the header of the selected object.
CornerStyle (Page 5306) *	-	-	Specifies the type of border lines for the selected object.
EdgeStyle *	-	-	Specifies the line style of the selected object.
Enabled (Page 5323)	A	A	Specifies whether the selected object can be operated in runtime.
FitToLargest	-	-	Specifies that the size of the object depends on the text length in all languages.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340) *	-	-	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342) *	-	-	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343) *	-	-	Specifies whether the border line of the selected object flashes in runtime.
FlashingOnLimitViolation	-	-	Specifies that flashing is activated when a limit is violated.
FlashingRate (Page 5344) *	-	-	Specifies the flash rate of the border line for the selected object.
FocusColor (Page 5346)	A	A	Specifies the color for the focus frame of the selected object when it is in focus.
FocusWidth (Page 5347)	A	A	Specifies the border width of the specified object when the object is in focus.
Font (Page 5348)	-	-	Specifies or returns the font.
ForeColor (Page 5353)	A	A	Specifies the font color of the text for the selected object.
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HelpText	A	-	Returns the tooltip that is shown in runtime as user help for the specified object.
HorizontalAlignment (Page 5372)	A	A	Specifies the horizontal alignment of the text within the selected object.
InnerBackColorOff (Page 5376)	A	A	Sets the color underneath the slider of the "Switch" object for the object in OFF state.
InnerBackColorOn (Page 5377)	A	A	Sets the color underneath the slider of the "Switch" object for the object in ON state.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399) *	-	-	Specifies the shape of the line end.
Mode (Page 5424)	-	-	Specifies the field type of the selected object.
Name (Page 5426)	A	-	Returns the object name as STRING.
OnValue	-	-	Specifies the value for the "ON" state in "two-state" mode.
PictureOff (Page 5452)	-	-	Specifies the screen to be displayed in the "Off" state.

Properties	Read	Write	Description
PictureOn (Page 5453)	-	-	Specifies the screen to be displayed in the "On" state.
ProcessValue	-	-	Specifies the default for the value to be displayed.
SwitchOrientation	-	-	Specifies the line color of the object.
TextOff (Page 5534)	A	A	Specifies the text to be displayed in the "Off" state of the selected object.
TextOn	A	A	Specifies the text to be displayed in the "On" state of the selected object.
TextOrientation (Page 5536) *	-	-	Specifies the text orientation of the selected object.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
VerticalAlignment (Page 5643)	A	A	Specifies the vertical alignment of the text for the selected object.
Visible (Page 5645) *	A	A	Specifies whether the selected object is visible.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-114 Methods

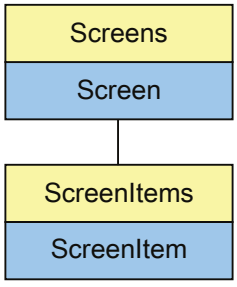
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.

See also

Screen object (list) (Page 5007)

SymbolicIOField

Description



Represents the "SymbolicIOField" object. The SymbolicIOField object is an element of the ScreenItems list.

The availability of the following object properties depends on the selected mode:

Property	"Input"	"Output"	"Input/Output"	"Two states"
BackColor	x	x	x	x
BorderColor	--	x	--	x
BorderWidth	--	--	--	x
BorderStyle3D	--	x	--	--
Enabled	x	--	x	--
HelpText	x	--	x	--
VerticalAlignment	--	x	--	x
HorizontalAlignment	--	x	--	x

Type identifier in VBS

HMISymbolicIOField

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-115 Properties

Properties	Read	Write	Description
AboveUpperLimitColor (Page 5224)	-	-	Sets the color of the specified object for the "High limit violated" event.
AcceptOnExit (Page 5225)	P	P	Specifies whether the input field is confirmed automatically when it is left.
AdaptBorder (Page 5232)	-	-	Specifies whether the border of the specified object is dynamically adapted to the text size.
AskOperationMotive (Page 5244)	P	P	Specifies whether the reason for operating this object is logged.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackFlashingColorOff (Page 5256)	P	P	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 5257)	P	P	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258)	P	P	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258)	P	P	Specifies the flash rate of the background for the selected object.
BelowLowerLimitColor (Page 5263)	-	-	Sets the color of the specified object for the "Low limit violated" event.
BitNumber (Page 5264)	P	P	Specifies the bit whose status must change to trigger a value change.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P, A	P, A	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderStyle3D	A	A	Specifies whether the object has a 3D border shading.
BorderWidth (Page 5280)	P, A	P, A	Specifies the line thickness of the object.

Properties	Read	Write	Description
BottomMargin	-	-	Sets the margin between the text and the bottom edge of the object.
CaptionBackColor (Page 5287)	P	P	Specifies the background color of the caption for the selected object.
CaptionColor (Page 5288)	P	P	Specifies the color of the text that is displayed in the header of the selected object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
CountVisibleItems (Page 5309)	P	P	Specifies the number of lines contained in the selection list.
CursorControl (Page 5310)	P	P	Specifies whether the mouse cursor jumps to the next field of the TAB sequence after leaving a field.
DrawInsideFrame (Page 5319)	-	-	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	P	P	Specifies the line style of the selected object.
EditOnFocus (Page 5322)	P	P	Specifies whether data input is immediately possible if the input field is selected using the "Tab" key.
Enabled (Page 5323)	P, A	P, A	Specifies whether the selected object can be operated in runtime.
FieldLength	-	-	Specifies that the "field length string" field is read-only.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
FitToLargest	-	-	Specifies that the size of the object depends on the text length in all languages.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingOnLimitViolation	-	-	Specifies that flashing is activated when a limit is violated.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
FocusColor (Page 5346)	-	-	Specifies the color for the focus frame of the selected object when it is in focus.
FocusWidth (Page 5347)	-	-	Specifies the border width of the specified object when the object is in focus.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	P	P	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 5350)	P	P	Specifies whether the text of the selected object is shown in italics.
FontName (Page 5351)	P	P	Specifies the font of the selected object.
FontSize (Page 5351)	P	P	Specifies the font size of the text for the selected object.
FontUnderline (Page 5352)	P	P	Specifies whether the text of the selected object is underlined.
ForeColor (Page 5353)	P, A	P, A	Specifies the font color of the text for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Properties	Read	Write	Description
HelpText	A	-	Returns the tooltip that is shown in runtime as user help for the specified object.
HorizontalAlignment (Page 5372)	P, A	P, A	Specifies the horizontal alignment of the text within the selected object.
InputValue (Page 5377)	-	-	Defines the value entered by the user in the IO field.
ItemBorderStyle (Page 5379)	P	P	Sets the separation line style in the selection list of the specified object.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LeftMargin	-	-	Sets the margin between the text and the left edge of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
LogOperation (Page 5403)	P	P	Specifies whether, after operating this object, an alarm is output to the alarm system.
Mode (Page 5424)	P	P	Specifies the field type of the selected object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
OnValue	-	-	Specifies the value for the "ON" state in "two-state" mode.
ProcessValue (Page 5456)	P	P	Specifies the default for the value to be displayed.
RightMargin	-	-	Sets the margin between the text and the right edge of the object.
SelectBackColor (Page 5478)	P	P	Specifies the background color of the selected cells.
SelectForeColor (Page 5483)	P	P	Specifies the foreground color of the selected cells.
SeparatorBackColor (Page 5488)	P	P	Specifies the background color of the broken separation lines in the selection list of the specified object.
SeparatorColor (Page 5489)	P	P	Sets the color of the separation lines in the selection list of the specified object.
SeparatorCornerStyle (Page 5490)	P	P	Sets the shape of corners for the object of the type "ScreenItem" with "SymbolicIOField" format.
SeparatorLineEndShapeStyle (Page 5491)	P	P	Sets the shape of line ends for the object of the type "ScreenItem" with "SymbolicIOField" format.
SeparatorStyle (Page 5490)	P	P	Sets the separation line style in the selection list of the specified object.
SeparatorWidth (Page 5490)	P	P	Sets the width separation lines in the selection list of the specified object.
ShowBadTagState (Page 5493)	P	P	Specifies whether an object is grayed out if its associated tag has a bad quality code or tag state.
ShowDropDownButton	-	-	Sets the display of a button for the selection list.
ShowDropDownList	-	-	Specifies selection of the entry from a selection list.
TextList (Page 5534)	-	-	Specifies the label for the text field.
TextOff (Page 5534)	-	-	Specifies the text to be displayed in the "Off" state of the selected object.
TextOn	-	-	Specifies the text to be displayed in the "On" state of the selected object.
TextOrientation (Page 5536)	P	P	Specifies the text orientation of the selected object.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.

Properties	Read	Write	Description
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
TopMargin	-	-	Sets the margin between the text and the top edge of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
VerticalAlignment (Page 5643)	P, A	P, A	Specifies the vertical alignment of the text for the selected object.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-116 Methods

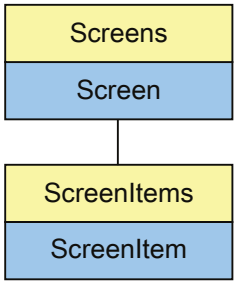
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

SymbolLibrary

Description



Represents the "SymbolLibrary" object. The SymbolLibrary object is an element of the ScreenItems list.

Type identifier in VBS

HMISymbolLibrary

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-117 Properties

Properties	Read	Write	Description
AboveUpperLimitColor (Page 5224)	-	-	Sets the color of the specified object for the "High limit violated" event.
Authorization (Page 5245)	-	-	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.

Properties	Read	Write	Description
BelowLowerLimitColor (Page 5263)	-	-	Sets the color of the specified object for the "Low limit violated" event.
BlinkColor (Page 5264)	P, A	P, A	Sets the flashing color for the "SymbolLibrary" object.
BlinkMode (Page 5265)	P	P	Sets the type of flashing graphic for the specified object.
BlinkSpeed (Page 5265)	P	P	Sets the flash rate.
ChangeMouseCursor (Page 5293)	P	P	Specifies the appearance of the cursor after its transformation in mouse over icon mode in runtime.
Enabled (Page 5323)	P, A	P, A	Specifies whether the selected object can be operated in runtime.
FillColorMode (Page 5336)	P	P	Sets the type of foreground for the specified object.
FixedAspectRatio (Page 5340)	P	P	Specifies whether the aspect ratio should be retained on changes of the icon size, or follow the change dynamically.
Flashing	-	-	Specifies that the text block flashes.
FlashingOnLimitViolation	-	-	Specifies that flashing is activated when a limit is violated.
Flip (Page 5346)	P	P	Flips the symbol on the vertical and / or horizontal center axis of the icon.
ForeColor (Page 5353)	P, A	P, A	Specifies the font color of the text for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
ProcessValue	-	-	Specifies the default for the value to be displayed.
Rotation (Page 5462)	P	P	Specifies the angle of rotation in degrees of the selected object.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-118 Methods

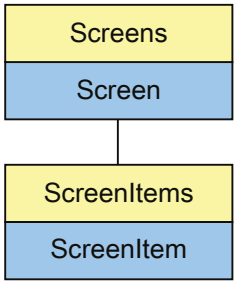
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

SystemDiagnoseView

Description



Represents the "System diagnostic view" object. The SystemDiagnoseView object is an element of the ScreenItems list.

Type identifier in VBS

HMISysDiagView

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-119 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	A	A	Specifies the operating rights of the selected object in runtime.
BackgroundColor	A	A	Specifies the background color.
ComponentInfoText	A	A	Specifies the tooltip of the system diagnostics view in runtime.

Properties	Read	Write	Description
Enabled	A	A	Specifies whether the selected object can be operated in runtime.
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
InfoArea_BackgroundColor	A	A	Specifies the background for grid control elements.
InfoArea_ColumnsMovable	A	A	Specifies the mobility of columns.
InfoArea_DefaultTextColor	A	A	Sets the text color.
InfoArea_ErrorTextBackgroundColor	A	A	Specifies the background color of a cell in error case.
InfoArea_ErrorTextColor	A	A	Sets the text color of a cell in error case.
InfoArea_FocusFrameColor	A	A	Sets the color of the focus frame.
InfoArea_FocusFrameWidth	A	A	Sets the line thickness of the focus rectangle.
InfoArea_Font	A	A	Sets the fonts of the grid views.
InfoArea_ShowGridLines	A	A	Sets the grid lines.
InfoArea_TableHeaderBackgroundCol-or	A	A	Specifies the background color of the table header.
InfoArea_TableHeaderTextColor	A	A	Sets the text color of the table header.
InputAddressText	A	A	Sets the input address.
ItemText_AKZ	A	A	Specifies the header for the "AKZ" column.
ItemText_Descriptor	A	A	Specifies the header for the "Descriptor" column.
ItemText_ErrorText	A	A	Specifies the header for the "ErrorText" column.
ItemText_InstallationDate	A	A	Specifies the header for the "InstallationDate" column.
ItemText_LADDR	A	A	Specifies the header for the "LADDR" column.
ItemText_Name	A	A	Sets the labeling of the "Name" column.
ItemText_OKZ	A	A	Specifies the header for the "OKZ" column.
ItemText_OperationState	A	A	Specifies the header for the "OperationState" column.
ItemText_OrderID	A	A	Specifies the header for the "OrderID" column.
ItemText_Rack	A	A	Specifies the header for the "Rack" column.
ItemText_Slot	A	A	Specifies the header for the "Slot" column.
ItemText_SoftwareRevision	A	A	Specifies the header for the "SoftwareRevision" column.
ItemText_State	A	A	Specifies the header for the "State" column.
ItemText_Station	A	A	Specifies the header for the "Station" column.
ItemText_SubAddress	A	A	Specifies the header for the "SubAddress" column.
ItemText_SubSlot	A	A	Specifies the header for the "SubSlot" column.
ItemText_SubSystem	A	A	Specifies the header for the "SubSystem" column.
ItemText_Type	A	A	Specifies the header for the "Type" column.
IV_ShowItem_AKZ	A	A	Enables the display of the "AKZ" column.
IV_ShowItem_Descriptor	A	A	Enables the display of the "Descriptor" column.
IV_ShowItem_ErrorText	A	A	Enables the display of the "ErrorText" column.
IV_ShowItem_InstallationDate	A	A	Enables the display of the "InstallationDate" column.
IV_ShowItem_LADDR	A	A	Enables the display of the "LADDR" column.
IV_ShowItem_Name	A	A	Enables the display of the "Name" column.
IV_ShowItem_OKZ	A	A	Enables the display of the "OKZ" column.
IV_ShowItem_OperationState	A	A	Enables the display of the "OperationState" column.

Properties	Read	Write	Description
IV_ShowItem_OrderID	A	A	Enables the display of the "OrderID" column.
IV_ShowItem_Rack	A	A	Enables the display of the "Rack" column.
IV_ShowItem_Slot	A	A	Enables the display of the "Slot" column.
IV_ShowItem_SoftwareRevision	A	A	Enables the display of the "SoftwareRevision" column.
IV_ShowItem_State	A	A	Enables the display of the "State" column.
IV_ShowItem_Station	A	A	Enables the display of the "Station" column.
IV_ShowItem_SubAddress	A	A	Enables the display of the "SubAddress" column.
IV_ShowItem_SubSlot	A	A	Enables the display of the "SubSlot" column.
IV_ShowItem_SubSystem	A	A	Enables the display of the "SubSystem" column.
IV_ShowItem_Type	A	A	Enables the display of the "Type" column.
Layer (Page 5384)	A	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
NavigationPath_Font	A	A	Sets the font type of the navigation path.
NavigationPath_TextColor	A	A	Sets the text color of the navigation path.
ObjectName (Page 5706)	A	-	Returns the object name as STRING.
OutputAddressText	A	A	Specifies the text for the output address.
ShowNavigationButtons	A	A	Enables the display of navigation buttons.
ShowPathInformation	A	A	Enables the display of the navigation path.
ShowSplittedView	A	A	Enables a split view.
TableHeaderFont	A	A	Specifies the text color of the header.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Type (Page 5809)	A	-	Returns the type of the specified object as STRING.
Parent (Page 5791)	A	-	Returns the screen in which the specified object is embedded.
UV_ShowItem_AKZ	A	A	Enables the display of the "AKZ" column.
UV_ShowItem_Descriptor	A	A	Enables the display of the "Descriptor" column.
UV_ShowItem_InstallationDate	A	A	Enables the display of the "InstallationDate" column.
UV_ShowItem_LADDR	A	A	Enables the display of the "LADDR" column.
UV_ShowItem_Name	A	A	Enables the display of the "Name" column.
UV_ShowItem_OKZ	A	A	Enables the display of the "OKZ" column.
UV_ShowItem_OperationState	A	A	Enables the display of the "OperationState" column.
UV_ShowItem_OrderID	A	A	Enables the display of the "OrderID" column.
UV_ShowItem_Rack	A	A	Enables the display of the "Rack" column.
UV_ShowItem_Slot	A	A	Enables the display of the "Slot" column.
UV_ShowItem_SoftwareRevision	A	A	Enables the display of the "SoftwareRevision" column.
UV_ShowItem_State	A	A	Enables the display of the "State" column.
UV_ShowItem_Station	A	A	Enables the display of the "Station" column.
UV_ShowItem_SubAddress	A	A	Enables the display of the "SubAddress" column.
UV_ShowItem_SubSlot	A	A	Enables the display of the "SubSlot" column.
UV_ShowItem_SubSystem	A	A	Enables the display of the "SubSystem" column.
UV_ShowItem_Type	A	A	Enables the display of the "Type" column.

Properties	Read	Write	Description
Visible (Page 5645)	A	A	Specifies whether the selected object is visible.
Width (Page 5655)	A	A	Enables the display of the "Slot" column.

Table 12-120 Methods

Methods	Valid	Description
Activate (Page 5825)	A	Activates the permanent window or the root screen.

See also

Screen object (list) (Page 4975)

Screen (Page 4976)

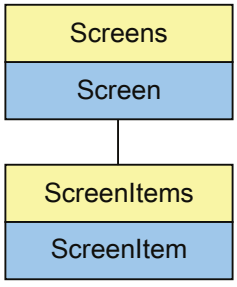
ScreenItem (Page 4977)

ScreenItems (Page 4979)

Name (Page 5426)

SystemDiagnoseWindow

Description



Represents the "System diagnostic window" object. The SystemDiagnoseWindow object is an element of the ScreenItems list.

Type identifier in VBS

HMISysDiagWindow

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-121 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	A	A	Specifies the operating rights of the selected object in runtime.
BackgroundColor	A	A	Specifies the background color.
Caption (Page 5287)	A	A	Specifies the text that is displayed in the header of the selected object.
CaptionActive	A	A	Enables display of the caption.
Closable (Page 5295)	A	A	Specifies that the control can be closed in runtime.
ComponentInfoText	A	A	Specifies the tooltip of the system diagnostics view in runtime.
Enabled	A	A	Specifies whether the selected object can be operated in runtime.
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
InfoArea_BackgroundColor	A	A	Specifies the background for grid control elements.
InfoArea_ColumnsMovable	A	A	Specifies the mobility of columns.
InfoArea_DefaultTextColor	A	A	Sets the text color.
InfoArea_ErrorTextBackgroundColor	A	A	Specifies the background color of a cell in error case.
InfoArea_ErrorTextColor	A	A	Sets the text color of a cell in error case.
InfoArea_FocusFrameColor	A	A	Sets the color of the focus frame.
InfoArea_FocusFrameWidth	A	A	Sets the line thickness of the focus rectangle.
InfoArea_Font	A	A	Sets the fonts of the grid views.
InfoArea_ShowGridLines	A	A	Sets the grid lines.
InfoArea_TableHeaderBackgroundColor	A	A	Specifies the background color of the table header.
InfoArea_TableHeaderTextColor	A	A	Sets the text color of the table header.
InputAddressText	A	A	Sets the input address.
ItemText_AKZ	A	A	Specifies the header for the "AKZ" column.
ItemText_Descriptor	A	A	Specifies the header for the "Descriptor" column.
ItemText_ErrorText	A	A	Specifies the header for the "ErrorText" column.
ItemText_InstallationDate	A	A	Specifies the header for the "InstallationDate" column.
ItemText_LADDR	A	A	Specifies the header for the "LADDR" column.
ItemText_Name	A	A	Sets the labeling of the "Name" column.

Properties	Read	Write	Description
ItemText_OKZ	A	A	Specifies the header for the "OKZ" column.
ItemText_OperationState	A	A	Specifies the header for the "OperationState" column.
ItemText_OrderID	A	A	Specifies the header for the "OrderID" column.
ItemText_Rack	A	A	Specifies the header for the "Rack" column.
ItemText_Slot	A	A	Specifies the header for the "Slot" column.
ItemText_SoftwareRevision	A	A	Specifies the header for the "SoftwareRevision" column.
ItemText_State	A	A	Specifies the header for the "State" column.
ItemText_Station	A	A	Specifies the header for the "Station" column.
ItemText_SubAddress	A	A	Specifies the header for the "SubAddress" column.
ItemText_SubSlot	A	A	Specifies the header for the "SubSlot" column.
ItemText_SubSystem	A	A	Specifies the header for the "SubSystem" column.
ItemText_Type	A	A	Specifies the header for the "Type" column.
IV_ShowItem_AKZ	A	A	Enables the display of the "AKZ" column.
IV_ShowItem_Descriptor	A	A	Enables the display of the "Descriptor" column.
IV_ShowItem_ErrorText	A	A	Enables the display of the "ErrorText" column.
IV_ShowItem_InstallationDate	A	A	Enables the display of the "InstallationDate" column.
IV_ShowItem_LADDR	A	A	Enables the display of the "LADDR" column.
IV_ShowItem_Name	A	A	Enables the display of the "Name" column.
IV_ShowItem_OKZ	A	A	Enables the display of the "OKZ" column.
IV_ShowItem_OperationState	A	A	Enables the display of the "OperationState" column.
IV_ShowItem_OrderID	A	A	Enables the display of the "OrderID" column.
IV_ShowItem_Rack	A	A	Enables the display of the "Rack" column.
IV_ShowItem_Slot	A	A	Enables the display of the "Slot" column.
IV_ShowItem_SoftwareRevision	A	A	Enables the display of the "SoftwareRevision" column.
IV_ShowItem_State	A	A	Enables the display of the "State" column.
IV_ShowItem_Station	A	A	Enables the display of the "Station" column.
IV_ShowItem_SubAddress	A	A	Enables the display of the "SubAddress" column.
IV_ShowItem_SubSlot	A	A	Enables the display of the "SubSlot" column.
IV_ShowItem_SubSystem	A	A	Enables the display of the "SubSystem" column.
IV_ShowItem_Type	A	A	Enables the display of the "Type" column.
Layer (Page 5384)	A	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Modal	A	A	Specifies that you must acknowledge the alarm view before continuing with your work.
NavigationPath_Font	A	A	Sets the font type of the navigation path.
NavigationPath_TextColor	A	A	Sets the text color of the navigation path.
ObjectName (Page 5706)	A	-	Returns the object name as STRING.
OutputAddressText	A	A	Specifies the text for the output address.
Resizable	A	A	Enables resizing of the window.
ShowNavigationButtons	A	A	Enables the display of navigation buttons.
ShowPathInformation	A	A	Enables the display of the navigation path.
ShowSplittedView	A	A	Enables a split view.

Properties	Read	Write	Description
TableHeaderFont	A	A	Specifies the text color of the header.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Type (Page 5809)	A	-	Returns the type of the specified object as STRING.
Parent (Page 5791)	A	-	Returns the screen in which the specified object is embedded.
UV_ShowItem_AKZ	A	A	Enables the display of the "AKZ" column.
UV_ShowItem_Descriptor	A	A	Enables the display of the "Descriptor" column.
UV_ShowItem_InstallationDate	A	A	Enables the display of the "InstallationDate" column.
UV_ShowItem_LADDR	A	A	Enables the display of the "LADDR" column.
UV_ShowItem_Name	A	A	Enables the display of the "Name" column.
UV_ShowItem_OKZ	A	A	Enables the display of the "OKZ" column.
UV_ShowItem_OperationState	A	A	Enables the display of the "OperationState" column.
UV_ShowItem_OrderID	A	A	Enables the display of the "OrderID" column.
UV_ShowItem_Rack	A	A	Enables the display of the "Rack" column.
UV_ShowItem_Slot	A	A	Enables the display of the "Slot" column.
UV_ShowItem_SoftwareRevision	A	A	Enables the display of the "SoftwareRevision" column.
UV_ShowItem_State	A	A	Enables the display of the "State" column.
UV_ShowItem_Station	A	A	Enables the display of the "Station" column.
UV_ShowItem_SubAddress	A	A	Enables the display of the "SubAddress" column.
UV_ShowItem_SubSlot	A	A	Enables the display of the "SubSlot" column.
UV_ShowItem_SubSystem	A	A	Enables the display of the "SubSystem" column.
UV_ShowItem_Type	A	A	Enables the display of the "Type" column.
Visible (Page 5645)	A	A	Specifies whether the selected object is visible.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.
WindowMode	A	A	Sets the window mode.

Table 12-122 Methods

Methods	Valid	Description
Activate (Page 5825)	A	Activates the permanent window or the root screen.

See also

Screen object (list) (Page 4975)

Screen (Page 4976)

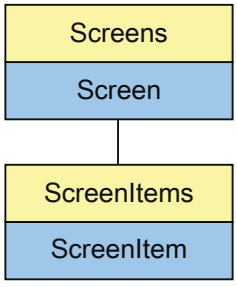
ScreenItem (Page 4977)

ScreenItems (Page 4979)

Name (Page 5426)

TextField

Description



Represents the "TextField" object. The TextField object is an element of the ScreenItems list.

Type identifier in VBS

HMITextField

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-123 Properties

Properties	Read	Write	Description
AdaptBorder (Page 5232)	P	P	Specifies whether the border of the specified object is dynamically adapted to the text size.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P, A	P, A	Specifies the background color of the selected object.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackFlashingColorOff (Page 5256)	P	P	Specifies the background color for flash state "Off".

Properties	Read	Write	Description
BackFlashingColorOn (Page 5257)	P	P	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258)	P	P	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258)	P	P	Specifies the flash rate of the background for the selected object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P, A	P, A	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderStyle3D	A	A	Specifies whether the object has a 3D border shading.
BorderWidth (Page 5280)	P, A	P, A	Specifies the line thickness of the object.
BottomMargin	-	-	Sets the margin between the text and the bottom edge of the object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	A	A	Specifies the line style of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
FitToLargest	-	-	Specifies that the size of the object depends on the text length in all languages.
Flashing	-	-	Specifies that the text block flashes.
FlashingColorOff (Page 5340)	P	P	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 5342)	P	P	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 5343)	P	P	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 5344)	P	P	Specifies the flash rate of the border line for the selected object.
Font (Page 5348)	-	-	Specifies or returns the font.
FontBold (Page 5349)	P	P	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 5350)	P	P	Specifies whether the text of the selected object is shown in italics.
FontName (Page 5351)	P	P	Specifies the font of the selected object.
FontSize (Page 5351)	P	P	Specifies the font size of the text for the selected object.
FontUnderline (Page 5352)	P	P	Specifies whether the text of the selected object is underlined.

Properties	Read	Write	Description
ForeColor (Page 5353)	P, A	P, A	Specifies the font color of the text for the selected object.
Height (Page 5358)	P, A	P, A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HorizontalAlignment (Page 5372)	P, A	P, A	Specifies the horizontal alignment of the text within the selected object.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P, A	P, A	Specifies the value of the X coordinate of the object.
LeftMargin	-	-	Sets the margin between the text and the left edge of the object.
LineEndShapeStyle (Page 5399)	P	P	Specifies the shape of the line end.
LineWrap *	-	-	Specifies the line break within the object.
Name (Page 5426)	P, A	-	Returns the object name as STRING.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
RightMargin	-	-	Sets the margin between the text and the right edge of the object.
RotationAngle (Page 5463)	P	P	Specifies the angle of rotation in degrees.
RotationCenterLeft (Page 5463)	P	P	Specifies the X coordinate of the pivot point for the object in runtime.
RotationCenterTop (Page 5464)	P	P	Specifies the Y coordinate of the pivot point for the object in runtime.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
Text (Page 5533)	P, A	P, A	Specifies the label for the text field.
TextOrientation (Page 5536)	P	P	Specifies the text orientation of the selected object.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P, A	P, A	Specifies the value of the Y coordinate of the object.
TopMargin	-	-	Sets the margin between the text and the top edge of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
VerticalAlignment (Page 5643)	P, A	P, A	Specifies the vertical alignment of the text for the selected object.
Visible (Page 5645)	P, A	P, A	Specifies whether the selected object is visible.
Width (Page 5655)	P, A	P, A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-124 Methods

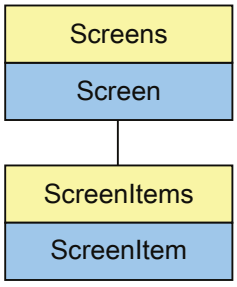
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

TrendRulerControl

Description



Represents the "Value table" object. The TrendRulerControl object is an element of the ScreenItems list.

Type identifier in VBS

HMITrendRulerControl

Example

In the following example, the object with the name "Control1" is moved 16 pixels to the right:

```
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 16
```

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-125 Properties

Properties	Read	Write	Description
ApplyProjectSettingsForDesignMode	-	-	Specifies that the project settings are used for the design.
AutoCompleteColumns (Page 5247)	P	P	Specifies whether empty columns are shown if the control is wider than the configured columns.
AutoCompleteRows (Page 5247)	P	P	Specifies whether empty rows are shown if the control is longer than the number of configured rows.
AutoPosition	P	P	Sets the automatic repositioning mode.
AutoSelectionColors (Page 5248)	P	P	Specifies whether the colors defined by the system are used as the selection colors for cells and rows.
AutoSelectionRectColor (Page 5249)	P	P	Specifies whether the selection frame is shown with the color defined by the system.
AutoShow	P	P	Specifies the automatic display of the value table.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BlockAlign *	P	P	Sets the alignment.
BlockAutoPrecisions *	P	P	Specifies that the number of decimal places displayed for an axis is adjusted automatically.
BlockCaption *	P	P	Sets the caption.
BlockCount *	P	P	Sets the number of blocks.
BlockDateFormat *	P	P	Specifies the date format.
BlockExponentialFormat *	P	P	Enables the use of the exponential notation.
BlockHideText *	P	P	Specifies whether the text is displayed.
BlockHideTitleText *	P	P	Specifies whether the text is displayed.
BlockId *	P	P	Sets the ID.
BlockIndex	P	P	The index specifies the block to which other properties such as names are referenced.
BlockLength *	P	P	Specifies the number of displayed characters.
BlockName *	P	P	Sets the name of a block.
BlockPrecisions *	P	P	Specifies the number of displayed decimal places.

Properties	Read	Write	Description
Blocks	-	-	Sets the alarm blocks.
BlockShowDate *	P	P	Specifies that the date is to be displayed.
BlockShowIcon *	P	P	Specifies the tooltip text.
BlockShowTitleIcon *	P	P	Enables the display of title as icon.
BlockTimeFormat *	P	P	Specifies the time format.
BlockUseSourceFormat *	P	P	Enables the use of the source format.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
Caption (Page 5287)	P	P	Specifies the text that is displayed in the header of the selected object.
CellCut (Page 5289)	P	P	Specifies whether the contents of the cells are abbreviated if the cells are too narrow.
CellSpaceBottom (Page 5290)	P	P	Specifies the bottom margin of the table cells.
CellSpaceLeft (Page 5290)	P	P	Sets the left indent for the table cells.
CellSpaceRight (Page 5291)	P	P	Specifies the right indent of the table cells.
CellSpaceTop (Page 5291)	P	P	Specifies the top margin of the table cells.
Closeable (Page 5295)	P	P	Specifies that the control can be closed in runtime.
ColumnAdd *	P	P	Adds a column.
ColumnCount *	P	P	Displays the number of columns.
ColumnIndex	P	P	The index specifies on which column other properties, e.g. initial value are based.
ColumnName *	P	P	Specifies the name of a column.
ColumnRemove *	P	P	Removes a column.
ColumnRepos *	P	P	Repositions the column.
ColumnResize (Page 5299)	P	P	Enables changes to the width of columns.
ColumnScrollbar (Page 5300)	P	P	Specifies the scroll bar type.
ColumnSort *	P	P	Sets the type of sorting.
ColumnSortIndex *	P	P	Specifies the sorting order.
ColumnTitleAlignment	P	P	Specifies the type of column title alignment.
ColumnTitles (Page 5303)	P	P	Specifies whether the column header is displayed.
ColumnVisible *	P	P	Enables the visibility of a column.
ControlDesignMode (Page 5507)	P	P	Specifies the design.
ExportDirectoryChangeable (Page 5328)	P	P	Specifies whether the data export directory can be changed in runtime.
ExportDirectoryname (Page 5329)	P	P	Specifies the target directory for the exported Runtime data.
ExportFileExtension (Page 5330)	P	P	Specifies the file extension of the export file. Only the "csv" file extension is currently supported.
ExportFilename (Page 5330)	P	P	Specifies the name of the target file for the exported Runtime data.
ExportFilenameChangeable (Page 5331)	P	P	Specifies whether the export file name can be changed in runtime.
ExportFormatGuid (Page 5331)	P	P	Default assignment of the ID number and export provider.
ExportFormatName (Page 5332)	P	P	Specifies the export file format. Only the "csv" file format is currently available for the export.

Properties	Read	Write	Description
ExportParameters (Page 5332)	P	P	Specifies the parameters of the selected format by means of the "Properties" dialog.
ExportSelection (Page 5333)	P	P	Specifies the file extension of the export file. Only the "csv" file extension is currently supported.
ExportShowDialog (Page 5334)	P	P	Specifies whether the data export dialog is shown in runtime.
Font (Page 5348)	-	-	Specifies or returns the font.
GridLineColor (Page 5356)	P	P	Specifies the color for the grid lines.
GridLineWidth (Page 5357)	P	P	Specifies the width of the separation lines in pixels.
Height (Page 5358)	P	-	Sets the height of the specified "Ellipse", "Circle" and "Rectangle" objects.
HorizontalGridLines (Page 5373)	P	P	Specifies whether horizontal separation lines are displayed.
IconSpace (Page 5375)	P	P	Specifies the spacing between icons and text in the table cells.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineColor (Page 5398)	P	P	Sets the color of the window separation lines.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
Moveable (Page 5425)	P	P	Specifies whether the window can be moved in runtime.
Name (Page 5426)	P	-	Returns the object name as STRING.
PrintJob	P	P	Specifies a print job
RowScrollbar (Page 5465)	P	P	Specifies whether row scroll bars are displayed.
RowTitleAlignment	P	P	Specifies the type of row title alignment.
RowTitles	P	P	Specifies that row headers will be displayed.
RTPersistence (Page 5466)	P	P	Specifies how online configurations of WinCC are retained.
RTPersistencePasswordLevel (Page 5716)	P	P	Specifies the authorization required for online configuration in runtime.
RTPersistenceType (Page 5467)	-	-	Specifies how online configurations of WinCC are retained.
RulerType	P	P	Sets the ruler type.
SelectedCellColor (Page 5478)	P	P	Specifies the background color of the selected cell.
SelectedCellForeColor (Page 5479)	P	P	Specifies the font color of the selected cell.
SelectedRowColor (Page 5480)	P	P	Specifies the background color of the selected row.
SelectedRowForeColor (Page 5481)	P	P	Specifies the font color of the selected row.
SelectedTitleColor (Page 5482)	P	P	Specifies the background color of a selected table header.
SelectedTitleForeColor (Page 5482)	P	P	Specifies the font color of the selected table header.
SelectionColoring (Page 5485)	P	P	Specifies whether selection colors are used for cells or rows.
SelectionRect (Page 5486)	P	P	Specifies whether a selection frame is used for the selected cells or rows.
SelectionRectColor (Page 5486)	P	P	Specifies the color of the selection rectangle in the alarm window if SelectionType equals "1".

Properties	Read	Write	Description
SelectionRectWidth (Page 5487)	P	P	Specifies the line width of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionType (Page 5488)	P	P	Specifies the number of lines you can mark.
ShareSpaceWithSourceControl	P	P	Enables the display of the value table in the visualization range of the data source.
ShowSortButton (Page 5500)	P	P	Specifies whether the sorting button is shown above the vertical scroll bar.
ShowSortIcon (Page 5501)	P	P	Specifies whether the sorting icon is displayed.
ShowSortIndex (Page 5501)	P	P	
ShowTitle (Page 5504)	P	P	Specifies the layout of the control window's header.
Sizeable (Page 5506)	P	P	Specifies that the size of an object can be changed in runtime.
SortSequence (Page 5508)	P	P	Specifies how the sorting order can be changed by mouse click.
SourceControl	P	P	Sets the data source.
SourceControlType	P	P	Sets the data source type.
StatusbarBackColor (Page 5511)	P	P	Specifies the background color of the status bar.
StatusbarElementAdd (Page 5512) *	P	P	Creates a new, user-defined status bar element.
StatusbarElementAutoSize (Page 5512) *	P	P	Specifies whether the width of the selected status bar element is set automatically.
StatusbarElementCount (Page 5513) *	P	P	Specifies the number of configurable status bar elements.
StatusbarElementIconId (Page 5514) *	P	P	Default assignment of the ID number and icon of a status bar element.
StatusbarElementId (Page 5514) *	P	P	Unique ID of the selected status bar element.
StatusbarElementIndex (Page 5515)	P	P	References a status bar element. You can use this property to assign the values of other properties to a specific element of the status bar.
StatusbarElementName (Page 5516) *	P	P	Displays the object name of the selected status bar element.
StatusbarElementRemove (Page 5516) *	P	P	Removes the selected status bar element.
StatusbarElementRename (Page 5517) *	P	P	Renames the user-defined status bar element that is referenced by means of the "StatusbarElementIndex" property.
StatusbarElementRepos *	P	P	Specifies the index that is assigned to the element.
StatusbarElements (Page 5517)	-	-	Specifies the elements that are to be shown in the status bar.
StatusbarElementText	P	P	Specifies the text of the currently selected status bar element.
StatusbarElementTooltipText (Page 5518) *	P	P	Specifies the tooltip text for the user-defined status bar element.
StatusbarElementUserDefined (Page 5518) *	P	P	Indicates whether the configuration engineer has added the status bar element as a new user-defined element.
StatusbarElementVisible (Page 5519) *	P	P	In the list, activate the status bar elements that you wish to display in runtime.
StatusbarElementWidth (Page 5519) *	P	P	Specifies the width of the selected status bar element in pixels.

Properties	Read	Write	Description
StatusbarFont	-	-	Status bar font
StatusbarFontColor (Page 5520)	P	P	Specifies the color of the text in the status bar.
StatusbarShowTooltips (Page 5521)	P	P	Specifies whether tooltips are displayed for the elements of the status bar in runtime.
StatusbarText (Page 5521)	P	P	Default text in the status bar.
StatusbarUseBackColor (Page 5522)	P	P	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 5523)	P	P	Specifies whether the status bar of the control is displayed.
TableColor (Page 5526)	P	P	Specifies the background color of the rows. Use this to open the color selection dialog.
TableColor2 (Page 5526)	P	P	Specifies the background color of "Row color 2".
TableForeColor (Page 5527)	P	P	
TableForeColor2 (Page 5528)	P	P	Specifies the font color of "Row color 2".
TimeBase (Page 5540)	P	P	Sets the timebase.
TitleColor	P	P	Specifies the color of the header.
TitleCut (Page 5557)	P	P	Specifies whether the contents of the fields of a header should be abbreviated if the column is too narrow.
TitleDarkShadowColor (Page 5558)	P	P	Specifies the color of the dark side of shading.
TitleForeColor (Page 5559)	P	P	Specifies the font color of the table header for the selected status.
TitleGridLineColor (Page 5559)	P	P	Specifies the color of separation lines in the table header.
TitleLightShadowColor (Page 5560)	P	P	Specifies the color of the bright side of shading.
TitleSort (Page 5561)	P	P	Specifies how sorting by column header is triggered.
TitleStyle (Page 5561)	P	P	Specifies whether a shading color is used for the table header.
ToolbarAlignment (Page 5568)	P	P	Specifies or returns the position of the toolbar.
ToolbarBackColor (Page 5568)	P	P	Specifies the background color of the toolbar.
ToolbarButtonActive (Page 5569) *	P	P	Specifies whether the function linked to the button is activated in runtime.
ToolbarButtonAdd (Page 5570) *	P	P	Creates a new, user-defined button function.
ToolbarButtonBeginGroup (Page 5570) *	P	P	Specifies whether a separator is inserted before the selected button function.
ToolbarButtonClick	P	P	Clicks a toolbar button.
ToolbarButtonCount (Page 5571) *	P	P	Specifies the number of configurable button functions.
ToolbarButtonEnabled (Page 5572) *	P	P	Specifies whether a user-defined toolbar button can be operated.
ToolbarButtonHotKey (Page 5572) *	P	P	Shows the hotkey for the selected button function.
ToolbarButtonId (Page 5573) *	P	P	Unique ID number for the selected button function.
ToolbarButtonIndex (Page 5573)	P	P	References a button function.
ToolbarButtonLocked (Page 5574) *	P	P	Specifies whether the clicked, pressed in state is displayed for a user-defined toolbar button.
ToolbarButtonName (Page 5575) *	P	P	Shows the name of the selected button function.
ToolbarButtonPasswordLevel (Page 5575) *	P	P	Shows the authorization for the selected button function.
ToolbarButtonRemove (Page 5576) *	P	P	Removes the selected button function from the list.

Properties	Read	Write	Description
ToolbarButtonRename (Page 5576) *	P	P	Renames the user-defined toolbar element that is referenced by means of the "ToolbarButtonIndex" property.
ToolbarButtonRepos (Page 5577) *	P	P	Changes the sequence of button functions.
ToolbarButtons	-	-	Specifies which buttons are available for the toolbar.
ToolbarButtonTooltipText (Page 5577) *	P	P	Specifies the tooltip text for the button.
ToolbarButtonUserDefined (Page 5578) *	P	P	Indicates whether the configuration engineer has added the toolbar button as a new user-defined button.
ToolbarButtonVisible *	P	P	Specifies that the button is visible.
ToolbarShowTooltips (Page 5579)	P	P	Specifies whether the tooltips for the button functions are displayed in runtime.
ToolbarUseBackColor (Page 5579)	P	P	Specifies whether the background color for the toolbar is shown.
ToolbarUseHotKeys (Page 5580)	P	P	Specifies whether the hotkeys for the button functions are activated in runtime.
ToolbarVisible (Page 5580)	P	P	Specifies whether the toolbar of the control is displayed.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
UseSelectedTitleColor (Page 5630)	P	P	Specifies whether a selection color is used for the headers of selected table cells.
UseSourceBackColors	P	P	Specifies the use of the background color of the data source.
UseSourceForeColors	P	P	Specifies the use of the foreground color of the data source.
UseTableColor2 (Page 5631)	P	P	Specifies whether a second row color is used for the representation of the table.
VerticalGridLines (Page 5644)	P	P	Specifies whether vertical separation lines are displayed.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	-	Sets the width of the specified "Ellipse", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-126 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.
Export (Page 5838)	P, A	Executes the "Export log" or "Export data" button function of the control.
GetRow (Page 5850)	P	Returns the row object defined by its row number in the table-based controls as "ICCAxDataRow" type.
GetRowCollection (Page 5851)	P	Returns the list of all row objects of the table-based controls as "ICCAxDataRowCollection" type.
GetRulerBlock (Page 5853)	P	Returns the block object of the evaluation table designated by name or index as "ICCAxRulerBlock" type.

Methods	Valid	Description
GetRulerBlockCollection (Page 5854)	P	Returns the list of all block objects of the evaluation table as "ICCAxCollection" type.
GetRulerColumn (Page 5855)	P	Returns the column object of the evaluation table designated by name or index as "ICCAxRulerColumn" type.
GetRulerColumnCollection (Page 5856)	P	Returns the list of all column objects of the evaluation table as "ICCAxCollection" type.
GetSelectedRow (Page 5858)	P	Returns the selected row object of a table-based control as "ICCAxDataRow" type.
GetSelectedRows (Page 5859)	P	Returns the selected row objects of a table-based control as type "ICCAxDataRow" for multiple selection.
GetStatisticAreaColumn (Page 5861)	P	Returns the column object of the statistic area window of the evaluation table designated by name or index as "ICCAxRulerColumn" type.
GetStatisticAreaColumnCollection (Page 5862)	P	Returns the list of all column objects of the statistic area window of the evaluation table as "ICCAxCollection" type.
GetStatisticResultColumn (Page 5863)	P	Returns the column object of the statistic window of the evaluation table designated by name or index as "ICCAxRulerColumn" type.
GetStatisticResultColumnCollection (Page 5864)	P	Returns the list of all column objects of the statistic window of the evaluation table as "ICCAxCollection" type.
GetStatusbarElement (Page 5865)	P, A	Returns the element of the control's status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 5866)	P, A	Returns the list of all status bar elements of the control as "ICCAxCollection" type.
GetToolbarButton (Page 5873)	P, A	Returns the button function designated by name or index on the control toolbar as "ICCAxToolbarButton" type.
GetToolbarButtonCollection (Page 5874)	P, A	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.
Print (Page 5903)	P, A	Executes the "Print" button function of the control.
SelectAll (Page 5918)	P	Selects all rows in a table-based control.
SelectRow (Page 5919)	P	Selects a specific row in a table-based control.
ShowHelp (Page 5923)	P, A	Executes the "Help" button function of the control.
ShowPropertyDialog (Page 5927)	P, A	Executes the "Configuration dialog" button function of the control.
UnselectAll (Page 5936)	P	Removes all selections from the cells of a table-based control.
UnselectRow (Page 5936)	P	Removes the selections from a specific cell of a table-based control.

See also

Screen (Page 5000)

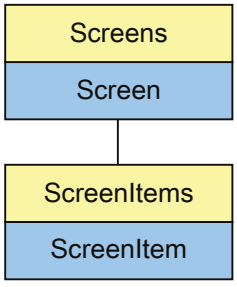
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

TrendView

Description



Represents the "TrendView" object. The TrendView object is an element of the ScreenItems list.

Type identifier in VBS

HMITrendView

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-127 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	-	-	Specifies the operating rights of the selected object in runtime.
AxisXBunchCount	-	-	Sets the step width of the large ticks for the X axis.
AxisXMarkCount	-	-	Sets the step width of the ticks for the X axis.
AxisXNoOfDigits	-	-	
AxisXShowBunchValues	-	-	Enables scale labeling of the interim values for the X axis.

Properties	Read	Write	Description
AxisXStyle *	-	-	
AxisY1BunchCount	-	-	Sets the step width of the large ticks for the Y axis.
AxisY1MarkCount	-	-	Sets the step width of the large ticks for the left Y axis.
AxisY1ShowBunchValues	-	-	Enables scale labeling of the interim values for the left Y axis.
AxisY2BunchCount	-	-	Sets the step width of the large ticks for the right Y axis.
AxisY2MarkCount	-	-	Sets the step width of the ticks for the right Y axis.
AxisY2ShowBunchValues	-	-	Enables scale labeling of the interim values for the right Y axis.
BackColor	A	A	Specifies the background color of the selected object.
ColumnsMoveable	-	-	Specifies that the columns can be moved.
CountVisibleItems (Page 5309)	-	-	Specifies the number of lines contained in the selection list.
Enabled	A	A	Specifies whether the selected object can be operated in runtime.
EnableNavigateButtons	-	-	
EnableNavigateKeys	-	-	Specifies keyboard operation.
FitToSize	-	-	Specifies that the size of the object should be automatically adapted to the content.
Flashing	-	-	Specifies that the text block flashes.
FocusColor	A	A	Specifies the color for the focus frame of the selected object when it is in focus.
FocusWidth	A	A	Specifies the border width of the specified object when the object is in focus.
Font (Page 5348)	-	-	Specifies or returns the font.
Height (Page 5358)	A	-	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HelpText (Page 5362)	A	-	Returns the tooltip that is shown in runtime as user help for the specified object.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Look3D	-	-	Activates 3D visualization.
Name (Page 5426)	A	-	Returns the object name as STRING.
RulerColor (Page 5468)	A	A	Specifies the color of the scale gradation (auxiliary line) of the axis labelling in the "OnlineTrendControl" object.
ScaleColor (Page 5469)	A	A	Specifies the color of the scale of the selected object.
SelectionBackColor *	-	-	Specifies the background color of the selected cells.
SelectionForeColor *	-	-	Specifies the foreground color of the selected cells.
ShowRuler	A	A	Specifies whether a scale gradation (auxiliary line) is shown for the axis label of the "OnlineTrendControl" object.
ShowTableGridLines	-	--	Enables the display of grid lines in the table.
ShowTimeAxis	-	-	Enables the display of the X axis.
ShowTimeAxisLabeling	-	-	Enables the display of the X axis labeling.
ShowValueAxis1	-	-	Enables the display of the left Y axis.
ShowValueAxis1Label	-	-	Enables the display of the labeling for left Y axis.
ShowValueAxis2	-	-	Enables the display of the right Y axis.

Properties	Read	Write	Description
ShowValueAxis2Label	-	-	Sets the window title.
ShowValueTable	-	-	Enables the display of the table of values.
ShowY1HlpLine	-	-	Enables the display of the auxiliary line.
ShowY2HlpLine	-	-	Enables the display of the auxiliary line.
TableBackColor (Page 5525)	A	A	Specifies the background color of the table cells for the selected object.
TableFont	-	-	Specifies the font in the table.
TableGridLineColor (Page 5528)	A	A	Sets the color of the gridlines in the table of the specified object.
TableHeaderBackColor (Page 5529)	A	A	Specifies the background color in the header of the table of the selected object.
TableHeaderFont	-	-	Specifies the text color of the header.
TableHeaderForeColor (Page 5530)	A	A	Specifies the text color in the header of the table of the selected object.
TagForExternalTime	-	-	Sets the tag for the external time.
TimeAxisBegin	-	-	Sets the start value for the time axis.
TimeAxisBeginTime	-	-	Specifies the starting time for the display of the selected trend.
TimeAxisCountPoints	-	-	Sets the number of values for the time axis.
TimeAxisEnd	-	-	Sets the end value for the time axis.
TimeAxisMode	-	-	Sets the time axis mode.
TimeAxisRange (Page 5540)	-	-	Period displayed by the time axis.
TimeAxisSide	-	-	Sets the page for new values on the time axis.
ToolbarButtons	-	-	Specifies which buttons are available for the toolbar.
ToolbarEnabled *	-	-	Enables the use of a toolbar in the trend view.
ToolbarStyle	-	-	Sets the appearance of the toolbar.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
TrendsForPrinting *	-	-	
ValueAxis1AutoRange *	-	-	Sets the automatic adaptation of the data area.
ValueAxis1Begin	-	-	Sets the start value for the axis.
ValueAxis1End	-	-	Sets the end value for the axis.
ValueAxis1LabelLength	-	-	Sets the length of axis labeling.
ValueAxis1Style *	-	-	
ValueAxis2AutoRange *	-	-	Sets the automatic adaptation of the data area.
ValueAxis2Begin	-	-	Sets the start value for the axis.
ValueAxis2End	-	-	Sets the end value for the axis.
ValueAxis2LabelLength	-	-	Sets the length of axis labeling.
ValueAxis2Style *	-	-	Sets the automatic adaptation of the data area.
ValueY1HlpLine	-	-	Sets the value for the auxiliary line.
ValueY2HlpLine	-	-	Value for the auxiliary line on the right Y axis.
Visible (Page 5645)	A	A	Specifies whether the selected object is visible.
Width (Page 5655)	A	-	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-128 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.

See also

Screen object (list) (Page 4975)

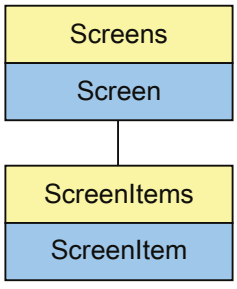
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

TubeArcObject

Description



Represents the "TubeArc" object. The TubeArcObject is an element of the ScreenItems list.

Type identifier in VBS

HMITubeArcObject

Application

In the following example, the object with the name "TubeArcObject1" is moved 10 pixels to the right:

```
'VBS24
Dim objTubeArcObject
Set objTubeArcObject = ScreenItems("TubeArcObject1")
objTubeArcObject.Left = objTubeArcObject.Left + 10
```

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-129 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
Color (Page 5296)	P	P	Specifies the line color of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
EndAngle (Page 5325)	P	P	Specifies the angle at which the end point of the selected object deviates from the zero position (0°).
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
Name (Page 5426)	P	-	Returns the object name as STRING.
RadiusHeight	P	P	Specifies the minor axis of the "Ellipsis" object.
RadiusWidth	P	P	Specifies the major axis of the "Ellipsis" object.
StartAngle (Page 5509)	P	P	Specifies the SQL statement.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-130 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen (Page 5000)

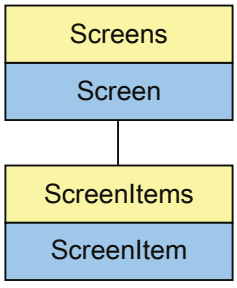
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

TubeDoubleTeeObject

Description



Represents the "DoubleTee" object. The TubeDoubleTeeObject is an element of the ScreenItems list.

Type identifier in VBS

HMITubeDoubleTeeObject

Application

In the following example, the object with the name "TubeDoubleTeeObject1" is moved 10 pixels to the right:

```
'VBS21
Dim objTubeDoubleTeeObject
Set objTubeDoubleTeeObject = ScreenItems("TubeDoubleTeeObject1")
objTubeDoubleTeeObject.Left = objTubeDoubleTeeObject.Left + 10
```

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-131 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
Color (Page 5296)	P	P	Specifies the line color of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
Name (Page 5426)	P	-	Returns the object name as STRING.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-132 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen (Page 5000)

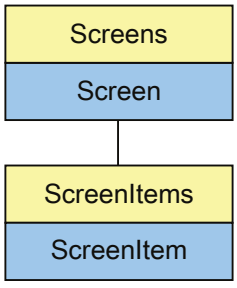
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

TubePolyline

Description



Represents the "TubePolyline" object. The TubePolyline object is an element of the ScreenItems list.

Type identifier in VBS

HMITubePolyline

Application

In the following example, the object with the name "TubePolyline1" is moved 10 pixels to the right:

```
'VBS24
Dim objTubePolyline
Set objTubePolyline = ScreenItems("TubePolyline1")
objTubePolyline.Left = objTubePolyline.Left + 10
```

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-133 Properties

Properties	Read	Write	Description
ActualPointIndex (Page 5230)	P	P	Specifies the number of the current corner point of the selected object.
ActualPointLeft (Page 5230)	P	P	Specifies the X coordinate of the current corner point with reference to the screen origin.
ActualPointTop (Page 5231)	P	P	Specifies the Y coordinate of the current corner point with reference to the screen origin.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
Color (Page 5296)	P	P	Specifies the line color of the selected object.
CornerStyle (Page 5306)	P	P	Specifies the type of border lines for the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
Name (Page 5426)	P	-	Returns the object name as STRING.
Points	-	-	Specifies the corner points.
PointsCount (Page 5454)	P	P	Specifies the number of corner points of the polyline or of the polygon.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.

Properties	Read	Write	Description
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-134 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen (Page 5000)

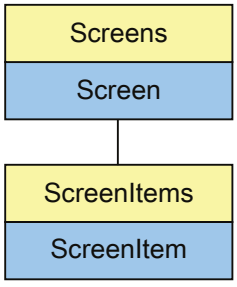
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

TubeTeeObject

Description



Represents the "Tee" object. The TubeTeeObject is an element of the ScreenItems list.

Object Type of ScreenItem Object. Represents the "T-piece" graphic object.

Type identifier in VBS

HMITubeTeeObject

Application

In the following example, the object with the name "TubeTeeObject1" is moved 10 pixels to the right:

```
'VBS21
Dim objTubeTeeObject
Set objTubeTeeObject = ScreenItems("TubeTeeObject1")
objTubeTeeObject.Left = objTubeTeeObject.Left + 10
```

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-135 Properties

Properties	Read	Write	Description
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
Color (Page 5296)	P	P	Specifies the line color of the selected object.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
Name (Page 5426)	P	-	Returns the object name as STRING.
RotationAngle (Page 5463)	P	P	Specifies the angle of rotation in degrees.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-136 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen (Page 5000)

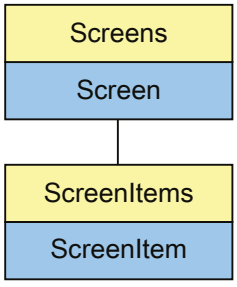
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

UserArchiveControl

Description



Displays the "Recipe view" object. The UserArchiveControl object is an element of the ScreenItems list.

Type identifier in VBS

HMIUserArchiveControl

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-137 Properties

Properties	Read	Write	Description
ApplyProjectSettingsForDesignMode	-	-	Specifies that the project settings are used for the design.
ArchiveName *	P	P	Sets the log name.
ArchiveType	P	-	Sets the log type.
AutoCompleteColumns (Page 5247)	P	P	Specifies whether empty columns are shown if the control is wider than the configured columns.
AutoCompleteRows (Page 5247)	P	P	Specifies whether empty rows are shown if the control is longer than the number of configured rows.

Properties	Read	Write	Description
AutoSelectionColors (Page 5248)	P	P	Specifies whether the colors defined by the system are used as the selection colors for cells and rows.
AutoSelectionRectColor (Page 5249)	P	P	Specifies whether the selection frame is shown with the color defined by the system.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
Blocks	-	-	Sets the alarm blocks.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
Caption (Page 5287)	P	P	Specifies the text that is displayed in the header of the selected object.
CellCut (Page 5289)	P	P	Specifies whether the contents of the cells are abbreviated if the cells are too narrow.
CellSpaceBottom (Page 5290)	P	P	Specifies the bottom margin of the table cells.
CellSpaceLeft (Page 5290)	P	P	Sets the left indent for the table cells.
CellSpaceRight (Page 5291)	P	P	Specifies the right indent of the table cells.
CellSpaceTop (Page 5291)	P	P	Specifies the top margin of the table cells.
Closeable (Page 5295)	P	P	Specifies that the control can be closed in runtime.
ColumnAlign (Page 5298) *	P	P	Specifies how the selected column is aligned.
ColumnAutoPrecisions *	P	P	Specifies that the number of decimal places displayed for an axis is adjusted automatically.
ColumnCaption *	P	P	Sets the caption.
ColumnCount *	P	-	Displays the number of columns.
ColumnDateFormat (Page 5298) *	-	P	Sets the display format for date information.
ColumnExponentialFormat *	P	P	Specifies that exponential notation is used in a column.
ColumnHideText *	P	P	Specifies whether the text is displayed.
ColumnHideTitleText *	P	P	Specifies whether the text is displayed.
ColumnIndex	P	P	The index specifies on which column other properties, e.g. initial value are based.
ColumnLeadingZeros *	P	P	Specifies that leading zeros should be displayed.
ColumnLength *	P	P	Specifies the number of displayed characters.
ColumnName *	P	-	Displays the name of the selected column.
ColumnPrecisions *	P	P	Sets the number of decimal places.
ColumnReadonly *	P	P	Sets the read-only mode for the values in the column.
ColumnRepos *	P	P	Repositions the column.
ColumnResize (Page 5299)	P	P	Enables changes to the width of columns.
ColumnScrollbar (Page 5300)	P	P	Specifies the scroll bar type.
ColumnShowDate *	P	P	Specifies that the date is to be displayed.
ColumnShowIcon *	P	P	Enables the display of an icon.
ColumnShowTitleIcon *	P	P	Enables the display of an icon.
ColumnSort *	P	P	Sets the type of sorting.
ColumnSortIndex *	P	P	Specifies the sorting order.
ColumnTimeFormat *	-	P	Sets the display format for time information.
ColumnTitleAlignment	P	P	Specifies the type of column title alignment.

Properties	Read	Write	Description
ColumnTitles (Page 5303)	P	P	Specifies whether the column header is displayed.
ColumnVisible *	P	P	Enables the visibility of a column.
ControlDesignMode (Page 5507)	-	P	Specifies the design.
DataSource	-	-	Sets the data source.
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
EnableDelete	P	P	Enables the deletion of data.
EnableEdit (Page 5324)	P	P	Specifies whether the data shown can be edited in runtime.
EnableInsert	P	P	Enables insertion of data.
ExportDirectoryChangeable (Page 5328)	P	P	Specifies whether the data export directory can be changed in runtime.
ExportDirectoryname (Page 5329)	P	P	Specifies the target directory for the exported Runtime data.
ExportFileExtension (Page 5330)	P	P	Specifies the file extension of the export file. Only the "csv" file extension is currently supported.
ExportFilename (Page 5330)	P	P	Specifies the name of the target file for the exported Runtime data.
ExportFilenameChangeable (Page 5331)	P	P	Specifies whether the export file name can be changed in runtime.
ExportFormatGuid (Page 5331)	P	P	Default assignment of the ID number and export provider.
ExportFormatName (Page 5332)	P	P	Specifies the export file format. Only the "csv" file format is currently available for the export.
ExportParameters (Page 5332)	-	-	Specifies the parameters of the selected format by means of the "Properties" dialog.
ExportSelection (Page 5333)	P	P	Specifies which runtime data of the control is exported.
ExportShowDialog (Page 5334)	P	P	Specifies whether the data export dialog is shown in runtime.
Filters	-	-	Specifies database criteria in SQL syntax.
FilterSQL	P	-	Sets an SQL statement for the filter criterion.
Font (Page 5348)	-	-	Specifies or returns the font.
GridLineColor (Page 5356)	P	P	Specifies the color for the grid lines.
GridLineWidth (Page 5357)	P	P	Specifies the width of the separation lines in pixels.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipse", "Circle" and "Rectangle" objects.
HorizontalGridLines (Page 5373)	P	P	Specifies whether horizontal separation lines are displayed.
IconSpace (Page 5375)	P	P	Specifies the spacing between icons and text in the table cells.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineColor (Page 5398)	P	P	Sets the color of the window separation lines.
LineWidth (Page 5400)	P	P	Specifies the line thickness of the selected object.
Moveable (Page 5425)	P	P	Specifies whether the window can be moved in runtime.
ObjectName (Page 5706)	P	-	Returns the object name as STRING.
PrintJobName (Page 5714)	P	P	Specifies a print job.

Properties	Read	Write	Description
RowScrollbar (Page 5465)	P	P	Specifies whether row scroll bars are displayed.
RowTitleAlignment	P	P	Specifies the type of row title alignment.
RowTitles	P	P	Specifies that row headers will be displayed.
RTPersistence (Page 5466)	P	P	Specifies how online configurations of WinCC are retained.
RTPersistencePasswordLevel (Page 5716)	P	P	Specifies the authorization required for online configuration in runtime.
RTPersistenceType (Page 5467)	-	-	Specifies how online configurations of WinCC are retained.
SelectArchiveName *	P	P	Specifies definition of the log by name.
SelectedCellColor (Page 5478)	P	P	Specifies the background color of the selected cell.
SelectedCellForeColor (Page 5479)	P	P	Specifies the font color of the selected cell.
SelectedRowColor (Page 5480)	P	P	Specifies the background color of the selected row.
SelectedRowForeColor (Page 5481)	P	P	Specifies the font color of the selected row.
SelectedTitleColor (Page 5482)	P	P	Specifies the background color of a selected table header.
SelectedTitleForeColor (Page 5482)	P	P	Specifies the font color of the selected table header.
SelectionColoring (Page 5485)	P	P	Specifies whether selection colors are used for cells or rows.
SelectionRect (Page 5486)	P	P	Specifies whether a selection frame is used for the selected cells or rows.
SelectionRectColor (Page 5486)	P	P	Specifies the color of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionRectWidth (Page 5487)	P	P	Specifies the line width of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionType (Page 5488)	P	P	Specifies the number of lines you can mark.
ShowSortButton (Page 5500)	P	P	Specifies whether the sorting button is shown above the vertical scroll bar.
ShowSortIcon (Page 5501)	P	P	Specifies whether the sorting icon is displayed.
ShowSortIndex (Page 5501)	P	P	Specifies whether a sort index is displayed.
ShowTitle (Page 5504)	P	P	Specifies the layout of the control window's header.
Sizeable	P	P	Specifies that the size of an object can be changed in runtime.
SortSequence	P	P	Specifies how the sorting order can be changed by mouse click.
StatusbarBackColor (Page 5511)	P	P	Specifies the background color of the status bar.
StatusbarElementAdd (Page 5512) *	-	P	Creates a new, user-defined status bar element.
StatusbarElementAutoSize (Page 5512) *	P	P	Specifies whether the width of the selected status bar element is set automatically.
StatusbarElementCount (Page 5513) *	P	-	Specifies the number of configurable status bar elements.
StatusbarElementIconId (Page 5514) *	P	P	Default assignment of the ID number and icon of a status bar element.
StatusbarElementId (Page 5514) *	P	-	Returns the ID number of the selected status bar element.
StatusbarElementIndex (Page 5515)	P	P	References a status bar element. You can use this property to assign the values of other properties to a specific element of the status bar.

Properties	Read	Write	Description
StatusbarElementName (Page 5516) *	P	P	Displays the object name of the selected status bar element.
StatusbarElementRemove (Page 5516) *	P	P	Removes the selected status bar element.
StatusbarElementRename (Page 5517) *	P	P	Renames the user-defined status bar element that is referenced by means of the "StatusbarElementIndex" property.
StatusbarElementRepos *	P	P	Specifies the index that is assigned to the element.
StatusbarElements (Page 5517)	-	-	Specifies the elements that are to be shown in the status bar.
StatusbarElementText *	P	P	Specifies the text of the currently selected status bar element.
StatusbarElementTooltipText (Page 5518) *	P	P	Specifies the tooltip text for the user-defined status bar element.
StatusbarElementUserDefined (Page 5518) *	P	-	Indicates whether the configuration engineer has added the status bar element as a new user-defined element.
StatusbarElementVisible (Page 5519) *	P	P	In the list, activate the status bar elements that you wish to display in runtime.
StatusbarElementWidth (Page 5519) *	P	P	Specifies the width of the selected status bar element in pixels.
StatusbarFont	-	-	Status bar font
StatusbarFontColor (Page 5520)	P	P	Specifies the color of the text in the status bar.
StatusbarShowTooltips (Page 5521)	P	P	Specifies whether tooltips are displayed for the elements of the status bar in runtime.
StatusbarText (Page 5521)	P	-	Returns the standard text of the status bar.
StatusbarUseBackColor (Page 5522)	P	P	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 5523)	P	P	Specifies whether the status bar of the control is displayed.
TableColor (Page 5526)	P	P	Specifies the background color of the rows. Use this to open the color selection dialog.
TableColor2 (Page 5526)	P	P	Specifies the background color of "Row color 2".
TableForeColor (Page 5527)	P	P	
TableForeColor2 (Page 5528)	P	P	Specifies the font color of "Row color 2".
TimeBase (Page 5540)	P	P	Sets the timebase.
TitleColor	P	P	Specifies the color of the header.
TitleCut (Page 5557)	P	P	Specifies whether the contents of the fields of a header should be abbreviated if the column is too narrow.
TitleDarkShadowColor (Page 5558)	P	P	Specifies the color of the dark side of shading.
TitleForeColor (Page 5559)	P	P	Specifies the font color of the table header for the selected status.
TitleGridLineColor (Page 5559)	P	P	Specifies the color of separation lines in the table header.
TitleLightShadowColor (Page 5560)	P	P	Specifies the color of the bright side of shading.
TitleSort (Page 5561)	P	P	Specifies how sorting by column header is triggered.
TitleStyle (Page 5561)	P	P	Specifies whether a shading color is used for the table header.
ToolbarAlignment (Page 5568)	P	P	Specifies or returns the position of the toolbar.

Properties	Read	Write	Description
ToolbarBackColor (Page 5568)	P	P	Specifies the background color of the toolbar.
ToolbarButtonActive (Page 5569) *	P	P	Specifies whether the function linked to the button is activated in runtime.
ToolbarButtonAdd (Page 5570) *	P	P	Creates a new, user-defined button function.
ToolbarButtonBeginGroup (Page 5570) *	P	P	Specifies whether a separator is inserted before the selected button function.
ToolbarButtonClick	P	P	Clicks a toolbar button.
ToolbarButtonCount (Page 5571) *	P	-	Returns the number of buttons contained in the toolbar.
ToolbarButtonEnabled (Page 5572) *	P	P	Specifies whether a user-defined toolbar button can be operated.
ToolbarButtonHotKey (Page 5572) *	P	P	Shows the hotkey for the selected button function.
ToolbarButtonId (Page 5573) *	P	P	Unique ID number for the selected button function.
ToolbarButtonIndex (Page 5573)	P	P	References a button function.
ToolbarButtonLocked (Page 5574) *	P	P	Specifies whether the clicked, pressed in state is displayed for a user-defined toolbar button.
ToolbarButtonName (Page 5575) *	P	-	Displays the name of the selected button of the toolbar.
ToolbarButtonPasswordLevel (Page 5575) *	P	P	Shows the authorization for the selected button function.
ToolbarButtonRemove (Page 5576) *	-	P	Removes the selected button function from the list.
ToolbarButtonRename (Page 5576) *	-	P	Renames the user-defined toolbar element that is referenced by means of the "ToolbarButtonIndex" property.
ToolbarButtonRepos (Page 5577) *	P	P	Changes the sequence of button functions.
ToolbarButtons	-	-	Specifies which buttons are available for the toolbar.
ToolbarButtonTooltipText (Page 5577) *	P	P	Specifies the tooltip text for the button.
ToolbarButtonUserDefined (Page 5578) *	P	-	Indicates whether the configuration engineer has added the toolbar button as a new user-defined button.
ToolbarButtonVisible *	P	P	Specifies that the button is visible.
ToolbarShowTooltips (Page 5579)	P	P	Specifies whether the tooltips for the button functions are displayed in runtime.
ToolbarUseBackColor (Page 5579)	P	P	Specifies whether the background color for the toolbar is shown.
ToolbarUseHotKeys (Page 5580)	P	P	Specifies whether the hotkeys for the button functions are activated in runtime.
ToolbarVisible (Page 5580)	P	P	Specifies whether the toolbar of the control is displayed.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
UseSelectedTitleColor (Page 5630)	P	P	Specifies whether a selection color is used for the headers of selected table cells.
UseTableColor2 (Page 5631)	P	P	Specifies whether a second row color is used for the representation of the table.
VerticalGridLines (Page 5644)	P	P	Specifies whether vertical separation lines are displayed.
Visibility	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipse", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-138 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property with VBScript during runtime.
CopyRows (Page 5833)	P	Executes the "Copy rows" button function of the control.
CutRows (Page 5834)	P	Executes the "Cut rows" button function of the recipe view.
DeleteRows (Page 5837)	P	Executes the "Delete rows" button function of the recipe view.
Export (Page 5838)	P, A	Executes the "Export log" or "Export data" button function of the control.
GetColumn (Page 5839)	P	Returns the column object of the recipe view designated by name or index as "ICCAxUAColumn" type.
GetColumnCollection (Page 5840)	P	Returns the list of all column objects of the recipe view as "ICCAxCollection" type.
GetRow (Page 5850)	P	Returns the row object defined by its row number in the table-based controls as "ICCAxDataRow" type.
GetRowCollection (Page 5851)	P	Returns the list of all row objects of the table-based controls as "ICCAxDataRowCollection" type.
GetSelectedRow (Page 5858)	P	Returns the selected row object of a table-based control as "ICCAxDataRow" type.
GetSelectedRows (Page 5859)	P	Returns the selected row objects of a table-based control as type "ICCAxDataRow" for multiple selection.
GetStatusbarElement (Page 5865)	P, A	Returns the element of the control's status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 5866)	P, A	Returns the list of all status bar elements of the control as "ICCAxCollection" type.
GetToolbarButton (Page 5873)	P, A	Returns the button function designated by name or index on the control toolbar as "ICCAxToolbarButton" type.
GetToolbarButtonCollection (Page 5874)	P, A	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.
MoveToFirst (Page 5894)	P	Executes the "First line" button function of the control.
MoveToLast (Page 5896)	P	Executes the "Last data record" button function of the control.
MoveToNext (Page 5897)	P	Executes the "Next data record" button function of the control.
MoveToPrevious (Page 5899)	P	Executes the "Previous data record" button function of the control.
PasteRows (Page 5902)	P	Executes the "Insert rows" button function of the recipe view.
Print (Page 5903)	P, A	Executes the "Print" button function of the control.
ReadTags (Page 5905)	P	Executes the "Read tags" button function of the recipe view.
SelectAll (Page 5918)	P	Selects all rows in a table-based control.
SelectRow (Page 5919)	P	Selects a specific row in a table-based control.
ServerExport (Page 5920)	P	Executes the "Export log" button function of the recipe view.
ServerImport (Page 5920)	P	Executes the "Import log" button function of the recipe view.
ShowHelp (Page 5923)	P, A	Executes the "Help" button function of the control.
ShowPropertyDialog (Page 5927)	P, A	Executes the "Configuration dialog" button function of the control.
ShowSelectArchive (Page 5928)	P	Executes the "Select data connection" button function of the recipe view.
ShowSelection (Page 5928)	P	Executes the "Selection dialog" button function of the recipe view.

Methods	Valid	Description
ShowSelectTimeBase (Page 5929)	P	Executes the "Timebase dialog" button function of the recipe view.
ShowSort (Page 5930)	P	Executes the "Sorting dialog" button function of the recipe view.
UnselectAll (Page 5936)	P	Removes all selections from the cells of a table-based control.
UnselectRow (Page 5936)	P	Removes the selections from a specific cell of a table-based control.
WriteTags (Page 5940)	P	Executes the "Write tags" button function of the recipe view.

See also

Screen (Page 5000)

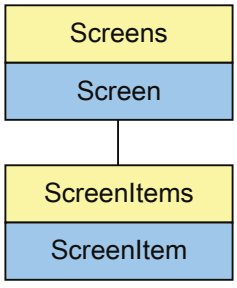
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

UserView

Description



Represents the "UserView" object. The UserView object is an element of the ScreenItems list.

Note

The object "Simple UserView" cannot be dynamized with a user-defined function.

Type identifier in VBS

HMIUserView

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-139 Properties

Properties	Read	Write	Description
Appearance (Page 5000)	-	-	Specifies the layout for this NC subroutine.
Authorization (Page 5245) *	-	-	Specifies the operating rights of the selected object in runtime.
BackColor	A	A	Specifies the background color of the selected object.
Columns	-	-	Specifies the columns to be displayed.
ColumnsMoveable	-	-	Specifies that the columns can be moved.
CountVisibleItems (Page 5309)	-	-	Specifies the number of lines contained in the selection list.
Enabled	A	A	Specifies whether the selected object can be operated in runtime.
FitToSize	-	-	Specifies that the size of the object should be automatically adapted to the content.
Flashing	-	-	Specifies that the text block flashes.
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	A	-	Returns the object name as STRING.
SelectionBackColor (Page 5484)	A	A	Specifies the background color of the selected cells.
SelectionForeColor (Page 5485)	A	A	Specifies the foreground color of the selected cells.
ShowTableGridlines (Page 5502)	-	-	Enables the display of gridlines in the table of the specified object.
TableBackColor (Page 5525)	A	A	Specifies the background color of the table cells for the selected object.
TableFont	-	-	Specifies the font in the table.
TableForeColor	A	A	
TableGridlineColor (Page 5528)	A	A	Specifies the color for the grid lines.
TableHeaderBackColor (Page 5529)	A	A	Specifies the background color in the header of the table of the selected object.
TableHeaderFont	-	-	Specifies the text color of the header.
TableHeaderForeColor (Page 5530)	A	A	Specifies the text color in the header of the table of the selected object.

Properties	Read	Write	Description
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Visible (Page 5645) *	A	A	Specifies whether the selected object is visible.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

* not visible in the ES

Table 12-140 Methods

Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

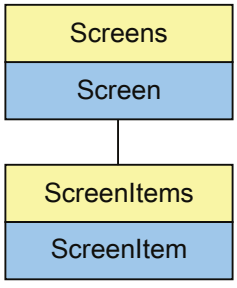
ScreenItem (Page 5003)

ScreenItems (list) (Page 5005)

Screen object (list) (Page 5007)

WindowSlider

Description



Represents the "Window slider" object. The WindowSlider object is an element of the ScreenItems list.

Type identifier in VBS

HMIWindowSlider

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-141 Properties

Properties	Read	Write	Description
AskOperationMotive (Page 5244)	P	P	Specifies whether the reason for operating this object is logged.
Authorization (Page 5245)	P	P	Specifies the operating rights of the selected object in runtime.
BackColor (Page 5251)	P	P	Specifies the background color of the selected object.
BackColorBottom (Page 5253)	P	P	Specifies the color for the lower / right part of the .
BackColorTop (Page 5254)	P	P	Sets the color for the upper / left part of the slider.
BackFillStyle (Page 5254)	P	P	Defines the fill pattern of the specified object.
BackFlashingColorOff (Page 5256)	P	P	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 5257)	P	P	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 5258)	P	P	Specifies whether the background of the specified object flashes in runtime.
BackFlashingRate (Page 5258)	P	P	Specifies the flash rate of the background for the selected object.
BorderBackColor (Page 5266)	P	P	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 5268)	P	P	Specifies the line color of the object.
BorderFlashingColorOff (Page 5271)	P	P	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 5272)	P	P	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 5273)	P	P	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 5274)	P	P	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 5278)	P	P	Specifies the type of border lines for the selected object.
BorderWidth (Page 5280)	P	P	Specifies the line thickness of the object.
CornerStyle (Page 5306)	-	-	Specifies the type of border lines for the selected object.
DrawInsideFrame (Page 5319)	P	P	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 5320)	-	-	Specifies the line style of the selected object.

Properties	Read	Write	Description
Enabled (Page 5323)	P	P	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 5336)	P	P	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	Specifies that the text block flashes.
Height (Page 5358)	P	P	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
HighLimitColor (Page 5363)	P	P	Specifies the color of the top or right scroll button in a scroll bar.
JumpToLimitsAfterMouseClick (Page 5379)	P	P	Specifies the length of the long tick marks of a scale.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	P	P	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 5399)	-	-	Specifies the shape of the line end.
LogOperation (Page 5403)	P	P	Specifies whether, after operating this object, an alarm is output to the alarm system.
LowLimitColor (Page 5404)	P	P	Specifies the color of the bottom or left scroll button in a scroll bar.
MarginToBorder (Page 5405)	P	P	Specifies the width of the 3D border in pixels.
MaximumValue (Page 5406)	P	P	Specifies the maximum value of the scale in the selected object.
MinimumValue (Page 5422)	P	P	Specifies the minimum value of the scale in the selected object.
Name (Page 5426)	P	-	Returns the object name as STRING.
OperationSteps (Page 5435)	P	P	Specifies by how many steps the slider of the scrollbar is moved with one mouseclick.
ProcessValue (Page 5456)	P	P	Specifies the default for the value to be displayed.
RelativeFillLevel (Page 5461)	P	P	Specifies the fill percentage for the object.
ShowBadTagState (Page 5493)	P	P	Specifies whether an object is grayed out if its associated tag has a bad quality code or tag state.
ShowFillLevel (Page 5495)	P	P	Specifies whether the selected object is filled.
StyleSettings (Page 5524)	P	P	Specifies the display style for the object:
TextOrientation (Page 5536)	P	P	Specifies the text orientation of the selected object.
ThumbBackColor (Page 5536)	P	P	Specifies the background color of the slider in the "Slider" object.
ToolTipText (Page 5581)	P	P	Specifies the tooltip text.
Top (Page 5582)	P	P	Specifies the value of the Y coordinate of the object.
Transparency (Page 5585)	P	P	Specifies and returns the transparency percentage of the object.
UseDesignColorScheme (Page 5625)	P	P	Specifies the font used for printing.
UseDesignShadowSettings (Page 5627)	P	P	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 5645)	P	P	Specifies whether the selected object is visible.
Width (Page 5655)	P	P	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.
WindowsStyle (Page 5659)	P	P	Specifies whether the object is displayed in the general Windows style.

Table 12-142 Methods

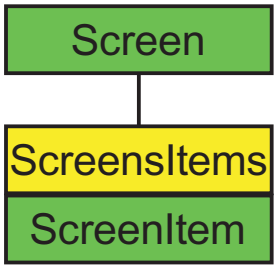
Methods	Valid	Description
Activate (Page 5825)	P, A	Activates the permanent window or the root screen.
ActivateDynamic (Page 5828)	P	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 5835)	P	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during Runtime.

See also

Screen object (list) (Page 5007)

WLanQualityView

Description



Represents the "WLAN reception" object. The WLanQualityView object is an element of the ScreenItems list.

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-143 Properties

Properties	Read	Write	Description
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	A	-	Returns the object name as STRING.

Properties	Read	Write	Description
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-144 Methods

Methods	Valid	Description
????		

See also

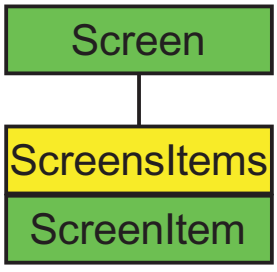
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

ZoneLabelView

Description



Represents the "Zone name" object. The ZoneLabelView object is an element of the ScreenItems list.

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-145 Properties

Properties	Read	Write	Description
Font (Page 5348)	-	-	Specifies or returns the font.
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	A	-	Returns the object name as STRING.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-146 Methods

Methods	Valid	Description
???		

See also

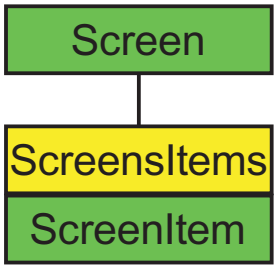
Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

ZoneQualityView

Description



Represents the "ZoneSignal" object. The ZoneQualityView object is an element of the ScreenItems list.

Abbreviation	Validity
Pa	Panels
A	Runtime Advanced
P	Runtime Professional

Table 12-147 Properties

Properties	Read	Write	Description
Height (Page 5358)	A	A	Sets the height of the specified "Ellipsis", "Circle" and "Rectangle" objects.
Layer (Page 5384)	-	-	Returns the layer that contains an object as LONG in the screen.
Left (Page 5388)	A	A	Specifies the value of the X coordinate of the object.
Name (Page 5426)	A	-	Returns the object name as STRING.
Top (Page 5582)	A	A	Specifies the value of the Y coordinate of the object.
Width (Page 5655)	A	A	Sets the width of the specified "Ellipsis", "Circle" and "Rectangle" objects.

Table 12-148 Methods

Methods	Valid	Description
?????		

See also

Screen (Page 4976)

ScreenItem (Page 4977)

ScreenItems (Page 4979)

Properties

Properties A

AboveUpperLimitColor

Description

Defines the color of the specified object for the case "High limit exceeded".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**AboveUpperLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- IOField

You have no access in runtime with the following format:

- GraphicIOField
- Switch
- SymbolLibrary
- SymbolicIOField

Color

Optional A value or a constant that specifies the color of the specified object for the "Higher limit violated" case.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

AcceptOnExit

Description

Specifies whether the input field will be confirmed automatically when it is left.

Access in runtime: Read and write

Syntax

Object.**AcceptOnExit**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField
- SymbolicIOField

BOOLEAN

Optional TRUE, if the input field will be confirmed automatically when it is left.

AcceptOnFull

Description

Specifies whether the input field will be left and confirmed automatically when the set number of values have been entered.

Access in runtime: Read and write

Syntax

Object.**AcceptOnFull**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField

BOOLEAN

Optional TRUE, if the input field will be left and confirmed automatically when the set number of values have been entered.

AccessPath

Description

Returns the storage path of a screen.

Access during runtime: Read

Syntax

Object.**AccessPath**

Object

Necessary. A "Screen" object.

Example

In the following example, the path of the picture "ScreenWindow1" is issued:

```
'VBS67
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
MsgBox objScreen.AccessPath
```

See also

Screen (Page 5000)

ActiveProject

Description

Returns the specified project.

Access in Runtime: Read

Syntax

Object.**ActiveProject**

Object

Required A "HMIRuntime" object.

See also

HMIRuntime (Page 4993)

Project (Page 4999)

ActiveScreen

Description

Returns a "Screen" type object, which represents the screen which has the focus at the moment.

Note

If you query the "ActiveScreen" property in a user-defined function, it may be due to a screen saver that the property does not return a valid "screen" object but "Nothing". A system message will be issued.

Access in Runtime: Read

Syntax

Object.ActiveScreen

Object

Required An object of the "HMIRuntime" type.

Comments

Which screen is returned depends on whether the root screen or the permanent window has the focus.

The ActiveScreen property returns NOTHING if no screen has the focus. That is the case, for example, when another window has the focus. With the statement "If not [expression] is nothing" you are able to interrogate whether a screen is to be returned:

```
VBS example ActiveScreen
Dim objActiveScreen
Set objActiveScreen = HmiRuntime.ActiveScreen
If not objActiveScreen is nothing then
'found an active screen
HmiRuntime.Trace("There is an active screen." & vbCrLf)
Else
'found NO active screen
HmiRuntime.Trace("There is NO active screen." & vbCrLf)
End if
```

ActiveScreen

Description

Returns an object of type "Screen" which shows the screen that currently has the focus.

Note

If you query the "ActiveScreen" property in a function, it may be due to a ScreenSavers that the property does not return a valid "Screen" object but "Nothing" and a system alarm is displayed.

Access during runtime: Read

Syntax

Object.ActiveScreen

Object

Necessary. An object of the "HMIRuntime" type.

Comments

Which screen is returned depends on whether the root screen or the permanent window has the focus.

The ActiveScreen property returns NOTHING if no screen has the focus. This is the case, for example, if a different window has the focus. With the instruction "If not [printout] Is Nothing" you can query whether a screen is going to be returned:

```
'VBS_Example_ActiveScreen
Dim objActiveScreen
Set objActiveScreen = HmiRuntime.ActiveScreen
If Not objActiveScreen Is Nothing Then
'found an active screen
HmiRuntime.Trace("There is an active screen." & vbCrLf)
Else
'found NO active screen
HmiRuntime.Trace("There is NO active screen." & vbCrLf)
End If
```

ActiveScreenItem

Description

References the screen object that currently has the focus.

It is only if the screen of the respective "Screen" object is currently selected and has an input field that the "ActiveScreenItem" property of the "Screen" object will be occupied with a valid

"ScreenItem" object. In all other cases if, for example, another screen from the "Screens" list, an independent window within WinCC or another application is selected, this property is not supplied on the screens, i.e. assigned "Nothing".

Application

You use the "ActiveScreenItem" object to address the properties of the object that has the focus in Runtime.

ActiveScreenItem

Description

References the screen object which currently has the focus.

The "ActiveScreenItem" property of the "Screen" object is only assigned a valid "ScreenItem" object if the screen of the corresponding "Screen" object is active and has an input field. In all other cases, for example, with a different screen from the "Screens" list, an independent window within WinCC or a different application will be selected, this property will not apply to the screens and will be occupied with "Nothing".

Application

The "ActiveScreenItem" object is used to address the object property which has the focus in runtime.

ActualPointIndex

Description

Specifies the number of the current corner point of the selected object.

Access in runtime: Read and write

Syntax

Object.**ActualPointIndex**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- Polygon
- Polyline
- Tubepolyline

You have no access in runtime with the following format:

- Line

Int32

Optional A value or a constant that specifies the number of the current corner point of the selected object.

ActualPointLeft

Description

Specifies the X coordinate of the current corner point with reference to the screen origin. The screen source is at the top left of the object. Every corner is identified by an index which is derived from the number ("PointCount") of the existing corners.

Access in runtime: Read and write

Syntax

Object.**ActualPointLeft**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- Polygon
- Polyline
- Tubepolyline

You have no access in runtime with the following format:

- Line

Int32

Optional A value or a constant that specifies the X coordinate of the current corner point with reference to the screen origin.

Comments

Changing the value can have an effect on the properties "Width" (object width) and "Left" (X coordinate of the object position).

ActualPointTop

Description

Specifies the Y coordinate of the current corner point with reference to the screen origin. The screen source is at the top left of the object.

Access in runtime: Read and write

Syntax

Object.**ActualPointTop**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- Polygon
- Polyline
- Tubepolyline

You have no access in runtime with the following format:

- Line

Int32

Optional A value or a constant that specifies the Y coordinate of the current corner point with reference to the screen origin.

Comments

Changing the value can have an effect on the properties "Height" (object height) and "Top" (X coordinate of the object position).

AdaptBorder

Description

Specifies whether the border of the selected object will be dynamically adapted to the text size.

Access in runtime: Read and write

Syntax

Object.**AdaptBorder**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- Button
- IOField
- OptionGroup
- SymbolicIOField
- TextField

You have no access in runtime with the following format:

- Checkbox

BOOLEAN

Optional TRUE, if the border of the selected object will be dynamically adapted to the text size.

AdaptPicture

Description

No access in runtime.

AdaptScreenToWindow

Description

Specifies whether or not the screen displayed in a screen window adapts to the size of the screen window in Runtime.

Access in runtime: Read

Syntax

Object.**AdaptScreenToWindow**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Screenwindow

BOOLEAN

TRUE, if the screen adapts to the screen window size.

FALSE, if the screen does not adapt to the screen window size.

AdaptWindowtoScreen

Description

Specifies whether the screen window adapts to the screen it displays in Runtime.

Access in runtime: Read

Syntax

Object.**AdaptWindowtoScreen**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Screenwindow

BOOLEAN

TRUE, if the screen window size adapts to the screen.

FALSE, if the screen window size does not adapt to the screen.

Address

Description

Specifies the web address that will be opened in the HTML browser.

Access in Runtime: Read and write

Syntax

Object.**Address**[=STRING]

Object

Required. An object of the "ScreenItem" type with the format:

- HTML-Browser

STRING

Optional A value or a constant that contains the web address.

AddressEnabled

Description

No access in runtime.

AdressPreview

Description

No access in runtime.

Alarm

Description

Specifies the limit for the storage space display as of which an alarm will be reported.

Access in runtime: Read and write

Syntax

Object.**Alarm**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

Int32

Optional A value or a constant that specifies the limit for the disk space view as of which an alarm will be reported.

AlarmClasses

Description

No access in runtime.

AlarmColor

Description

Specifies the color in which the bars will be shown as soon as the alarm area is reached.

Access in runtime: Read and write

Syntax

Object.**AlarmColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

Color

Optional A value or a constant that specifies the color in which the bar will be shown as soon as the warning area is exceeded.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0-255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

AlarmID

Description

Returns the AlarmID of the Alarm object. The AlarmID is unique and is assigned by the system.

AlarmID (readonly)

See also

Alarms (list) (Page 4984)

AlarmLog

Description

No access in runtime.

AlarmLogs

Description

Returns an object of type "AlarmLogs".

Access in Runtime: Read

Syntax

Object.**AlarmLogs**

Object

Required A "Logging" object.

See also

Logging (Page 4998)

AlarmLowerLimit

Description

Specifies the low limit at which the alarm is triggered.

Access in runtime: Read and write

Syntax

Object.**AlarmLowerLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the low limit at which the alarm is triggered.

Comments

The type of evaluation (percentage or absolute) is defined using the "AlarmLowerLimitRelative" property.

The "AlarmLowerLimitEnable" property defines whether or not monitoring of this limit is enabled.

AlarmLowerLimitColor

Description

Specifies the bar color of the "AlarmLowerLimit" limit.

The "AlarmLowerLimitEnable" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime: Read and write

Syntax

Object.**AlarmLowerLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar

You have no access in runtime with the following format:

- Slider

Color

Optional A value or a constant that specifies the bar color for the "AlarmLowerLimit" limit.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

AlarmLowerLimitEnabled

Description

Specifies whether the "AlarmLowerLimit" limit is monitored.

Access in runtime: Read and write

Syntax

Object.**AlarmLowerLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the "AlarmLowerLimit" limit is monitored.

Comments

The following values will be defined via the properties "AlarmLowerLimit", "AlarmLowerLimitColor" and "AlarmLowerLimitRelative":

Limit

Representation upon reaching the limit

Type of evaluation

AlarmLowerLimitRelative

Description

Establish whether the lower limit at which the interrupt is triggered is evaluated as a percentage or an absolute value.

Access in runtime: Read and write

Syntax

Object.**AlarmLowerLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the lower limit at which the interrupt is triggered is evaluated as a percentage.

AlarmSource

Description

No access in runtime.

AlarmUpperLimit

Description

Establishes the upper limit at which the interrupt will be triggered.

Access in runtime: Read and write

Syntax

Object.**AlarmUpperLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that defines the upper limit at which an interrupt will be triggered.

Comments

The type of evaluation (percentage or absolute) is defined using the "AlarmUpperLimitRelative" property.

The "AlarmUpperLimitEnable" property defines whether or not monitoring of this limit is enabled.

AlarmUpperLimitColor

Description

Defines the bar color for the "AlarmUpperLimit" limit.

The "AlarmUpperLimitEnable" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime: Read and write

Syntax

Object.**AlarmUpperLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Bar

You have no access in runtime with the following format:

- Slider

Color

Optional A value or a constant that defines the bar color for the "AlarmUpperLimit" limit.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

AlarmUpperLimitEnabled

Description

Specifies whether the "AlarmUpperLimit" limit is monitored.

Access in runtime: Read and write

Syntax

Object.**AlarmUpperLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the "AlarmUpperLimit" limit is monitored.

Comments

The following values will be defined via the properties "AlarmUpperLimit", "AlarmUpperLimitColor" and "AlarmUpperLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

AlarmUpperLimitRelative

Description

Establish whether the upper limit at which the interrupt is triggered is evaluated as a percentage or an absolute value.

Access in runtime: Read and write

Syntax

Object.**AlarmUpperLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the high limit at which the interrupt is triggered is evaluated as a percentage.

AllFilters

Description

No access in runtime.

AllFiltersForHitlist

Description

No access in runtime.

AllowEdit

Description

No access in runtime.

AllowMenu

Description

No access in runtime

Analog

Description

Specifies whether the clock should be shown as an analog clock.

Access in runtime: Read and write

Syntax

Object.**Analog**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Clock

BOOLEAN

Optional TRUE, if the clock should be shown as an analog clock.

AngleMax

Description

Specifies the angle of the scale end of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**AngleMax**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the angle in degrees.

Comments

The start and end of the scale gradations are described with the properties "AngleMin" and "AngleMax" in angle degrees.

The value of the AngleMin property must always be less than the value of the AngleMax property. The zero degree angle is located at the 3 o´clock position on the scale. Positive angle values are counted clockwise.

AngleMin

Description

Specifies the angle of the scale start of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**AngleMin**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the angle in degrees.

Comments

The start and end of the scale gradations are described with the properties "AngleMin" and "AngleMax" in angle degrees.

The value of the AngleMin property must always be less than the value of the AngleMax property.

The zero degree angle is located at the 3 o´clock position on the scale. Positive angle values are counted clockwise.

Appearance

Description

No access in runtime.

ApplyProjectSettings

Description

Specifies whether the project settings are applied from the "HMI alarms" editor.

Access in runtime: Read and write

Syntax

Object.**ApplyProjectSettings**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: The alarm text blocks configured in the "HMI alarms" editor are applied in the alarm view with their properties. The alarm text blocks are displayed with these properties in the alarm view.

FALSE: The properties are not applied.

ApplyProjectSettingsForDesignMode

Description

No access in runtime.

AskOperationMotive

Description

Specifies whether the reason for operating this object will be logged. When operating the specified object in Runtime, a dialog opens in which the operator enters the reason for the operation in text format.

Access in runtime: Read and write

Syntax

Object.**AskOperationMotive**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- ComboBox
- IOField
- ListBox
- SymbolicIOField
- WindowsSlider

BOOLEAN

Optional TRUE, if the reason for operating this object will be logged.

Assignments

Description

Specifies a list which contains the assignments between the output value and the output text actually to be output. The assignments are dependent on the set list type. You define the list type with the ListType property.

Access in runtime: Read and write

Syntax

Object.**Assignments**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- SymbolicIOField

STRING

Optional Specifies a list which contains the assignments between the output value and the output text actually to be output.

AssociatedS7GraphDBTag

Description

No access in runtime.

Authorization

Description

Specifies the operating rights of the selected object in Runtime.

Access in runtime: Read and write

Syntax

Object.**Authorization**[=HMIRTAuthorization]

Object

Required. An object of the "ScreenItem" type with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- DateTimeField
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- RecipeView
- Rectangle
- RoundButton
- Slider
- StatusForce

- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TextField
- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserView
- WindowsSlider

You have no access in runtime with the following format:

- AlarmView
- PLCCodeViewer
- ProtectedAreaNameView
- RangeLabelView
- S7GraphOverview

HMIRTAuthorization

Optional A value or a constant that establishes the operating rights of the specified object in Runtime.

AutoCompleteColumns

Description

Adds empty columns if the Control width is greater than the width of columns configured.

Access in runtime: Read and write

Syntax

Object.**AutoCompleteColumns**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl

- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The empty columns are displayed.

FALSE: The empty columns are not displayed.

AutoCompleteRows

Description

Enables the insertion of empty rows if the Control length is greater than the number of rows configured.

Access in runtime: Read and write

Syntax

Object.**AutoCompleteRows**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The empty rows are displayed.

FALSE: The empty rows are not displayed.

AutoScroll

Description

Specifies how the alarm window reacts when new alarms appear.

The specific choice of alarm lines is only possible if "AutoScroll" is not enabled.

The "AutoScroll" property is disabled if the "MsgCtrlFlag" = "-1" attribute is set. The most recent alarm is shown at the top in the alarm window.

Access in runtime: Read and write

Syntax

Object.**AutoScroll**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

Optional

TRUE: A new alarm is appended to and selected from the list shown in the alarm window. If necessary, the visible area of the alarm window can be moved.

FALSE: A new alarm is not selected. The visible area of the alarm window is not altered.

AutoSelectionColors

Description

Enables the display of default system colors as selection color for cells and rows.

Access in runtime: Read and write

Syntax

Object.**AutoSelectionColors**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The system color is used.

FALSE: The customized color is used.

AutoSelectionRectColor

Description

Specifies whether the selection frame is shown with the specified color.

Access in runtime: Read and write

Syntax

Object.**AutoSelectionRectColors**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The system color is used.

FALSE: The customized color is used.

AutoSizing

Description

No access in runtime.

AverageLast15Values

Description

Specifies whether a mean value is shown from the last 15 values.

Access in Runtime: Read and write

Syntax

Object.**AverageLast15Value**s[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format "Bar".

BOOLEAN

Optional. TRUE, if a mean value is shown from the last 15 values.

AxisXBunchCount

Description

No access in runtime.

AxisXMarkCount

Description

No access in runtime.

AxisXShowBunchValues

Description

No access in runtime.

AxisY1BunchCount

Description

No access in runtime.

AxisY1MarkCount

Description

No access in runtime.

AxisY1ShowBunchValues

Description

No access in runtime.

AxisY2BunchCount

Description

No access in runtime.

AxisY2MarkCount

Description

No access in runtime.

AxisY2ShowBunchValues

Description

No access in runtime.

Properties B

BackColor

Description

Specifies the background color of the selected object.

Access in runtime: Read and write

Syntax

Object.**BackColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- AlarmView
- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- DateTimeField
- Ellipse
- EllipseSegment
- EllipticalArc

- FunctionTrendControl
- Gauge
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- Polygon
- Polyline
- RecipeView
- Rectangle
- RoundButton
- Slider
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- TextField
- TrendRulerControl
- TrendView
- TubeArcObject
- UserArchiveControl
- UserView
- WindowsSlider

You have no access in runtime with the following format:

- GraphicIOField

Color

Optional A value or a constant that specifies the background color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

The background color is not visible if the "BorderStyle" property has the value "0".

BackColorBottom

Description

Specifies the color for the lower / right part of the Slider object.

Access in runtime: Read and write

Syntax

Object.**BackColorBottom**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- WindowsSlider

Color

Optional A value or a constant that specifies the color for the lower / right part of the Slider object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0-255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

BackColorTop

Description

Specifies the color for the upper / left part of the Slider object.

Access in runtime: Read and write

Syntax

Object.**BackColorTop**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- WindowsSlider

Color

Optional A value or a constant that specifies the color for the upper / left part of the Slider object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0-255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

BackFillStyle

Description

Determines the fill style of the specified object.

Access in runtime: Read and write

Syntax

Object.**BackFillStyle**[=FillStyle]

Object

Required. A "ScreenItem" object with the following format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- Clock
- ComboBox
- Ellipse
- EllipseSegment
- Gauge
- GraphicView
- IOField
- ListBox

- OptionGroup
- Polygon
- Rectangle
- RoundButton
- Slider
- SymbolLibrary
- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- DateTimeField
- GraphicIOField
- Switch

FillStyle

Optional A value or a constant that specifies the fill style.

hmiFillStyleTransparent (65536): Transparent fill

hmiFillStyleSolid (0): Object is filled with the specified color

Default setting: hmiFillStyleSolid

BackFlashingColorOff

Description

Specifies the color of the background for the flashing status "off".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BackFlashingColorOff**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- GraphicView
- IOField

- OptionGroup
- RoundButton
- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- CheckBox
- Switch

Color

Optional A value or a constant that specifies the color of the background for the flashing status "off".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

BackFlashingColorOn

Description

Specifies the color of the background for the flashing status "on".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BackFlashingColorOn**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- GraphicView
- IOField
- OptionGroup
- RoundButton

- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- CheckBox
- Switch

Color

Optional A value or a constant that specifies the color of the background for the flashing status "on".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

BackFlashingEnabled

Description

Specifies whether the background of the specified object will flash in Runtime.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BackFlashingEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- GraphicView
- IOField
- OptionGroup
- RoundButton
- SymbolicIOField

- TextField
- WindowsSlider

You have no access in runtime with the following format:

- CheckBox
- Switch

BOOLEAN

Optional TRUE, if the background of the specified object is flashing in Runtime.

BackFlashingRate

Description

Specifies the flash rate of the background for the specified object.

Access in Runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BackFlashingRate**[=FlashingRate]

Object

Required. An object of the "ScreenItem" type with the format:

- Bar
- Button
- GraphicView
- IOField
- OptionGroup
- RoundButton
- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- CheckBox
- Switch

FlashingRate

hmiFlashingRateSlow (0): The length of the flashing interval is 250 ms.

hmiFlashingRateMedium (1): The length of the flashing interval is 500 ms.

hmiFlashingRateFast (2): The length of the flashing interval is 1000 ms.

BackgroundColor

Description

No access in runtime.

BarBackColor

Description

Specifies the color of the bar background for the selected object.

Access in runtime: Read and write

Syntax

Object.**BarBackColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar
- Slider

Color

Optional A value or a constant that specifies the color of the bar background.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

BarBackFillStyle

Description

Specifies the fill style for the bar.

Access in runtime: Read and write

Syntax

Object.**BarBackFillStyle**[=FillStyle]

Object

Required. A "ScreenItem" object with the following format:

- Bar

FillStyle

hmiFillStyleTransparent (65536): Transparent fill style

hmiFillStyleSolid (0): Solid fill style

hmiFillStyleBackwardDiagonal (131075): Diagonal fill style striped towards the top right

hmiFillStyleCross (131076): Checked fill style

hmiFillStyleDiagonalCross (131077): Diagonal checked fill style

hmiFillStyleForwardDiagonal (131074): Diagonal fill style striped towards the top left

hmiFillStyleHorizontal (131072): Horizontal striped fill style

hmiFillStyleVertical (131073): Vertical striped fill style

hmiFillStylePattern1 (196608): Default fill style

hmiFillStylePattern2 (196609): Default fill style

hmiFillStylePattern3 (196610): Default fill style

hmiFillStylePattern4 (196611): Default fill style

hmiFillStylePattern5 (196612): Default fill style

hmiFillStylePattern6 (196613): Default fill style

hmiFillStylePattern7 (196614): Default fill style

hmiFillStylePattern8 (196615): Default fill style

hmiFillStylePattern9 (196616): Default fill style

hmiFillStylePattern10 (196617): Default fill style

hmiFillStylePattern11 (196618): Default fill style

hmiFillStylePattern12 (196619): Default fill style

hmiFillStylePattern13 (196620): Default fill style

hmiFillStylePattern14 (196621): Default fill style

hmiFillStylePattern15 (196622): Default fill style

hmiFillStylePattern16 (196623): Default fill style

hmiFillStylePattern17 (196624): Default fill style

hmiFillStylePattern18 (196625): Default fill style

hmiFillStylePattern19 (196626): Default fill style

hmiFillStylePattern20 (196627): Default fill style

hmiFillStylePattern21 (196628): Default fill style

hmiFillStylePattern22 (196629): Default fill style
hmiFillStylePattern23 (196630): Default fill style
hmiFillStylePattern24 (196631): Default fill style
hmiFillStylePattern25 (196632): Default fill style
hmiFillStylePattern26 (196633): Default fill style
hmiFillStylePattern27 (196634): Default fill style
hmiFillStylePattern28 (196635): Default fill style
hmiFillStylePattern29 (196636): Default fill style
hmiFillStylePattern30 (196637): Default fill style
hmiFillStylePattern31 (196638): Default fill style
hmiFillStylePattern32 (196639): Default fill style
hmiFillStylePattern33 (196640): Default fill style
hmiFillStylePattern34 (196641): Default fill style
hmiFillStylePattern35 (196642): Default fill style
hmiFillStylePattern36 (196643): Default fill style
hmiFillStylePattern37 (196644): Default fill style

BarBackFlashingColorOff

Description

No access in runtime.

BarBackFlashingColorOn

Description

No access in runtime.

BarBackFlashingEnabled

Description

No access in runtime.

BarBackFlashingRate

Description

No access in runtime.

BarColor

Description

Specifies the color of the bar in the "Slider" object.

Access in runtime: Read and write

Syntax

Object.**BarColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Slider

Color

Optional A value or a constant that specifies the color of the slider.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

The range extends from "RangeMin" to the position of the controller.

BarEdgeStyle

Description

No access in runtime.

BaseScreenName

Description

Reads the name of the current root screen or triggers a root screen change by setting a new screen name.

Access in Runtime: Read and write

Syntax

Object.**BaseScreenName**[= STRING]

Object

Required An "HMIRuntime" object.

STRING

Optional A value or a constant that contains the screen name.

Comments

You can also use the property to establish which screen is currently being displayed.

BelowLowerLimitColor

Description

No access in runtime.

BitNumber

Description

Specifies the bit whose status must change to trigger a value change.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BitNumber**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- SymbolicIOField

You have no access in runtime with the following format:

- Button
- GraphicIOField

Int32

Optional Specifies the bit whose status must change to trigger a value change.

Comments

The tag being used must be of type BYTE, WORD or DWORD.

BlinkColor

Description

Specifies the color in which the "SymbolLibrary" object will flash.

Access in runtime: Read and write

Syntax

Object.**BlinkColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- SymbolLibrary

Color

Optional A value or a constant that specifies the flash color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

BlinkMode

Description

Sets the type of flash picture for the specified object.

Access in runtime: Read and write

Syntax

Object.**BlinkMode**[=SymbolLibraryBlinkMode]

Object

Required. A "ScreenItem" object with the following format:

- SymbolLibrary

SymbolLibraryBlinkMode

hmiSymbolLibraryFlashingNone (0): Flashing is switched off.

hmiSymbolLibraryFlashingInvisible (1): The flash picture is invisible.

hmiSymbolLibraryFlashingShaded (2): The flash picture is given a colored, shaded surface. The color of the surface corresponds to the settings in the property "BlinkColor".

hmiSymbolLibraryFlashingSolid (3): The flash picture is given a colored, unshaded surface. The color of the surface corresponds to the settings in the property "BlinkColor".

BlinkSpeed

Description

Specifies the flash rate.

Fast - 250: The length of the flashing rate is 250 ms. Medium - 500: The length of the flashing rate is 500 ms.

Slow - 1000: The length of the flashing rate is 1000 ms. The default value is medium - 500.

Access in runtime: Read and write

Syntax

Object.**BlinkSpeed**[=FlashingRate]

Object

Required. A "ScreenItem" object with the following format:

- SymbolLibrary

FlashingRate

hmiFlashingRateSlow (0): The length of the flashing rate is 250 ms.

hmiFlashingRateMedium (1): The length of the flashing rate is 500 ms.

hmiFlashingRateFast (2): The length of the flashing rate is 1000 ms.

Blocks

Description

No access in runtime.

BorderBackColor

Description

Specifies the background color of the broken border line for the specified object.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderBackColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- ComboBox
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- AlarmView
- Clock

- DateTimeField
- Gauge
- RecipeView
- Slider
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

Color

Optional A value or a constant that specifies the background color of the broken border line for the specified object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

BorderBrightColor3D

Description

Specifies the frame color of the following frame parts in case of a 3D display of the specified object:

- External frame parts at the top and bottom
- Internal frame parts at the top and right

Access in runtime: Read and write

Syntax

Object.**BorderBrightColor3D**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Button
- RoundButton
- Slider

You have no access in runtime with the following format:

- Switch

Color

Optional A value or a constant that specifies the frame color. Default setting is white.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

BorderColor

Description

Specifies the line color of the specified object.

Access in runtime: Read and write

Syntax

Object.**BorderColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- AlarmView
- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- ComboBox
- DateTimeField
- Ellipse
- EllipseSegment
- FunctionTrendControl
- GraphicIOField
- GraphicView
- IOField

- ListBox
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- TrendRulerControl
- UserArchiveControl
- WindowsSlider

You have no access in runtime with the following format:

- Clock
- Gauge
- RecipeView
- Slider
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

Color

Optional A value or a constant that specifies the line color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

BorderEnabled

Description

Returns whether the window is displayed with a border in runtime.

Access in runtime: Read

Syntax

Object.**BorderEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- ApplicationWindow
- Screenwindow

BOOLEAN

Optional TRUE if the window is displayed with a border in runtime.

BorderEndStyle

Description

Specifies the type of line-end shapes for the specified object.

Access in runtime: Read and write

Syntax

Object.**BorderEndStyle**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Line
- Polyline

Int32

Optional A value or a constant that specifies the type of line-end shapes for the specified object.

BorderFlashingColorOff

Description

Specifies the color of the border line for the selected object for the flashing status "off".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderFlashingColorOff**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- CheckBox
- Switch

Color

Optional A value or a constant that specifies the color of the border line of the selected object for the flashing status "off".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

BorderFlashingColorOn

Description

Specifies the color of the border line for the selected object for the flashing status "on".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderFlashingColorOn**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- CheckBox
- Switch

Color

Optional A value or a constant that specifies the color of the border line of the selected object for the flashing status "on".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

BorderFlashingEnabled

Description

Specifies whether the border line of the selected object will flash in Runtime.

Access in Runtime: Read and write

Syntax

Object.**BorderFlashingEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the formats "Ellipse", "Circle", "EllipseSegment", "CircleSegment", "Rectangle", "Polygon", "TextField", "IOField", "SymbolicIOField", "Button", "Roundbutton", "Switch", "GraphicView", "GraphicIOField", "Bar", "Checkbox", "OptionGroup" or "WindowSlider".

BOOLEAN

Optional. TRUE, if the border line of the object is flashing.

See also

Ellipse (Page 5065)

Circle (Page 5048)

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

Rectangle (Page 5149)

Polygon (Page 5134)

TextField (Page 5184)

IOField (Page 5098)

SymbolicIOField (Page 5170)

Button (Page 5040)

Switch (Page 5166)

GraphicView (Page 5091)

GraphicIOField (Page 5088)

Bar (Page 5033)

CheckBox (Page 5045)

OptionGroup (Page 5131)

WindowSlider (Page 5216)

RoundButton (Page 5151)

BorderFlashingRate

Description

Specifies the flash rate of the border line for the selected object.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read/Write

Syntax

Object.**BorderFlashingRate**[=FlashingRate]

Object

Required. An object of the "ScreenItem" type with the format:

- Bar
- Button
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- CheckBox
- Switch

FlashingRate

hmiFlashingRateSlow (0): The length of the flashing interval is 250 ms.

hmiFlashingRateMedium (1): The length of the flashing interval is 500 ms.

hmiFlashingRateFast (2): The length of the flashing interval is 1000 ms.

BorderInnerStyle3D

Description

Specifies the display of the internal part of the object border.

Access in Runtime: Read and write

Syntax

Object.**BorderInnerStyle3D**[=SliderBorder3DStyle]

Object

Required A "ScreenItem" object with following format:

- Gauge
- Slider

SliderBorder3DStyle

hmiGaugeBorder3DStyleNone (0): There is no inside part of the object border.

hmiGaugeBorder3DStyleRecessed (1): The object border is shown in relief.

hmiGaugeBorder3DStyleRaised (2): The object border is shown raised.

hmiGaugeBorder3DStyleGray (3): The object border has a uniform gray border.

hmiGaugeBorder3DStyleColored (4): The object border has a uniform colored border. The border color corresponds to the background color.

BorderInnerWidth3D

Description

Specifies the width of the inner border for the 3D presentation of the selected object.

Access in runtime: Read and write

Syntax

Object.**BorderInnerWidth3D**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Slider

Int32

Optional A value or a constant that specifies the width of the inner border in pixels.

BorderOuterStyle3D

Description

Specifies the display of the external part of the object border.

Access in Runtime: Read and write

Syntax

Object.**BorderOuterStyle3D**[=GaugeBorder3DStyle]

Object

Required A "ScreenItem" object with following format:

- Gauge
- Slider

GaugeBorder3DStyle

hmiGaugeBorder3DStyleNone (0): There is no inside part of the object border.

hmiGaugeBorder3DStyleRecessed (1): The object border is shown in relief.

hmiGaugeBorder3DStyleRaised (2): The object border is shown raised.

hmiGaugeBorder3DStyleGray (3): The object border has a uniform gray border.

hmiGaugeBorder3DStyleColored (4): The object border has a uniform colored border. The border color corresponds to the background color.

BorderOuterWidth3D

Description

Specifies the width of the outer border for the 3D presentation of the selected object.

Access in runtime: Read and write

Syntax

Object.**BorderOuterWidth3D**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Slider

Int32

Optional A value or a constant that specifies the width of the outer border in pixels.

BorderShadeColor3D

Description

Specifies the frame color of the following frame parts in case of a 3D display of the specified object:

- Internal frame parts at the top and bottom
- External frame parts at the bottom and right

Access in runtime: Read and write

Syntax

Object.**BorderShadeColor3D**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Button
- RoundButton
- Slider

You have no access in runtime with the following format:

- Switch

Color

Optional A value or a constant that specifies the color of the shading.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

BorderStyle

Description

Specifies the type of border lines for the specified object.

Access in runtime: Read and write

Syntax

Object.**BorderStyle**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar
- Button
- CheckBox
- ComboBox
- GraphicIOField
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField
- WindowsSlider

Int32

Optional A value or a constant that specifies the type of border lines for the specified object.

0 = filled line

1 = broken line

2 = dotted line

3 = dashed and dotted line

4 = dash - dot - dot line

BorderStyle3D

Description

Determines whether the given object has 3D border shading.

Access in runtime:

- RT Advanced: Read and write
- RT Professional: No access

Syntax

Object.**BorderStyle3D**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- DateTimeField
- IOField
- Switch
- TextField

You have no access in runtime with the following format:

- Bar
- Button
- GraphicIOField
- GraphicView
- SymbolicIOField

BOOLEAN

Optional TRUE if the object has a 3D border shadow.

BorderWidth

Description

Specifies the line weight of the selected object.

Access in runtime: Read and write

Syntax

Object.**BorderWidth**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- AlarmView
- Bar

- Button
- CheckBox
- Circle
- CircleSegment
- ComboBox
- Ellipse
- EllipseSegment
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- IOField
- ListBox
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- Slider
- SymbolicIOField
- TextField
- TrendRulerControl
- UserArchiveControl
- WindowsSlider

You have no access in runtime with the following format:

- Clock
- DateTimeField
- RecipeView
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

Int32

Optional A value or a constant that specifies the line weight in pixels.

BorderWidth3D

Description

Specifies the width of the inner border for the 3D presentation of the selected object.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderWidth3D**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- Button
- Gauge
- RoundButton

You have no access in runtime with the following format:

- Bar
- IOField
- Switch

Int32

Optional A value or constant that determines the width of the inside border for 3D display in pixels.

BottomMargin

Description

No access in runtime.

BusyText

Description

No access in runtime.

ButtonBackColor

Description

No access in runtime.

ButtonBackFillStyle

Description

No access in runtime.

ButtonBarElements

Description

No access in runtime.

ButtonBarStyle

Description

No access in runtime.

ButtonBorderBackColor

Description

No access in runtime.

ButtonBorderColor

Description

No access in runtime.

ButtonBorderWidth

Description

No access in runtime.

ButtonCornerRadius

Description

No access in runtime.

ButtonEdgeStyle

Description

No access in runtime.

ButtonFirstGradientColor

Description

No access in runtime.

ButtonFirstGradientOffset

Description

No access in runtime.

ButtonMiddleGradientColor

Description

No access in runtime.

ButtonSecondGradientColor

Description

No access in runtime.

ButtonSecondGradientOffset

Description

No access in runtime.

BV_ColumnWidth_Date

Description

No access in runtime.

BV_ColumnWidth_Event

Description

No access in runtime.

BV_ColumnWidth_EventSeverity

Description

No access in runtime.

BV_ColumnWidth_EventState

Description

No access in runtime.

BV_ColumnWidth_Number

Description

No access in runtime.

BV_ColumnWidth_Time

Description

No access in runtime.

BV_ItemText_Date

Description

No access in runtime.

BV_ItemText_Event

Description

No access in runtime.

BV_ItemText_EventSeverity

Description

No access in runtime.

BV_ItemText_EventState

Description

No access in runtime.

BV_ItemText_Number

Description

No access in runtime.

BV_ItemText_Time

Description

No access in runtime.

BV_ShowItem_Date

Description

No access in runtime.

BV_ShowItem_Event

Description

No access in runtime.

BV_ShowItem_EventSeverity

Description

No access in runtime.

BV_ShowItem_EventState

Description

No access in runtime.

BV_ShowItem_Number

Description

No access in runtime.

BV_ShowItem_Time

Description

No access in runtime.

Properties C

Caption

Description

Specifies the text that will be displayed in the title line of the selected object.

Access in runtime: Read and write

Syntax

Object.**Caption**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl

- OnlineTrendControl
- TrendRulerControl
- Slider
- UserArchiveControl

STRING

Optional A value or a constant that contains the text that will be shown in the title line.

CaptionBackColor

Description

Specifies the background color of the title line for the selected object.

Access in runtime: Read and write

Syntax

Object.**CaptionBackColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- SymbolicIOField

Color

Optional A value or a constant that specifies the background color of the title line for the selected object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0-255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

CaptionColor

Description

Specifies the color of the text that will be displayed in the title line of the selected object.

Access in runtime: Read and write

Syntax

Object.**CaptionColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Gauge
- Switch
- SymbolicIOField

Color

Optional A value or a constant that specifies the text color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

CaptionText

Description

Specifies the text that will be displayed in the title line of the selected object.

Access in Runtime: Read and write

Syntax

Object.**CaptionText**[= STRING]

Object

Required. An object of the "ScreenItem" type with the format:

- Gauge
- ScreenWindow
- Switch

STRING

Optional A value or a constant that contains the text that will be shown in the title line.

CaptionTop

Description

Specifies the distance of the instrument labels to the upper edge of the selected object. You can position the instrument labels only in line with the vertical diameter of the scale. The value of the attribute refers to the height of the specified object. The height specifies the upper edge of the specified object and the lower edge of the lettering.

Access in runtime: Read and write

Syntax

Object.**CaptionTop**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the distance of the instrument labels to the upper edge of the selected object.

Value range from 0 to 1

0: The lower edge of the lettering is positioned on the upper limit of the selected object. The text is no longer visible as it is positioned outside of the selected object.

1: The lower edge of the lettering is positioned on the lower limit of the selected object.

CellCut

Description

Specifies whether the contents of the cells are abbreviated if the cells are too narrow.

Access in runtime: Read and write

Syntax

Object.**CellCut**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE, if the contents are abbreviated.

CellSpaceBottom

Description

Defines the bottom margin of the table cells.

Access in runtime: Read and write

Syntax

Object.**CellSpaceBottom**[=Int32]

Object

Required. A "ScreenItem" object with the following formats:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchivControl

Int32

Optional A value or constant that defines the bottom margin used in the table cells.

CellSpaceLeft

Description

Specifies the left indent used in the table cells.

Access in runtime: Read and write

Syntax

Object.**CellSpaceLeft**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or parameter that defines the left indent in the table cells.

CellSpaceRight

Description

Specifies the right indent used in the table cells.

Access in runtime: Read and write

Syntax

Object.**CellSpaceRight**[=Int32]

Object

Required. A "ScreenItem" object with the following formats:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or constant that defines the right indent used in the table cells.

CellSpaceTop

Description

Defines the top margin of the table cells.

Access in runtime: Read and write

Syntax

Object.**CellSpaceTop**[=Int32]

Object

Required. A "ScreenItem" object with the following formats:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or constant that defines the top margin used in the table cells.

CenterColor

Description

Specifies the color of the center of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**CenterColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

Color

Optional A value or a constant that specifies the color of the center.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

CenterSize

Description

Specifies the diameter of the scale center point.

Access in runtime: Read and write

Syntax

Object.**CenterSize**[=SINGLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

SINGLE

Optional A value or a constant that specifies the diameter of the round scale center point.

Value range from 0.03 to 1

1: The diameter corresponds to the smaller value of the geometry attribute "Width" or "Height".

ChangeMouseCursor

Description

Specifies how the appearance of the cursor changes in Runtime when it is above the icon.

Access in runtime: Read and write

Syntax

Object.**ChangeMouseCursor**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- SymbolLibrary

BOOLEAN

Optional TRUE, if the mouse cursor has the appearance of an arrow even when it is positioned above the icon.

FALSE, if the mouse cursor has the appearance of a 3D arrow with a green lightning symbol. This indicates in Runtime that the respective objective can be operated.

CheckMarkAlignment

Description

Specifies whether the fields are right aligned.

Access in runtime: Read and write

Syntax

Object.**CheckMarkAlignment**[=ContentAlignment]

Object

Required. An object of the "ScreenItem" type with the format:

- OptionGroup

You have no access in runtime with the following format:

- CheckBox

ContentAlignment

(0): The fields are arranged left aligned.

(1): The fields are arranged right aligned.

CheckMarkCount

Description

Specifies the number of fields.

Access in runtime: Read and write

Syntax

Object.**CheckMarkCount**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- OptionGroup

You have no access in runtime with the following format:

- CheckBox

Int32

Optional A value or a constant that specifies the number of fields. Value range from 0 to 31

ClearOnError

Description

Specifies whether an invalid input in this object will be deleted automatically.

Access in runtime: Read and write

Syntax

Object.**ClearOnError**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField

BOOLEAN

Optional TRUE, if an invalid input in this object will be deleted automatically.

ClearOnFocus

Description

Specifies whether the field entry will be deleted as soon as the I/O field is activated.

Access in runtime: Read and write

Syntax

Object.**ClearOnFocus**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField

BOOLEAN

Optional TRUE, if the field entry is deleted as soon as the I/O field is activated.

Closeable

Description

Specifies whether the window can be closed in Runtime.

Access in Runtime: Read and write

Syntax

Object.**Closeable**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format "UserArchiveControl".

BOOLEAN

Optional. TRUE, if the window can be closed in Runtime.

See also

AlarmControl (Page 5018)

UserArchiveControl (Page 5207)

SystemDiagnoseWindow (Page 5180)

TrendRulerControl (Page 5187)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

OnlineTrendControl (Page 5122)

Color

Description

Specifies the line color of the specified object.

Access in runtime: Read and write

Syntax

Object.**Color**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- CircularArc
- EllipticalArc
- Line
- Polyline
- TubeArcObject
- TubeDoubleTeeObject
- Tubepolyline
- TubeTeeObject

Color

Optional A value or a constant that specifies the line color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

ColorChangeHysteresis

Description

Specifies hysteresis as a percentage of the display value.

The "ColorChangeHysteresisEnable" property must have the value TRUE so that the hysteresis can be calculated.

Access in runtime: Read and write

Syntax

Object.**ColorChangeHysteresis**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the hysteresis as a percentage of the display value.

ColorChangeHysteresisEnabled

Description

Determines whether the object will be displayed with a hysteresis.

Access in runtime: Read and write

Syntax

Object.**ColorChangeHysteresisEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the object will be displayed with a hysteresis.

ColumnAlignment

Description

Specifies how the selected column is aligned.

The following settings are possible:

Value	Description	Explanation
0	Left	The column selected is displayed left-justified.
1	Centered	The column selected is displayed centered.
2	Right	The column selected is displayed right-justified.

The attribute can be assigned dynamic properties by means of the name **ColumnAlignment**. The data type is LONG.

ColumnDateFormat

Description

Defines the date format to be used for visualization. The "ColumnIndex" property is used to reference the date column of the recipe view. The attribute can be assigned dynamic properties by means of the name **ColumnDateFormat**.

Access in Runtime: Write

Syntax

Object.**ColumnDateFormat**(=String)

Object

Required An object of the "ScreenItem" type with the characteristic "UserArchiveControl".

String

The following date formats are available:

Value	Explanation
dd.MM.yy	Day.Month.Year, e.g., 24.12.13.
dd.yyyyd.MM	Day.Month.Year, e.g., 24.12.2013.
dd/MM/yy	Day/Month/Year, e.g., 24/12/13.
dd/MM/yyyy	Day/Month/Year, e.g., 24/12/2013.

See also

UserArchiveControl (Page 5207)

ColumnResize

Description

Enables changes to the width of columns.

Access in runtime: Read and write

Syntax

Object.**ColumnResize**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: You can change the width of the columns.

FALSE: You cannot change the width of the columns.

Columns

Description

No access in runtime.

ColumnScrollbar

Description

Enables the display of column scroll bars.

Access in Runtime: Read

Syntax

Object.**ColumnScrollbar**[=Scrollbar]

Object

Required An object of the "ScreenItem" type with the format "AlarmControl", "TrendRulerControl", "OnlineTableControl", "UserArchivControl".

Scrollbar

0 = (No) The column scrollbars are not displayed.

1 = (if required) The column scrollbars are displayed if the space requirement of the control is greater in vertical direction than the available display area.

2 = (always) The column scrollbars are always displayed.

See also

UserArchiveControl (Page 5207)

TrendRulerControl (Page 5187)

OnlineTableControl (Page 5114)

AlarmControl (Page 5018)

ColumnSettings

Description

No access in runtime.

ColumnSettingsBufferView

Description

No access in runtime.

ColumnsMoveable

Description

No access in runtime.

ColumnTextAckGroup

Description

No access in runtime.

ColumnTextAlarmState

Description

No access in runtime.

ColumnTextAlarmText

Description

No access in runtime.

ColumnTextClassName

Description

No access in runtime.

ColumnTextDate

Description

No access in runtime.

ColumnTextDevice

Description

No access in runtime.

ColumnTextDiagnosable

Description

No access in runtime.

ColumnTextNumber

Description

No access in runtime.

ColumnTextTime

Description

No access in runtime.

ColumnTimeFormat

Description

Defines the time format to be used for visualization. The "ColumnIndex" property is used to reference the time column of the recipe view. The attribute can be assigned dynamic properties by means of the name **ColumnTimeFormat**.

Access in Runtime: Write

Syntax

Object.**ColumnTimeFormat**(=String)

Object

Required. An object of the "ScreenItem" type with the characteristic "UserArchiveControl".

String

The following time formats are available:

Value	Explanation
HH:mm:ss.ms	Hours:Minutes:Seconds, e.g. 15:35:44.240.
hh:mm:ss tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.
hh:mm:ss.ms tt	Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.

ColumnTitleAlignment

Description

Specifies the type of column title alignment.

Access in Runtime: Read

Syntax

Object.**ColumnTitleAlignment**[=GridColumnHeaderHorizontalAlignment]

Object

Required. An object of the "ScreenItem" type with the format "AlarmControl", "TrendRulerControl", "OnlineTableControl", "UserArchivControl".

GridColumnHeaderHorizontalAlignment

0 = The column headers are shown left aligned.

1 = The column headers are shown centered.

2 = The column headers are shown right aligned.

3 = The column headers are aligned as the corresponding content.

See also

AlarmControl (Page 5018)

ColumnTitles

Description

Enables the display of the column header.

Access in runtime: Read and write

Syntax

Object.**ColumnTitles**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchivControl

BOOLEAN

TRUE: The column header is displayed.

FALSEThe column header is not displayed.

ComboboxFont

Description

No access in runtime.

Dublette_CompatibilityMode

Description

No access in runtime.

ComputerName

Description

Returns the name of the computer on which the alarm object was triggered.

ComputerName (readonly)

ConnectionType

Description

No access in runtime

ConnectOnStart

Description

No access in runtime.

ConnectTrendWindows

Description

Specifies whether the configured trend views are connected. The requirement is that you have configured several trend views.

The connected trend views have the following properties:

- A common X axis
- A scrollbar
- A ruler

Access in runtime: Read and write

Syntax

Object.**ConnectTrendWindows**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: All configured trend views are connected.

FALSE: The trend views are displayed separately.

Context

Description

Reads or sets the alarm object server prefix.

ContinousChange

Description

Specifies whether the value of the "ProcessValue" property will be transferred when the mouse button is released or as soon as the slider position changes in Runtime.

Access in runtime: Read and write

Syntax

Object.**ContinousChange**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Slider

BOOLEAN

Optional TRUE, if the value of the property "ProcessValue" will be transferred when the mouse button is released or as soon as the slider position changes in Runtime.

CornerRadius

Description

No access in runtime.

CornerStyle

Description

Specifies the type of border lines for the specified object.

Access in Runtime: Read-only

Syntax

Object.**CornerStyle**[=Int]

Object

Required. An object of the "ScreenItem" type with the format:

- Bar
- Button
- CheckBox
- Circle

- CircleSegment
- CircularArc
- ComboBox
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- Tubepolyline
- WindowsSlider

You have no access in runtime with the following format:

- Ellipse
- EllipticalArc
- Switch
- TubeArcObject

Int

Optional A value or a constant that specifies the type of border lines for the specified object.

0 = filled line

1 = broken line

2 = dotted line

3 = dashed and dotted line

4 = dash - dot - dot line

Count

Description

Returns the number of elements in the specified list.

Access in Runtime: Read

Syntax

Object.Count

Object

Required A "Collection" object.

CountDivisions

Description

Specifies the number of segments in which the bar will be divided by the large marking lengths of the scale.

Access in runtime: Read and write

Syntax

Object.**CountDivisions**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Int32

Optional A value or a constant that specifies the number of segments in which the bar will be divided by the large marking lengths of the scale.

0-100: An object can be divided into a maximum of 100 segments

= 0: The optimum number of segments will be established automatically.

See also

Bar (Page 5033)

CountOfLinesPerAlarms

Description

No access in runtime.

CountOfVisibleAlarms

Description

No access in runtime.

CountSubDivisions

Description

No access in runtime.

CountVisibleItems

Description

Specifies how many lines the selection list will contain. If the number of configured texts is greater than this value, then the selection list will have a vertical scroll bar.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**CountVisibleItems**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- ComboBox
- ListBox
- SymbolicIOField

You have no access in runtime with the following format:

- StatusForce
- TrendView
- UserView

Int32

Optional A value or a constant that specifies how many lines the selection list will contain.

CurrentContext

Description

Returns a character string depending on the use of the function.

If the function is contained in a screen in the screen window, CurrentContext returns the symbolic server name which supplies the screen. Example: "WinCCProject_MyComputer::"

If the function is contained in the main screen, an empty character string is returned.

Access in Runtime: Read

Syntax

Object.**CurrentContext**

Object

Required A "HMIRuntime" object.

See also

HMIRuntime (Page 4993)

CursorControl

Description

Specifies whether the mouse cursor jumps to the next field of the TAB sequence after leaving the field.

Access in runtime: Read and write

Syntax

Object.**CursorControl**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField
- SymbolicIOField

BOOLEAN

Optional TRUE, if the mouse cursor jumps to the next field of the TAB-order after leaving the field.

Comments

For this purpose, the "CursorMode" property must be set to TRUE.

Curves

Description

No access in runtime.

Properties D

DangerRangeColor

Description

Specifies the color of the warning range of the scale of the "Gauge" object.

The "DangerRangeVisible" property must have the value TRUE so that the warning range is displayed.

Access in runtime: Read and write

Syntax

Object.**DangerRangeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

Color

Optional A value or a constant that specifies the color of the warning range.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

DangerRangeStart

Description

Specifies at which scale value the warning range of the "Gauge" object will start.

The "DangerRangeColor" property must have the value TRUE so that the warning range is displayed.

Access in runtime: Read and write

Syntax

Object.**DangerRangeStart**[=SINGLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

SINGLE

Optional A value or a constant that specifies the scale value for the start of the warning range.

Comments

The range extends from the value "Gefahr" through to the end of the scale.

DangerRangeVisible

Description

Specifies whether the warning range in the scale of the "Gauge" object will be displayed.

Access in runtime: Read and write

Syntax

Object.**DangerRangeVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

BOOLEAN

Optional TRUE, if the warning range will be displayed in the scale.

Comments

Specifies the color of the warning range with the "DangerRangeColor" property.

Specifies the start of the warning range with the "DangerRangeStart" property.

DataFormat

Description

Returns the data type of the I/O field object.

Access in runtime: Read

Syntax

Object.**DataFormat**[=IOFieldDataFormat]

Object

Required. An object of the "ScreenItem" type with the following format:

- IOField

IOFieldDataFormat

Optional. A value or a constant that returns the data type of the I/O field object.

Value range from 0 to 3.

- 0: Binary
- 1: Decimal
- 2: String
- 3: Hexadecimal

DataLogs

Description

Returns an object of type "DataLogs".

Access in Runtime: Read

Syntax

Object.**DataLogs**

Object

Required A "Logging" object.

See also

Logging (Page 4998)

DataRecordNameCaption

Description

No access in runtime.

DataRecordNrCaption

Description

No access in runtime.

DataSet

Description

Returns an object of type "DataSet".

Access in Runtime: Read

Syntax

Object.**DataSet**

Object

Required A "Screen" object.

See also

HMIRuntime (Page 4993)

Screen (Page 5000)

DialColor

Description

Specifies the color of the dial for the selected object.

Access in runtime: Read and write

Syntax

Object.**DialColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Clock
- Gauge

Color

Optional A value or a constant that specifies the color of the dial.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

DialFillStyle

Description

Specifies the type of background of the selected object.

Access in Runtime: Read and write

Syntax

Object.**DialFillStyle**[=GaugeFillStyle]

Object

Required A "ScreenItem" object with following format:

- Gauge

GaugeFillStyle

hmiGaugeBackStyleSolid (0): The rectangular background of the display is filled with the specified border color. The scale is filled with the specified background color.

hmiGaugeBackStyleFrameTransparent (1): The rectangular background of the Gauge is transparent. The scale is filled with the specified background color. As a result, a circular display can be shown.

hmiGaugeBackStyleTransparent (2): The rectangular background and the scale are transparent.

DialSize

Description

Specifies the diameter of the scale in relation to the smaller value of the geometry attributes "Width" and "Height".

Access in runtime: Read and write

Syntax

Object.**DialSize**[=SINGLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

SINGLE

Optional A value or a constant that specifies the diameter of the scale in relation to the smaller value of the geometry attributes "Width" and "Height".

DisplayButton2Plc

Description

No access in runtime.

DisplayButtonComparison

Description

No access in runtime.

DisplayButtonDelete

Description

No access in runtime.

DisplayButtonFromPlc

Description

No access in runtime.

DisplayButtonHelp

Description

No access in runtime.

DisplayButtonNew

Description

No access in runtime.

DisplayButtonSave

Description

No access in runtime.

DisplayButtonSaveAs

Description

No access in runtime.

DisplayComboBox

Description

No access in runtime.

DisplayGridLines

Description

No access in runtime.

DisplayLabeling

Description

No access in runtime.

DisplayNumbers

Description

No access in runtime.

DisplayOptions

Description

Specifies whether alarms whose view was suppressed are displayed.

Access in Runtime: Read and write

Syntax

Object.**DisplayOptions**[=AlarmDisplayOptions]

Object

Required. An object of the "ScreenItem" type with the format:

- AlarmControl

AlarmDisplayOptions

0: All messages

1: Only displayed messages

2: Only hidden messages

DisplayStatusBar

Description

No access in runtime.

DisplayTable

Description

No access in runtime.

DoubleClickAction

Description

Specifies the action to be executed in Runtime by double-clicking on a message line.

Access in Runtime: Read and write

Syntax

Object.**DoubleClickAction**[=AlarmControlActions]

Object

Required A "ScreenItem" object with following format:

- AlarmControl

AlarmControlActions

Value	Description	Explanation
0	None	No action.
1	Loop-in-alarm	Calls the "Loop-in-alarm" function.
2	Open comments dialog	Calls the "Comments dialog" button function.
3	Open Infotext dialog	Calls the "Infotext dialog" button function.
4	Column-dependent	The action is determined by the column in which you double-clicked.

DrawInsideFrame

Description

Specifies whether the border line of the selected object should be illustrated with a line weight greater than 1 within the border or symmetrically to the border.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**DrawInsideFrame**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- Circle
- CircleSegment
- CircularArc
- Ellipse
- EllipseSegment

- EllipticalArc
- GraphicIOField
- GraphicView
- OptionGroup
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- CheckBox
- Switch
- TubeArcObject

BOOLEAN

Optional TRUE, if the border line of the specified object will be illustrated with a line weight greater than 1 within the border.

Drive

Description

Specifies the characters of the drive that is to be monitored.

Access in runtime: Read and write

Syntax

Object.**Drive**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

STRING

Optional A value or a constant that specifies the characters of the drive that is to be monitored.

Properties E-F

EdgeStyle

Description

Specifies the line style of the selected object.

Access in runtime: Read and write

Syntax

Object.**EdgeStyle**[=LineStyle]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- ComboBox
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- AlarmView
- Clock

- DateTimeField
- Gauge
- RecipeView
- Slider
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

LineStyle

Optional A value or a constant that specifies the line style. Value range from -1 to 4.

The objects "Ellipse", "Circle", "Rectangle" and "Polygon" support the line styles:

- hmiLineStyleSolid (0): solid line
- hmiLineStyleDash (1): broken line
- hmiLineStyleDot (2): dotted line
- hmiLineStyleDashDot (3): dash - dot line
- hmiLineStyleDashDotDot (4): dash - dot - dot line

The objects "TextField" and "IOField" support only the line styles:

- hmiLineStyleNone (-1): no line
- hmiLineStyleSolid (0): solid line

Default setting: hmiLineStyleSolid

EditOnFocus

Description

Specifies whether the data input is immediately possible if the input field is selected using the <Tab> key.

Access in runtime: Read and write

Syntax

Object.**EditOnFocus**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField
- SymbolicIOField

BOOLEAN

Optional TRUE, if the data input is immediately possible and the input field has been selected using the <Tab> key.

Enabled

Description

Specifies whether the selected object can be operated in Runtime.

Access in runtime: Read and write

Syntax

Object.**Enabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- AlarmView
- Bar
- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- DateTimeField
- DiscSpaceView
- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- HTML-Browser

- IOField
- Line
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- Polygon
- Polyline
- RecipeView
- Rectangle
- RoundButton
- S7GraphOverview
- Slider
- SmartClientView
- StatusForce
- Switch
- SymbolicIOField
- SymbolLibrary
- SysDiagControl
- TextField
- TrendRulerControl
- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- Tubepolyline
- TubeTeeObject
- UserArchiveControl
- UserView
- WindowsSlider

BOOLEAN

Optional TRUE, if the specified object can be operated in Runtime.

EnableEdit

Description

Enables editing of the data displayed during runtime.

Access in runtime: Read and write

Syntax

Object.**EnableEdit**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl
- UserArchiveControl

BOOLEAN

TRUE: The data can be changed in Runtime.

FALSE: The data cannot be changed.

EnableNavigateKeys

Description

No access in runtime.

EndAngle

Description

Specifies the angle at which the end point of the selected object deviates from the zero position (0°).

Access in runtime: Read and write

Syntax

Object.**EndAngle**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- CircleSegment
- CircularArc
- EllipseSegment

- EllipticalArc
- TubeArcObject

Int32

Optional A value or a constant that specifies the angle at which the end point of the selected object deviates from the zero position (0°).

EndLeft

Description

No access in runtime.

EndStyle

Description

Specifies how the line-end shapes of the selected object will be displayed.

Access in runtime: Read and write

Syntax

Object.**EndStyle**[=LineEndStyle]

Object

Required. A "ScreenItem" object with the following format:

- Line
- Polyline

LineEndStyle

Optional A value or a constant that specifies the line-end shapes. Value range from 0 to 6.

hmiLineEndStyleNone (0): The line has no end symbol.

hmiLineEndStyleArrow (1): The line ends with an arrow.

hmiLineEndStyleFilledArrow (2): The line ends with a filled arrowhead.

hmiLineEndStyleFilledArrowReversed (3): The line ends with a reversed arrow.

hmiLineEndStyleLine (4): The line ends with a vertical line.

hmiLineEndStyleCircle (5): The line ends with a circle.

hmiLineEndStyleFilledCircle (6): The line ends with a filled circle.

EndTop

Description

No access in runtime.

EntryNameCaption

Description

No access in runtime.

ErrorDescription

Description

Returns one of the following error descriptions in English:

Output	Description
" "	OK
"Operation Failed"	Execution error
"Variable not found"	Tag error
"Server down"	Server not available.
"An error occured for one or several tags"	Multi Tag Error (Error in one or several tags)

To obtain an error description, first of all carry out the Read method.

Note

If the error occurs when accessing via the TagSet object, evaluate the ErrorDescription property for each tag of the TagSet object.

Access in Runtime: Read

Syntax

Object.**ErrorDescription**

Object

Required A "Tag" object.

Example

The following example shows the error description for the "Tag1" tag:

```
'VBS72
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objtag.Read
MsgBox objTag.ErrorDescription
```

The following example adds two tags to the TagSet list and outputs the ErrorDescription property as Trace:

```
'VBS179
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMIRuntime.Trace "ErrorDescription: " & group.ErrorDescription & vbNewLine
```

The ErrorDescription property of a tag contained in the list may be accessed as follows:

```
HMIRuntime.Trace "ErrorDescription: " & group("Motor1").ErrorDescription & vbNewLine
```

See also

Tag (Page 5012)

TagSet (list) (Page 5016)

EvenRowBackColor

Description

No access in runtime.

ExportDelimiter

Description

No access in runtime.

ExportDirectoryChangeable

Description

Enables changing of the directory for data export in Runtime.

Syntax

Object.**ExportDirectoryChangeable**=[BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The directory for data export can be changed in runtime.

FALSE: The directory for data export cannot be changed in runtime.

ExportDirectoryname

Description

Defines the directory to which the exported Runtime data is written.

Access in runtime: Read and write

Syntax

Object.**ExportDirectoryname**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant which specifies the directory.

ExportFileExtension

Description

Specifies the file extension of the export file. Only the file extension "csv" is supported.

Access in runtime: Read and write

Syntax

Object.**ExportFileExtension**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional Specifies the file extension of the export file.

ExportFilename

Description

Defines the name of the file which is to receive the exported Runtime data.

Access in runtime: Read and write

Syntax

Object.**ExportFilename**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional Defines the name of the file which is to receive the exported Runtime data.

ExportFilenameChangeable

Description

Enables renaming of the export file in Runtime.

Syntax

Object.**ExportFilenameChangeable**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The file name of the export file can be changed in runtime.

FALSE: The file name of the export file cannot be changed in runtime.

ExportFormatGuid

Description

Specifies the assignment of ID number and export provider.

Access in runtime: Read and write

Syntax

Object.**ExportFormatGuid**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or constant that specifies the assignment of ID number and export provider.

ExportFormatName

Description

Defines the export file format. Only the "csv" file format is currently available for the export.

Access in runtime: Read and write

Syntax

Object.**ExportFormatName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that defines the file format for export.

ExportParameters

Description

Specifies the parameters of the selected format by means of the properties dialog.

Access in runtime: Read and write

Syntax

Object.**ExportParameters**

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

ExportParameters

Optional A value or constant that specifies the parameters of the selected format in the Properties dialog.

ExportSelection

Description

Specifies which runtime data of the control is exported.

Access in Runtime: Read and write

Syntax

Object.**ExportSelection**[=ExportRange]

Object

Required A "ScreenItem" object with following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

- TrendRulerControl
- UserArchiveControl

ExportRange

Value	Description	Explanation
0	all	All Runtime data of the control is exported.
1	Selection	Selected Runtime data of the control is exported.

ExportShowDialog

Description

Enables the display of the export dialog during runtime.

Access in runtime: Read and write

Syntax

Object.**ExportShowDialog**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional TRUE, if the dialog box is shown in Runtime.

ExtendedZoomingEnable

Description

Specifies whether the operator can zoom the screen in or out in runtime by turning the mouse wheel.

Access during runtime: Read and write

Syntax

Object.**ExtendedZoomingEnable**[=BOOLEAN]

Object

Necessary. A "Screen" object.

BOOLEAN

Optional TRUE, if the operator can zoom the screen in and out in runtime.

Example

The following example shows how the extended zoom can be enabled for the NewPDL1 screen:

```
'VBS155
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
objScreen.ExtendedZoomingEnable = 1
```

See also

Screen (Page 5000)

FieldLength

Description

Specifies that the "field length string" field is read-only.

Access in Runtime: Read and write

Syntax

Object.**FieldLength**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format "IOField".

BOOLEAN

Optional. TRUE, if the "field length string" can only be read.

Used for the following object types

IOField

See also

IOField (Page 5098)

FieldLength

Description

No access in runtime.

FillColorMode

Description

Specifies the type of foreground for the selected object.

Access in Runtime: Read and write

Syntax

Object.**FillColorMode**[=SymbolLibraryColorMode]

Object

Required A "ScreenItem" object with following format:

- SymbolLibrary

SymbolLibraryColorMode

hmiSymbolLibraryAppearanceOriginal (0): The surface is gray.

hmiSymbolLibraryAppearanceShaded (1): The display is shaded.

hmiSymbolLibraryAppearanceSolid (2): The display is solid.

hmiSymbolLibraryAppearanceTransparent (3): The display is gray.

FillPatternColor

Description

Specifies the color of the fill pattern for the selected object.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**FillPatternColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- ComboBox
- Ellipse
- EllipseSegment
- GraphicView
- IOField
- ListBox
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- AlarmControl
- DateTimeField
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant that establishes the color of the fill pattern for the specified object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

FillStyle

Description

Determines the fill style of the specified object.

Access in runtime: Read and write

Syntax

Object.**FillStyle**[=LineFillStyle]

Object

Required. A "ScreenItem" object with the following format:

- Line
- Polyline

LineFillStyle

Optional A value or a constant that specifies the fill style.

hmiLineFillStyleTransparent (0): Transparent fill

hmiLineFillStyleSolid (1): Object is filled with the specified color

Default setting: hmiLineFillStyleSolid

FillstyleAlignment

Description

Specifies the alignment within the specified object.

Filter

Description

No access in runtime.

FilterTag

Description

No access in runtime.

FilterText

Description

No access in runtime.

FirstConnectedObjectIndex

Description

Specifies the index number of the upper connector point.

Access in Runtime: Read and write

Syntax

Object.**FirstConnectedObjectIndex**[=Int]

Object

Required A "ScreenItem" object with the format "Connector".

Int

Int

See also

Connector (Page 5058)

FirstConnectedObjectName

Description

Specifies the object name of the object that is docked at the upper connector point.

Access in Runtime: Read and write

Syntax

Object.**FirstConnectedObjectName**[=String]

Object

Required A "ScreenItem" object with the format "Connector".

String

String

See also

Connector (Page 5058)

FitToLargest

Description

No access in runtime.

FitToSize

Description

No access in runtime.

FixedAspectRatio

Description

Specifies whether the aspect ratio should be maintained or changed when the symbol size is altered.

Access in runtime: Read and write

Syntax

Object.**FixedAspectRatio**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- SymbolLibrary

BOOLEAN

Optional TRUE, if the aspect ratio should be maintained when the symbol size is altered.

FlashingColorOff

Description

Specifies the color of the border line for the selected object for the flashing status "off".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**FlashingColorOff**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Button
- Circle
- CircleSegment
- CircularArc
- Ellipse
- EllipseSegment
- EllipticalArc
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField
- TextField

You have no access in runtime with the following format:

- Bar
- CheckBox
- Switch
- TubeArcObject

Color

Optional A value or a constant that specifies the color of the border line of the selected object for the flashing status "off".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

FlashingColorOn

Description

Specifies the color of the border line for the selected object for the flashing status "on".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**FlashingColorOn**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Button
- Circle
- CircleSegment
- CircularArc
- Ellipse
- EllipseSegment
- EllipticalArc
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField
- TextField

You have no access in runtime with the following format:

- Bar
- CheckBox
- Switch
- TubeArcObject

Color

Optional A value or a constant that specifies the color of the border line of the selected object for the flashing status "on".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

FlashingEnabled

Description

Specifies whether the border line of the selected object will flash in Runtime.

Access in Runtime: Read and write

Syntax

Object.**FlashingEnable**d[=BOOLEAN]

Object

Required. A "ScreenItem" object with the formats "Line", "Polyline", "Ellipse", "Circle", "EllipseSegment", "CircleSegment", "EllipticalArc", "CircularArc", "Rectangle", "Polygon", "TextField", "IOField", "SymbolicIOField", "Button", "Switch", "GraphicIOField", "Bar", "Checkbox", "OptionGroup" or "Connector".

BOOLEAN

Optional. TRUE, if the border line of the object is flashing.

See also

Line (Page 5102)

Polyline (Page 5137)

Ellipse (Page 5065)

Circle (Page 5048)

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

EllipticalArc (Page 5070)

CircularArc (Page 5053)

Rectangle (Page 5149)

Polygon (Page 5134)

TextField (Page 5184)
IOField (Page 5098)
SymbolicIOField (Page 5170)
Button (Page 5040)
Switch (Page 5166)
GraphicIOField (Page 5088)
Bar (Page 5033)
CheckBox (Page 5045)
OptionGroup (Page 5131)
Connector (Page 5058)
RoundButton (Page 5151)

FlashingOnLimitViolation

Description

No access in runtime.

FlashingRate

Description

Specifies the flash rate of the border line for the selected object.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read/Write

Syntax

Object.**FlashingRate**[=FlashingRate]

Object

Required. An object of the "ScreenItem" type with the format:

- Button
- Circle
- CircleSegment
- CircularArc
- Ellipse
- EllipseSegment

- EllipticalArc
- GraphicIOField
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField
- TextField

You have no access in runtime with the following format:

- Bar
- CheckBox
- Switch
- TubeArcObject

FlashingRate

hmiFlashingRateSlow (0): The length of the flashing interval is 1000 ms.

hmiFlashingRateMedium (1): The length of the flashing interval is 500 ms.

hmiFlashingRateFast (2): The length of the flashing interval is 250 ms.

FlashTransparentColor

Description

Specifies the color of the bitmap object of a flashing graphic that is set to "transparent".

Access in Runtime: Read and write

Syntax

Object.**FlashTransparentColor**[=Color]

Object

Required A "ScreenItem" object with the format "GraphicIOField".

Color

Optional A value or a constant that specifies the color of the bitmap object of a flashing graphic that is set to "transparent".

See also

GraphicIOField (Page 5088)

Flip

Description

Flips the symbol on the vertical and / or horizontal center axis of the symbol.

Access in Runtime: Read and write

Syntax

Object.**Flip**[=SymbolLibraryFlip]

Object

Required A "ScreenItem" object with following format:

- SymbolLibrary

SymbolLibraryFlip

hmiSymbolLibraryFlipNone (0): The symbol is not flipped.

hmiSymbolLibraryFlipHorizontal (1): The symbol is flipped horizontally.

hmiSymbolLibraryFlipVertical (2): The symbol is flipped vertically.

hmiSymbolLibraryFlipBoth (3): The symbol is flipped horizontally and vertically.

FocusColor

Description

Specifies that the color for the focus border of the selected object is currently active.

Access in runtime: Read and write

Syntax

Object.**FocusColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- AlarmView
- RecipeView
- Slider
- StatusForce

- Switch
- TrendView

You have no access in runtime with the following format:

- Button
- GraphicIOField
- SymbolicIOField

Color

Optional A value or a constant that specifies the color of the focus border.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

FocusWidth

Description

Specifies the border width of the specified object if the object is currently active.

Access in runtime: Read and write

Syntax

Object.**FocusWidth**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- AlarmView
- RecipeView
- Slider
- StatusForce
- Switch
- TrendView

You have no access in runtime with the following format:

- Button
- GraphicIOField
- SymbolicIOField

Int32

Optional A value or a constant that specifies the border width in pixels. Value range from 1 to 10.

Font

Description

Specifies or returns the font.

The font object has the following sub-properties

- Size (Font Size)
- Bold (yes/no)
- Name (font name)
- Italic (yes/no)
- Underline (underline yes/no)
- StrikeThrough (yes/no)

If two font properties are directly assigned, only the default property "Name" is assumed.

Access during runtime: Read and write

Syntax

Object.**Font**[=Font]

Object

Required. An object of the "ScreenItem" type with the format:

AlarmControl

Clock

FunctionTrendControl

OnlineTableControl

OnlineTrendControl

Slider

TrendRulerControl

UserArchiveControl

You have no access in runtime with the following format:

- Bar
- Button
- CheckBox
- ComboBox

- DateTimeField
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- ProtectedAreaNameView
- RangeLabelView
- RecipeView
- RoundButton
- SmartClientView
- Switch
- SymbolicIOField
- TextField
- TrendView
- ZoneLabelView

Font

Font

Example

```
'VBS74
Dim objControl1
Dim objControl2
Set objControl1 = ScreenItems("Control1")
Set objControl2 = ScreenItems("Control2")
objControl2.Font = objControl1.Font ' take over only the type of font
```

FontBold

Description

Specifies whether the text of the selected object should be shown in bold.

Access in runtime: Read and write

Syntax

Object.**FontBold**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- TextField
- Bar
- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField

BOOLEAN

Optional TRUE, if the text of the selected object should be shown in bold.

FontItalic

Description

Specifies whether the text of the selected object should be shown in italic.

Access in runtime: Read and write

Syntax

Object.**FontItalic**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- TextField
- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup

- RoundButton
- SymbolicIOField

BOOLEAN

Optional TRUE, if the text of the selected object should be shown in italic.

FontName

Description

Specifies the font of the selected object.

Access in runtime: Read and write

Syntax

Object.**FontName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- TextField
- Bar
- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField

STRING

Optional A value or a constant that specifies the font of the selected object.

FontSize

Description

Specifies the font size of the text for the selected object.

Access in runtime: Read and write

Syntax

Object.**FontSize**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- TextField
- Bar
- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField

Int32

Optional A value or a constant that specifies the font size of the text for the selected object.

FontUnderline

Description

Specifies whether the text of the selected object should be underlined.

Access in runtime: Read and write

Syntax

Object.**FontUnderline**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- TextField
- Button
- CheckBox
- ComboBox
- IOField
- ListBox

- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField

BOOLEAN

Optional TRUE, if the text of the selected object should be shown underlined.

ForeColor

Description

Specifies the font color of the text for the selected object.

Access in runtime: Read and write

Syntax

Object.**ForeColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- CheckBox
- ComboBox
- DateTimeField
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RecipeView
- RoundButton
- Slider
- Switch
- SymbolLibrary
- SymbolicIOField
- TextField

You have no access in runtime with the following format:

- AlarmView

Color

Optional A value or a constant that specifies the font color of the text for the selected object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

ForeColorTransparency

Description

No access in runtime.

FormatPattern

Description

Specifies the format of the output value.

Access in runtime: Read

Syntax

Object.**FormatPattern**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- IOField

STRING

Optional A value or a constant that specifies the format of the output value.

Free

Description

Returns the size of the free disk space.

Access in runtime: Read and write

Syntax

Object.**Free**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

DOUBLE

Optional A value or a constant that returns the size of the free disk space.

FreePercent

Description

Returns the measured values for the free disk space as a percentage. The values can be queried in Runtime. The values cannot be predefined.

Access in runtime: Read and write

Syntax

Object.**FreePercent**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

Int32

Optional A value or a constant that returns the measured values for the free disk space as a percentage.

Properties G-H

Gradation

Description

Specifies the value difference between two main marking lengths of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**Gradation**[=SINGLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

SINGLE

Optional A value or a constant that specifies the value difference.

GraphDirection

Description

Specifies at which border of the trend view the current values are displayed.

Access in Runtime: Read and write

Syntax

Object.**GraphDirection**[=GraphDirection]

Object

Required A "ScreenItem" object with following format:

- FunctionTrendControl
- OnlineTrendControl

GraphDirection

0: Positive values run to the right and upwards.

-1: Positive values run to the left and upwards.

-2: Positive values run to the right and upwards.

-3: Positive values run to the right and downwards.

GridLineColor

Description

Specifies the color for the grid lines.

Access in runtime: Read and write

Syntax

Object.**GridLineColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- AlarmView
- OnlineTableControl

- TrendRulerControl
- UserArchiveControl

You have no access in runtime with the following format:

- TrendView

Color

Optional A value or a constant that specifies the color of the grid lines.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

GridlineFillColor

Description

No access in runtime.

GridlineStyle

Description

No access in runtime.

GridLineWidth

Description

Defines the line weight of the row/column dividers in pixels.

Access in runtime: Read and write

Syntax

Object.**GridLineWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl

- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or a constant which specifies the width of the grid line.

HeaderFont

Description

No access in runtime.

Height

Description

Specifies the height of the specified object.

Access in runtime: Read and write

Syntax

Object.**Height**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- AlarmView
- ApplicationWindow
- Bar
- BatteryView
- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- DateTimeField
- DiscSpaceView

- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- HTML-Browser
- IOField
- Line
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- Polygon
- Polyline
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- Rectangle
- RoundButton
- Screenwindow
- Slider
- SmartClientView
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TextField
- TrendRulerControl

- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserArchiveControl
- UserView
- WLanQualityView
- WindowsSlider
- ZoneLabelView
- ZoneQualityView

You have no access in runtime with the following format:

- S7GraphOverview

Int32

Optional A value or a constant that specifies the height in pixels.

Example

The following example halves the height of all objects of the image "NewPDL1", whose name starts with "Circle":

```
'VBS75
Dim objScreen
Dim objCircle
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
'
'Searching all circles
strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
If "Circle" = Left(strName, 6) Then
'
'to halve the height of the circles
Set objCircle = objScreen.ScreenItems(strName)
objCircle.Height = objCircle.Height / 2
End If
Next
```

See also

Line (Page 5102)
Polyline (Page 5137)
Ellipse (Page 5065)
Circle (Page 5048)
EllipseSegment (Page 5067)
CircleSegment (Page 5051)
EllipticalArc (Page 5070)
CircularArc (Page 5053)
Rectangle (Page 5149)
Polygon (Page 5134)
TextField (Page 5184)
IOField (Page 5098)
SymbolicIOField (Page 5170)
Button (Page 5040)
Switch (Page 5166)
GraphicView (Page 5091)
GraphicIOField (Page 5088)
Bar (Page 5033)
Clock (Page 5056)
Gauge (Page 5084)
Slider (Page 5159)
SymbolLibrary (Page 5175)
OnlineTrendControl (Page 5122)
FunctionTrendControl (Page 5077)
OnlineTableControl (Page 5114)
AlarmControl (Page 5018)
HTMLBrowser (Page 5096)
CheckBox (Page 5045)
OptionGroup (Page 5131)
WindowSlider (Page 5216)
Connector (Page 5058)
ScreenWindow (Page 5155)
DiskSpaceView (Page 5063)
ChannelDiagnose (Page 5044)

ScriptDiagnostics (Page 5157)
Group (Page 5094)
ForeignControl (Page 5072)
ProtectedAreaName (Page 5141)
UserView (Page 5214)
TubeTeeObject (Page 5205)
TubePolyline (Page 5202)
TubeDoubleTeeObject (Page 5200)
TubeArcObject (Page 5198)
MultiLineEdit (Page 5109)
MediaPlayer (Page 5107)
Listbox (Page 5104)
DateTimeField (Page 5061)
UserArchiveControl (Page 5207)
TrendRulerControl (Page 5187)
ProjectName (Page 5139)
AlarmView (Page 5029)
BatteryView (Page 5038)
RangeLabelView (Page 5142)
ZoneQualityView (Page 5221)
ZoneLabelView (Page 5220)
WLanQualityView (Page 5219)
TrendView (Page 5195)
SystemDiagnoseWindow (Page 5180)
SystemDiagnoseView (Page 5177)
StatusForce (Page 5164)
SmartClientView (Page 5162)
RecipeView (Page 5145)
RangeQualityView (Page 5144)

HelpText

Description

Returns the tooltip that is shown in Runtime as an operating aid for the specified object.

Access in runtime: Read

Syntax

Object.**HelpText**[=STRING]

Object

Required. An object of the "ScreenItem" type with the format:

- Button
- DateTimeField
- GraphicIOField
- IOField
- Switch
- SymbolicIOField
- TrendView

STRING

Optional. A value or constant that specifies the contents of the tooltip that is shown in runtime as an operator aid for the specified object.

HiddenInput

Description

Specifies whether the input value or a * for each character will be shown during the input.

Access in runtime: Read and write

Syntax

Object.**HiddenInput**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField

BOOLEAN

Optional TRUE, if the input value is not shown during the input. An * is shown for each character.

HighLimitColor

Description

Specifies the color of the top or right scroll button in a scroll bar.

Access in runtime: Read and write

Syntax

Object.**HighLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- WindowsSlider

Color

Optional A value or a constant that specifies the color of the top or right scroll button in a scroll bar.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

Hitlist

Description

No access in runtime.

HitlistColumnAdd

Description

Transfers the selected alarm text block from the list of available alarm text blocks to the list of selected alarm text blocks.

Access in runtime: Read and write

Syntax

Object.**HitlistColumnAdd**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Transfers the selected alarm text block from the list of available alarm text blocks to the list of selected alarm text blocks.

HitlistColumnCount

Description

Specifies the number of alarm text blocks displayed in the hit list in Runtime.

Access in runtime: Read and write

Syntax

Object.**HitlistColumnCount**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Specifies the number of alarm text blocks displayed in the hit list in Runtime.

HitlistColumnIndex

Description

References a alarm text block selected for the hit list. Using this attribute you can assign the values of other attributes to a specific alarm text block of the hit list.

Values between 0 and "HitlistColumnCount" minus 1 are valid for "HitlistColumnIndex". The attribute "HitlistColumnCount" defines the number of alarm text blocks selected for the hit list. The "HitlistColumnIndex" attribute can be dynamized with the **HitlistColumnRepos** attribute.

Access in runtime: Read and write

Syntax

Object.**HitlistColumnIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional References a alarm text block selected for the hit list.

HitlistColumnName

Description

Specifies the name of the alarm text block of the hit list that is referenced with the "HitlistColumnIndex" attribute. You cannot edit this name.

Access in runtime: Read and write

Syntax

Object.**HitlistColumnName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies the name of the alarm text block of the hit list that is referenced with the "HitlistColumnIndex" attribute.

HitlistColumnRemove

Description

Removes the marked alarm text block from the list of selected alarm text blocks and inserts it in the list of existing alarm text blocks.

Access in runtime: Read and write

Syntax

Object.**HitlistColumnRemove**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Removes the marked alarm text block from the list of selected alarm text blocks and inserts it in the list of existing alarm text blocks.

HitlistColumnRepos

Description

Changes the order of the alarm text blocks. The "Up" and "Down" commands move the selected alarm text block accordingly in the list. This moves the alarm text block in Runtime Control towards the front or towards the back.

Access in runtime: Read and write

Syntax

Object.**HitlistColumnRepos**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Changes the order of the alarm text blocks.

HitlistColumnSort

Description

Specifies the sorting order in the hit list for the alarm text block that is referenced in "HitlistColumnIndex".

Access in Runtime: Read and write

Syntax

Object.**HitlistColumnSort**[**=**SortMode]

Object

Required A "ScreenItem" object with following format:

- AlarmControl

SortMode

0: No sorting.

1: Ascending sorting from the lowest to highest value

2 Descending sorting from the highest to lowest value

HitlistColumnSortIndex

Description

Defines the sorting order of the alarm text block referenced in "HitlistColumnIndex" in the hit list. If you set the value to "0", the sorting criterion is removed in "HitlistColumnSort".

Access in runtime: Read and write

Syntax

Object.**HitlistColumnSortIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Defines the sorting order of the alarm text block referenced in "HitlistColumnIndex" in the hit list. If you set the value to "0", the sorting criterion is removed in "HitlistColumnSort".

HitlistColumnVisible

Description

Specifies a list with the selected alarm text blocks of the hit list which are used in the control in Runtime.

Access in runtime: Read and write

Syntax

Object.**HitlistColumnVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

Optional Specifies a list with the selected alarm text blocks of the hit list which are used in the control in Runtime.

HitlistDefaultSort

Description

Defines the default sorting order in the table columns of the hit list.

Access in Runtime: Read and write

Syntax

Object.**HitlistDefaultSort**[=SortMode]

Object

Required A "ScreenItem" object with following format:

- AlarmControl

SortMode

0: The list is sorted in ascending order of frequency.

1:: The list is sorted in descending order of frequency.

HitlistMaxSourceItems

Description

Specifies the maximum number of data records that can be used from the alarm log to create the hit list. The value can be anything between "0 - 10000".

Access in runtime: Read and write

Syntax

Object.**HitlistMaxSourceItems**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional A value or a constant that specifies the maximum number of data records that can be used from the alarm log to create the hit list.

HitlistMaxSourceItemsWarn

Description

Specifies whether a warning should be given if the maximum number of data records that is defined in "HitlistMaxSourceItems" is reached.

Access in Runtime: Read and write

Syntax

Object.**HitlistMaxSourceItemsWarn**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format "AlarmControl".

BOOLEAN

Optional. TRUE, if a warning is given as soon as the maximum number of data records that is defined in "HitlistMaxSourceItems" is reached.

See also

AlarmControl (Page 5018)

HitListRelTimeFactor

Description

Specifies the time factor which, alongside the time type "HitlistRelTimeFactorType", determines the time period in which the statistics of the hit list will be created.

If you do not wish to specify a period of time, set the time factor to "0".

Access in Runtime: Read and write

Syntax

Object.**HitlistRelTimeFactor**[=Long]

Object

Required. A "ScreenItem" object with the format "AlarmControl".

Long

Optional. A value or a constant that specifies the time factor.

See also

AlarmControl (Page 5018)

HitlistRelTimeFactorType

Description

Specifies the time type which, alongside the time factor "HitlistRelTimeFactor", determines the time period in which the statistics of the hit list will be created.

Access in Runtime: Read and write

Syntax

Object.**HitlistRelTimeFactorType**[=AlarmViewAdvancedTimeFactorUnit]

Object

Required. A "ScreenItem" object with the format "AlarmControl".

AlarmViewAdvancedTimeFactorUnit

hmiAlarmViewAdvancedTimeFactorUnitMinute (0): The time range for the statistics will be measured in minutes.

hmiAlarmViewAdvancedTimeFactorUnitHour (1): The time range for the statistics will be measured in hours.

hmiAlarmViewAdvancedFactorUnitDay (2): The time range for the statistics will be measured in days.

hmiAlarmViewAdvancedTimeFactorUnitWeek (3): The time range for the statistics will be measured in weeks.

hmiAlarmViewAdvancedTimeFactorUnitMonth (4): The time range for the statistics will be measured in months.

See also

AlarmControl (Page 5018)

HitListRelTime

Description

Sets a time range for the statistics.

If you do not wish to specify a time period, set the time factor to "0".

Access in runtime: Read and write

Syntax

Object.**HitListRelTime** [=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: If no time range is specified in the selection, the specified time range is used for the statistics.

FALSE: The specified time range is not used.

HorizontalAlignment

Description

Specifies the horizontal alignment of the text within the specified object.

Access in runtime: Read and write

Syntax

Object.**HorizontalAlignment**[=HorizontalAlignment]

Object

Required. A "ScreenItem" object with the following format:

- Button
- CheckBox
- ComboBox
- DateTimeField
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- Switch
- SymbolicIOField
- TextField

HorizontalAlignment

Optional A value or a constant that specifies the horizontal alignment of the text.

hmiAlignmentLeft (0):The text is arranged flush left in the object.

hmiAlignmentCentered (1): The text is arranged horizontally centered in the object.

hmiAlignmentRight (2): The text is arranged flush right in the object.

HorizontalGridLines

Description

Defines whether horizontal separating lines will be displayed.

Access in runtime: Read and write

Syntax

Object.**HorizontalGridLines**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: Horizontal grid lines are displayed.

FALSE: Horizontal grid lines are not displayed.

HorizontalScrollBarPosition

Description

Specifies the horizontal offset of the scroll bar in a screen window with scroll bar.

The picture is displayed in the picture window by positioning the horizontal and vertical scroll bars. If you wish to display the screen as a cutout where the scroll bars are located at the left and top edge of the screen, use the OffsetLeft" and "OffsetTop" properties as the source of this cutout.

Access in runtime: Read and write

Syntax

Object.**HorizontalScrollBarPosition**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- ScreenWindow

Int32

Optional A value or constant that specifies the horizontal offset of the scroll bar in a screen window with scroll bar.

Hotkey

Description

Returns the function key related to the mouse operation in respect of a button object.

Read only access.

See also

Button (Page 5040)

HourNeedleHeight

Description

Specifies the length of the hour hand in the "Clock" object.

Access in runtime: Read and write

Syntax

Object.**HourNeedleHeight**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Clock

Int32

Optional A value or a constant that specifies the length of the hour hand.

Specify the length of the hour hand as a percentage relating to the radius of the clock face.

HourNeedleWidth

Description

Specifies the width of the hour hand in the "Clock" object.

Access in runtime: Read and write

Syntax

Object.**HourNeedleWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Clock

Int32

Optional A value or a constant that specifies the width of the hour hand.

Specify the width of the hour hand as a percentage relating to double the length of the clock face.

Properties I-J

IconSpace

Description

Defines the spacing between the icons and text in the table cells. The value is active if and icon and text are displayed.

Access in runtime: Read and write

Syntax

Object.**IconSpace**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional Value that specifies the spacing.

Index

Description

Specifies the index for the selected text field.

Access in runtime: Read and write

Syntax

Object.**Index**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- CheckBox
- ComboBox
- ListBox
- OptionGroup

Int32

Optional A value or a constant that specifies the index of the selected text field.

InnerBackColorOff

Description

Specifies the color under the slider of the "Switch" object if the object is in OFF status.

Access in runtime: Read and write

Syntax

Object.**InnerBackColorOff**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Switch

Color

Optional A value or a constant that specifies the color for the OFF status.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

InnerBackColorOn

Description

Specifies the color under the slider of the "Switch" object if the object is in ON status.

Access in runtime: Read and write

Syntax

Object.**InnerBackColorOn**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Switch

Color

Optional A value or a constant that specifies the color for the ON status.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

InputValue

Description

Defines the value entered by the user in the IO field. The value is not displayed in the I/O field when the property is set.

If you want the value to be displayed in the I/O field after confirmation with the <Return> key, configure a direct connection between the "input value" and "output value" properties. The direct connection is only useful when no tag connection is configured at the output value but the user still wants to query the specified value, for example with a script.

Access in runtime: Read and write

Syntax

Object.**InputValue**[=Object]

Object

Required. A "ScreenItem" object with the following format:

- IOField
- SymbolicIOField

Object

Optional A value or a constant that specifies the input value.

Instance

Description

Returns an instance of the alarm object.

IntegerDigits

Description

Establishes the number of integer digits (0 to 20).

Access in runtime: Read and write

Syntax

Object.**IntegerDigits**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Int32

Optional A value or a constant that specifies the number of integer digits (0 to 20).

Interval

Description

Specifies the time interval for updating the displayed measured value. You enter the value in minutes.

Access in runtime: Read and write

Syntax

Object.**Interval**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

Int32

Optional A value or a constant that specifies the time interval for updating the measured values.

ItemBorderStyle

Description

Specifies the line style of the separation lines in the selection list of the selected object.

Access in runtime: Read and write

Syntax

Object.**ItemBorderStyle**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- SymbolicIOField

Int32

hmiLineStyleNone (-1): The selection list has no separation lines.

hmiLineStyleSolid (0): The selection list has solid separation lines.

hmiLineStyleDash (1): The selection list has broken separation lines.

hmiLineStyleDot (2): The selection list has dotted separation lines.

hmiLineStyleDashDot (3): The selection list has dash - dot lines as separation lines.

hmiLineStyleDashDotDot (4): The selection list has dash - dot - dot lines as separation lines.

JumpToLimitsAfterMouseClick

Description

Specifies whether the slider will be set to the associated end value. The end value is the minimum or maximum value. To set the slider to the end value, users click on the area outside of the current slider setting.

Access in runtime: Read and write

Syntax

Object.**JumpToLimitsAfterMouseClick**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- WindowsSlider

BOOLEAN

Optional TRUE, if the slider will be set to the associated end value.

Properties K-L

LabelColor

Description

Specifies the color of the scale label in the "Slider" object.

Access in runtime: Read and write

Syntax

Object.**LabelColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Slider

Color

Optional A value or a constant that specifies the color of the scale label.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

Language

Description

Determines the current Runtime language.

Access in Runtime: Read and write

Syntax

Object.**Language**[= LONG]

Object

Required A "HMIRuntime" object.

LONG

Optional A value or a constant that specifies the national code.

Comments

Specify the Runtime language in VBS with a national code, e.g. 1031 for German, 2057 for English etc.. Refer to the VBScript basics under the topic "Current local ID (LCID) diagram" for an overview of all the national codes.

See also

HMIRuntime (Page 4993)

LargeTicksBold

Description

Specifies whether the long marking lengths of a scale are shown in bold.

Access in runtime: Read and write

Syntax

Object.**LargeTicksBold**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the long marking lengths of a scale are shown in bold.

LargeTicksSize

Description

Specifies the length of the long marking lengths of a scale.

Access in runtime: Read and write

Syntax

Object.**LargeTicksSize**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Int32

Optional A value or a constant that specifies the length of the long marking lengths of a scale.

LastConnectedObjectIndex

Description

Specifies the index number of the connection point for the last, connected object.

Access in Runtime: Read and write

Syntax

Object.**LastConnectedObjectIndex**[=Int]

Object

Required A "ScreenItem" object with the format "Connector".

Int

Int

See also

Connector (Page 5058)

LastConnectedObjectName

Description

Specifies the object name of the object that is docked at the lower connector point.

Access in Runtime: Read and write

Syntax

Object.**LastConnectedObjectName**[=String]

Object

Required A "ScreenItem" object with the format "Connector".

String

String

See also

Connector (Page 5058)

LastError

Description

Returns an error code regarding the success of the last operation, e.g. information on a tag write or read process:

Code in hexadecimal notation	Description
0x00000000	OK
0x80040001	Execution error
0x80040002	Tag error
0x80040003	Server not available.
0x80040004	Multi Tag Error; error in one or several tags

To obtain an error description, first of all carry out the Read method.

Note

If the error occurs when accessing via the TagSet object, evaluate the LastError property for each tag of the TagSet object.

To obtain a statement about the quality of the supplied value, use the "QualityCode" property. To obtain a description of the error, use the "ErrorDescription" property.

Access during runtime: Read

Syntax

Object.**LastError**

Object

Necessary. A "Tag" object.

Example

The following example shows the error code for the "Tag1" tag:

```
'VBS77
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
MsgBox objTag.LastError
```

The following example adds two tags to the TagSet list and outputs the LastError property as Trace:

```
'VBS178
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMIRuntime.Trace "LastError: " & group.LastError & vbNewLine
```

The LastError property of a tag contained in the list may be accessed as follows:

```
HMIRuntime.Trace "LastError: " & group("Motor1").LastError & vbNewLine
```

See also

Tag (Page 5012)

TagSet (list) (Page 5016)

Layer

Description

Returns the layer that contains an object as LONG in the screen. There is a total of 32 layers available, whereby Layer "0" is the bottom layer and Layer "31" the top layer.

The configured objects are initially in the background of a layer.

Access during runtime: Read

Syntax

Object.**Layer**

Object

Necessary. A "ScreenItem" object.

Note

The Layer property specifies the layer in which the object is located. The layer "0" is output as layer "0".

When accessed, the layers are counted up from 1 in VBS. Therefore, address the level "1" with layers(2).

Example

The following example displays the name and layer of all objects in the screen "NewPDL1":

```
'VBS78
Dim objScreen
Dim objScrItem
Dim lngAnswer
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
Set objScrItem = objScreen.ScreenItems(strName)
lngAnswer = MsgBox(strName & " is in layer  " & objScrItem.Layer,vbOKCancel)
If vbCancel = lngAnswer Then Exit For
Next
```

See also

ScreenItem (Page 5003)

HTMLBrowser (Page 5096)

Group (Page 5094)

ForeignControl (Page 5072)

DateTimeField (Page 5061)

ZoneQualityView (Page 5221)

ZoneLabelView (Page 5220)

WLanQualityView (Page 5219)

ProtectedAreaName (Page 5141)

RangeLabelView (Page 5142)

RangeQualityView (Page 5144)

ScreenWindow (Page 5155)

ScriptDiagnostics (Page 5157)

Slider (Page 5159)

SmartClientView (Page 5162)

StatusForce (Page 5164)

SymbolLibrary (Page 5175)

BatteryView (Page 5038)

ChannelDiagnose (Page 5044)

CheckBox (Page 5045)

GraphicIOField (Page 5088)

Button (Page 5040)
Bar (Page 5033)
WindowSlider (Page 5216)
UserView (Page 5214)
TubeTeeObject (Page 5205)
TubePolyline (Page 5202)
TubeDoubleTeeObject (Page 5200)
TubeArcObject (Page 5198)
Switch (Page 5166)
Rectangle (Page 5149)
MultiLineEdit (Page 5109)
MediaPlayer (Page 5107)
Line (Page 5102)
Listbox (Page 5104)
AlarmView (Page 5029)
Circle (Page 5048)
CircleSegment (Page 5051)
CircularArc (Page 5053)
Clock (Page 5056)
Connector (Page 5058)
DiskSpaceView (Page 5063)
Ellipse (Page 5065)
EllipseSegment (Page 5067)
EllipticalArc (Page 5070)
Gauge (Page 5084)
GraphicView (Page 5091)
Polygon (Page 5134)
Polyline (Page 5137)
RoundButton (Page 5151)
SymbolicIOField (Page 5170)
IOField (Page 5098)
TextField (Page 5184)
OptionGroup (Page 5131)
RecipeView (Page 5145)
TrendView (Page 5195)

UserArchiveControl (Page 5207)

SystemDiagnoseWindow (Page 5180)

SystemDiagnoseView (Page 5177)

TrendRulerControl (Page 5187)

ProjectName (Page 5139)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

LayerDeclutteringEnable

Description

Indicates whether the layers of a screen can be shown or hidden dependent on the set minimum and maximum zoom.

Access during runtime: Read

Syntax

Object.**LayerDeclutteringEnable**

Object

Necessary. A "Screen" object.

Example:

The example outputs the LayerDecluttering property of the "NewPDL1" screen as Trace.

```
'VBS156
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

See also

Screen (Page 5000)

Layers

Description

Returns an object of type "Layers".

Access in Runtime: Read

Syntax

Object.**Layers**

Object

Required A "Screen" object.

See also

Screen (Page 5000)

Left

Description

Specifies the value of the X coordinate of the selected object.

Access in runtime: Read and write

Syntax

Object.**Left**[=DOUBLE]

Object

Required. A "ScreenItem" object. This property is a standard property of the ScreenItem-Objekts and is therefore available for all formats.

DOUBLE

Optional A value or a constant that contains the value of the X coordinate in pixels (measured from the top left edge of the screen).

Comments

The X coordinate refers to the top left corner of the rectangle that surrounds the object. The screen limits are also monitored in runtime. If the assigned coordinate value exceeds the display size, the user-defined function is interrupted with an error message.

Used for the following object types:

Auto-Hotspot

Auto-Hotspot

Auto-Hotspot

LeftOffset

Description

Specifies the distance between the screen and the left screen window margin.

The picture is displayed as a cutout of the picture window. The picture scroll bars are located at the left and upper edge of the picture. If you wish to display the screen in the screen window with horizontal and vertical offset of the screen scroll bars, use the "HorizontalScrollBarPosition" and "VerticalScrollBarPosition properties for the offset.

Access in runtime: Read and write

Syntax

Object.**LeftOffset**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Screenwindow

Int32

Optional A value or a constant that specifies the distance between the screen and the left screen window margin.

Limit4LowerLimit

Description

Specifies the lower limit for ""Reserve4".

The "Limit4LowerLimitEnabled" property must be set to TRUE so that the limit "Reserve4"" can be monitored.

Access in runtime: Read and write

Syntax

Object.**Limit4LowerLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the low limit for "Reserve4".

Comments

The "Limit4LowerLimitRelative" property specifies whether the object is evaluated as a percentage or absolutely.

Limit4LowerLimitColor

Description

Specifies the color for the lower limit "Reserve4".

The "Limit4LowerLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime: Read and write

Syntax

Object.**Limit4LowerLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Color

Optional A value or a constant that specifies the color for the lower limit ""Reserve4".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

Limit4LowerLimitEnabled

Description

Specifies whether the lower limit ""Reserve4"" is to be monitored.

Access in runtime: Read and write

Syntax

Object.**Limit4LowerLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the lower limit ""Reserve4" is to be monitored.

Comments

The following values will be specified with the properties ""Limit4LowerLimit", "Limit4LowerLimitColor" and "Limit4LowerLimitRelative"":

- Limit
- Representation upon reaching the limit
- Type of evaluation

Limit4LowerLimitRelative

Description

Specifies whether the lower limit "Reserve4" is evaluated as a percentage or as an absolute value.

Access in runtime: Read and write

Syntax

Object.**Limit4LowerLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional

TRUE: The lower limit "Reserve4" is evaluated as a percentage.

FALSE: The lower limit "Reserve4" is evaluated as an absolute value.

Limit4UpperLimit

Description

Specifies the lower limit for "Reserve4".

The "Limit4UpperLimitEnabled"" property must be set to TRUE so that the limit ""Reserve4" can be monitored.

Access in runtime: Read and write

Syntax

Object.**Limit4UpperLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the upper limit for "Reserve4".

Comments

The "Limit4UpperLimitRelative" property specifies whether the object is evaluated as a percentage or absolutely.

Limit4UpperLimitColor

Description

Specifies the color for the upper limit ""Reserve4".

The "Limit4UpperLimitEnabled"" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime: Read and write

Syntax

Object.**Limit4UpperLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Color

Optional A value or a constant that specifies the color for the upper limit "Reserve4".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

Limit4UpperLimitEnabled

Description

Specifies whether the upper limit "Reserve4" is to be monitored.

Access in runtime: Read and write

Syntax

Object.**Limit4UpperLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the upper limit ""Reserve4" is to be monitored.

Comments

The following values are defined via the properties ""Limit4UpperLimit", "Limit4UpperLimitColor" and "Limit4UpperLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

Limit4UpperLimitRelative

Description

Specifies whether the upper limit "Reserve4" is to be evaluated as a percentage or as an absolute value.

Access in runtime: Read and write

Syntax

Object.**Limit4UpperLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional

TRUE: The lower limit "Reserve4" is evaluated as a percentage.

FALSE: The lower limit "Reserve4" is evaluated as an absolute value.

Limit5LowerLimit

Description

Specifies the lower limit for "Reserve5"".

The "Limit5LowerLimitEnabled" property must be set to TRUE so that the limit ""Reserve5" can be monitored.

Access in runtime: Read and write

Syntax

Object.**Limit5LowerLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the lower limit for "Reserve5"".

Comments

The "Limit5LowerLimitRelative" property specifies whether the object is evaluated as a percentage or absolutely.

Limit5LowerLimitColor

Description

Specifies the color for the lower limit ""Reserve5"".

The "Limit5LowerLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime: Read and write

Syntax

Object.**Limit5LowerLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Color

Optional A value or a constant that specifies the color for the lower limit ""Reserve5".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

Limit5LowerLimitEnabled

Description

Specifies whether the lower limit "Reserve5" is to be monitored.

Access in runtime: Read and write

Syntax

Object.**Limit5LowerLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the lower limit ""Reserve5" is to be monitored.

Comments

The following values will be defined via the properties "Limit5LowerLimit", "Limit5LowerLimitColor" and "Limit5LowerLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

Limit5LowerLimitRelative

Description

Specifies whether the lower limit ""Reserve5" is to be evaluated as a percentage or as an absolute value.

Access in runtime: Read and write

Syntax

Object.**Limit5LowerLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional

TRUE: The lower limit "Reserve5" is evaluated as a percentage.

FALSE: The lower limit "Reserve5" is evaluated as an absolute value.

Limit5UpperLimit

Description

Specifies the lower limit for "Reserve5".

The ""Limit5UpperLimitEnabled" property must be set to TRUE so that the limit ""Reserve5" can be monitored.

Access in runtime: Read and write

Syntax

Object.**Limit5UpperLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the upper limit for ""Reserve5".

Comments

The "TypeLimitHigh5" property specifies whether the object is evaluated as a percentage or absolutely.

Limit5UpperLimitColor

Description

Specifies the color for the upper limit "Reserve5"".

The "Limit5UpperLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime: Read and write

Syntax

Object.**Limit5UpperLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Color

Optional A value or a constant that specifies the color for the upper limit ""Reserve5".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

Limit5UpperLimitEnabled

Description

Specifies whether the upper limit "Reserve5" is to be monitored.

Access in runtime: Read and write

Syntax

Object.**Limit5UpperLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the upper limit ""Reserve5"" is to be monitored.

Limit5UpperLimitRelative

Description

Specifies whether the upper limit "Reserve5" is to be evaluated as a percentage or as an absolute value.

Access in runtime: Read and write

Syntax

Object.**Limit5UpperLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional

TRUE: The higher limit "Reserve5" is evaluated as a percentage.

FALSE: The higher limit "Reserve5" is evaluated as an absolute value.

LineColor

Description

Defines the color of the window separation lines. Open the "Color selection" dialog by clicking the button.

Access in runtime: Read and write

Syntax

Object.**LineColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant that specifies the color of the window separation lines.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

LineEndShapeStyle

Description

Specifies the shape of the line ends.

Access in runtime: Read and write

Syntax

Object.**LineEndShapeStyle**[=LineEndShapeStyle]

Object

Required. An object of the "ScreenItem" type with the following format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- Ellipse
- EllipseSegment

- EllipticalArc
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowsSlider
- Switch

You have no access in runtime with the following formats:

- Switch
- TubeArcObect

LineEndShapeStyle

Optional. A value or a constant that specifies the form of the line ends.

LineWidth

Description

Specifies the line weight of the selected object.

Access in runtime: Read and write

Syntax

Object.**LineWidth**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- CircularArc
- EllipticalArc

- FunctionTrendControl
- Line
- OnlineTableControl
- OnlineTrendControl
- Polyline
- TrendRulerControl
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserArchiveControl

You have no access in runtime with the following format:

- Polygon

Int32

Optional A value or a constant that specifies the line weight in pixels.

LoadDataImmediately

Description

Specifies whether in the event of a screen refresh the tag values should be downloaded from the archives for the time range that is to be shown.

Access in runtime: Read and write

Syntax

Object.**LoadDataImmediately**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

BOOLEAN

Optional TRUE, if in the event of a screen the tag values should be downloaded from the archives for the time range that is to be shown.

LocalCursor

Description

No access in runtime.

LockSquaredExtent

Description

Specifies whether the size of the clock can be adjusted with the mouse.

Access in runtime: Read and write

Syntax

Object.**LockSquaredExtent**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Clock
- Gauge

BOOLEAN

Optional TRUE, if the size of the clock can be adjusted by dragging the mouse on the selection points to the desired aspect ratio.

Logging

Description

Returns an object of type "Logging".

Access in Runtime: Read

Syntax

Object.**Logging**

Object

Required A "HMIRuntime" object.

See also

HMIRuntime (Page 4993)

LogOperation

Description

Specifies whether, after operating this object, an alarm is output on the alarm system.

Access in runtime: Read and write

Syntax

Object.**LogOperation**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- CheckBox
- ComboBox
- IOField
- ListBox
- OptionGroup
- SymbolicIOField
- WindowsSlider

BOOLEAN

Optional TRUE, if, after operating this object, an alarm is output on the alarm system.

LongTermArchiveConsistency

Description

This setting defines how alarms are displayed in the historical alarm list (long-term).

Access in runtime: Read and write

Syntax

Object.**LongTermArchiveConsistency**[= BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl

BOOLEAN

Value	Explanation
TRUE	The 1000 most recent alarms are displayed on the client of all servers or the redundant server pair in the historical alarm list (long-term)
FALSE	1000 alarms are displayed in the historical alarm list (long-term) on the single-user system, server or client for each server, or for each redundant server pair.

LowerLimit

Description

Specifies the lower limit for input values.

Access in runtime: Read and write

Syntax

Object.**LowerLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- IOField

DOUBLE

Optional A value or a constant that specifies the lower limit for input values.

LowLimitColor

Description

Specifies the color of the bottom or left scroll button in a scroll bar.

Access in runtime: Read and write

Syntax

Object.**LowLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- WindowsSlider

Color

Optional A value or a constant that specifies the color of the bottom or left scroll button in a scroll bar.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

Properties M-N

MachineName

Description

Specifies the network code of the device that is to be monitored.

Enter the name or the port of the device as the network code.

Access in runtime: Read and write

Syntax

Object.**MachineName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- SmartClientView

STRING

Optional A value or a constant that contains the network code.

MarginToBorder

Description

Specifies the width of the 3D border in pixels. The value of the width is dependent on the size of the object.

Access in runtime: Read and write

Syntax

Object.**MarginToBorder**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- WindowsSlider

Int32

Optional A value or a constant that specifies the width of the 3D border in pixels.

MaximumValue

Description

Specifies the maximum value of the scale in the selected object.

Access in runtime: Read and write

Syntax

Object.**MaximumValue**[=DOUBLE | Int32 | SINGLE]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Gauge
- Slider
- WindowsSlider

DOUBLE | Int32 | SINGLE

Optional A value or a constant that specifies the maximum value. The data type depends on the format.

- DOUBLE: Bar
- Int32: Slider, WindowsSlider
- SINGLE: Gauge

MenuToolBarConfig

Description

Loads the given configuration file with configured menu and toolbars or returns the name of the configuration file.

Access in Runtime: Read and write

Syntax

Object.**MenuToolBarConfig**[=HmiObjectHandle]

Object

Required. An object of the "ScreenItem" type with the format:

- Screenwindow

HmiObjectHandle

Optional The configuration file with user-defined menu and toolbars.

MessageBlockAlignment

Description

Aligns the contents of a selected message block in the table.

Access in Runtime: Read and write

Syntax

Object.**MessageBlockAlignment** [=HorizontalAlignment]

Object

Required. An object of the "ScreenItem" type with the format "AlarmContro", "TrendRulerControl", "OnlineTableControl", "UserArchiveControl".

HorizontalAlignment

Value	Description	Explanation
0	Left	Aligns the contents of a selected message block to the left.
1	Centered	Aligns the contents of a selected message block to the center.
2	Right	Aligns the contents of a selected message block to the right.

See also

AlarmControl (Page 5018)

MessageBlockAutoPrecisions

Description

Enables automatic setting of the number of decimal places.

Access in runtime: Read and write

Syntax

Object.**MessageBlockAutoPrecisions**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: The number of decimal places is specified automatically.

FALSE: The value in the "Decimal places" field is effective.

MessageBlockCaption

Description

Specifies the caption of the column header in the alarm view for the selected alarm text block. You can only edit the labels after having disabled the "Apply project settings" option. The entered caption is effective in all Runtime languages.

Access in runtime: Read and write

Syntax

Object.**MessageBlockCaption**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies the caption of the column header in the alarm view for the selected alarm text block.

MessageBlockCount

Description

Specifies the number of existing alarm text blocks that are available for the alarm list and hit list.

Access in runtime: Read and write

Syntax

Object.**MessageBlockCount**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Specifies the number of existing alarm text blocks that are available for the alarm list and hit list.

MessageBlockDateFormat

Description

Defines the date format for displaying messages.

Access in runtime: Read and write

Syntax

Object.**MessageBlockDateFormat**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Value	Explanation
Automatic	The date format is set automatically.
dd.MM.yy	Day.Month.Year, e.g. 24.12.10.
dd.MM.yyyy	Day.Month.Year, e.g. 24.12.2010.
dd/MM/yy	Day/Month/Year, e.g. 24/12/10.
dd/MM/yyyy	Day/Month/Year, e.g. 24/12/2010.

MessageBlockExponentialFormat

Description

Specifies the exponential notation for visualization of the values of a selected alarm text block.

Access in runtime: Read and write

Syntax

Object.**MessageBlockExponentialFormat**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: The values are displayed in exponential format.

FALSE: The values are displayed in decimal format.

MessageBlockFlashOn

Description

Enables flashing of the selected alarm text block in Runtime after a message was activated.

Access in runtime: Read and write

Syntax

Object.**MessageBlockFlashOn**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: The content of the alarm text block flashes.

FALSE: The content of the alarm text block does not flash.

MessageBlockHideText

Description

Enables the textual display of the content of a selected alarm text block.

Access in runtime: Read and write

Syntax

Object.**MessageBlockHideText**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: The content is not shown as a text. The option is not activated.

FALSE: The content is shown as a text. The option is activated.

MessageBlockHideTitleText

Description

Specifies whether the header of the selected alarm text block is displayed as a text.

Access in runtime: Read and write

Syntax

Object.**MessageBlockHideTitleText**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: The header is not shown as text. The option is not activated.

FALSE: The header is shown as text. The option is activated.

MessageBlockId

Description

Specifies the assignment of ID number and alarm text block in the alarm view.

Access in runtime: Read and write

Syntax

Object.**MessageBlockId**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Specifies the assignment of ID number and alarm text block in the alarm view.

MessageBlockIndex

Description

References an existing alarm text block. You can assign the values of other attributes to a certain alarm text block by using the attribute. Values between 0 and "MessageBlockCount minus 1 are valid for "MessageBlockIndex".

The "MessageBlockIndex" attribute can be dynamized with the **MessageBlockRepos** attribute.

Access in runtime: Read and write

Syntax

Object.**MessageBlockIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional References an existing alarm text block. Values between 0 and "MessageBlockCount minus 1 are valid for "MessageBlockIndex".

MessageBlockLeadingZeros

Description

Enables the display of selected alarm text blocks with leading zero format.

Access in Runtime: Read and write

Syntax

Object.**MessageBlockLeadingZeros**[=Int32]

Object

Required A "ScreenItem" object with following format:

- AlarmControl

Int32

TRUE: Leading zeros are displayed. You specify the number of leading zeros in the properties.

FALSE: Leading zeros are not displayed.

MessageBlockLength

Description

Defines the length of the alarm text block selected based on the number of characters. You can only enter a value if the "Apply project settings" field is disabled.

Access in runtime: Read and write

Syntax

Object.**MessageBlockLength**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Value which specifies the length of the alarm text block.

MessageBlockName

Description

Specifies the name for the selected alarm text block.

Access in Runtime: Read and write

Syntax

Object.**MessageBlockName**[=STRING]

Object

Required A "ScreenItem" object with the format "AlarmControl".

STRING

Optional Specifies the name for the selected alarm text block.

MessageBlockPrecisions

Description

Specifies the decimal precision of the values of a selected alarm text block. You can only enter the value if the "Automatic" option is deactivated.

Access in runtime: Read and write

Syntax

Object.**MessageBlockPrecisions**[=Int16]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int16

Optional Specifies the number of decimal places.

MessageBlockSelected

Description

The available alarm text blocks can be used in Runtime control for the alarm list or hit list.

Select the "Alarm text blocks" tab to activate existing alarm text blocks as required in the Control. Select the "Hitlist" and "Alarm list" tabs to configure the hit list and alarm list based on the available blocks.

Access in Runtime: Read and write

Syntax

Object.**MessageBlockSelected**[=BOOLEAN]

Object

Required A "ScreenItem" object with following format:

- AlarmControl

BOOLEAN

Optional Select the "Alarm text blocks" tab to activate existing alarm text blocks as required in the Control. Select the "Hitlist" and "Alarm list" tabs to configure the hit list and alarm list based on the available blocks.

MessageBlockShowDate

Description

Enables the display of a date in the "Time" alarm text block in addition to the time.

Access in runtime: Read and write

Syntax

Object.**MessageBlockShowDate**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: The date and the time are displayed.

FALSE: The time is displayed.

MessageBlockShowIcon

Description

Enables the display of the content of a selected alarm text block as icon.

Access in runtime: Read and write

Syntax

Object.**MessageBlockShowIcon**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: The content is shown as a symbol.

FALSE: The content is not shown as a symbol.

MessageBlockShowTitleIcon

Description

Specifies whether the header of the selected alarm text block is displayed as a text.

Access in runtime: Read and write

Syntax

Object.**MessageBlockShowTitleIcon**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: The header is shown as a symbol.

FALSE: The header is not shown as a symbol.

MessageBlockTextId

Description

Specifies the caption of the selected alarm text block using a Text ID which was derived from the text library. The caption is adapted automatically if a user changes the Runtime language.

You can only enter a text ID after having disabled the "Apply project settings" option.

Access in runtime: Read and write

Syntax

Object.**MessageBlockTextId**[=Int32]

Object

Required. A "ScreenItem" object with the format "AlarmControl".

Int32

Optional Specifies the name of the selected alarm text block using a text ID number.

MessageBlockTimeFormat

Description

Defines which time format or duration format is used for displaying the messages.

Access in runtime: Read and write

Syntax

Object.**MessageBlockTimeFormat**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

The following time formats are available:

Value	Explanation
Automatic	The time format is set automatically.
HH:mm:ss	Hours:Minutes:Seconds, e.g. 15:35:44
HH:mm:ss.ms	Hours:Minutes:Seconds.Milliseconds, e.g. 15:35:44.240.
hh:mm:ss tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.
hh:mm:ss.ms tt	Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.

The following duration formats are available:

Value	Explanation
Automatic	The duration format is determined automatically.
d H:mm:ss	Day Hours:Minutes:Seconds, e.g. 1 2:03:55.
H:mm:ss.	Hours:Minutes:Seconds, e.g. 26:03:55.
m:ss	Minutes:Seconds, Example: 1563:55.
s	Seconds, e.g. 93835.

MessageBlockType

Description

Specifies the number of available message blocks which are available for the message list and hit list.

Access in runtime: Read

Syntax

Object.**MessageBlockType**[=AlarmBlockType]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl

AlarmBlockType

Value	Description	Description
0	System block	The message block belongs to the system block category
1	Text block	The message block belongs to the user text block category
2	Process value block	The message block belongs to the process value block category
3	Hitlist block	The message block belongs to the message blocks of the hitlist.

MessageColumnAdd

Description

Adds the selected alarm text block from the list of existing alarm text blocks to the list of selected alarm text blocks.

Access in runtime: Read and write

Syntax

Object.**MessageColumnAdd**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Applies the selected alarm text block from the list of existing alarm text blocks to the list of selected alarm text blocks.

MessageColumnCount

Description

Specifies the number of selected alarm text blocks displayed in the alarm list in Runtime.

Access in runtime: Read and write

Syntax

Object.**MessageColumnCount**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Specifies the number of selected alarm text blocks displayed in the alarm list in Runtime.

MessageColumnIndex

Description

References a alarm text block selected for the alarm list. Using this attribute you can assign the values of other attributes to a specific alarm text block of the alarm list.

Valid values for "MessageColumnIndex" are between 0 and "MessageColumnCount" minus 1. The "MessageColumnIndex" attribute can be dynamized by the attribute **MessageColumnRepos**.

Access in runtime: Read and write

Syntax

Object.**MessageColumnIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional References a alarm text block selected for the alarm list. Valid values for "MessageColumnIndex" are between 0 and "MessageColumnCount" minus 1.

MessageColumnName

Description

Specifies the name of the alarm text block in the alarm list which is referenced with the "MessageColumnIndex" property.

Access in runtime: Read and write

Syntax

Object.**MessageColumnName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies the name of the alarm text block in the alarm list which is referenced with the "MessageColumnIndex" property.

MessageColumnRemove

Description

Removes the marked alarm text block from the list of selected alarm text blocks and inserts it in the list of existing alarm text blocks.

Access in runtime: Read and write

Syntax

Object.**MessageColumnRemove**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Removes the marked alarm text block from the list of selected alarm text blocks and inserts it in the list of existing alarm text blocks.

MessageColumnRepos

Description

Specifies the order of the alarm text blocks. The "Up" and "Down" commands move the selected alarm text block accordingly in the list. This places the alarm text block further forward or back in the alarm view in Runtime.

Access in runtime: Read and write

Syntax

Object.**MessageColumnRepos**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Value which specifies the order of the alarm text blocks in the list.

MessageColumnSort

Description

Defines the sorting order of the alarm text block referenced in "MessageColumnIndex" .

Access in Runtime: Read and write

Syntax

Object.**MessageColumnSort**[=SortMode]

Object

Required A "ScreenItem" object with following format:

- AlarmControl

SortMode

0: No sorting.

1: Ascending sorting from the lowest to highest value

2 Descending sorting from the highest to lowest value

MessageColumnSortIndex

Description

Defines the sorting order of the alarm text block referenced in "MessageColumnIndex". If you set the value to "0", the sorting criterion is removed in "MessageColumnSort".

Access in runtime: Read and write

Syntax

Object.**MessageColumnSortIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Defines the sorting order of the alarm text block referenced in "MessageColumnIndex".

MessageColumnVisible

Description

Specifies whether alarm text blocks are displayed in the alarm view.

Access in runtime: Read and write

Syntax

Object.**MessageColumnVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

Optional

TRUE: Alarm text blocks are displayed in the alarm view.

FALSE: Alarm text blocks are not displayed in the alarm view.

MessageListType

Description

This setting defines which alarm lists are displayed in the picture.

Access in runtime: Read and write

Syntax

Object.**MessageListType**[=AlarmListType]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

AlarmListType

Value	Description	Explanation
0	Alarm list	The currently active messages are displayed after a picture was called.
1	Short-term archive list	A short-term archive list displays the logged messages after the picture was called. The display is updated immediately on activation of new messages.
2	Long-term archive list	A long-term archive list displays the logged messages after a picture was called.
3	Lock list	Only the currently locked messages are displayed after a picture was called.
4	Hit list	The configured statistics data is displayed after a picture was called.
5	List of messages to be hidden	The messages to be hidden are displayed at the call of a picture.

MinimumValue

Description

Specifies the minimum value of the scale in the selected object.

Access in runtime: Read and write

Syntax

Object.**MinimumValue**[=DOUBLE | Int32 | SINGLE]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Gauge
- Slider
- WindowsSlider

DOUBLE | Int32 | SINGLE

Optional A value or a constant that specifies the minimum value. The data type depends on the format.

- DOUBLE: Bar
- Int32: Slider, WindowsSlider
- SINGLE: Gauge

MinuteNeedleHeight

Description

Specifies the length of the minute hand in the "Clock" object.

Access in runtime: Read and write

Syntax

Object.**MinuteNeedleHeight**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Clock

Int32

Optional A value or a constant that specifies the length of the minute hand.

Specify the length of the minute hand as a percentage relating to the radius of the clock face.

MinuteNeedleWidth

Description

Specifies the width of the minute hand in the "Clock" object.

Access in runtime: Read and write

Syntax

Object.**MinuteNeedleWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Clock

Int32

Optional A value or a constant that specifies the width of the minute hand.

Specify the width as a percentage relating to double the length of the minute hand.

Mode

Description

Specifies the field type of the selected object.

Access in Runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**Mode**[=IOFieldType]

Object

Required A "ScreenItem" object with following format:

- Button
- IOField
- RoundButton
- SymbolicIOField

You have no access in runtime with the following format:

- DateTimeField
- GraphicIOField
- Switch

IOFieldType

hmiIOFieldInput (1): Input field

hmiIOFieldOutput (0): Output field

hmiIOFieldInOutput (2): Input and output field

Moveable

Description

Specifies whether the window can be moved in Runtime.

Access in Runtime: Read and write

Syntax

Object.**Moveable**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format "UserArchiveControl".

BOOLEAN

Optional. TRUE, if the window can be moved in Runtime.

See also

AlarmControl (Page 5018)

UserArchiveControl (Page 5207)

TrendRulerControl (Page 5187)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

OnlineTrendControl (Page 5122)

MsgFilterSQL

Description

Specifies one or more SQL statements for the customized selection of the messages. Several customized selections are connected with "OR". If you have configured a fixed selection "DefaultMsgFilterSQL", the SQL statements of "DefaultMsgFilterSQL" and "MsgFilterSQL" are connected with "AND".

Access in runtime: Read and write

Syntax

Object.**MsgFilterSQL**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional A value or constant that specifies the SQL statements for user-defined selection of messages.

Name

Description

Returns the object name as STRING. The returned value is dependent upon the used object.

Access during runtime: Read

Syntax

Object.**Name**[=STRING]

Object

Required. An object of the "ScreenItem" type with the format:

- BatteryView
- Clock
- Ellipse
- Gauge
- Slider

You have no access in runtime with the following format:

- AlarmControl
- AlarmView
- ApplicationWindow
- Bar
- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- DateTimeField
- DiscSpaceView
- EllipseSegment
- EllipticalArc

- FunctionTrendControl
- GraphicIOField
- GraphicView
- HTML-Browser
- IOField
- Line
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- Polygon
- Polyline
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- Rectangle
- RoundButton
- S7GraphOverview
- Screenwindow
- SmartClientView
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TextField
- TrendRulerControl
- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject

- Tubepolyline
- UserArchiveControl
- UserView
- WLanQualityView
- WindowsSlider
- ZoneLabelView
- ZoneQualityView

Comments

Dependent on the specified object, the following object names will be returned:

- Tag: Name of the tag without server and tag prefix.
- Project: Name of the current Runtime project.
- DataItem: Name of the DataItem object.
- Layer: Name of the layer.
- FunctionTrendControl : Name of the trend referenced by the "Index" property.

Note

Use the "Name" property to address a tag in the Tags" list. Tag names are structured in WinCC according to the following scheme:

<Tag prefix><Name of tag>

If you only specify the tag name, the tag prefix is applied from the screen window shortcut.

Example

The following example returns the name of the current Runtime project as Trace:

```
'VBS160
HMIRuntime.Trace "Name: " & HMIRuntime.ActiveProject.Name & vbNewLine
```

NeedleBorderColor

Description

Specifies the line color of the pointer in the "Clock" object.

Access in runtime: Read and write

Syntax

Object.**NeedleBorderColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Clock

Color

Optional A value or a constant that specifies the line color of the pointer.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

NeedleColor

Description

Specifies the hand color of the "Clock" object.

You must have configured the "NeedleFillStyle" property as "Transparent" for the pointer color to be shown.

Access in runtime: Read and write

Syntax

Object.**NeedleColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Clock

Color

Optional A value or a constant that specifies the pointer color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

NeedleFillStyle

Description

Specifies whether the pointers are displayed as filled or transparent:

Access in runtime: Read and write

Syntax

Object.**NeedleFillStyle**[=THmiFillStyle]

Object

Required. A "ScreenItem" object with the following format:

- Clock

THmiFillStyle

hmiFillStyleTransparent (65536): The pointers are filled in the pointer fill color and shown with a border in the foreground color.

hmiFillStyleSolid (0): The pointers are displayed as transparent with a border in the foreground color.

NormalRangeColor

Description

Specifies the color of the normal range on the scale of the "Gauge" object.

The "NormalRangeVisible" property must have the value TRUE for the normal range to be displayed.

Access in runtime: Read and write

Syntax

Object.**NormalRangeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

Color

Optional A value or a constant that specifies the color of the normal range.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

NormalRangeVisible

Description

Specifies whether the normal range in the scale of the "Gauge" object is displayed.

Access in runtime: Read and write

Syntax

Object.**NormalRangeVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

BOOLEAN

Optional TRUE, if the normal range will be displayed in the scale.

Comments

Specify the color of the normal range with the "NormalRangeColor" property.

Properties O-P

Object

Description

If you use a Control that is not a standard WinCC control, it could be that the properties provided by the Control have the same names as the general ScreenItem properties. In this case, the ScreenItem properties take priority. You can access the "hidden" properties of an outside provider Controls with the additional "object" property.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**Object**

Object

Required. An object of the "ScreenItem" type with the format:

- AlarmControl
- AlarmView
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl

You have no access in runtime with the following format:

- RecipeView
- SmartClientView
- StatusForce
- SymbolLibrary
- UserArchiveControl

Application example

To do this, address the properties of an outside provider control in the following way:

Control.object.type

If you are using only the form Control.type, the properties of the ScreenItem object will be used in case of identical names.

ObjectSizeDeclutteringEnable

Description

Specifies whether only the objects within a set size range are displayed.

Access during runtime: Read

Syntax

Object.**ObjectSizeDeclutteringEnable**

Object

Necessary. A "Screen" object.

Example

The example outputs the Decluttering properties of the screen "NewPDL1" as Trace.

```
'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

See also

Screen (Page 5000)

ObjectSizeDeclutteringMax

Description

Returns the upper size range for the display suppression of objects of the specified screen as LONG.

Access during runtime: Read

Syntax

Object.**ObjectSizeDeclutteringMax**

Object

Necessary. A "Screen" object.

Example

The example outputs the Decluttering properties of the screen "NewPDL1" as Trace.

```
'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

See also

Screen (Page 5000)

ObjectSizeDeclutteringMin

Description

Returns the lower size range for the display suppression of objects of the specified screen as LONG.

Access during runtime: Read

Syntax

Object.**ObjectSizeDeclutteringMin**

Object

Necessary. A "Screen" object.

Example

The example outputs the Decluttering properties of the screen "NewPDL1" as Trace.

```
'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

See also

Screen (Page 5000)

OCXState

Description

No access in runtime.

Online

Description

Specifies start and stop of the updating.

Access in runtime: Read and write

Syntax

Object.**Online**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTrendControl
- FunctionTrendControl
- OnlineTableControl

BOOLEAN

Optional

TRUE: The updated display is stopped. The values are buffered and updated when the button is clicked again.

FALSE: The updated display is continued.

OperationSteps

Description

Specifies by how many steps the slider of the scrollbar is moved with one mouse click.

Access in runtime: Read and write

Syntax

Object.**OperationSteps**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- WindowSlider

Int32

Optional A value or a constant which specifies by how many steps the slider of the scrollbar is moved with one mouse-click.

OperatorMessageId

Description

Specifies the assignment of ID number and trigger event in the alarm view.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageId**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Value	Description	Explanation
0	Lock	Trigger event "Lock"
1	Unlock	"Unlock" Trigger Event
2	Hide	Trigger event "Hide"
3	Unhide	Trigger event "Unhide"
4	Ackn	Trigger event "Acknowledge"

OperatorMessageIndex

Description

References an operator message event. You can assign the values of other properties to a specific operator message by using the property.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Value	Explanation
0	Message event "Lock"
1	"Unlock" Message Event
2	Message event "Hide"
3	Message event "Unhide"
4	Message event "Acknowledge"

OperatorMessageName

Description

Specifies the name which is referenced with the "OperatorMessageIndex" event in message events for operator messages.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Value	Explanation
Lock	Message event "Lock"
Unlock	Message event "Enable"
Hide	Message event "Hide"
Unhide	Message event "Unhide"
Quit	Message event "Ackn."

OperatorMessageNumber

Description

Specifies a message number for the operator message of the selected message event if you do *not* use the operator message of WinCC.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageNumber**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Specifies a message number for the operator message of the selected message event.

OperatorMessageSelected

Description

Activates the message events in the list that trigger operator messages.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSelected**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

Optional Activates the message events in the list that trigger operator messages.

OperatorMessageSource1

Description

Specifies a alarm text block of the operated message to be added to "Process value block 1" of the operator message configured here.

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "1" at process value as the alarm text block lock of the operated alarm "User text block 1".

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource1**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies a alarm text block of the operated message to be added to "Process value block 1" of the operator message configured here.

OperatorMessageSource2

Description

Specifies a alarm text block of the operated message to be added to "Process value block 2" of the operator message configured here.

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 2" of the operator message. Select "2" at process value as the alarm text block of the operated alarm "User text block 1".

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource2**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies a alarm text block of the operated message to be added to "Process value block 2" of the operator message configured here.

OperatorMessageSource3

Description

Specifies a alarm text block of the operated message to be added to "Process value block 3" of the operator message configured here.

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 3" of the operator message. Select "3" at process value as the alarm text block of the operated alarm "User text block 1".

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource3**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies a alarm text block of the operated message to be added to "Process value block 3" of the operator message configured here.

OperatorMessageSource4

Description

Specifies a alarm text block of the operated message to be added to "Process value block 4" of the operator message configured here.

An operator message is to be generated to indicate that a message was locked. The content of "User text block 4" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 4" of the operator message. Select "4" at process value as the alarm text block of the operated alarm "User text block 1".

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource4**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies a alarm text block of the operated message to be added to "Process value block 4" of the operator message configured here.

OperatorMessageSource5

Description

Specifies a alarm text block of the operated message to be added to "Process value block 5" of the operator message configured here.

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 5" of the operator message. Select "5" at process value as the alarm text block of the operated alarm "User text block 1".

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource5**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies a alarm text block of the operated message to be added to "Process value block 5" of the operator message configured here.

OperatorMessageSource6

Description

Specifies a alarm text block of the operated message to be added to "Process value block 6" of the operator message configured here.

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 6" of the operator message. Select "6" at process value as the alarm text block of the operated alarm "User text block 1".

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource6**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies a alarm text block of the operated message to be added to "Process value block 6" of the operator message configured here.

OperatorMessageSource7

Description

Specifies a alarm text block of the operated message to be added to "Process value block 7" of the operator message configured here.

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 7" of the operator message. Select "7" at process value as the alarm text block of the operated alarm "User text block 1".

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource7**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies a alarm text block of the operated message to be added to "Process value block 7" of the operator message configured here.

OperatorMessageSource8

Description

Specifies a alarm text block of the operated message to be added to "Process value block 8" of the operator message configured here.

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 8" of the operator message. Select "8" at process value as the alarm text block of the operated alarm "User text block 1".

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource8**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies a alarm text block of the operated message to be added to "Process value block 8" of the operator message configured here.

OperatorMessageSource9

Description

Specifies a alarm text block of the operated message to be added to "Process value block 9" of the operator message configured here.

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process

value block 9" of the operator message. Select "9" at process value as the alarm text block of the operated alarm "User text block 1".

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource9**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional Specifies a alarm text block of the operated message to be added to "Process value block 9" of the operator message configured here.

OperatorMessageSource10

Description

Specifies a alarm text block of the operated message to be added to "Process value block 10" of the operator message configured here.

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 10" of the operator message. Select "10" at process value as the alarm text block of the operated alarm "User text block 1".

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource10**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Specifies a alarm text block of the operated message to be added to "Process value block 10" of the operator message configured here.

OperatorMessageSourceType1

Description

Specifies the format of the source content for the transfer.

Syntax

Object.**OperatorMessageType1**[=Type]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Type

Value	Description	Description
0	Text	Transfer the source content in text format.
1	Value	Transfer the source content as value.

See also

AlarmControl (Page 5018)

OperatorMessageSourceType2

Description

Specifies in what format the content of the source is transferred.

Syntax

Object.**OperatorMessageType2**[=Type]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Type

Value	Description	Explanation
0	Text	The content of the source is transferred as a text.
1	Value	The content of the source is transferred as a value.

See also

AlarmControl (Page 5018)

OperatorMessageSourceType3

Description

Specifies in what format the content of the source is transferred.

Syntax

Object.**OperatorMessageType1**[=Type]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Type

Value	Description	Explanation
0	Text	The content of the source is transferred as a text.
1	Value	The content of the source is transferred as a value.

See also

AlarmControl (Page 5018)

OperatorMessageSourceType4

Description

Specifies in what format the content of the source is transferred.

Syntax

Object.**OperatorMessageType4**[=Type]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Type

Value	Description	Explanation
0	Text	The content of the source is transferred as a text.
1	Value	The content of the source is transferred as a value.

See also

AlarmControl (Page 5018)

OperatorMessageSourceType5

Description

Specifies in what format the content of the source is transferred.

Syntax

Object.**OperatorMessageType5**[=Type]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Type

Value	Description	Explanation
0	Text	The content of the source is transferred as a text.
1	Value	The content of the source is transferred as a value.

See also

AlarmControl (Page 5018)

OperatorMessageSourceType6

Description

Specifies in what format the content of the source is transferred.

Syntax

Object.**OperatorMessageType6**[=Type]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Type

Value	Description	Explanation
0	Text	The content of the source is transferred as a text.
1	Value	The content of the source is transferred as a value.

See also

AlarmControl (Page 5018)

OperatorMessageSourceType7

Description

Specifies in what format the content of the source is transferred.

Syntax

Object.**OperatorMessageType7**[=Type]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Type

Value	Description	Explanation
0	Text	The content of the source is transferred as a text.
1	Value	The content of the source is transferred as a value.

See also

AlarmControl (Page 5018)

OperatorMessageSourceType8

Description

Specifies in what format the content of the source is transferred.

Syntax

Object.**OperatorMessageType8**[=Type]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Type

Value	Description	Explanation
0	Text	The content of the source is transferred as a text.
1	Value	The content of the source is transferred as a value.

See also

AlarmControl (Page 5018)

OperatorMessageSourceType9

Description

Specifies in what format the content of the source is transferred.

Syntax

Object.**OperatorMessageType9**[=Type]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Type

Value	Description	Explanation
0	Text	The content of the source is transferred as a text.
1	Value	The content of the source is transferred as a value.

See also

AlarmControl (Page 5018)

OperatorMessageSourceType10

Description

Specifies in what format the content of the source is transferred.

Syntax

Object.**OperatorMessageType10**[=Type]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Type

Value	Description	Explanation
0	Text	The content of the source is transferred as a text.
1	Value	The content of the source is transferred as a value.

See also

AlarmControl (Page 5018)

PageMode

Description

Enables paging is in the long-term archive list. Allows you to display all messages of the short-term archive in the long-term archive list. Use the "PageModeMessageNumber" property to determine the number of messages displayed per page.

The page up/down buttons of the toolbar can be used if paging is enabled.

Access in runtime: Read and write

Syntax

Object.**PageMode**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: Scrolling in the long-term log list is possible.

FALSE: Scrolling in the long-term log list is not possible.

PageModeMessageNumber

Description

Defines the number of messages shown per page when paging the long-term archive list.

Access in runtime: Read and write

Syntax

Object.**PageModeMessageNumber**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional A value or constant that specifies the number of messages per page.

Path

Description

Returns the path of the current project without the file names as STRING. For a WinCC client without its own path, the path is returned in UNC format, otherwise the local path is returned.

Access during runtime: Read

Syntax

Object.**Path**

Object

Necessary. A "Project" object.

Example

The following example returns the project path as Trace:

```
'VBS161
HMIRuntime.Trace "Path: " & HMIRuntime.ActiveProject.Path & vbNewLine
```

See also

Project (Page 4999)

Password

Description

Determines the password for the loading of the remote monitor.

Access in runtime: Read and write

Syntax

Object.**Password**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- SmartClientView

STRING

Optional A value or constant that contains the password for establishing remote monitoring.

PicAlignment

Description

Specifies the alignment within the specified object.

Picture

Description

Specifies the screen that will be displayed in the graphics object in Runtime.

Access in Runtime: Read and write

Syntax

Object.**Picture**[=Image]

Object

Required A ""ScreenItem" object with following format:

- GraphicView
- Clock

Image

Optional A value or a constant that specifies the screen that will be displayed in the graphics object in Runtime.

Comments

The picture (*.BMP or *.DIB) must be located in the "GraCS" folder of the current project so that it can be integrated.

PictureAlignment

Description

Specifies the display type for the background screen in the process image.

Access in runtime: Read and write

Syntax

Object.**PictureAlignment**[=PictureAlignment]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton

PictureAlignment

Optional A value or a constant that specifies the display type for the background screen in the process image.

PictureDeactivated

Description

Specifies the picture that is displayed in the "Deactivated" state.

Access in Runtime: Read and write

Syntax

Object.**PictureDeactivated**[=Image]

Object

Required A ""ScreenItem" object with following format:

- Roundbutton

Image

Optional A value or a constant that specifies the picture that will be displayed in the "Deactivated" status.

Comments

The picture must be located in the "GraCS" folder of the current project in order to integrate it.

PictureOff

Description

Specifies the picture that is displayed in the "Off" status.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**PictureOff**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the format:

- Button
- GraphicIOField
- RoundButton

You have no access in runtime with the following format:

- Switch

HmiObjectHandle

Optional A value or a constant that specifies the picture that will be displayed in the "Off" status.

Comments

The picture (*.BMP or *.DIB) must be in the "GraCS" folder of the current project so that you can integrate it.

PictureOn

Description

Specifies the screen that will be displayed in the "on" state.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**PictureOn**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the format:

- Button
- GraphicIOField
- RoundButton

You have no access in runtime with the following format:

- Switch

HmiObjectHandle

Optional A value or a constant that specifies the screen that will be displayed in the "on" state.

Comments

The screen (*.BMP or *.DIB) must be in the "GraCS" folder of the current project in order to be able to integrate it.

PointerColor

Description

Specifies the pointer color of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**PointerColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

Color

Optional A value or a constant that specifies the pointer color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

PointsCount

Description

Specifies the number of corner points of the polyline or of the polygon.

Access in runtime: Read and write

Syntax

Object.**PointsCount**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- Polygon
- Polyline
- Tubepolyline

You have no access in runtime with the following format:

- Line

Int32

Optional A value or a constant that specifies the number of corner points of the polyline.

Precision

Description

Specifies the number of decimal places (0 to 20).

Access in runtime: Read and write

Syntax

Object.**Precision**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Int32

Optional A value or a constant that specifies the number of decimal places (0 to 20).

Pressed

Description

Specifies whether the selected object is displayed in a pressed state.

Access in runtime: Read and write

Syntax

Object.**Pressed**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton

BOOLEAN

Optional TRUE, if the selected object is displayed in a pressed state.

ProcessValue

Description

Specifies the default for the value to be displayed.

This value will be used in runtime if the associated tag is not connected or has not been updated when the screen starts.

Access in runtime: Read and write

Syntax

Object.**ProcessValue**[=DOUBLE | Int32 | Object | SINGLE]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Gauge
- GraphicIOField
- IOField
- OptionGroup
- Slider
- SymbolicIOField
- WindowsSlider

You have no access in runtime with the following format:

- Button
- CheckBox
- DateTimeField
- Switch
- SymbolLibrary

DOUBLE | Int32 | SINGLE

Optional A value or a constant that includes the default value. The data type depends on the format.

- DOUBLE: Bar
- Int32: GraphicIOField, OptionGroup, Slider, SymbolicIOField, WindowsSlider
- Object: IOField
- SINGLE: Gauge

Comments

If you want to assign the "ProcessValue" SmartTags property, then you must formulate the assignment as follows:

'Examples for the assignment of SmartTags

```
'Example 1
IOField.ProcessValue = SmartTags("TagName").Value
'Example 2
HmiRuntime.Screens("Screen_1").ScreenItems("IOField_1").ProcessValue =
SmartTags("Tag_1").Value
```

ProgID

Description

In the case of non-WinCC controls, the version-independent ProgID is returned as the type.

See also

ForeignControl (Page 5072)

Properties Q-R

QualityCode

Description

Returns the quality of a tag value after reading the tag as SHORT. After a tag has been written, the value is invalid.

Access during runtime: Read

Syntax

Object.**QualityCode**

Object

Necessary. An object of the "HMIRuntime" type.

Example

The following example indicates the quality of the read value when no errors have occurred during the reading process:

```
'VBS83
Dim objTag
Dim lngLastErr
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
lngLastErr = objTag.LastError
If 0 = lngLastErr Then
MsgBox objTag.QualityCode
End If
```

See also

Tag (Page 5012)

Radius

Description

Determines the radius of the given object "Circle."

Access in runtime: Read and write

Syntax

Object.**Radius**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Circle
- CircleSegment
- CircularArc
- RoundButton

Int32

Optional A value or constant which determines the radius in pixels.

RadiusHeight

Description

Specifies the secondary axis of the specified object.

Access in runtime: Read and write

Syntax

Object.**RadiusHeight**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- Ellipse
- EllipseSegment
- EllipticalArc
- TubeArcObject

You have no access in runtime with the following format:

- Circle

Int32

Optional A value or constant which determines the secondary axis in pixels.

RadiusWidth

Description

Specifies the main axis of the specified object.

Access in runtime: Read and write

Syntax

Object.**RadiusWidth**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- Ellipse
- EllipseSegment
- EllipticalArc
- TubeArcObject

You have no access in runtime with the following format:

- Circle

Int32

Optional A value or constant which determines the main axis in pixels.

RecipeName

Description

Returns the name of the recipe that is currently being displayed in the "Recipe view".

Access in runtime: Read

Syntax

Object.**RecipeName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- RecipeView

STRING

Optional A value or a constant that returns the name of the recipe.

RecipeNumber

Description

Returns the number of the recipe that is currently being displayed in the "Recipe view".

Access in runtime: Read

Syntax

Object.**RecipeNumber**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- RecipeView

Int32

Optional A value or a constant that returns the number of the recipe.

RecordName

Description

Returns the name of the recipe data record that is currently being displayed in the "Recipe view".

Access in runtime: Read

Syntax

Object.**RecordName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- RecipeView

STRING

Optional A value or a constant that returns the name of the recipe data record.

RecordNumber

Description

Returns the number of the recipe data record that is currently being displayed in the "Recipe view".

Access in runtime: Read

Syntax

Object.**RecordNumber**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- RecipeView

Int32

Optional A value or a constant that returns the number of the recipe data record.

RelativeFillLevel

Description

Specifies the fill percentage for the object.

Access in runtime: Read and write

Syntax

Object.**RelativeFillLevel**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- Button
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicView
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- CheckBox

Int32

Optional A value or a constant that specifies the fill percentage of the object.

Rotation

Description

Specifies the angle of rotation in degrees of the selected object. The angle of rotation is measured counterclockwise.

Access in runtime: Read and write

Syntax

Object.**Rotation**[=SymbolLibraryRotation]

Object

Required. A "ScreenItem" object with the following format:

- SymbolLibrary

SymbolLibraryRotation

hmiSymbolLibraryRotationNone (0): The object rotates by 0 degrees.

hmiSymbolLibraryRotation90Degree (90): The object rotates by 90 degrees.

hmiSymbolLibraryRotation180Degree (180): The object rotates by 180 degrees.

hmiSymbolLibraryRotation270Degree (270): The object rotates by 270 degrees.

RotationAngle

Description

Specifies the angle of rotation in degrees.

Access in runtime: Read and write

Syntax

Object.**RotationAngle**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Line
- Polygon
- Polyline
- TextField
- TubeTeeObject

Int32

Optional A value or a constant that specifies the angle of rotation in degrees.

Comments

In Runtime the object rotates clockwise on the reference point.

RotationCenterLeft

Description

Establishes the X coordinates of the pivot point to rotate the object in Runtime.

Access in Runtime: Read and write

Syntax

Object.**RotationCenterLeft**[=Int32]

Object

Required. An object of the "ScreenItem" type with the format:

- Line
- Polygon
- Polyline
- TextField

Int32

Optional A value or a constant that specifies the X coordinates of the pivot point to rotate the object in runtime.

Comments

The value of the X coordinate is relative to the object width. Specify the value as a percentage from the left edge of the rectangle that surrounds the object.

RotationCenterTop

Description

Specifies the Y coordinates of the pivot point to rotate the object in Runtime.

Access in Runtime: Read and write

Syntax

Object.**RotationCenterTop**[=Int32]

Object

Required. An object of the "ScreenItem" type with the format:

- Line
- Polygon
- Polyline
- TextField

Int32

Optional A value or a constant that specifies the Y coordinates of the pivot point to rotate the object in Runtime.

Comments

The value of the Y coordinate is relative to the object width. Specify the value as a percentage from the upper edge of the rectangle that surrounds the object.

RoundCornerHeight

Description

Specifies the corner radius. Enter the value as a percentage of half the height of the object.

Access in runtime: Read and write

Syntax

Object.**RoundCornerHeight**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Rectangle

Int32

Optional A value or a constant that specifies the corner radius.

RoundCornerWidth

Description

Specifies the corner radius. Enter the value as a percentage of half the width of the object.

Access in runtime: Read and write

Syntax

Object.**RoundCornerWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Rectangle

Int32

Optional A value or a constant that specifies the corner radius.

RowScrollbar

Description

Enables the display of row scroll bars.

Access in Runtime: Read and write

Syntax

Object.**RowScrollbar**[=ScrollbarVisibility]

Object

Required A "ScreenItem" object with following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

ScrollbarVisibility

Value	Description	Explanation
0	No	No row scroll bars.
1	as required	Row scroll bars are displayed if horizontal space requirements of the control are greater than the actually available display area.
2	always	Row scroll bars are always displayed.

RowTitleAlignment

Description

Specifies the type of row label alignment.

The following settings are available:

Value	Description	Explanation
0	Left	The row headers are aligned left.
1	Centered	The row headers are aligned to center.
2	Right	The row headers are aligned right.

The attribute can be assigned dynamic properties by means of the name **RowTitleAlignment** .

See also

AlarmControl (Page 5018)

RTPersistence

Description

Specifies how online configurations of WinCC are retained.

Access in runtime: Read and write

Syntax

Object.**RTPersistence**[=RTPersistence]

Object

Required. An object of the "ScreenItem" type with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

You have no access in runtime with the following format:

- SysDiagControl
- UserView

RTPersistence

Value	Description	Explanation
0	Discard	The current online configurations are discarded at the next picture change.
1	Retain	The current online configurations are retained at the next picture change.
2	Reset	All online configurations made are lost. The picture is set to the contents found in the configuration system.

RTPersistenceType

Description

Specifies how online configurations of WinCC are retained.

Syntax

Object.**RTPersistenceType**[=RTPersistenceType]

Object

Required. An object of the "ScreenItem" type with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

You have no access in runtime with the following format:

- SysDiagControl
- UserView

RTPersistenceType

Value	Description	Explanation
0	Do not retain	Online configurations are not retained. These are lost at the next picture change.
1	Retain during Runtime	Online configurations are retained during Runtime. These are lost on exiting.
2	Retain permanently	Online configurations are retained permanently. These are also available after restart.

RulerColor

Description

Specifies the color of the scale gradation (help line) of the axis label in the "OnlineTrendControl" object.

Access in runtime: Read and write

Syntax

Object.**RulerColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- TrendView

Color

Optional A value or a constant that specifies the color of the scale gradation.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

Properties S

ScaleColor

Description

Specifies the color of the scale of the selected object.

Access in runtime: Read and write

Syntax

Object.**ScaleColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- TrendView
- Bar

Color

Optional A value or a constant that specifies the color of the scale.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

ScreenItem object with the format "Bar": For the color to be displayed, the property ""ShowScale" must have the value TRUE.

ScaleGradation

Description

Specifies the distance between two large marking lengths of the scale.

Access in runtime: Read and write

Syntax

Object.**ScaleGradation**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the distance between two large marking lengths of the scale.

ScaleLabelColor

Description

Specifies the color of the label for the scale gradation of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**ScaleLabelColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

Color

Optional A value or a constant that specifies the label color of the scale gradation.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

ScaleNumerator

Description

No access in runtime.

ScalePosition

Description

Specifies the position of the scale of the selected object. The "ShowScale" property must be set to TRUE so that the scale can be displayed.

Access in Runtime: Read and write

Syntax

Object.**ScalePosition**[=ScalePosition]

Object

Required. An object of the "ScreenItem" type with the format:

- Bar

You have no access in runtime with the following format:

- Slider

ScalePosition

hmiScalePositionLeftUp (0): For a vertical bar, the scale is displayed at the top. For a horizontal bar the scale will be shown at the left.

hmiScalePositionRightDown (1): For a vertical bar, the scale is displayed at the bottom. For a horizontal bar the scale will be shown at the right.

ScaleTickColor

Description

Specifies the color of the scale gradation of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**ScaleTickColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

Gauge

Color

Optional A value or a constant that specifies the color of the scale gradation.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

ScaleTickLabelPosition

Description

Specifies the diameter of the assumed circle on which the label of the scale divisions is located.

Access in runtime: Read and write

Syntax

Object.**ScaleTickLabelPosition**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the diameter of the assumed circle on which the label of the scale division is located.

Value range from 0 to 1

0: The label is located in the center of the scale.

1: The diameter of the assumed circle for the label is the smaller value of the geometry properties "Width" and "Height". A part of the label can lie outside the object limits and is therefore invisible.

ScaleTickLength

Description

Specifies the length of the main markings for the scale division. The value refers to half the smaller value of the geometry properties "Width" and "Height".

The length of the marking lengths for the fine divisions is 0.5* scale width.

Access in runtime: Read and write

Syntax

Object.**ScaleTickLength**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the length of the main markings of the scale.

Value range 0 to scale end value

0: There are no scale divisions. The divisions of the scale in the areas are not visible.

Scale end value: The scale division ranges from the mid-point of the scale disc to the value specified by the scale end value.

ScaleTickPosition

Description

Specifies the diameter of the assumed circle on which the scale divisions are located.

The main markings of the scale divisions lie with their outward-facing ends on this circle.

Access in runtime: Read and write

Syntax

Object.**ScaleTickPosition**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the diameter of the assumed circle on which the scale divisions are located.

Value range from 0 to 1

0: The scale divisions are located in the center of the scale.

1: The diameter of the assumed circle for the label is the scale divisions is the smaller value of the geometry properties ""Width" and "Height".

ScaleDenominator

Description

Defines the counter for scaling on the client.

Syntax

Object.ScaleDenominator=[Int]

Object

Required. A "ScreenItem" object with the format "SmartClientView".

Int

Optional. A value or a constant that specifies the value.

See also

SmartClientView (Page 5162)

Scaling

Description

No access in runtime.

ScalingType

Description

Specifies the type of bar scaling.

Access in Runtime: Read and write

Syntax

Object.**ScalingType**[=BarScalingType]

Object

Required A "ScreenItem" object with following format:

- Bar

BarScalingType

Optional A value or a constant that specifies the type of bar scaling.

- hmiBarScalingLinear (0): Linear
  The large marks are distributed evenly over the scale. The distance between the large marks corresponds to the value of the "axis section" attribute.
- hmiBarScalingLogarithmic (1): Logarithmic
  The distribution of the large marks on the scale follows a logarithmic function.
  The representation of low values is very strongly highlighted.
- hmiBarScalingNegativeLogarithmic (2): Negative logarithmic
  The distribution of the large marks on the scale follows a negative logarithmic function. The representation of high values is very strongly highlighted.
- hmiBarScalingAutomatic (3): Automatic
  The large marks are distributed evenly over the scale. The distance between the large marks is specified automatically.

- hmiBarScalingTangent (4): Tangents
  The distribution of the large marks on the scale highlights the representation of the low and high values.
- hmiBarScalingQuadratic (5): Square
  The distribution of the large marks follows a square function. The representation of high values is highlighted.
- hmiBarScalingCubic (6): Cubic
  The distribution of the large marks on the scale follows a cubic function. This highlights the representation of large values.

Comments

For the color to be displayed, the property "ShowScale" must have the value TRUE.

See also

Bar (Page 5033)

ScreenItems

Description

Returns the ScreenItems list.

Access in Runtime: Read

Syntax

Object.ScreenItems

Object

Required A "ScreenItems"." object.

ScreenName

Description

Specifies the screen to be displayed in the screen window in runtime.

Access in runtime: Read and write

Syntax

Object.**ScreenName**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the following format:

- ScreenWindow

HmiObjectHandle

Optional A value or a constant that specifies the screen to be displayed in the screen window in runtime.

Screens

Description

Returns the Screens list. The Screens list contains two elements: The first element with the index 0 represents the permanent window. The second element with the index 1 represents the root screen. Alternatively, the two elements can be addressed by means of their names. Use "Overview" for the permanent window and "Base" for the root screen.

Note

The alarm window and the alarm indicator are not contained in the Screens list, even if they have the focus in Runtime.

Access in Runtime: Read

Syntax

Object.Screens

Object

Required A "Screens" object.

SecondNeedleHeight

Description

Specifies the length of the second hand in the "Clock" object.

Access in runtime: Read and write

Syntax

Object.**SecondNeedleHeight**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Clock

Int32

Optional A value or a constant that specifies the length of the second hand.

Specify the length of the second hand as a percentage relating to the radius of the clock face.

SecondNeedleWidth

Description

Specifies the width of the second hand in the "Clock" object.

Access in runtime: Read and write

Syntax

Object.**SecondNeedleWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Clock

Int32

Optional A value or a constant that specifies the width of the second hand. Specify the width as a percentage relating to double the length of the second hand.

SegmentColoring

Description

Specifies the type of color change that should be displayed in the "Bar" if the limits are exceeded.

Access in runtime: Read and write

Syntax

Object.**SegmentColoring**[=THmiBarColorType]

Object

Required. A "ScreenItem" object with the following format:

- Bar

THmiBarColorType

Optional A value or a constant that specifies the type of color change. Value range from 0 to 1.

hmiBarColorEntire (0): Color change applied to the entire bar.

hmiBarColorSegmented (1): Color change applied to segments.

SelectedText

Description

Shows the text defined with the "Selected field" (SelIndex) attribute which is highlighted in the combobox or list box.

See also

Listbox (Page 5104)

SelectBackColor

Description

Specifies the background color of the selected text entry for the selected object.

Access in runtime: Read and write

Syntax

Object.**SelectBackColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- SymbolicIOField

Color

Optional A value or a constant that specifies the background color of the selected text entry of the selected object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

SelectedCellColor

Description

Specifies the background color of the selected cell. You open a color selection dialog box with the button.

Access in runtime: Read and write

Syntax

Object.**SelectedCellColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant which specifies the background color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

SelectedCellForeColor

Description

Specifies the font color of the selected cell. You open a color selection dialog box with the button.

Access in runtime: Read and write

Syntax

Object.**SelectedCellForeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant which specifies the font color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

SelectedIndex

Description

Defines the index of which the associated text is highlighted in the combo box or list box.

The maximum value is the number of lines (NumberLines) of the object.

Access in runtime: Read and write

Syntax

Object.**SelectedIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- ComboBox
- ListBox

Int32

Optional A value or a constant that specifies the index of the text highlighted.

SelectedRowColor

Description

Specifies the background color of the selected row. You open a color selection dialog box with the button.

Access in runtime: Read and write

Syntax

Object.**SelectedRowColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant which specifies the background color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

SelectedRowForeColor

Description

Specifies the font color of the selected row. You open a color selection dialog box with the button.

Access in runtime: Read and write

Syntax

Object.**SelectedRowForeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant which specifies the font color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

SelectedTitleColor

Description

Specifies the background color of the selected table header. You open a color selection dialog box with the button.

The setting is only effective in Runtime when the "Marking color" option is activated.

Access in runtime: Read and write

Syntax

Object.**SelectedTitleColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant which specifies the background color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

SelectedTitleForeColor

Description

Specifies the font color of the selected table header. You open a color selection dialog box with the button.

The setting is only effective in Runtime when the "Marking color" option is activated.

Access in runtime: Read and write

Syntax

Object.**SelectedTitleForeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant which specifies the font color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

See also

UserArchiveControl (Page 5207)

TrendRulerControl (Page 5187)

OnlineTableControl (Page 5114)

AlarmControl (Page 5018)

SelectForeColor

Description

Specifies the color of the selected text entry for the selected object.

Access in runtime: Read and write

Syntax

Object.**SelectForeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- SymbolicIOField

Color

Optional A value or a constant that specifies the color of the selected text entry of the selected object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

SelectionBackColor

Description

Determines the background color of the selected cells.

Access in runtime: Read and write

Syntax

Object.**SelectionBackColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- AlarmView
- RecipeView
- StatusForce
- UserView

You have no access in runtime with the following format:

- TrendView

Color

Optional A value or constant that specifies the background color of the selected row.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

SelectionColoring

Description

Enables the use of selection colors for cells or rows.

Access in Runtime: Read and write

Syntax

Object.**SelectionColoring**[=GridSelectionColoring]

Object

Required A "ScreenItem" object with following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSelectionColoring

Value	Description	Explanation
0	None	No selection colors for cells and rows.
1	Cell	Selection color for cell.
2	Row	Selection color for row.
3	Cell and row	Selection colors for cell and row.

SelectionForeColor

Description

Determines the foreground color of the selected cells.

Access in runtime: Read and write

Syntax

Object.**SelectionForeColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- AlarmView
- RecipeView
- StatusForce
- UserView

You have no access in runtime with the following format:

- TrendView

Color

Optional A value or constant that specifies the background color of the selected row.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

SelectionRect

Description

Enables the use of a selection border for selected cells or rows.

Access in Runtime: Read and write

Syntax

Object.**SelectionRect**[=GridSelectionBorder]

Object

Required A "ScreenItem" object with following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSelectionBorder

Value	Description	Explanation
0	None	No selection border is drawn for selected cells or rows.
1	Cell	A selection border is drawn for the selected cell.
2	Row	A selection border is drawn for the selected row.

SelectionRectColor

Description

Specifies the color of the rectangle in the alarm window if SelectionType is "1".

Access in runtime: Read and write

Syntax

Object.**SelectionRectColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant which specifies the color.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

SelectionRectWidth

Description

Specifies the line width of the selection rectangle in the alarm window if SelectionType equals "1".

Access in runtime: Read and write

Syntax

Object.**SelectionRectWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or a constant which specifies the line width.

SelectionType

Description

Specifies the number of lines you can mark.

Access in runtime: Read and write

Syntax

Object.**SelectionType**[=GridSelectionType]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSelectionType

Optional A value or a constant that specifies the number of lines that you can select.

The following settings are available:

Value	Description	Explanation
0	None	No line is selected.
1	Single selection	One line can be selected.
2	Multiple selection	Several lines can be selected.

SeparatorBackColor

Description

Specifies the background color of the broken separation lines in the selection list of the specified object.

Access in runtime: Read and write

Syntax

Object.**SeparatorBackColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- SymbolicIOField

Color

Optional A value or a constant that specifies the background color of the broken separation lines in the selection list of the specified object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

SeparatorColor

Description

Specifies the color of the separation lines in the selection list of the specified object.

Access in runtime: Read and write

Syntax

Object.**SeparatorColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- SymbolicIOField

You have no access in runtime with the following format:

- S7GraphOverview

Color

Optional A value or a constant that specifies the color of the separation lines in the selection list of the specified object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

SeparatorCornerStyle

Description

No access in runtime

SeparatorStyle

Description

Specifies the line style of the separation lines in the selection list of the selected object.

Access in runtime: Read and write

Syntax

Object.**SeparatorStyle**[=LineStyle]

Object

Required. A "ScreenItem" object with the following format:

- SymbolicIOField

LineStyle

hmiLineStyleNone (-1): The selection list has no separation lines.

hmiLineStyleSolid (0): The selection list has solid separation lines.

hmiLineStyleDash (1): The selection list has dashed separation lines.

hmiLineStyleDot (2): The selection list has dotted separation lines.

hmiLineStyleDashDot (3): The selection list has dash - dot lines as separation lines.

hmiLineStyleDashDotDot (4): The selection list has dash - dot - dot lines as separation lines.

SeparatorWidth

Description

Specifies the width of the separation lines in the selection list of the specified object.

Access in runtime: Read and write

Syntax

Object.**SeparatorWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- SymbolicIOField

Int32

Optional A value or a constant that specifies the width of the separation lines in the selection list of the specified object.

SeparatorLineEndShapeStyle

Description

Specifies the shape of the line ends for the object of the type "ScreenItem" with the format "SymbolicIOField".

See also

SymbolicIOField (Page 5170)

ServerScale

Description

No access in runtime.

ServerNames

Description

Specifies the servers of a distributed system from which the alarm view draws data. The information is given in form of: NameServer1;NameServer2;NameServer3.

Access in runtime: Read and write

Syntax

Object.**ServerNames**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

STRING

Optional A value or a constant that specifies the server of a distributed system from which the alarm window receives information.

ServerPrefix

Description

No access in runtime

Shared

Description

No access in runtime.

ShiftDecimalPoint

Description

Specifies that the "Shift decimal point" field can only be read.

Access in Runtime: Read and write

Syntax

Object.**ShiftDecimalPoint**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format "IOField".

BOOLEAN

Optional. TRUE, if the "Shift decimal point" field can only be read.

See also

IOField (Page 5098)

ShowAlarmsFromDate

Description

Specifies that only those message events are displayed that are saved in this tag.

Access in runtime: Read and write

Syntax

Object.**ShowAlarmsFromDate**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the format:

- AlarmView

HmiObjectHandle

Optional A value or a constant which specifies that only those message events are displayed that are saved in this tag.

ShowBadTagState

Description

Defines whether the object is grayed out when a bad quality code or tag status is detected.

Access in runtime: Read and write

Syntax

Object.**ShowBadTagState**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar
- IOField
- OptionGroup
- SymbolicIOField
- WindowsSlider

You have no access in runtime with the following formats:

- CheckBox

BOOLEAN

TRUE	If the quality code or the the tag status are poor, the object is grayed out or the settings for the grid color are used.
FALSE	If the quality code or the the tag status are poor, the object is not grayed out or the settings for the grid color are not used.

ShowBar

Description

Specifies whether the displayed process value in the "Slider" object is also shown by a filled bar.

Access in runtime: Read and write

Syntax

Object.**ShowBar**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Slider

BOOLEAN

Optional TRUE, if the process value is also shown by a filled bar.

ShowCaption

Description

Specifies whether the title line is hidden or shown.

Access in runtime: Read and write

Syntax

Object.**ShowCaption**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- ApplicationWindow
- Screenwindow

You have no access in runtime with the following format:

- Switch

BOOLEAN

Optional TRUE, if the title line is shown.

ShowDecimalPoint

Description

Specifies whether the scale is identified with decimal figures (decimal point and one decimal place) or with whole integers.

Access in runtime: Read and write

Syntax

Object.**ShowDecimalPoint**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

BOOLEAN

Optional TRUE, if the scale is identified with decimal figures (decimal point and one decimal place).

ShowFillLevel

Description

Specifies whether the selected object is filled.

Access in runtime: Read and write

Syntax

Object.**ShowFillLevel**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- Button
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicView
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- TextField
- WindowsSlider

You have no access in runtime with the following format:

- CheckBox

BOOLEAN

Optional TRUE, if the selected object is filled.

ShowFocusRectangle

Description

Specifies whether the button is given a selection border when it is activated in Runtime.

Access in runtime: Read and write

Syntax

Object.**ShowFocusRectangle**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Clock

BOOLEAN

Optional TRUE, if the button is given a selection border when it is activated in Runtime.

ShowLargeTicksOnly

Description

Specifies whether only large scale marks are shown.

Access in runtime: Read and write

Syntax

Object.**ShowLargeTicksOnly**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE if only large scale marks are shown.

ShowLimitMarkers

Description

Specifies whether the limit value is shown as a scale value.

Access in runtime: Read and write

Syntax

Object.**ShowLimitMarkers**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- Bar

You have no access in runtime with the following format:

- Slider

BOOLEAN

Optional TRUE, if the limit value is shown as a scale value.

ShowPeakValuePointer

Description

Specifies whether a slave pointer will be used for the selected object.

The slave pointer shows the maximum pointer deflection provided in Runtime while the process screen is loaded. If you reload the process screen, the slave pointer will be reset.

Access in runtime: Read and write

Syntax

Object.**ShowPeakValuePointer**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

BOOLEAN

Optional TRUE, if the slave pointer is used.

ShowPosition

Description

Specifies whether the value of the current slider position should also be displayed numerically. The value is then displayed beneath the slider.

Access in runtime: Read and write

Syntax

Object.**ShowPosition**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Slider

BOOLEAN

Optional TRUE if the value is also shown numerically.

RowTitles

Description

Specifies whether the message view contains a column with consecutive numbering of the existing messages.

Access in Runtime: Read and write

Syntax

Object.**RowTitles**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format "AlarmControl".

BOOLEAN

Optional. TRUE, if the message view contains a column with consecutive numbering of the existing messages.

See also

AlarmControl (Page 5018)

ShowRuler

Description

Specifies whether a scale division (help line) is displayed for an axis label of the object "OnlineTrendControl".

Access in runtime: Read and write

Syntax

Object.**ShowRuler**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl
- TrendView

BOOLEAN

Optional TRUE, if the scale gradation is shown.

ShowRulerInAxis

Description

Enables the display of rulers in the time axis.

Access in runtime: Read and write

Syntax

Object.**ShowRulerInAxis**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The rulers are also displayed in the time axes.

FALSE: The rulers are not displayed in the time axes.

ShowScale

Description

Specifies whether the values will also be shown in a scale.

Access in runtime: Read and write

Syntax

Object.**ShowScale**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- Bar

You have no access in runtime with the following format:

- Slider

BOOLEAN

Optional TRUE, if the values will also be shown in a scale.

ShowScrollbars

Description

Enables the display of scroll bars.

Access in runtime: Read and write

Syntax

Object.**ShowScrollbars**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the format:

- FunctionTrendControl
- OnlineTrendControl
- Screenwindow

ShowScrollbars

Optional. TRUE if scroll bars are shown.

ShowSortButton

Description

Enables the display of a sorting button above the vertical scroll bar. Click this sorting button to sort the selected column based on the configured sorting criteria. The sorting button is not displayed if the table does not contain a vertical scroll bar.

Access in runtime: Read and write

Syntax

Object.**ShowSortButton**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The sorting key is displayed. You can sort the selected column.

FALSE: The sorting key is not displayed.

ShowSortIcon

Description

Enables the display of the sorting icon.

Access in runtime: Read and write

Syntax

Object.**ShowSortIcon**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The sorting icon is displayed.

FALSE: The sorting icon is not displayed.

ShowSortIndex

Description

Specifies whether a sort index is displayed.

Access in runtime: Read and write

Syntax

Object.**ShowSortIndex**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: A sort index is displayed.

FALSE: A sort index is not displayed.

ShowStatusBar

Description

Specifies whether the status bar is shown.

Access in runtime: Read and write

Syntax

Object.**ShowStatusBar**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- HTML-Browser

You have no access in runtime with the following format:

- MediaPlayer

BOOLEAN

Optional TRUE, if the status bar is shown.

ShowTableGridlines

Description

Determines whether gridlines are shown in the table of the given object.

Access in runtime: Read and write

Syntax

Object.**ShowTableGridlines**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- StatusForce

You have no access in runtime with the following format:

- TrendView
- UserView

BOOLEAN

Optional TRUE if gridlines in the table are shown.

ShowThumb

Description

Specifies whether the slider of the "Slider" object will be displayed.

Access in runtime: Read and write

Syntax

Object.**ShowThumb**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Slider

BOOLEAN

Optional TRUE, if the slider is shown.

ShowTickLabels

Description

Specifies whether the label is shown in the scale.

Access in runtime: Read and write

Syntax

Object.**ShowTickLabels**[= BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the format:

- Slider

You have no access in runtime with the following format:

- Bar

BOOLEAN

Optional TRUE, if the label is shown.

Comments

The increments for the measured value are automatically established dependent upon the specified measurement range and the size of the object.

ShowTicks

Description

Specifies whether the marking lengths are shown in the scale of the selected object.

Access in runtime: Read and write

Syntax

Object.**ShowTicks**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Clock
- Slider

BOOLEAN

Optional TRUE, if the marking lengths are shown.

ShowTitle

Description

Defines representation the Control window header.

Access in Runtime: Read and write

Syntax

Object.**ShowTitle**[=WindowHeaderStyle]

Object

Required A "ScreenItem" object with following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

WindowHeaderStyle

Value	Designation	Description
0	No	No window title.
1	Normal	The window title consists of a WinCC icon and text. The text is entered in the "Text" field.
2	Narrow	The window title consists only of text. The text is entered in the "Text" field.

ShowToolBar

Description

Specifies whether the toolbar is shown.

Access in runtime: Read and write

Syntax

Object.**ShowToolBar**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- HTMLBrowser

BOOLEAN

Optional TRUE, if a toolbar will be displayed.

ShowTrendIcon

Description

Enables the display of an icon below the value axes. The icon indicates the trend currently displayed in the foreground.

Access in runtime: Read and write

Syntax

Object.**ShowTrendIcon**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The icon is displayed.

FALSE: The icon is not displayed.

ShowTrendIndicator

Description

Specifies whether the tendency (increasing or falling) of the measurement value that is to be monitored is shown with a small arrow.

Access in runtime: Read and write

Syntax

Object.**ShowTrendIndicator**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the tendency (increasing or falling) of the measurement value that is to be monitored is shown with a small arrow.

Sizeable

Description

TRUE, when it should be possible to change the size of the object in Runtime. BOOLEAN write-read access.

In the case of application window and picture window: Read only access

See also

TrendRulerControl (Page 5187)

OnlineTableControl (Page 5114)

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

ControlDesignMode

Description

The control style can be defined in this selection field.

Access in Runtime: Write

Syntax

Object.**ControlDesignMode**[=RTControlModes]

Object

Required. An object of the "ScreenItem" type with the format "MessageView", "OnlineTableControl", "OnlineTrendControl", "FunctionTrendControl", "RecipeView", "RulerControl".

RTControlModes

The following settings are available:

Value	Designation	Description
	Project setting	The style corresponds to the project settings in WinCC Explorer.
0	Classic	"Classic" WinCC style
1	Standard	New WinCC V7 style

SmartTags

Description

Returns the SmartTags list.

Access in Runtime: Read

Syntax

Object.SmartTags

Object

Required A ""HMIRuntime" object.

SortByTimeDirection

Description

Specifies whether the last incoming message is shown at the top of the "AlarmControl" object (ascending sorting order).

Access in Runtime: Read and write

Syntax

Object.**SortByTimeDirection**[=SortByTimeDirection]

Object

Required A "ScreenItems" object with following format:

- AlarmView

SortByTimeDirection

Optional TRUE, if the last incoming alarm is shown at the top.

SortByTimeEnabled

Description

Specifies whether the sorting of the messages can be altered according to the time in the "AlarmView" object.

Access in runtime: Read and write

Syntax

Object.**SortByTimeEnabled**[=BOOLEAN]

Object

Required. A "ScreenItems" object with the following format:

- AlarmView

BOOLEAN

Optional TRUE, if the sorting can be altered by the device operator.

SortSequence

Description

Specifies how to change the sorting order by mouse click.

Access in Runtime: Read and write

Syntax

Object.**SortSequence**[=GridSortSequence]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveConrol

GridSortSequence

Value	Description	Explanation
0	Up/down/none	You can toggle between ascending, descending and no sorting by means of mouse click.
1	Up/down	You can toggle between ascending and descending sorting order by means of mouse click.

StartAngle

Description

Specifies the angle at which the start point of the selected object deviates from the zero position (0°).

Access in runtime: Read and write

Syntax

Object.**StartAngle**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- CircleSegment
- CircularArc
- EllipseSegment
- EllipticalArc
- TubeArcObject

Int32

Optional A value or a constant that specifies the angle at which the start point of the specified object deviates from the zero position (0°).

StartStyle

Description

Determines how the line start of the given object is displayed.

Access in runtime: Read and write

Syntax

Object.**StartStyle**[=LineEndStyle]

Object

Required. A "ScreenItem" object with the following format:

- Line
- Polyline

LineEndStyle

Optional A value or constant which determines the line start. Value range from 0 to 6.

hmiLineEndStyleNone (0): The line has no start symbol.

hmiLineEndStyleArrow (1): The line starts with an arrow.

hmiLineEndStyleFilledArrow (2): The line starts with a filled arrow.

hmiLineEndStyleFilledArrowReversed (3): The line starts with a reversed arrow.

hmiLineEndStyleLine (4): The line starts with a vertical line.

hmiLineEndStyleCircle (5): The line starts with a circle.

hmiLineEndStyleFilledCircle (6): The line starts with a filled circle.

StartTop

Description

No access in runtime.

StartValue

Description

Defines the absolute value of the zero point of the scale indicator.

Access in runtime: Read and write

Syntax

Object.**StartValue**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the absolute value of the zero point for the scale indicator.

State

Description

Returns the status of a message.

The following table shows the possible states of a message:

State	Alarm Log Status
1	Came In
2	Went Out
5	Came in and comment
6	Gone and comment

StatusbarBackColor

Description

Specifies the background color for status bars. You open a color selection dialog box with the button. Enable the "Display" option to active the setting.

Access in runtime: Read and write

Syntax

Object.**StatusbarBackColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant that specifies the background color of the status bar.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

StatusbarElementAdd

Description

Creates a new, user-defined status bar element. You can change the name assigned by WinCC in the "Object name" field.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementAdd**[=STRING]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that specifies a new, user-defined element in the status bar.

StatusbarElementAutoSize

Description

Enables autosizing of the width of a status bar element selected.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementAutoSize**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The width of the selected element is set automatically.

FALSE: The width of the selected element is not set automatically.

See also

UserArchiveControl (Page 5207)

TrendRulerControl (Page 5187)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

StatusbarElementCount

Description

Specifies the number of configurable status bar elements.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementCount**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- FunctionTrendControl

- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or a constant that specifies the number of configurable status bar elements.

StatusbarElementIconId

Description

Specifies the ID number and symbol assignment of a status bar element.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementIconId**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or constant that specifies the ID number and symbol assignment of a status bar element.

StatusbarElementID

Description

Specifies the ID number of the selected status bar element.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementID**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or constant that specifies the ID number of the selected status bar element.

StatusbarElementIndex

Description

Specifies the reference for a status bar element.

Values between 0 and "StatusbarElementCount minus 1 are valid for "StatusbarElementIndex".

Access in runtime: Read and write

Syntax

Object.**StatusbarElementIndex**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or a constant that specifies the reference for a status bar element.

StatusbarElementName

Description

Specifies the object name of the selected status bar element.

You can rename the objects of custom status bar elements. The "StatusbarElementName" property for user-defined elements can be dynamized with the "StatusbarElementRename" property.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementName**[=STRING]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or constant that specifies the object name of the selected status bar element.

StatusbarElementRemove

Description

Removes the selected user-defined status bar element.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementRemove**[=STRING]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl

- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that specifies the name of the user-defined status bar element to be removed.

StatusbarElementRename

Description

Changes the name of the user-defined status bar element that is referenced by the "StatusbarElementIndex" property.

The property with the name "StatusbarElementRename" can be dynamized for user-defined elements. With "StatusbarElementRename" you also dynamize the "StatusbarElementName" property.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementRename**[=STRING]

Object

Required. An object of the "ScreenItem" type with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or constant that specifies the new object name of the selected status bar element.

StatusbarElements

Description

Specifies the elements that are to be shown in the status bar.

No access in runtime.

StatusbarElementTooltipText

Description

Specifies the tooltip text for the selected user-defined status bar element.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementTooltipText**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that specifies the tooltip text for the selected user-defined status bar element.

StatusbarElementUserDefined

Description

Specifies whether the configuration engineer has added the status bar element as a new user-defined element.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementUserDefined**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The element of the status bar is customized.

FALSE: The element of the status bar is specified by the system.

StatusbarElementVisible

Description

Specifies whether the selected user-defined status bar element is displayed in runtime.

In the list, select the status bar elements that you want to display in runtime.

Click a list entry to adapt the properties, or to change its position in the status bar of the Control by means of the "Up" and "Down" buttons.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementVisible**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The selected user-defined status bar element is displayed.

FALSE: The selected user-defined status bar element is not displayed.

StatusbarElementWidth

Description

Defines the width of the selected status bar element in pixels.

You can define the width if the "Automatic" option is not activated.

Access in runtime: Read and write

Syntax

Object.**StatusbarElementWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or constant that defines the width of the selected user-defined status bar element in pixels.

StatusbarFontColor

Description

Specifies the color of the texts in the status bar.

Access in runtime: Read and write

Syntax

Object.**StatusbarFontColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or constant that specifies the color of the texts in the status bar.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

StatusbarShowTooltips

Description

Enables the display of tooltips for the status bar elements in Runtime.

Access in runtime: Read and write

Syntax

Object.**StatusbarShowTooltips**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The tooltips are displayed.

FALSE: The tooltips are not displayed.

StatusbarText

Description

Specifies the default text in the status bar.

Access in runtime: Read and write

Syntax

Object.**StatusbarText**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that specifies the default text in the status bar.

StatusbarUseBackColor

Description

Sets a background color for the status bar.

Access in runtime: Read and write

Syntax

Object.**StatusbarUseBackColor**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The background color of the status bar is displayed.

FALSE: The background color of the status bar is not displayed.

StatusbarVisible

Description

Enables the display of the status bar of a control.

Access in Runtime: Read and write

Syntax

Object.**StatusbarVisible**[=BOOLEAN]

Object

Required A "ScreenItem" object with following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The status bar is displayed.

FALSE: The status bar is not displayed.

Style

Description

Specifies the line style of the selected object.

Access in runtime: Read and write

Syntax

Object.**Style**[=LineStyle]

Object

Required. A "ScreenItem" object with the following format:

- CircularArc
- EllipticalArc
- Line
- Polyline

LineStyle

Optional A value or a constant that specifies the line style. Value range from 0 to 4.

hmiLineStyleSolid (0): solid line

hmiLineStyleDash (1): broken line

hmiLineStyleDot (2): dotted line

hmiLineStyleDashDot (3): dash - dot line

hmiLineStyleDashDotDot (4): dash - dot - dot line

Default setting: hmiLineStyleSolid

StyleSettings

Description

Defines the style in which the object is displayed.

Access in runtime: Read and write

Syntax

Object.**StyleSettings**[=WinCCStyle]

Object

Required. An object of the "ScreenItem" type with the following format:

- Button
- RoundButton
- WindowsSlider

WinCCStyle

Optional. A value or a constant that specifies the style in which the object is displayed.

User Defined	Shows the object according to the respective settings.
Global	Shows the object in a globally defined design.
Windows Style	Shows the object in Windows style.

SwapFirstWithLastConnection

Description

Specifies whether the text in the object is shown horizontally.

Access in Runtime: Read and write

Syntax

Object.**SwapFirstWithLastConnection**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Connector".".

BOOLEAN

Optional TRUE, if the text in the object is shown horizontally.

See also

Connector (Page 5058)

Properties T

TableBackColor

Description

Determines the background color of the table cells of the given object.

Access in runtime: Read and write

Syntax

Object.**TableBackColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmView
- RecipeView
- StatusForce
- TrendView
- UserView

Color

Optional A value or constant which determines the background color of the table cells.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TableColor

Description

Specifies the background color of the rows.

You open a color selection dialog box with the button.

Access in runtime: Read and write

Syntax

Object.**TableColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or constant that specifies the background color of the table rows.

Comments

You can use the "RGB" function to define the colors in RGB format (red, green, blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TableColor2

Description

Specifies the second background color of the rows.

You open a color selection dialog box with the button.

The settings are only active in Runtime if the "Row color 2" option is enabled. "Row color 1" and "Row color 2" are then used alternately for the row background.

Access in runtime: Read and write

Syntax

Object.**TableColor2**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or constant that specifies the second background color of the table rows.

Comments

You can use the "RGB" function to define the colors in RGB format (red, green, blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TableForeColor

Description

Specifies the text color used in the table cells of the selected object.

Access in runtime: Read and write

Syntax

Object.**TableForeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- AlarmView
- OnlineTableControl
- RecipeView
- StatusForce
- TrendRulerControl
- UserArchiveControl
- UserView

Color

Optional A value or a constant that specifies the text color used in the table cells.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TableForeColor2

Description

Specifies the second font color.

You open a color selection dialog box with the button.

The settings are only active in Runtime if the "Row color 2" option is enabled. The "Row color 1" and "Row color 2" font colors are then used alternately.

Access in runtime: Read and write

Syntax

Object.**TableForeColor2**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the second font color used in the table cells.

Comments

You can use the "RGB" function to define the color in RGB format (red, green blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TableGridLineColor

Description

Determines the color of the gridlines in the table of the given object.

Access during runtime: Read and Write

Syntax

Object.**TableGridLineColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the format:

- RecipeView
- TrendView
- UserView

You have no access in runtime with the following format:

- StatusForce
- SysDiagControl

Color

Optional A value or constant which determines the color of the gridlines of the table.

Comments

You can use the "RGB" function to define the color in RGB format (red, green blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TableHeaderBackColor

Description

Specifies the background color in the header of the table of the selected object.

Access in runtime: Read and write

Syntax

Object.**TableHeaderBackColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmView
- RecipeView
- StatusForce
- TrendView
- UserView

Color

Optional A value or a constant that specifies the background color in the header.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TableHeaderForeColor

Description

Specifies the text color in the header of the table of the selected object.

Access in runtime: Read and write

Syntax

Object.**TableHeaderForeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmView
- RecipeView
- StatusForce
- TrendView
- UserView

Color

Optional A value or a constant that specifies the text color in the header.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TagPrefix

Description

Specifies a prefix for all tags used in the screen.

A tag prefix can be assigned for a screen window. This prefix is then prefixed to all tags used in the screen. In this way, a picture that is embedded in a picture window retains access to its own tags while another accesses other tags.

The change to the tag prefix does not become effective until the screen is reloaded. This happens automatically upon screen change; otherwise, the screen name must be reassigned.

The prefix is freely definable but must match the name of the structure tag.

Access in runtime: Read and write

Syntax

Object.**TagPrefix**[= STRING]

Object

Required. An object of the "ScreenItem" type with the following format:

- Screenwindow

STRING

Optional. A value or a constant that specifies the tag prefix.

Example

The "InOutput" screen should be displayed in the screen window. The "InOutput" screen contains three I/O fields that are linked to a structure tag. The structure tag consists of the elements IO1, IO2, IO3; one element for each I/O field.

Three such structure tags with the structure names Struct1, Struct2 and Struct3 are defined in the project, for example.

In this case, the tag prefix is the structure name with a following period. For example, if you specify Struct2. as the tag prefix (the period is necessary for addressing the elements of the structure tags as structure elements with the correct syntax), the I/O fields in the "InOutput" screen are linked to the elements of structure tag Struct2:

Tag prefix: "Struct2."

- Output value (first I/O field): EA1
- Output value (second I/O field): EA2
- Output value (third I/O field): EA3

The current tag binding in the screen window is therefore

- Output value (first I/O field): Struct2.EA1
- Output value (second I/O field): Struct2.EA2
- Output value (third I/O field): Struct2.EA3

Tags

Description

Returns an object of type "Tags".

Access during runtime: Read

Syntax

Object.**Tags**

Object

Necessary. An object of the "HMIRuntime" type.

Example

The following example accesses the tag "Tag1":

```
'VBS86
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
```

See also

HMIRuntime (Page 4993)

Template

Description

Specifies the template for displaying window content in the "Application window" object.

Access in runtime: Read and write

Syntax

Object.**Template**[=TemplateType]

Object

Required. An object of the "ScreenItem" type with the following format:

- ApplicationWindow

TemplateType

Optional. A value or a constant that specifies the template.

The following templates are possible depending on the property value:

Window Contents = Global Script

- "GSC diagnostics"
  The application window is supplied by applications of the Global Script. The results of the diagnosis system are displayed.
- "GSC Runtime"
  The application window is supplied by applications of the Global Script. The analysis results regarding characteristics in Runtime are displayed.

Window Contents = Print Jobs

- "All Jobs":
  The application window is supplied by the logging system. The available reports are displayed as a list.
- "All Jobs - Context Menu":
  The application window is supplied by the logging system. The available reports are displayed as a list. The shortcut menu enables the selection of print options, display of a print preview as well as a printout of the log.
- "Job Detail View":
  The application window is supplied by the logging system. The available reports are displayed in a selection menu. Detailed information is displayed for the selected report.
- "Selected Jobs - Context Menu":
  The application window is supplied by the logging system. The available reports are displayed as a list. This list only contains reports which you have activated the option "Mark for print job list" in the "Print Job Properties" dialog. The shortcut menu enables the selection of print options, display of a print preview as well as a printout of the log.

Text

Description

Specifies the labeling for the text field.

Access in runtime: Read and write

Syntax

Object.**Text**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- TextField
- CheckBox
- ComboBox
- ListBox
- MultiLineEdit

- OptionGroup
- RoundButton

STRING

Optional A value or a constant that specifies the labeling.

TextList

Description

Returns a list containing the output text actually to be output for each output value.

The assignments are dependent on the set list type. You define the list type with the "ListType" property.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read

Syntax

Object.**TextList**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the following format:

- SymbolicIOField

You have no access in runtime with the following formats:

- Button

HmiObjectHandle

A list which contains the assignments between the output value and the output text actually to be output.

TextOff

Description

Specifies the text that will be displayed in the "off" status of the selected object.

Access in runtime: Read and write

Syntax

Object.**TextOff**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- Button
- Switch

You have no access in runtime with the following formats:

- SymbolicIOField

STRING

Optional A value or a constant that specifies the labeling for the "off" status.

Comments

The property is only available if the referenced object "SymbolicIOField", Button" or "Switch" is of the "Text" type.

TextOn

Description

Specifies the text that will be displayed in the "on" status of the selected object.

Access in runtime: Read and write

Syntax

Object.**TextOn**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- Switch

You have no access in runtime with the following formats:

- Button
- SymbolicIOField

STRING

Optional A value or a constant that specifies the labeling for the "on" status.

Comments

The property is only available if the referenced object "SymbolicIOField", Button" or "Switch" is of the "Text" type.

TextOrientation

Description

Specifies the text direction (orientation) of the selected object.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read/Write

Syntax

Object.**TextOrientation**[=TextOrientation]

Object

Required. An object of the "ScreenItem" type with the following format:

- Button
- IOField
- OptionGroup
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

You have no access in runtime with the following formats:

- CheckBox
- DateTimeField
- Switch

TextOrientation

hmiTextHorizontal (0): The text is shown horizontally.

hmiTextRotated90Degree (-1): The text is shown vertically and left justified.

hmiTextRotated270Degree (1): The text is shown vertically and right-justified.

ThumbBackColor

Description

Specifies the background color of the slider in the ""Slider" object.

Access in runtime: Read and write

Syntax

Object.**ThumbBackColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Slider
- WindowsSlider

Color

Optional A value or a constant that specifies the background color of the slider.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TicksColor

Description

Specifies the color of the hour markers on the face of the "Clock" object.

Access in runtime: Read and write

Syntax

Object.**TicksColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Clock

Color

Optional A value or a constant that specifies the color of the hour markers.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TickStyle

Description

Specifies the scale display.

Access in runtime: Read and write

Syntax

Object.**TickStyle**[=SliderTickStyle]

Object

Required. A "ScreenItem" object with the following format:

- Slider

You have no access in runtime with the following formats:

- Clock

SliderTickStyle

hmiSliderTickStyleNone (0): The object has no scale.

hmiSliderTickStyleEffect1(1): The scale consists of main gradations only. The scale is black on a white background.

hmiSliderTickStyleEffect2 (2): The scale consists of main gradations only. The scale is white on a black background.

hmiSliderTickStyleNormal (3): The scale consists of simple gradations.

Comments

Because of automatic scaling, two scale marks may in some cases be positioned directly adjacent to one another (this looks like a thicker scale mark). You can correct this effect by slightly lengthening or shortening the slider object.

You you can also completely suppress the display of the scale ("WithAxes").

TimeAxisBeginTime(i)

Description

Specifies the start time for the display of the selected trend. The ""UseTimeRange(i)" and "ShareTimeAxis" properties determine whether or not the information is evaluated.

Access in runtime: Read and write

Syntax

Object.**TimeAxisBeginTime**(i)[=DateTime]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTrendControl

You have no access in runtime with the following format:

- TrendView

DateTime

Optional A value or a constant that specifies the start time for the display of the specified trend.

TimeAxisEndTime

Description

Specifies the end time for the display of the selected trend. Whether or not the information is evaluated depends on the "Autorange", "UseTimeRange(i)" and "ShareTimeAxis" properties.

Access in runtime: Read and write

Syntax

Object.**TimeAxisEndTime**[=DateTime]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTrendControl

DateTime

Optional A value or a constant that specifies the end time for the display of the specified trend.

TimeAxisLabel(i)

Description

Specifies the labeling of the time axis. Whether or not the information is evaluated depends on the "ConfigureTimeAxis(i)" property.

Access in runtime: Read and write

Syntax

Object.**TimeAxisLabel**(i)[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTrendControl

STRING

Optional A value or a constant that specifies the labeling of the time axis.

Comments

The parameter i indicates the number of the trend.

TimeAxisRange

Description

No access in runtime.

TimeAxisTimeFormat(i)

Description

Specifies the format of the information along the time axis for the selected trend.

Access in runtime: Read and write

Syntax

Object.**TimeAxisTimeFormat**(i)[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTrendControl

STRING

Optional A value or a constant which defines the format for the time axis.

Comments

The parameter i indicates the number of the trend.

TimeBase

Description

Specifies the time zone in which the time values will be displayed.

Access in Runtime: Read and write

Syntax

Object.**TimeBase**[=TimeBase]

Object

Required A "ScreenItem" object with following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- UserArchiveControl

TimeBase

hmiTimeBaseLocalTimezone (0): Local time

hmiTimeBaseServerTimezone (1): Time zone of the server

hmiTimeBaseUTC (2): UTC (Universal Time Coordinated)

hmiTimeBaseProjectSetting (3): Project settings

Comments

Set the time mode in accordance with the computer settings on the properties page of the computer in WinCC Explorer.

TimeColumnActualize

Description

Specifies whether the values of the selected column are updated.

Access in runtime: Read and write

Syntax

Object.**TimeColumnActualize**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

BOOLEAN

TRUE: The time column is actualized.

FALSE: The time column is not actualized. This setting can be useful when comparing tables.

TimeColumnAdd

Description

Creates a new time column.

Access in runtime: Read and write

Syntax

Object.**TimeColumnAdd**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

STRING

Optional A value or constant which specifies a new time column.

TimeColumnAlignment

Description

Defines the mode of alignment of the time column selected.

Access in runtime: Read and write

Syntax

Object.**TimeColumnAlignment** [=HorizontalAlignment]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

HorizontalAlignment

Value	Description	Explanation
0	Left	The time column selected is displayed on the left.
1	Centered	The time column selected is aligned to center.
2	Right	The time column selected is displayed on the right.

See also

OnlineTableControl (Page 5114)

TimeColumnAlignment(i)

Description

Specifies the alignment of the time column of a column pair i. The parameter i indicates the number of the column pair.

Access in Runtime: Read and write

Syntax

Object.**TimeColumnAlignment**(i)[=AlignmentHorizontal]

Object

Required A "ScreenItem" object with the format "TableView".".

AlignmentHorizontal

hmiAlignmentLeft (0): The text is shown left justified.

hmiAlignmentCentered (1): The text is shown centered.

hmiAlignmentRight (2): The text is shown right justified.

See also

OnlineTableControl (Page 5114)

TimeColumnBackColor

Description

Specifies the background color of the selected time column.

The setting is effective:

- When the option "in the colors of the value column" is not activated.
- When the "Font color" option is activated in the "Use column color" area of the "General" tab.

Access in runtime: Read and write

Syntax

Object.**TimeColumnBackColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

Color

Optional A value or a constant that specifies the background color of the selected time column.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TimeColumnBeginTime

Description

Defines the start of the time range for a selected time column.

Access in runtime: Read and write

Syntax

Object.**TimeColumnBeginTime**[=DateTime]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

DateTime

Optional A value or a constant that specifies the starting time for the selected time column.

TimeColumnCaption

Description

Defines the caption of the time column.

Access in runtime: Read and write

Syntax

Object.**TimeColumnCaption**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

STRING

Optional A value or a constant that specifies the caption of the time column.

TimeColumnCount

Description

Specifies the number of configured time columns.

Access in runtime: Read and write

Syntax

Object.**TimeColumnCount**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

Int32

Optional A value or a constant that specifies the number of configured time columns.

TimeColumnDateFormat

Description

Defines the date format for visualizing a selected time column.

Access in runtime: Read and write

Syntax

Object.**TimeColumnDateFormat**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

STRING

The following date formats are available:

Value	Explanation
dd.MM.yy	Day.Month.Year, e.g., 24.12.13.
dd.yyyyd.MM	Day.Month.Year, e.g., 24.12.2013.
dd/MM/yy	Day/Month/Year, e.g., 24/12/13.
dd/MM/yyyy	Day/Month/Year, e.g., 24/12/2013.

TimeColumnEndTime

Description

Defines the end of the time range of a selected time column.

Access in runtime: Read and write

Syntax

Object.**TimeColumnBeginTime**[=DateTime]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

DateTime

Optional A value or constant that specifies the end time.

TimeColumnForeColor

Description

Specifies the font color of the selected time column.

The setting is effective:

- When the option "in the colors of the value column" is not activated.
- When the "Font color" option is activated in the "Use column color" area of the "General" tab.

Access in runtime: Read and write

Syntax

Object.**TimeColumnForeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

OnlineTableControl

Color

Optional A value or a constant that specifies the font color of the selected time column.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TimeColumnHideText

Description

Sets text format for displaying the content of a time column.

Access in runtime: Read and write

Syntax

Object.**TimeColumnHideText**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

BOOLEAN

TRUE: The content is not shown as a text.

FALSE: The content is shown as a text.

TimeColumnHideTitleText

Description

Sets text format for displaying the time column header.

Access in runtime: Read and write

Syntax

Object.**TimeColumnHideTitleText**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

BOOLEAN

TRUE: The header is not shown as text.

FALSE: The header is shown as text.

TimeColumnIndex

Description

References a configured time column. You can assign the values of other properties to a specific time column by using the property.

Valid values for "TimeColumnIndex" are between 0 and "TimeColumnCount" minus 1. The "TimeColumnCount" property indicates the number of configured time columns.

Syntax

Object.**TimeColumnIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

Int32

Optional A value or constant which references a configured time span.

TimeColumnLength

Description

Specifies the width of a selected time column.

Access in runtime: Read and write

Syntax

Object.**TimeColumnLength**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- OnlineTableControl

Int32

Optional A value or a constant that specifies the width of the selected time column.

TimeColumnMeasurePoints

Description

Defines the number of measurement points to be displayed in the time column selected.

Access in runtime: Read and write

Syntax

Object.**TimeColumnMeasurePoints**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

Int32

Optional A value or constant that specifies the number of measurement points.

TimeColumnName

Description

Specifies the name of a selected time column.

Access in runtime: Read and write

Syntax

Object.**TimeColumnName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

STRING

Optional A value or a constant that specifies the name of the selected time column.

TimeColumnRangeType

Description

Defines the time range setting for the time column selected.

Access in Runtime: Read and write

Syntax

Object.**TimeColumnRangeType**[=TimeRangeMode]

Object

Required A "ScreenItem" object with following format:

- OnlineTableControl

TimeRangeMode

Value	Description	Explanation
0	Time range	Defines the start time and time range of the time column.
1	Start to end time	Defines the start and end time for the time column.
2	Number of measurement points	Defines the start time and the number of measurement points for the time column.

TimeColumnRemove

Description

Removes the selected time column from the list.

Access in runtime: Read and write

Syntax

Object.**TimeColumnRemove**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

STRING

Optional Removes the selected time column from the list.

TimeColumnRename

Description

Changes the name of the time column which is referenced by the "TimeColumnIndex" property. With "TimeColumnRename" you also dynamize the "TimeColumnName" property.

Access in runtime: Read and write

Syntax

Object.**TimeColumnRename**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

STRING

Optional Changes the name of the time column which is referenced by the "TimeColumnIndex" property.

TimeColumnRepos

Description

Repositions the order of time columns and of corresponding value columns. "Up" and "Down" move the time column selected up or down in the list. This moves the time column and corresponding value columns in the table towards the front or towards the back.

Access in runtime: Read and write

Syntax

Object.**TimeColumnRepos**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

Int32

Optional Value or constant that changes the order of time columns and corresponding value columns.

TimeColumnShowDate

Description

Enables the display of the date and time in the time column selected.

Access in runtime: Read and write

Syntax

Object.**TimeColumnShowDate**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

BOOLEAN

TRUE: The date and the time are displayed. The date format is defined in the "Date format" field.

FALSE: The date is not displayed. Only the time is displayed.

TimeColumnShowIcon

Description

Enables the display of time column contents as icon.

Access in runtime: Read and write

Syntax

Object.**TimeColumnActualize**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the format:

- OnlineTableControl

BOOLEAN

TRUE: The content is shown as a symbol.

FALSE: The content is not shown as an icon.

TimeColumnShowTitleIcon

Description

Specifies whether the header is shown as a symbol.

Access in runtime: Read and write

Syntax

Object.**TimeColumnShowTitleIcon**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

BOOLEAN

TRUE: The header is shown as a symbol.

FALSE: The header is not shown as a symbol.

TimeColumnSort

Description

Specifies how the time column referenced in "TimeColumnIndex" is sorted.

Access in Runtime: Read and write

Syntax

Object.**TimeColumnSort**[=SortMode]

Object

Required A "ScreenItem" object with following format:

- OnlineTableControl

SortMode

Value	Description	Explanation
0	No	No sorting
1	Ascending	Ascending order, starting at the lowest value.
2	Descending	Descending order, starting at the highest value.

TimeColumnSortIndex

Description

Specifies the sorting order of the time column referenced in "TimeColumnIndex". If you set the value to "0", the sorting criterion is removed in "TimeColumnSort".

Syntax

Object.**TimeColumnSortIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

Int32

Optional Value or constant that specifies the sorting order of the time column referenced in "TimeColumnIndex". If you set the value to "0", the sorting criterion is removed in "TimeColumnSort".

TimeColumnTimeFormat

Description

Defines the time format for visualizing a selected time column.

Access in runtime: Read and write

Syntax

Object.**TimeColumnTimeFormat**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

STRING

Value	Explanation
Automatic	The time format is set automatically.
HH:mm:ss.ms	Hours:Minutes:Seconds, e.g. 15:35:44.240.
hh:mm:ss tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.
hh:mm:ss.ms tt	Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.

TimeColumnTimeRangeBase

Description

Specifies the time unit for calculating the time range.

Access in Runtime: Read and write

Syntax

Object.**TimeColumnTimeRangeBase**[=TagLoggingTimeUnit]

Object

Required A "ScreenItem" object with following format:

- OnlineTableControl

TagLoggingTimeUnit

Value	Description
500	500 ms
1000	1 second
60000	1 minute

Value	Description
3600000	1 hour
86400000	1 day

TimeColumnTimeRangeFactor

Description

Defines the factor for calculating the time range. Only integer factors are valid.

Access in runtime: Read and write

Syntax

Object.**TimeColumnTimeRangeFactor**[=Int16]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

Int16

Optional A value or a constant that specifies the factor to determine the time range.

TimeColumnUseValueColumnColors

Description

Defines whether the selected time column will be displayed in the value column colors.

Access in runtime: Read and write

Syntax

Object.**TimeColumnUseValueColumnColors**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

BOOLEAN

TRUE: The selected time column is displayed in the colors of the value column. The settings in the "Font color" and "Background color" fields are disabled.

FALSE: The selected time column is displayed in the colors which are specified in the "Font color" and "Background color" fields.

TimeColumnVisible

Description

The list shows the time columns you created. Select the time columns to be displayed in the table from the list.

Click a time column entry in the list to adapt the properties and to define the time range of the time column.

Access in Runtime: Read and write

Syntax

Object.**TimeColumnVisible**[=BOOLEAN]

Object

Required A "ScreenItem" object with following format:

- OnlineTableControl

BOOLEAN

Optional. TRUE if the time column is displayed in the table.

Timestamp

Description

Returns the time stamp of the last read access of a tag in local time as DATE.

Access in Runtime: Read

Syntax

Object.**TimeStamp**

Object

Required A "Tag" object.

Comments

To show the TimeStamp property in plain text, use the VBS function "FormatDateTime(Date[, NamedFormat])". The output is dependent on the language setting. To adjust the language, use the VBS function "SetLocale´()".

If you want to return the time stamp sorted by date, day and time, use the NamedFormat parameter or the VBS functions like Year, WeekDay, Day, Hour, Minute, Second. The name of a week day can be obtained using the VBS function WeekdayName.

Examples

The following example issues the time stamp of the tag "Tag11" with the aid of the function "FormatDateTime":

```
'VBS87
Dim objTag
Dim lngCount
lngCount = 0
Set objTag = HMIRuntime.Tags("Tag11")
objTag.Read
SetLocale("en-gb")
MsgBox FormatDateTime(objTag.TimeStamp)    'Output: e.g. 06/08/2002 9:07:50
MsgBox Year(objTag.TimeStamp)    'Output: e.g. 2002
MsgBox Month(objTag.TimeStamp)    'Output: e.g. 8
MsgBox Weekday(objTag.TimeStamp)    'Output: e.g. 3
MsgBox WeekdayName(Weekday(objTag.TimeStamp))    'Output: e.g. Tuesday
MsgBox Day(objTag.TimeStamp)    'Output: e.g. 6
MsgBox Hour(objTag.TimeStamp)    'Output: e.g. 9
MsgBox Minute(objTag.TimeStamp)    'Output: e.g. 7
MsgBox Second(objTag.TimeStamp)    'Output: e.g. 50
For lngCount = 0 To 4
MsgBox FormatDateTime(objTag.TimeStamp, lngCount)
Next
'lngCount = 0: Output: e.g. 06/08/2002 9:07:50
'lngCount = 1: Output: e.g. 06 August 2002
'lngCount = 2: Output: e.g. 06/08/2002
'lngCount = 3: Output: e.g. 9:07:50
'lngCount = 4: Output: e.g. 9:07
```

The following example issues the time stamp of the tag "Tag1":

```
'VBS88
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
MsgBox objTag.TimeStamp
```

See also

Tag (Page 5012)

TitleCut

Description

Specifies whether the content of the fields in a title bar is to be shortened if the column is too narrow.

Access in runtime: Read and write

Syntax

Object.**TitleCut**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional TRUE if the column headers in the title bar are shortened if the column is too narrow.

TitleDarkShadowColor

Description

Specifies the color for the dark side of the shading.

You open a color selection with the button. The settings are only active if the "Shading Color" option is enabled.

Access in runtime: Read and write

Syntax

Object.**TitleDarkShadowColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the color for the dark side of the shading.

Comments

You can use the "RGB" function to define the colors in RGB format (red, green, blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TitleForeColor

Description

Specifies the font color of the table header for the selected status.

You open a color selection dialog box with the button.

Access in runtime: Read and write

Syntax

Object.**TitleForeColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or constant that specifies the font color in the title bar of the table.

Comments

You can use the "RGB" function to define the colors in RGB format (red, green, blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TitleGridLineColor

Description

Specifies the color of the separation lines in the title bar of the table.

You open a color selection dialog box with the button.

Access in runtime: Read and write

Syntax

Object.**TitleGridLineColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the color of the separation lines in the title bar of the table.

Comments

You can use the "RGB" function to define the colors in RGB format (red, green, blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TitleLightShadowColor

Description

Specifies the color for the light side of the shading.

You open a color selection dialog box with the button. The setting is only effective when the "Shading color" option is activated.

Access in runtime: Read and write

Syntax

Object.**TitleLightShadowColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the color for the light side of the shading.

Comments

You can use the "RGB" function to define the colors in RGB format (red, green, blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBScript color constants such as vbRed and vbGreen.

TitleSort

Description

Defines how to trigger sorting by column title. The "Auto-scrolling" option must be disabled to sort by column title.

Access in Runtime: Read and write

Syntax

Object.**TitleSort**[=GridSortTrigger]

Object

Required An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSortTrigger

Value	Description	Explanation
0	No	Sorting by column header is disabled.
1	With click	Sorting is triggered by clicking in the column header.
2	With double-click	Sorting is triggered by double-clicking in the column title.

TitleStyle

Description

Specifies whether to set a shading color for the table header.

Access in Runtime: Read and write

Syntax

Object.**TitleStyle**[=GridHeaderStyle]

Object

Required A "ScreenItem" object with following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridHeaderStyle

Value	Description	Explanation
0	Flat	Disables the use of shading colors. Flat header style.
1	Button	Enables the use of shading colors. 3D representation of the header.

Toggle

Description

Specifies whether the selected object engages after it has been operated in Runtime.

Access in runtime: Read and write

Syntax

Object.**Toggle**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton

BOOLEAN

Optional TRUE, if the selected object engages after it has been operated in Runtime.

Tolerance

Description

Specifies the limit for the disk space view as of which a deviation will be reported.

Access in runtime: Read and write

Syntax

Object.**Tolerance**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

Int32

Optional A value or a constant that specifies the limit for the disk space view as of which a deviation will be reported.

ToleranceColor

Description

Specifies the colors in which the bars of the storage space display will be shown as soon as the tolerance range is exceeded.

Access in runtime: Read and write

Syntax

Object.**ToleranceColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

Color

Optional A value or a constant that specifies the colors in which the bars of the storage space display will be shown as soon as the tolerance range is exceeded.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

ToleranceLowerLimit

Description

Sets the low limit for tolerance 1.

Access in runtime: Read and write

Syntax

Object.**ToleranceLowerLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the format:

- Bar

DOUBLE

Optional A value or a constant that specifies the low limit for tolerance 1.

Comments

The following values will be defined via the properties "ToleranceLowerLimit", "ToleranceLowerLimitColor" and "ToleranceLowerLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

ToleranceLowerLimitColor

Description

Specifies the color for the lower limit "ToleranceLowerLimit"".

Access in runtime: Read and write

Syntax

Object.**ToleranceLowerLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Color

Optional A value or a constant that specifies the color for the lower limit "ToleranceLowerLimit" ".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

The "ToleranceLowerLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

ToleranceLowerLimitEnabled

Description

Specifies whether the "ToleranceLowerLimit" limit is monitored. The limit, the display when the limit has been reached, and the type of evaluation are set with the properties "ToleranceLowerLimit", "ToleranceLowerLimitColor" and "ToleranceLowerLimitRelative".

Access in runtime: Read and write

Syntax

Object.**ToleranceLowerLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the "ToleranceLowerLimit" limit is monitored.

ToleranceLowerLimitRelative

Description

Specifies whether the lower limit "ToleranceLowerLimit" is evaluated as a percentage or as an absolute value.

Access in runtime: Read and write

Syntax

Object.**ToleranceLowerLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional

TRUE: The lower limit "ToleranceLowerLimit" is evaluated as a percentage.

FALSE: The lower limit "ToleranceLowerLimit" is evaluated as an absolute value.

ToleranceUpperLimit

Description

Sets the high limit for tolerance 1.

Access in runtime: Read and write

Syntax

Object.**ToleranceUpperLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the format:

- Bar

DOUBLE

Optional A value or a constant that specifies the high limit for tolerance 1.

Comments

The following values will be specified via the properties "ToleranceUpperLimit", "ToleranceUpperLimitColor" and "ToleranceUpperLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

ToleranceUpperLimitColor

Description

Specifies the color for the upper limit "ToleranceUpperLimit".

Access in runtime: Read and write

Syntax

Object.**ToleranceUpperLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Color

Optional A value or a constant that specifies the color for the upper limit "ToleranceUpperLimit"".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

ToleranceUpperLimitEnabled

Description

Specifies whether the "ToleranceUpperLimit"" limit is to be monitored.

Access in runtime: Read and write

Syntax

Object.**ToleranceUpperLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the "ToleranceUpperLimit" limit is to be monitored.

Comments

The following values will be specified with the properties "ToleranceUpperLimit",", "ToleranceUpperLimitColor" and "ToleranceUpperLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

ToleranceUpperLimitRelative

Description

Specifies whether the higher limit "ToleranceUpperLimit"" is to be evaluated as a percentage or as an absolute value.

Access in runtime: Read and write

Syntax

Object.**ToleranceUpperLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional

TRUE: The higher limit "ToleranceUpperLimit" is evaluated as a percentage.

FALSE: The higher limit "ToleranceUpperLimit" is evaluated as an absolute value.

ToolbarAlignment

Description

Specifies the position of the toolbar.

Access in runtime: Read and write

Syntax

Object.**ToolbarAlignment**[=ToolbarPosition]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

You have no access in runtime with the following format:

- SysDiagControl

ToolbarPosition

Optional. A value or a constant that specifies the position of the toolbar.

ToolbarBackColor

Description

Background color - ToolbarBackColor

Specifies the background color of the toolbar. Open the "Color selection" dialog by clicking the button.

The background color you configured is only displayed if the "Display" option is enabled.

Access in runtime: Read and write

Syntax

Object.**ToolbarBackColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

Optional A value or a constant that specifies the background color of the toolbar.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

ToolbarButtonActive

Description

Activates a button function in Runtime. Clicking the button in Runtime triggers the corresponding function.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonActive**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl

- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The function connected to the key is active.

FALSE: The function connected to the key is not active. You can connect your own functions to the key by local scripts.

ToolbarButtonAdd

Description

Creates a new, user-defined key function.

You can change the name assigned by WinCC in the "Object name" field.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonAdd**[=STRING]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or a constant that specifies the name of the new key function.

ToolbarButtonBeginGroup

Description

Inserts a leading separator (vertical line) for the selected button function on the toolbar. These separators can be used to group the icons of the button functions.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonBeginGroup**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The separator is inserted in front of the selected key function.

FALSE: The separator is not inserted in front of the selected key function.

ToolbarButtonCount

Description

Specifies the number of buttons in the toolbar.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonCount**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or a constant that specifies the number of buttons in the toolbar.

ToolbarButtonEnabled

Description

Enables operation of custom toolbar buttons.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional TRUE if operation is enabled for the selected user-defined key in the toolbar.

ToolbarButtonHotKey

Description

Specifies the hotkey for the selected button function.

You create or edit a hotkey by clicking in the "Hotkey" field and pressing the key or key shortcut required.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonHotKey**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or a constant that specifies the hotkey for the selected key function.

ToolbarButtonID

Description

Specifies a unique ID number for the selected button function.

WinCC sets this read only ID.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonID**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or constant that specifies the ID number of the selected key function.

ToolbarButtonIndex

Description

References a button function.

Using this attribute you can assign the values of other attributes to a specific button function.

Values between 0 and "ToolbarButtonCount" minus 1 are valid for "ToolbarButtonIndex". The attribute "ToolbarButtonCount" specifies the number of configurable key functions.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or a constant that specifies the number of the selected key function.

ToolbarButtonLocked

Description

Enables/disables display of the locked, pressed state for a user-defined toolbar button.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonLocked**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional TRUE if the pressed state is displayed for the selected user-defined button in the toolbar.

ToolbarButtonName

Description

Specifies the name of the selected user-defined button.

You can change the name of a user-defined button. The "ToolbarButtonName" property for user-defined buttons can be dynamized with the "ToolbarButtonRename" property.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that specifies the name of the selected user-defined button.

ToolbarButtonAuthorization

Description

Shows the authorization for a button function selected. You can edit the authorization using the selection button. The authorizations are configured in the user administration.

See also

UserArchiveControl (Page 5207)

TrendRulerControl (Page 5187)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

OnlineTrendControl (Page 5122)

ToolbarButtonRemove

Description

Removes the selected user-defined button.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonRemove**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that specifies the name of the user-defined button to be removed.

ToolbarButtonRename

Description

Returns the name of the user-defined toolbar button that is being referenced via the "ToolbarButtonID" property.

With "ToolbarButtonRename" you also dynamize the "ToolbarButtonName" property.

Access in Runtime: Read and write

Syntax

Object.**ToolbarButtonRename**[=STRING]

Object

Required. An object of the "ScreenItem" type with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl

- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or a constant that specifies the new name of the selected user-defined button.

ToolbarButtonRepos

Description

Changes the sorting order of user-defined key functions in the toolbar.

"Up" and "Down" move the button function selected up or down in the list. This moves the button function in the toolbar of a Control towards the front or towards the back.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonRepos**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or a constant that specifies the position of the selected user-defined button in the toolbar.

ToolbarButtonTooltipText

Description

Specifies the tooltip text for the user-defined button in the toolbar.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonTooltipText**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that specifies the tooltip text for the selected user-defined button.

ToolbarButtonUserDefined

Description

Specifies whether the project engineer has added the toolbar button as new user-defined button.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonUserDefined**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Boolean

TRUE: The key of the toolbar is customized.

FALSE: The key of the toolbar is specified by the system.

ToolbarShowTooltips

Description

Enables the display of tooltips for the button functions in Runtime. The property can be dynamized with the ToolbarShowTooltips name. The property for specifying the tooltip text is "ToolbarButtonTooltipText".

Access in runtime: Read and write

Syntax

Object.**ToolbarShowTooltips**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The tooltips are displayed.

FALSE: The tooltips are not displayed.

ToolbarUseBackColor

Description

Enables the display of the background color for a toolbar.

Access in runtime: Read and write

Syntax

Object.**ToolbarUseBackColor**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl

- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The background color of the toolbar is displayed.

FALSE: The background color of the toolbar is not displayed.

ToolbarUseHotKeys

Description

Activates the hotkeys for button functions in Runtime. You insert the hotkeys for button functions in the "Hotkey" field.

Access in runtime: Read and write

Syntax

Object.**ToolbarUseHotKeys**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The hotkeys are activated.

FALSE: The hotkeys are not activated.

ToolbarVisible

Description

Specifies whether the toolbar of the control is displayed.

Access in runtime: Read and write

Syntax

Object.**ToolbarVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The toolbar is displayed.

FALSE: The toolbar is not displayed.

ToolTipText

Description

Specifies the tooltip text.

Access in Runtime: Read and write

Syntax

Object.**ToolTipText**[=STRING]

ToolTipText

Required. An object of the "ScreenItem" type with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- Ellipse
- EllipseSegment
- EllipticalArc

- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- StatusForce
- Switch
- SymbolicIOField
- TextField
- TubeArcObject

STRING

Optional A value or a constant that specifies the tooltip text.

Top

Description

Specifies the value of the Y coordinate of the selected object.

Access in runtime: Read and write

Syntax

Object.**Top**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- AlarmView
- ApplicationWindow
- Bar
- BatteryView
- Button

- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- DateTimeField
- DiscSpaceView
- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- HTML-Browser
- IOField
- Line
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- Polygon
- Polyline
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- Rectangle
- RoundButton
- Screenwindow

- Slider
- SmartClientView
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TextField
- TrendRulerControl
- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserArchiveControl
- UserView
- WLanQualityView
- WindowsSlider
- ZoneLabelView
- ZoneQualityView

You have no access in runtime with the following format:

- S7GraphOverview

Int32

Optional A value or a constant that contains the value of the Y coordinate in pixels (measured from the top left edge of the screen).

Comments

The Y coordinate refers to the top left corner of the rectangle that surrounds the object. The screen limits are also monitored in runtime. If the assigned coordinate value exceeds the display size, the user-defined function is interrupted with an error message.

TopOffset

Description

Specifies the distance between the screen and the top screen window margin.

The picture is displayed as a cutout of the picture window. The picture scroll bars are located at the left and upper edge of the picture. If you wish to display the screen in the screen window with horizontal and vertical offset of the screen scroll bars, use the "HorizontalScrollBarPosition" and "VerticalScrollBarPosition properties for the offset.

Access in runtime: Read and write

Syntax

Object.**TopOffset**[=Int32]

Object

Required. An object of the "ScreenItem" type with the following format:

- Screenwindow

Int32

Optional. A value or a constant that specifies the distance between the screen and the top screen window margin.

Total

Description

Specifies the memory capacity.

Access in runtime: Read and write

Syntax

Object.**Total**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- DiskSpaceView

DOUBLE

Optional A value or a constant that specifies the memory capacity.

Transparency

Description

Defines the object transparency in percent.

0 = no transparency; 100 = complete transparency (invisible).

The text and fields of the graphic objects are only transparent with "100."

In runtime, a completely transparent object (invisible) is also functional.

Access in runtime: Read and write

Syntax

Object.**Transparency**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Ellipse
- EllipseSegment
- EllipticalArc
- Gauge
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- Slider
- SymbolicIOField
- TextField
- TubeArcObject
- TubeDoubleTeeObject

- TubeTeeObject
- Tubepolyline
- WindowsSlider

Int32

Optional A value or constant that specifies the transparency of the object in percent.

TransparentColor

Description

Specifies which color of the allocated graphic (*.bmp, *dib) of the specified object will be set to "transparent".

The "UseTransparentColor" property must have the value TRUE so that the color will be shown as transparent.

Access in runtime: Read and write

Syntax

Object.**TransparentColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- GraphicView
- GraphicIOField

Color

Optional A value or a constant that specifies the color that will be shown transparent.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TransparentColorDeactivatedPicture

Description

Sets the color of the assigned bitmap object to "transparent" for the "disabled" status.

Access in runtime: Read and write

Syntax

Object.**TransparentColorDeactivatedPicture**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- RoundButton

Color

Optional A value or a constant that specifies which color of the allocated bitmap object will be set to transparent" for the status "disabled".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

The "PicDeactUseTransColor" property must have the value TRUE so that the color can be set to "transparent".

TransparentColorPictureOff

Description

Specifies which color of the allocated bitmap object will be set to transparent" for the "off" status.

Access in runtime: Read and write

Syntax

Object.**TransparentColorPictureOff**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton

Color

Optional A value or a constant that specifies which color of the allocated bitmap object will be set to transparent" for the "off" status.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TransparentColorPictureOn

Description

Specifies which color of the allocated bitmap object will be set to transparent" for the "on" status.

Access in runtime: Read and write

Syntax

Object.**TransparentColorPictureOn**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton

Color

Optional A value or a constant that specifies which color of the allocated bitmap object will be set to "transparent" for the "on" status.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

The "PicDownUseTransColor" property must have the value TRUE so that the color can be set to "transparent".

TrendActualize

Description

Enables the update of a selected trend.

Access in runtime: Read and write

Syntax

Object.**TrendActualize**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

BOOLEAN

TRUE: The selected trend is always actualized.

FALSE: The selected trend is not actualized. This setting is useful when a logged trend is compared with a current trend.

TrendAdd

Description

Creates a new trend.

Access in runtime: Read and write

Syntax

Object.**TrendAdd**[=STRING]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional. The name of the new trend.

TrendBeginTime

Description

Defines the start time of the time range for data transfer to the selected trend.

Access in runtime: Read and write

Syntax

Object.**TrendBeginTime**[=DateTime]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

DateTime

Optional Defines the start time for data supply for the selected trend.

TrendColor

Description

Specifies or returns the object color.

The trend indicator shows the trend (rising or falling) in the measurement value to be monitored with a small arrow. In order to activate the trend display, the "Trend" property must be set to "TRUE".

Access in runtime: Read and write

Syntax

Object.**TrendColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the object.

Comments

You can use the "RGB" function to define the color in RGB format (red, green blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TrendCount

Description

Specifies the number of configured trends.

Access in runtime: Read and write

Syntax

Object.**TrendCount**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional A value or a constant that specifies the number of trends in the selected object.

TrendEndTime

Description

Defines the end of the time range for data connections of a selected trend.

Access in runtime: Read and write

Syntax

Object.**TrendEndTime**[=DateTime]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

DateTime

Optional Specifies the end time for data supply for the selected trend.

TrendExtendedColorSet

Description

Enables configuration of the point and fill colors and the display of colors in Runtime.

Access in runtime: Read and write

Syntax

Object.**TrendExtendedColorSet**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The settings in the "Point color" and "Fill color" fields are configurable and effective in Runtime.

FALSE: The settings in the "Point color" and "Fill color" fields are configurable and not effective in Runtime.

See also

FunctionTrendControl (Page 5077)

OnlineTrendControl (Page 5122)

TrendFill

Description

Specifies if the area beneath the trend is to be filled.

Access in runtime: Read and write

Syntax

Object.**TrendFill**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The area beneath the trend is shown filled. f the "Advanced" option is disabled, the trend color is used as fill color.

FALSE: The trend is not shown filled.

TrendFillColor

Description

Defines the fill color of the trend.

The settings are only active if the "Filled" option is enabled. Open the "Color selection" dialog by clicking the button.

The settings are only active if the "Advanced" option is enabled.

Access in runtime: Read and write

Syntax

Object.**TrendFillColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the fill color of the selected trend.

Comments

You can use the "RGB" function to define the color in RGB format (red, green blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TrendIndex

Description

References a configured trend. You can assign the values of other properties to a specific trend using this property. The index must always be set before you change the properties of a trend in runtime.

Values of between 0 and "TrendCount" minus 1 are valid for "TrendIndex". The property "TrendCount" specifies the number of configured trends.

Access in runtime: Read and write

Syntax

Object.**TrendIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional A value or a constant that specifies the number of the selected trend.

TrendIndicatorColor

Description

Specifies the color for the trend indicator. The trend indicator represents the tendency (rising or falling) of the measurement value that is to be monitored with a small arrow. To activate the trend indicator, the property "ShowTrendIndicator" must have the value "TRUE".

Access in runtime: Read and write

Syntax

Object.**TrendIndicatorColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Color

Optional A value or a constant that specifies the color of the trend indicator.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TrendLabel

Description

Defines the label of the trend selected. The label is displayed in runtime if the value at the "UseTrendNameAsLabel" attribute is "FALSE".

Access in runtime: Read and write

Syntax

Object.**TrendLabel**[=STRING]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the label of the selected trend.

TrendLineStyle

Example

Defines the line style for trend visualization.

Access in Runtime: Read and write

Syntax

Object.**TrendLineStyle**[=LineStyle]

Object

Required. An object of the "ScreenItem" type with the format:

- FunctionTrendControl
- OnlineTrendControl

LineStyle

Value	Description	Explanation
0	Solid	The trend is visualized as solid line.
1	Dashed	The trend is visualized as dashed line.
2	Dots	The trend is visualized as dotted line.
3	Dash dot	The trend is visualized as dot-dash line.
4	Dash Dot Dot	The trend is visualized as dash-dot-dot line.

TrendLineType

Description

Defines how to visualize a trend.

Access in Runtime: Read and write

Syntax

Object.**TrendLineType**[=TrendLineTypeScada]

Object

Required. An object of the "ScreenItem" type with the format:

FunctionTrendControl

OnlineTrendControl

TrendLineTypeScada

Value	Description	Explanation
0	None	Only the dots are displayed.
1	Line	Visualizes a trend with linear interconnection of points.
2	Stepped	Visualizes a stepped trend and its interconnected points.

TrendLineWidth

Description

Defines the line weight of the selected trend in pixels.

Access in runtime: Read and write

Syntax

Object.**TrendLineWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional A value or a constant that specifies the line weight of the selected trend in pixels.

TrendLowerLimit

Description

Specifies the low limit of a tag. If the tag drops below the value of "TrendLowerLimit", the values are marked in the color set in "TrendLowerLimitColor". The specification is effective if the "TrendLowerLimitColoring" property has the value "TRUE".

Access in runtime: Read and write

Syntax

Object.**TrendLowerLimit**[=DOUBLE]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

DOUBLE

Optional. A value or a constant that specifies the lower limit for a tag.

TrendLowerLimitColor

Description

Specifies the color which marks tag values which are below the value of "TrendLowerLimit". The setting is effective if the "TrendLowerLimitColoring" attribute has the value "TRUE".

Syntax

Object.**TrendLowerLimitColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of tag values below the "TrendLowerLimit" value.

Comments

You can use the "RGB" function to define the color in RGB format (red, green blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TrendLowerLimitColoring

Description

Specifies if the "TrendLowerLimitColor" attribute is used to identify tag values which are less than the value at "TrendLowerLimit".

Syntax

Object.**TrendLowerLimitColoring**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The "TrendLowerLimitColor" property is effective.

FALSE: The "TrendLowerLimitColor" property is not effective.

TrendMeasurePoints

Description

Defines the number of measurement points for visualization of selected trends.

Defines the number of value pairs provided to the trend from a user archive.

Access in runtime: Read and write

Syntax

Object.**TrendMeasurePoints**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

Int32

Optional A value or a constant that specifies the number of measurement points or value pairs for the selected trend.

TrendName

Description

Displays the name of the selected trend. The name is defined on the "Trends" tab. The "TrendName" property can be dynamized with the "TrendRename" property.

Access in runtime: Read and write

Syntax

Object.**TrendName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional A value or a constant that specifies the name of the selected trend.

TrendPointColor

Description

Defines the color of the dots in the selected trend.

Open the "Color selection" dialog by clicking the button.

The settings are only active if the "Advanced" option is enabled.

Access in runtime: Read and write

Syntax

Object.**TrendPointColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the dots in the selected trend.

Comments

You can use the "RGB" function to define the color in RGB format (red, green blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TrendPointStyle

Description

Defines the dot style for trend visualization.

Access in Runtime: Read and write

Syntax

Object.**TrendPointStyle**[=PointStyle]

Object

Required A "ScreenItem" object with following format:

- FunctionTrendControl
- OnlineTrendControl

PointStyle

Value	Description	Explanation
0	None	The points are not displayed.
1	Dots	The trend points are visualized with a size of one pixel. The setting in the "Dot width" field is disabled.
2	Squares	The dots are displayed as square. The setting in the "Dot width" field is active.
3	Circles	The dots are displayed as circles. The setting in the "Dot width" field is active.

TrendPointWidth

Description

Specifies the dot weight in pixels. You can only define the dot weight for "square" and "circular" dots.

Access in runtime: Read and write

Syntax

Object.**TrendPointWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional A value or a constant that specifies the dot weight of the selected trend in pixels.

TrendProvider

Description

Specifies the data source for a selected trend.

Access in Runtime: Read and write

Syntax

Object.**TrendProvider**[=Provider]

Object

Required A "ScreenItem" object with following format:

- FunctionTrendControl
- OnlineTrendControl

Provider

Value	Description	Explanation
0	None	No data source configured for implementation in Runtime by means of a user-defined function.
1	Archive tags	Data source with archive tags of a process value archive.
2	HMI tags	Data supply with tag values of HMI tags
3	Recipe data	Data supply with columns of a recipe

TrendRangeType

Description

Specifies the time range for which the trend is supplied with data.

You can only define the number of measuring points if you select user archives as the data source.

Access in Runtime: Read and write

Syntax

Object.**TrendRangeType**[=RangeType]

Object

Required A "ScreenItem" object with following format:

- FunctionTrendControl

RangeType

Value	Description	Explanation
0	Time range	Defines the start time and the time range for the data connection.
1	Start to end time	Defines the start and end time for the data connection.
2	Number of measurement points	Defines the start time and the number of measurement points for the data connection.

TrendRemove

Description

Removes selected trends from the list.

Access in runtime: Read and write

Syntax

Object.**TrendRemove**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional A value or a constant that specifies the name of the trend to be removed.

TrendRename

Description

Changes the name of the trend referenced by the "TrendIndex" property.

With "TrendRename", you also dynamize the "TrendName" property.

Access in runtime: Read and write

Syntax

Object.**TrendRename**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional A value or a constant that specifies the new name of the selected trend.

TrendRepos

Description

Repositions the trend in the trend window.

"Up" and "Down" move the selected trend up or down in the list. This moves the trend towards the foreground or background for visualization in Runtime.

Access in runtime: Read and write

Syntax

Object.**TrendRepos**[=Int32]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the new position of the selected trend.

TrendSelectTagNameX

Description

Opens the dialog box for selecting the tag name for the data supply of the X axis in the f(x) trend view. Programmers can use the property to let the user select a tag name with a button for example.

Access in runtime: Read and write

Syntax

Object.**TrendSelectTagNameX**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

BOOLEAN

TRUE:

FALSE:

TrendSelectTagNameY

Description

Opens the dialog for selecting the tag name for data supply of the Y-axis in the f(x) trend view. Programmers can use the property to let the user select a tag name with a button for example.

Access in runtime: Read and write

Syntax

Object.**TrendSelectTagNameY**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

BOOLEAN

TRUE:

FALSE:

TrendTagNameX

Description

Specifies the name of the connected HMI tag or column for the X-axis. You select an HMI tag or a column with the selection button.

Access in runtime: Read and write

Syntax

Object.**TrendTagNameX**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the name of the HMI tag or column for the X axis.

TrendTagNameY

Description

Specifies the name of the connected HMI tag or column for the Y-axis. You select an HMI tag or a column with the selection button.

Access in runtime: Read and write

Syntax

Object.**TrendTagNameY**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the name of the HMI tag or column for the Y axis.

TrendTimeRangeBase

Description

Specifies the time unit for calculating the time range.

Access in Runtime: Read and write

Syntax

Object.**TrendTimeRangeBase**[=TagLoggingTimeUnit]

Object

Required A "ScreenItem" object with following format:

- FunctionTrendControl

TagLoggingTimeUnit

Value	Description
500	500 ms
1000	1 second
60000	1 minute
3600000	1 hour
86400000	1 day

TrendTimeRangeFactor

Description

Defines the factor for calculating the time range. Only integer factors are valid.

Access in runtime: Read and write

Syntax

Object.**TimeRangeFactor**[=Int32]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl

Int32

Optional. A value or a constant that specifies the factor for determining the time range.

TrendTrendWindow

Description

Defines the trend window for visualizing the trend selected.

Define the available trend windows in the "Trend window" tab.

Access in runtime: Read and write

Syntax

Object.**TrendTrendWindow**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional A value or constant that specifies the name of the trend window for the selected trend.

TrendUncertainColor

Description

Value are in uncertain state if the initial value is unknown after runtime has been activated, or if a substitute value is used.

With the "TrendUncertainColor" property you specify the color which is used for marking these values. Whether or not the information is evaluated depends on the "TrendUncertainColoring" property.

Access in runtime: Read and write

Syntax

Object.**TrendUncertainColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional A value or a constant that specifies the color of the values of uncertain status in the selected trend.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TrendUncertainColoring

Description

Value are in uncertain state if the initial value is unknown after runtime has been activated, or if a substitute value is used. The "TrendUncertainColoring" attribute is used to enable identification of such values based on the color set in "TrendUncertainColor".

Access in runtime: Read and write

Syntax

Object.**TrendUncertainColoring**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The setting of the "TrendUncertainColor" property is effective.

FALSE: The setting of the "TrendUncertainColor" property is not effective.

TrendUpperLimit

Description

Specifies the high limit of a tag.

If the tag drops below the value of "TrendUpperLimit", the values are shown in the color set in "TrendUpperLimitColor". The specification is effective if the "TrendUpperLimitColoring" attribute has the value "TRUE".

Access in runtime: Read and write

Syntax

Object.**TrendUpperLimit**[=DOUBLE]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

DOUBLE

Optional. A value or a constant that specifies the higher limit for values in the selected trend.

TrendUpperLimitColor

Description

Specifies the color that marks tag values that are below the value of "TrendLowerLimit". The setting is effective if the "TrendUpperLimitColoring" attribute has the value "TRUE".

Access in runtime: Read and write

Syntax

Object.**TrendUpperLimitColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color for values below the limit for the selected trend.

Comments

You can use the "RGB" function to define the color in RGB format (red, green blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TrendUpperLimitColoring

Description

Specifies whether the selection frame is shown with the specified color.

Access in runtime: Read and write

Syntax

Object.**TrendUpperLimitColoring**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The setting for the "TrendUpperLimitColor" attribute is effective.

FALSE: The setting for the "TrendUpperLimitColor" attribute is not effective.

TrendVisible

Description

Specifies whether or not the selected trend is displayed.

The list contains all the trends that you have created.

Select the trends to be displayed in the trend window from the list.

Click a trend entry in the list to adapt the properties and to assign axes and trend windows to the trend.

Access in runtime: Read and write

Syntax

Object.**TrendVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The selected trend is displayed.

FALSE: The selected trend is not displayed.

TrendWindowAdd

Description

Creates a new trend view.

Access in runtime: Read and write

Syntax

Object.**TrendWindowAdd**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional A value or a constant that specifies the name of the new trend view.

TrendWindowCoarseGrid

Description

Enables the display of grid lines for the main scale.

Access in runtime: Read and write

Syntax

Object.**TrendWindowCoarseGrid**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The grid lines for the main scaling are displayed.

FALSE: The grid lines for the main scaling are not displayed.

TrendWindowCoarseGridColor

Description

Specifies the grid color of the main scale.

Open the "Color selection" dialog by clicking the button.

Access in runtime: Read and write

Syntax

Object.**TrendWindowCoarseGridColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the grid line color for the main scale.

Comments

You can use the "RGB" function to define the color in RGB format (red, green blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TrendWindowFineGrid

Description

Enables the display of grid lines for the secondary scale.

Access in runtime: Read and write

Syntax

Object.**TrendWindowFineGrid**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The grid lines for the auxiliary scaling are displayed.

FALSE: The grid lines for the auxiliary scaling are not displayed.

TrendWindowFineGridColor

Description

Specifies the grid color of the main scale.

Open the "Color selection" dialog by clicking the button.

Access in runtime: Read and write

Syntax

Object.**TrendWindowFineGridColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the grid line color for the auxiliary scale.

Comments

You can use the "RGB" function to define the color in RGB format (red, green blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TrendWindowForegroundTrendGrid

Description

Enables the display of grid lines only for the foreground trend in the trend window.

Access in runtime: Read and write

Syntax

Object.**TrendWindowForegroundTrendGrid**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The grid lines for the foreground trend are displayed in the trend window.

FALSE: The grid lines for the foreground trend are not displayed in the trend window.

TrendWindowGridInTrendColor

Description

Sets the trend color for the visualization of the grid lines for the main scale.

Access in runtime: Read and write

Syntax

Object.**TrendWindowGridInTrendColor**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

BOOLEAN

TRUE: The grid lines are displayed in the trend color.

FALSE: The grid lines are displayed with the color set in the "Color" field.

TrendWindowHorizontalGrid

Description

Enables the display of horizontal grid lines.

Access in runtime: Read and write

Syntax

Object.**TrendWindowHorizontalGrid**[=BOOLEAN]

Object

- FunctionTrendControl

BOOLEAN

TRUE: The horizontal grid lines are displayed.

FALSE: The horizontal grid lines are not displayed.

TrendWindowIndex

Description

References a configured trend view. You can assign the values of other properties to a specific trend window by using the property.

Values of between 0 and "TrendWindowCount" minus 1 are valid for "TrendWindowIndex". The property "TrendWindowCount" specifies the number of configured trend windows.

Access in runtime: Read and write

Syntax

Object.**TrendWindowIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional A value or a constant that specifies the number of the selected trend window.

TrendWindowName

Description

Defines the name of the trend window selected.

Access in runtime: Read and write

Syntax

Object.**TrendWindowName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional A value or a constant that specifies the name of the selected trend window.

TrendWindowRemove

Description

Removes the selected trend view from the list.

Access in runtime: Read and write

Syntax

Object.**TrendWindowRemove**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional A value or a constant that specifies the name of the trend window to be removed.

TrendWindowRename

Description

Changes the name of the trend window referenced by the "TrendWindowIndex" property.

With "TrendWindowRename", you also dynamize the ""TrendWindowName" property.

Access in runtime: Read and write

Syntax

Object.**TrendWindowRename**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional A value or a constant that specifies the new name of the selected trend view.

TrendWindowRepos

Description

Changes the order of the trend view.

"Up" and "Down" move the selected trend up or down in the list. The sorting order in the list defines the position in the Control. The first trend view is shown in the lowest position, the last trend view in the highest position.

Access in runtime: Read and write

Syntax

Object.**TrendWindwRepos**[=Int32]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the new position of the selected trend view.

TrendWindowRulerColor

Description

Specifies the ruler color.

Open the "Color selection" dialog by clicking the button. The color can be configured and displayed if "1 - graphic" is set for visualization.

Access in runtime: Read and write

Syntax

Object.**TrendWindowRulerColor**[=Color]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the ruler.

Comments

You can use the "RGB" function to define the color in RGB format (red, green blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

TrendWindowRulerLayer

Description

Specifies the display layer of the ruler in the trend view.

Access in Runtime: Read and write

Syntax

Object.**TrendWindowRulerLayer**[=RulerLayer]

Object

Required A "ScreenItem" object with following format:

- FunctionTrendControl
- OnlineTrendControl

RulerLayer

Value	Description	Explanation
0	Under grid	The ruler is visualized on a layer under the grid.
1	Between grid and trend	The ruler is positioned on top of the trend and under the grid.
2	On top of trend	The ruler is positioned on top of the trend.

TrendWindowRulerStyle

Description

Defines the appearance of the ruler.

Access in Runtime: Read and write

Syntax

Object.**TrendWindowRulerStyle**[=RulerStyle]

Object

Required A "ScreenItem" object with following format:

- FunctionTrendControl
- OnlineTrendControl

RulerStyle

TRUE: The ruler is displayed as a single black line.

FALSE: The ruler is displayed in the configured "Color" and "Width".

TrendWindowRulerWidth

Description

Defines the width of the ruler in pixels.

The width can be configured and displayed if "1 - graphic" is set for visualization.

Access in runtime: Read and write

Syntax

Object.**TrendWindowRulerWidth**[=Int32]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the width of the ruler in pixels.

TrendWindowSpacePortion

Description

Specifies the proportion of the trend widow to be used for the selected curve.

Access in runtime: Read and write

Syntax

Object.**TrendWindowSpacePortion**[=Int32]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or constant that specifies the proportion of the selected trend window in the control in percent.

TrendWindowVerticalGrid

Description

Enables the display of vertical grid lines.

Access in runtime: Read and write

Syntax

Object.**TrendWindowVerticalGrid**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The vertical grid lines are displayed.

FALSE: The vertical grid lines are not displayed.

TrendWindowVisible

Description

Specifies whether or not the selected trend window is displayed.

The list contains the trend views that you have created.

Activate the trend views in the list which you want to display in the control.

Click a list entry to adapt the ruler and grid line properties.

Access in runtime: Read and write

Syntax

Object.**TrendWindowVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The selected trend window is displayed.

FALSE: The selected trend window is not displayed.

TrendXAxis

Description

Defines the X axis to be used for the trend selected.

Define the available X axes inn the "X Axes" tab.

Access in runtime: Read and write

Syntax

Object.**TrendXAxis**[=STRING]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl

STRING

Optional. A value or constant that specifies the name of the X-axis used for the selected trend.

TrendYAxis

Description

Defines the Y axis to be used for the trend selected.

Define the available Y axes inn the "Y Axes" tab.

Access in runtime: Read and write

Syntax

Object.**TrendYAxis**[=STRING]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl

STRING

Optional. A value or constant that specifies the name of the Y-axis used for the selected trend.

Properties U-W

Unit

Description

Specifies the unit of measurement.

Access in runtime: Read and write

Syntax

Object.**Unit**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- Bar
- IOField

STRING

Optional A value or a constant that specifies the unit of measurement.

UnitColor

Description

Specifies the text color for the unit of measurement in the Gauge" object

Access in runtime: Read and write

Syntax

Object.**UnitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

Color

Optional A value or a constant that specifies the text color for the measurement unit.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

UnitText

Description

Specifies the text for the measurement unit of the selected object.

Access in runtime: Read and write

Syntax

Object.**UnitText**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

STRING

Optional A value or a constant that specifies the text for the measurement unit.

Comments

Enter a text to show, for example, the physical unit of the displayed value. It is standard for no text to be set as a default.

UnitTop

Description

Specifies the distance of the measurement unit to the upper edge of the selected object. The lettering can be positioned only in line with the vertical diameter of the scale. The value of the property refers to the height of the selected object and is measured from the upper edge of the selected object to the lower edge of the lettering.

Access in runtime: Read and write

Syntax

Object.**UnitTop**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the distance of the measurement unit to the upper edge of the selected object.

Value range from 0 to 1

0: The lower edge of the lettering is positioned on the upper limit of the selected object. The text is no longer visible as it is positioned outside of the selected object.

1: The lower edge of the lettering is positioned on the lower limit of the selected object.

UpperLimit

Description

Specifies the upper limit for input values.

Access in runtime: Read and write

Syntax

Object.**UpperLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- IOField

DOUBLE

Optional A value or a constant that specifies the upper limit for input values.

Used

Description

Returns the size of the used disk space.

Access in runtime: Read and write

Syntax

Object.**Used**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

DOUBLE

Optional A value or a constant that returns the size of the used disk space.

UseDesignColorSchema

Description

Specifies whether the colors defined in the global color scheme of the current design are used for this object.

Access in runtime: Read and write

Syntax

Object.**UseDesignColorSchema**[=BOOLEAN]

Object

Required. An object of the "ScreenItem"" type with the following format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Ellipse
- EllipseSegment
- EllipticalArc
- Gauge
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- Slider
- SymbolicIOField
- TextField
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- WindowsSlider

You have no access in runtime with the following format:

- AlarmView
- DateTimeField

- RecipeView
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

BOOLEAN

Optional. TRUE if the object is displayed with the colors from the global color scheme defined for this object type.

FALSE if the object is displayed with the colors as per the settings in the object.

UseDesignShadowSettings

Description

Defines whether the object will be displayed with the shadowing defined in the active design.

Access in runtime: Read and write

Syntax

Object.**UseDesignShadowSettings**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Circle
- CircleSegment
- CircularArc
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicView
- Line
- Polygon
- Polyline
- Rectangle
- TextField
- TubeArcObject
- TubeDoubleTeeObject

- Tubepolyline
- TubeTeeObject
- Bar
- Button
- CheckBox
- Clock
- ComboBox
- Gauge
- GraphicIOField
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- Slider
- SymbolicIOField
- WindowsSlider

BOOLEAN

TRUE: The object is displayed with the global shading defined for this object type.

FALSE: The object is displayed without shadows.

UsedPercent

Description

Specifies the measured value for the used disk space as a percentage. The values can be queried in Runtime. The values cannot be predefined.

Access in runtime: Read and write

Syntax

Object.**UsedPercent**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

Int32

Optional A value or a constant that returns the measured values for the used disk space as a percentage.

UseExponentialFormat

Description

Specifies whether the figures are shown exponentially (e.g. "1.00e+000").

Access in runtime: Read and write

Syntax

Object.**UseExponentialFormat**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the figures are shown exponentially (e.g. "1.00e+000").

UseFlashTransparentColor

Description

Specifies whether the color of the bitmap object of a flashing graphic will be set to ""transparent".

Access in Runtime: Read and write

Syntax

Object.**UseFlashTransparentColor**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "GraphicIOField".

BOOLEAN

Optional TRUE, if the color of the bitmap object of a flashing graphic will be set to "transparent".

See also

GraphicIOField (Page 5088)

UseMessageColor

Description

Specifies whether the agreed colors of the message classes are displayed.

Access in runtime: Read and write

Syntax

Object.**UseMessageColor** [=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

BOOLEAN

TRUE: The colors are displayed.

FALSE: The color settings specified for the table content on the "Layout" tab are effective.

UseSelectedTitleColor

Description

Specifies whether to use a selection color for the headers of selected table cells.

Access in runtime: Read and write

Syntax

Object.**UseSelectedTitleColor**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: A marking color is used. The "Background" and "Font" settings are active in Runtime.

FALSE: A marking color is not used. The "Background" and "Font" settings are disabled in Runtime.

UseTableColor2

Description

Specifies whether to use a second row color for the representation of the table.

Access in runtime: Read and write

Syntax

Object.**UseTableColor2**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: The settings of "RowColor 2" are used in alternation with "RowColor1".

FALSE: The settings of "RowColor 1" are used for all rows.

UseTagLimitColors

Description

Specifies that the colors are used.

Access in runtime: Read and write

Syntax

Object.**UseTagLimitColors**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the following format:

- IOField

BOOLEAN

Optional. TRUE if the colors are used.

UseTransparentColor

Description

Specifies whether the color described with the property "TransparentColor" will be shown as transparent for the selected object.

Access in runtime: Read and write

Syntax

Object.**UseTransparentColor**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- GraphicView
- GraphicIOField

BOOLEAN

Optional TRUE, if the specified color is to be shown as transparent.

UseTransparentColorDeactivatedPicture

Description

Specifies if the transparent color defined with the property "TransparentColorDeactivatedPicture" for the status "disabled" will be used.

Access in runtime: Read and write

Syntax

Object.**UseTransparentColorDeactivatedPicture**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- RoundButton

BOOLEAN

Optional TRUE, if the transparent color defined with the property "TransparentColorDeactivatedPicture" for the status "disabled" will be used.

UseTransparentColorPictureOff

Description

Specifies if the transparent color defined with the property "TransparentColorPictureOff" for the "off" status will be used.

Access in runtime: Read and write

Syntax

Object.**UseTransparentColorPictureOff**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton

BOOLEAN

Optional TRUE, if the transparent color defined with the property "TransparentColorPictureOff" for the "off" status will be used.

UseTransparentColorPictureOn

Description

Specifies if the transparent color defined with the property "TransparentColorPictureOn" for the "on" status will be used.

Access in runtime: Read and write

Syntax

Object.**UseTransparentColorPictureOn**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton

BOOLEAN

Optional TRUE, if the transparent color defined with the property "TransparentColorPictureOn" for the "on" status will be used.

UseTrendNameAsLabel

Description

Specifies whether the "Name" or "Label" property is used as a designation for the trend in runtime.

Access in runtime: Read and write

Syntax

Object.**UseTrendNameAsLabel**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The name configured under "Properties > Properties > Trends > Name" is used.

FALSE: The name configured under "Properties > Properties > Trends > Label" is used.

UserName

Description

Returns the name of the user who triggered the alarm object.

UV_ColumnWidth_AKZ

Description

No access in runtime.

UV_ColumnWidth_Descriptor

Description

No access in runtime.

UV_ColumnWidth_InstallationDate

Description

No access in runtime.

UV_ColumnWidth_LADDR

Description

No access in runtime.

UV_ColumnWidth_Name

Description

No access in runtime.

UV_ColumnWidth_OKZ

Description

No access in runtime.

UV_ColumnWidth_OperationState

Description

No access in runtime.

UV_ColumnWidth_OrderID

Description

No access in runtime.

UV_ColumnWidth_ProfileID

Description

No access in runtime.

UV_ColumnWidth_Rack

Description

No access in runtime.

UV_ColumnWidth_Slot

Description

No access in runtime.

UV_ColumnWidth_SoftwareRevision

Description

No access in runtime.

UV_ColumnWidth_SpecificProfileData

Description

No access in runtime.

UV_ColumnWidth_State

Description

No access in runtime.

UV_ColumnWidth_Station

Description

No access in runtime.

UV_ColumnWidth_SubAddress

Description

No access in runtime.

UV_ColumnWidth_SubSlot

Description

No access in runtime.

UV_ColumnWidth_SubSystem

Description

No access in runtime.

UV_ColumnWidth_Type

Description

No access in runtime.

UV_ShowItem_AKZ

Description

No access in runtime.

UV_ShowItem_Descriptor

Description

No access in runtime.

UV_ShowItem_InstallationDate

Description

No access in runtime.

UV_ShowItem_LADDR

Description

No access in runtime.

UV_ShowItem_Name

Description

No access in runtime.

UV_ShowItem_OKZ

Description

No access in runtime.

UV_ShowItem_OperationState

Description

No access in runtime.

UV_ShowItem_OrderID

Description

No access in runtime.

UV_ShowItem_ProfileID

Description

No access in runtime.

UV_ShowItem_Rack

Description

No access in runtime.

UV_ShowItem_Slot

Description

No access in runtime.

UV_ShowItem_SoftwareRevision

Description

No access in runtime.

UV_ShowItem_SpecificProfileData

Description

No access in runtime.

UV_ShowItem_State

Description

No access in runtime.

UV_ShowItem_Station

Description

No access in runtime.

UV_ShowItem_SubAddress

Description

No access in runtime.

UV_ShowItem_SubSlot

Description

No access in runtime.

UV_ShowItem_SubSystem

Description

No access in runtime.

UV_ShowItem_Type

Description

No access in runtime.

Value

Description

Specifies a value for the object being used or returns it.

Access during runtime: Read and write

Syntax

Object.**Value**[=VARIANT]

Object

Necessary. An object of type "Tag", "DataItem" or "ScreenItem" with the format "Gauge".

VARIANT

Optional The value that is specified dependent on the object being used:

- Tag.Value: returns the tag value for the last read access or specifies the future tag value. Use the "Read" method to read the tag value from the "Value" property. You assign a new tag value to the "Value" property with the "Write" method.
- Dataset.Value: Specifies a value or returns a copy of the value or the object reference. When returning object references, ensure that the object reference is multithread-capable.
- ScreenItem("Gauge_1").Value: Specifies the value to which the pointer points. The value range within the values set via the properties "ValueMin",and "ValueMax".

Examples

The following example writes a new value in the "Tag1" tag:

```
'VBS94
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Value = 50
objTag.Write
```

The example shows how to add a value to a list of tags and output it as a trace. After that, the value is changed, output again and then removed. It make sense to perform this in several different actions:

```
'VBS198
HMIRuntime.DataSet.Add "motor1", 23
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet("motor1").Value = 55
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet.Remove("motor1")
```

See also

DataItem (Page 4987)

Tag (Page 5012)

ValueAxisAutoRange(i)

Description

Specifies whether the value range of the Y axis is determined automatically or via the values "ValueAxisBegin(i)" and "ValueAxisEnd(i)".

Access in runtime: Read and write

Syntax

Object.**ValueAxisAutoRange**(i)[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTrendControl

BOOLEAN

Optional TRUE, if the value range of the Y axis is established automatically or via the values "ValueAxisBegin(i)" and "ValueAxisEnd(i)"".

ValueAxisLabel(i)

Description

Specifies the labeling of the value axis. Whether or not the information is evaluated depends on the "ConfigureTimeAxis(i)" property.

Access in runtime: Read and write

Syntax

Object.ValueAxisLabel(i)[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTrendControl

STRING

Optional A value or a constant that specifies the labeling of the value axis.

ValueAxisScalingType(i)

Description

Specifies the type of axis scaling.

Access in Runtime: Read and write

Syntax

Object.ValueAxisScalingType(i)[=AxisScalingType]

Object

Required An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

AxisScalingType

hmiBarScalingLinear (0): Linear axis scaling

hmiBarScalingLogarithmic (1): Logarithmic axis scaling

hmiBarScalingNegativeLogarithmic (2): Negative logarithmic axis scaling

Comments

The parameter i indicates the number of the trend.

ValueColumnAlignment(i)

Description

Specifies the alignment of the tag values of this column pair. The parameter i indicates the number of the column pair.

Access in runtime: Read and write

Syntax

Object.ValueColumnAlignment(i)[=HorizontalAlignment]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

HorizontalAlignment

hmiAlignmentLeft (0): The text is left-justified.

hmiAlignmentCentered (1): The text is centered.

hmiAlignmentRight (2): The text is right-justified.

VerticalAlignment

Description

Specifies the vertical alignment of the text in the specified object.

Access in runtime: Read and write

Syntax

Object.**VerticalAlignment**[=VerticalAlignment]

Object

Required. A "ScreenItem" object with the following format:

- Button
- DateTimeField
- IOField
- OptionGroup
- RoundButton
- Switch
- SymbolicIOField
- TextField

You have no access in runtime with the following format:

- CheckBox

VerticalAlignment

Optional A value that specifies the vertical alignment.

hmiAlignmentTop (0): The text is shown at the top.

hmiAlignmentMiddle (1): The text is shown in the middle.

hmiAlignmentBottom (2): The text is shown at the bottom.

VerticalGridLines

Description

Enables the display of vertical dividers.

Access in runtime: Read and write

Syntax

Object.**VerticalGridLines**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE: Vertical grid lines are displayed.

FALSE: Vertical grid lines are not displayed.

VerticalScrollbarPosition

Description

Specifies the vertical positioning of the scroll bar in a picture window with slider, or returns its value.

The picture is displayed in the picture window by positioning the horizontal and vertical scroll bars. If you wish to display the screen as a cutout where the scroll bars are located at the left and top edge of the screen, use the OffsetLeft" and "OffsetTop" properties as the source of this cutout.

Access in runtime: Read and write

Syntax

Object.**VerticalScrollBarPosition**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Screenwindow

Int32

Optional A value or constant that specifies the offset of the vertical scroll bar.

ViewOnly

Description

Specifies whether the Sm@rtClient display will be used for remote monitoring or remote maintenance.

Remote maintenance means that settings can be changed on the monitored device.

Remote monitoring means that settings of the monitored device cannot be changed.

Access in runtime: Read and write

Syntax

Object.**ViewOnly**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- SmartClientView

BOOLEAN

Optional TRUE, if the Sm@rtClient display is to be used only for remote monitoring.

Visible

Description

Specifies whether the selected object is visible.

Access in runtime: Read and write

Syntax

Object.**Visible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- AlarmView
- ApplicationWindow
- Bar
- BatteryView

- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- DateTimeField
- DiscSpaceView
- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge
- GraphicIOField
- HTML-Browser
- IOField
- Line
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- Polygon
- Polyline
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- Rectangle
- RoundButton
- Screenwindow

- Slider
- SmartClientView
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TextField
- TrendRulerControl
- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserArchiveControl
- UserView
- WLanQualityView
- WindowsSlider
- ZoneLabelView
- ZoneQualityView

You have no access in runtime with the following format:

- S7GraphOverview

BOOLEAN

Optional TRUE, if the object is visible.

Warning

Description

Specifies the percentage of used storage space as of which a warning is to be generated.

Access in runtime: Read and write

Syntax

Object.**Warning**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- DiskSpaceView

Int32

Optional A value or a constant that specifies the percentage of used storage space as of which a warning is generated.

WarningColor

Description

Specifies the color in which the bar of the storage space display will be shown as soon as the warning area is reached.

Access in runtime: Read and write

Syntax

Object.**WarningColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- DiskSpaceView

Color

Optional A value or a constant that specifies the color in which the bar of the storage space display will be shown as soon as the warning area is exceeded.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

WarningLowerLimit

Description

Specifies the lower limit for "WarningLowerLimit".

The "WarningLowerLimitEnable" property must have the value "TRUE" for the limit to be monitored.

Access in runtime: Read and write

Syntax

Object.**WarningLowerLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the lower limit for "WarningLowerLimit".

Comments

The following values will be specified with the properties "WarningLowerLimitColor" and "WarningLowerLimitRelative":

- Representation upon reaching the limit
- Type of evaluation

WarningLowerLimitColor

Description

Specifies the color for the lower limit "WarningLowerLimit".

The "WarningLowerLimitEnable" property must have the value "TRUE" if the bar color is to change when the limit is reached.

Access in runtime: Read and write

Syntax

Object.**WarningLowerLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Color

Optional A value or a constant that specifies the color for the lower limit "WarningLowerLimit".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

The following values are defined with the "WarningUpperLimit", "WarningUpperLimitColor" and "WarningUpperLimitRelative" properties:

- Limit
- Representation upon reaching the limit
- Type of evaluation

WarningLowerLimitEnabled

Description

Specifies whether the "WarningLowerLimit" limit is to be monitored.

Access in runtime: Read and write

Syntax

Object.**WarningLowerLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the "WarningLowerLimit" limit is monitored.

Comments

The following values will be defined via the properties "WarningLowerLimit", "WarningLowerLimitColor" and "WarningLowerLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

WarningLowerLimitRelative

Description

Specifies whether the lower limit "WarningLowerLimit" is evaluated as a percentage or as an absolute value.

Access in runtime: Read and write

Syntax

Object.**WarningLowerLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional

TRUE: The lower limit "WarningLowerLimit" is evaluated as a percentage.

FALSE: The lower limit "WarningLowerLimit" is evaluated as an absolute value.

WarningRangeColor

Description

Specifies the color of the warning range on the scale of the "Gauge" object.

The "WarningRangeVisible" property must have the value TRUE so that the warning range is displayed.

Access in runtime: Read and write

Syntax

Object.**WarningRangeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

Color

Optional A value or a constant that specifies the color of the warning range.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

WarningRangeStart

Description

Specifies at which scale value the warning range of the "Gauge" object will start.

The "WarningRangeVisible" property must have the value TRUE so that the warning range is displayed.

Access in runtime: Read and write

Syntax

Object.**WarningRangeStart**[=SINGLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

SINGLE

Optional A value or a constant that contains the scale value for the start of the warning range.

Comments

The range extends from the value "Warning" through to the value "Danger". If no range is activated for "Danger", the range for "Warning" extends to the end of the scale.

WarningRangeVisible

Description

Specifies whether the warning range in the scale of the "Gauge" object will be displayed.

Access in runtime: Read and write

Syntax

Object.**WarningRangeVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

BOOLEAN

Optional TRUE, if the warning range will be displayed in the scale.

Comments

Specifies the color of the warning range with the "WarningRangeColor" property.

Specifies the start of the warning range with the "WarningRangeStart" property.

WarningUpperLimit

Description

Specifies the upper warning limit.

The "WarningUpperLimitEnabled"" property must have the value TRUE so that the limit is monitored.

Access in runtime: Read and write

Syntax

Object.**WarningUpperLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the higher limit.

Comments

"WarningUpperLimitColor" defines the display for when the limit is reached.

"WarningUpperLimitRelative" specifies the type of evaluation.

WarningUpperLimitColor

Description

Defines the color for the lower warning limit.

The ""WarningUpperLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime: Read and write

Syntax

Object.**WarningUpperLimitColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Color

Optional A value or a constant that specifies the color for the higher limit.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

WarningUpperLimitEnabled

Description

Specifies whether the higher limit is to be monitored.

Access in runtime: Read and write

Syntax

Object.**WarningUpperLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE if the higher limit is monitored.

WarningUpperLimitRelative

Description

Determines whether the lower limit "WarningUpperLimit" is to be evaluated as a percentage or as an absolute value.

Access in runtime: Read and write

Syntax

Object.**WarningUpperLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the lower limit "WarningUpperLimit" is to be evaluated as a percentage.

Width

Description

Specifies the width of the object in pixels.

Access in runtime: Read and write

Syntax

Object.**Width**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- AlarmView
- ApplicationWindow
- Bar
- BatteryView
- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- DateTimeField
- DiscSpaceView
- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- HTML-Browser
- IOField
- Line

- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- Polygon
- Polyline
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- Rectangle
- RoundButton
- Screenwindow
- Slider
- SmartClientView
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TextField
- TrendRulerControl
- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserArchiveControl
- UserView
- WLanQualityView
- WindowsSlider

- ZoneLabelView
- ZoneQualityView

You have no access in runtime with the following format:

- S7GraphOverview

Int32

Optional A value or a constant that specifies the width of the object in pixels.

WindowCloseEnabled

Description

Indicates whether a window can be closed in runtime.

Access in runtime: Read

Syntax

Object.**WindowCloseEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- ApplicationWindow
- ScreenWindow

BOOLEAN

Optional TRUE, if the window can be closed in Runtime.

WindowMaximizeEnabled

Description

Returns whether the object can be maximized in runtime.

Access in runtime: Read

Syntax

Object.**WindowMaximizeEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- ApplicationWindow
- Screenwindow

BOOLEAN

TRUE: The object can be maximized in runtime.

FALSE: The object cannot be maximized in runtime.

WindowMovingEnabled

Description

Returns whether the object can be moved in runtime.

Access in runtime: Read

Syntax

Object.**WindowMovingEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- ApplicationWindow
- Screenwindow

BOOLEAN

TRUE: The object can be moved in runtime.

FALSE: The object cannot be moved in runtime.

WindowOnTop

Description

Returns whether the object always remains in the foreground in runtime.

Access in runtime: Read

Syntax

Object.**WindowOnTop**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- ApplicationWindow
- Screenwindow

BOOLEAN

TRUE: The object remains in the foreground in runtime.

FALSE: The object does not always remain in the foreground in runtime.

WindowsContents

Description

Supplies the content of the application window Read only access.

See also

ScriptDiagnostics (Page 5157)

WindowSizingEnabled

Description

Indicates whether the size of the specified object can be changed in runtime.

Access in runtime: Read

Syntax

Object.**WindowSizingEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- ApplicationWindow
- ScreenWindow

BOOLEAN

Optional TRUE, if the size of the selected object can be changed in runtime.

WindowsStyle

Description

Specifies whether the object will be displayed in the general Windows style.

Access in runtime: Read and write

Syntax

Object.**WindowsStyle**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Button
- WindowSlider

You have no access in runtime with the following format:

- Switch

BOOLEAN

Optional TRUE, if the object will be displayed in the general Windows style.

Properties X-Z

XAxisAdd

Description

Creates a new X axis.

Syntax

Object.**XAxisAdd**[=STRING]

Object

Required. An object of the "ScreenItem" type with the format:

- FunctionTrendControl

STRING

Optional. A value or a constant, which????

XAxisAlignment

Description

Specifies how the selected axis is aligned.

Syntax

Object.**XAxisAlignment**[=AxisAlignment]

Object

Required. A "ScreenItem" object with the format "FunctionTrendControl".

AxisAlignment

Value	Description	Explanation
0	Bottom	The X axis selected is displayed below the trend.
1	Top	The X axis selected is displayed above the trend.

XAxisAutoPrecisions

Description

Enables automatic setting of the decimal precision.

Access in runtime: Read and write

Syntax

Object.**XAxisAutoPrecisions**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl

BOOLEAN

TRUE: The number of decimal places is specified automatically.

FALSE: The value configured under "Properties > X axis > Format > Decimal places" is effective.

XAxisAutoRange(i)

Description

Specifies whether the value range of the X-axis is determined automatically or with the "XAxisBegin(i)" and "XAxisEnd(i)" attributes.

Access in runtime: Read and write

Syntax

Object.**XAxisAutoRange(i)**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

BOOLEAN

Optional TRUE if the value range of the X-axis is established automatically or with the "XAxisBegin(i)" and "XAxisEnd(i)" attributes.

XAxisBeginValue

Description

Specifies the lower end of the value range of the selected X-axis.

To configure the value, deselect the "Automatic" option under "Value range".

Access in runtime: Read and write

Syntax

Object.**XAxisBeginValue**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

DOUBLE

Optional A value or a constant that specifies the lower limit of the value range of the selected X-axis.

XAxisColor

Description

Use this attribute to define the color for the common X-axis.

Access in runtime: Read and write

Syntax

Object.**XAxisColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

Color

A value or a constant that specifies the color used for the common X-axis.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

XAxisCount

Description

Specifies the number of X-axes.

Access in runtime: Read and write

Syntax

Object.**XAxisCount**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

Int32

Optional A value or a constant that specifies the number of X-axes.

XAxisEndValue

Description

Specifies the upper end of the value range of the selected X-axis.

To configure the value, deselect the "Automatic" option under "Value range".

Access in runtime: Read and write

Syntax

Object.**XAxisEndValue**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

DOUBLE

Optional A value or a constant that specifies the upper limit of the value range of the selected X-axis.

XAxisExponentialFormat

Description

Enables the exponential notation for visualization of a selected axis.

Access in runtime: Read and write

Syntax

Object.**XAxisExponentialFormat**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

BOOLEAN

TRUE: The values are displayed in exponential format.

FALSE: The values are not displayed in exponential format.

XAxisIndex

Description

References a configured X axis. You can assign the values of other properties to a specific X axis by using the property.

Values of between 0 and "XAxisCount" minus 1 are valid for "XAxisIndex". The property "ToolbarButtonCount" specifies the number of configured X-axes.

Access in runtime: Read and write

Syntax

Object.**XAxisIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

Int32

Optional A value or a constant that specifies the number of the selected X-axis.

XAxisInTrendColor

Description

Specifies whether the selected axis is shown in the trend color. The color of the first trend is used if there are several trends displayed in the trend window. The order of the trends is defined in the "Trends" properties.

Access in runtime: Read and write

Syntax

Object.**XAxisInTrendColor**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl

BOOLEAN

TRUE: The selected axis is displayed in the trend color. The setting in the "Color" field is disabled.

FALSE: The selected axis is displayed in the color which is set in the "Color" field.

XAxisLabel(i)

Description

In line with the value of "XAxisMode(i)", specifies the X-axis labeling of a trend that is referenced with "CurrentCurveIndex".

Access in Runtime: Read and write

Syntax

Object.XAxisLabel(i)[=STRING]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the labeling of the X-axis of a trend referenced with "CurrentCurveIndex".

XAxisName

Description

Specifies the name of the selected axis.

Access in runtime: Read and write

Syntax

Object.**XAxisName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the name of the selected axis.

XAxisPrecisions

Description

Specifies the number of decimal places to be displayed for the values of the selected X-axis.

To enter the value, deselect the "Automatic" option under "Format".

Access in runtime: Read and write

Syntax

Object.**XAxisPrecisions**[=Int16]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

Int16

Optional A value or a constant that specifies the number of decimal places.

XAxisRemove

Description

Removes the selected axis from the list.

Access in runtime: Read and write

Syntax

Object.**XAxisRemove**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the axis that is to be removed from the list.

XAxisRepos

Description

Specifies the order of the axes. The order in the list determines the position of the axis in the trend view in Runtime. The axis is displayed further away from the trend if it is higher up the list and the alignment is the same.

Access in runtime: Read and write

Syntax

Object.**XAxisRepos**[=Int32]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl

Int32

Optional. A value or a constant that specifies the order of the axes???

XAxisScalingType(i)

Description

Specifies the type of scaling for the X axis. Whether the information is evaluated depends on the value of the "XAxisMode(i)" attribute.

Access in Runtime: Read and write

Syntax

Object.**XAxisScalingType**(i)[=AxisScalingType]

Object

Required An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

AxisScalingType

hmiBarScalingLinear (0): Linear axis scaling.

hmiBarScalingLogarithmic (1): Logarithmic axis scaling

hmiBarScalingNegativeLogarithmic (2): Negative logarithmic axis scaling

Comments

The parameter i indicates the number of the trend.

XAxisTrendWindow

Description

Specifies the trend window in which the selected axis is used. The available trend windows are specified in the properties under "Trends".

Access in runtime: Read and write

Syntax

Object.**XAxisTrendWindow**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the name of the trend window.

XAxisVisible

Description

The list shows all axes that you have created.

From the list, select the axes that are to be displayed in the trend views.

Click on an axis in the list to adapt the properties and to assign the axis to a trend view.

Access in runtime: Read and write

Syntax

Object.**XAxisVisible**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl

BOOLEAN

Optional. Specifies whether the X-axis is to be visible.

YAxisAdd

Description

Creates a new Y axis.

Syntax

Object.**YAxisAdd**[=STRING]

Object

Required An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies ???.

YAxisAlignment

Description

Specifies how the selected axis is aligned.

Syntax

Object.YAxisAlignment[=AxisAlignment]

Object

Required. A "ScreenItem" object with the format "FunctionTrendControl".

AxisAlignment

Value	Description	Explanation
0	Left	The X axis selected is displayed on left side of the trend.
1	Right	The selected Y axis is displayed to the right of the trend.

YAxisAutoPrecisions

Description

Enables automatic setting of the decimal precision.

Access in runtime: Read and write

Syntax

Object.**YAxisAutoPrecisions**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

BOOLEAN

TRUE: The number of decimal places is specified automatically.

FALSE: The value configured in the Inspector window under "Properties > Y-axis > Format > Decimal places" is effective.

YAxisAutoRange

Description

Specifies whether the value range of the selected Y-axis is calculated automatically.

Access in runtime: Read and write

Syntax

Object.**YAxisAutoRange**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

BOOLEAN

Value	Explanation
TRUE	The range of values is calculated automatically.
FALSE	The range of values is calculated based in the values configured in the "from" and "to" fields.

YAxisBeginValue

Description

Specifies the lower end of the value range of the selected Y-axis.

To configure the value, deselect the "Automatic" option under "Value range".

Access in runtime: Read and write

Syntax

Object.**YAxisBeginValue**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

DOUBLE

Optional A value or a constant that specifies the lower limit of the value range of the selected Y-axis.

YAxisColor

Description

Specifies the color of the selected axis. Open the "Color selection" dialog by clicking the button. The setting is only active if the "Use trend color" field is not selected.

Access in runtime: Read and write

Syntax

Object.**YAxisColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

Color

Optional A value or a constant that specifies the color of the selected axis.

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as vbRed and vbGreen.

YAxisCount

Description

Specifies the number of configured Y-axes.

Access in runtime: Read and write

Syntax

Object.**YAxisCount**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

Int32

Optional A value or a constant that specifies the number of configured Y-axes.

YAxisEndValue

Description

Specifies the upper end of the value range of the selected Y-axis.

To configure the value, deselect the "Automatic" option under "Value range".

Access in runtime: Read and write

Syntax

Object.**YAxisEndValue**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

DOUBLE

Optional A value or a constant that specifies the upper limit of the value range of the selected Y-axis.

YAxisExponentialFormat

Description

Enables the exponential notation for visualization of a selected axis.

Access in runtime: Read and write

Syntax

Object.**YAxisExponentialFormat**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

BOOLEAN

TRUE: The values are displayed in exponential format.

FALSE: The values are not displayed in exponential format.

YAxisIndex

Description

References a configured Y axis. You can assign the values of other properties to a specific Y axis by using the property.

Values of between 0 and "YAxisCount" minus 1 are valid for "YAxisIndex". The property "ToolbarButtonCount" specifies the number of configured Y-axes.

Access in runtime: Read and write

Syntax

Object.**YAxisIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

Int32

Optional A value or a constant that specifies the number of the selected Y-axis.

YAxisInTrendColor

Description

Specifies whether the selected axis is shown in the trend color. The color of the first trend is used if there are several trends displayed in the trend window. The order of the trends is defined in the "Trends" properties.

Access in runtime: Read and write

Syntax

Object.**YAxisInTrendColor**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

BOOLEAN

TRUE: The selected axis is displayed in the trend color. The setting in the "Color" field is disabled.

FALSE: The selected axis is displayed in the color set in the "Color" field.

YAxisLabel

Description

Defines the label text for a selected axis.

Access in runtime: Read and write

Syntax

Object.**YAxisLabel**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the label text for the selected Y-axis.

YAxisName

Description

Specifies the name of the selected axis.

Access in runtime: Read and write

Syntax

Object.**YAxisName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the name of the selected axis.

YAxisPrecisions

Description

Specifies the number of decimal places to be displayed for the values of the selected Y-axis.

To enter the value, deselect the "Automatic" option under "Format".

Access in runtime: Read and write

Syntax

Object.**YAxisPrecisions**[=Int16]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

Int16

Optional A value or a constant that specifies the number of decimal places.

YAxisRemove

Description

Removes the selected axis from the list.

Access in runtime: Read and write

Syntax

Object.**YAxisRemove**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the name of the selected axis.

YAxisRename

Description

Changes the name of the Y axis which is referenced by the "YAxisIndex" property.

Access in runtime: Read and write

Syntax

Object.**YAxisRename**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the new name of the selected Y-axis.

YAxisRepos

Description

Changes the sorting order of the axes. "Up" and "Down" move the axis selected up or down in the list.

The order in the list determines the position of the axis in the trend view in Runtime. The axis output position is moved away from the trend if the axis is moved further up in the list and the orientation is the same.

Access in runtime: Read and write

Syntax

Object.**YAxisRepos**[=Int32]

Object

Required An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Int32

Optional ##########.

YAxisScalingType

Description

Specifies how the Y-axis is scaled.

Access in runtime: Read and write

Syntax

Object.**YAxisScalingType**[=AxisScalingType]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

AxisScalingType

The following settings are available:

Value	Description
0	Linear
1	Logarithmic
2	Negative logarithmic

YAxisTrendWindow

Description

Specifies the trend window in which the selected axis is used. The available trend windows are specified in the properties under "Trends".

Access in runtime: Read and write

Syntax

Object.**YAxisTrendWindow**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the name of the trend window.

YAxisVisible

Description

The list shows all axes that you have created.

From the list, select the axes that are to be displayed in the trend views.

Click on an axis in the list to adapt the properties and to assign the axis to a trend view.

Access in runtime: Read and write

Syntax

Object.**YAxisVisible**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the following format:

- FunctionTrendControl

BOOLEAN

Optional. Specifies whether the selected Y-axis is to be visible.

ZeroPoint

Description

Specifies the position of the zero point as a percentage of the bar height. The zero point can also be located outside of the displayed area.

The "ScalingType" property must be set to "Auto".

The "ShowScale" property must be set to "TRUE".

Access in runtime: Read and write

Syntax

Object.**ZeroPoint**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Int32

Optional A value or a constant that specifies the position of the zero point as a percentage of the bar height.

Zoom/ZoomFactor

Description

Sets or reads the zoom factor of a screen or screen window.

If the indicated zoom factor is smaller than the minimum value, the zoom factor is automatically set to the minimum value. If the indicated zoom factor is larger than the minimum value, the zoom factor is automatically set to the maximum value.

The minimum value of the zoom factor is at 2%, the maximum value at 800%.

With the Screen Object the zoom factor is indicated as a numeric value and with a picture window object, it is indicated in percent.

Zoom: Zoom factor of picture

Zoomfactor: Zoom factor of picture window

Example

The following example doubles the zoom factor of the current picture:

```
'VBS97
HMIRuntime.ActiveScreen.Zoom = HMIRuntime.ActiveScreen.Zoom * 2
```

See also

ScreenWindow (Page 5155)

1

AlarmLow

Description

Defines the bottom limit value at which an alarm should be triggered or returned.
The type of evaluation (percentage or absolute) is specified with the "TypeAlarmLow" property.

Access in Runtime: Write and read

Syntax

Object.AlarmLow[=REAL]

Object

Required A "ScreenItem" object with the format "Bar".

REAL

Value for the lower limit

See also

Bar (Page 5033)

AlignmentTop

Description

Defines or returns the vertical alignment of the text. Value range from 0 to 2.

Access in Runtime: Read and write

Syntax

Object.**AlignmentTopt**[=VerticalAlignment]

Object

Required An object of the "ScreenItem" type with the format "TextField", "IOField", "SymbolicIOField", "Button", "RoundButton", "CheckBox", "OptionGroup".

VerticalAlignment

0: Alignment left

1: Alignment centered

2: Alignment right

Application

Description

Supplies the content of the application window Read only access.

See also

ScriptDiagnostics (Page 5157)

AssumeOnExit

Description

Specifies whether the entered text is applied after leaving the input field, e.g. with the <TAB> key or a mouseclick.

Access in Runtime: Read and write

Syntax

Object.**AssumeOnExit**

Object

Required An object of the "ScreenItem" type with the format "IOField", "SymbolicIOField".

See also

IOField (Page 5098)

SymbolicIOField (Page 5170)

AssumeOnFull

Description

Specifies whether the input field is left automatically after a complete input (the specified number of characters was entered) and the input is applied.

Access in Runtime: Read and write

Syntax

Object.AssumeOnFull[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "IOField", "SymbolicIOField".

BOOLEAN

TRUE: is applied.

FALSE: is not applied.

See also

IOField (Page 5098)

SymbolicIOField (Page 5170)

BackBorderWidth

Description

Defines or returns the width of the 3D border in pixels. The value for the width is dependent on the size of the object.

BackColor3 property

Description

Defines or returns the color of the bar background. LONG write-read access.

BackFlashColorOff

Description

Defines or returns the color of the object background for the flash status "Off". LONG write-read access.

BackFlashColorOn

Description

Defines or returns the color of the object background for the flash status "On". LONG write-read access.

BorderFlashColorOff

Description

Defines or returns the color of the object lines for the flashing status "Off". LONG write-read access.

BorderFlashColorOn

Description

Defines or returns the color of the object lines for the flashing status "On". LONG write-read access.

BottomConnectedConnectionPointIndex

Description

Specifies or sets the index number of the bottom connecting point.

LONG write-read access.

BottomConnectedObjectName

Description

Specifies or sets the object name of the object which is docked on at the bottom connecting point.

LONG write-read access.

BoxAlignment

Description

TRUE, when the fields are arranged aligned to the right. BOOLEAN write-read access.

BoxCount

Description

Defines or returns the number of fields. Value range from 0 to 31.

BoxType

Description

Defines or returns the field type. Value range from 0 to 2:

- 0: Edition
- 1: Input
- 2: I/O field

Button1Width

Description

Defines or returns the width of the Button 1 in pixels.
When the SameSize property is set to TRUE, all the buttons are specified the same width.

Button2Width

Description

Defines or returns the width of the Button 2 in pixels.
When the SameSize property is set to TRUE, all the buttons are specified the same width.

Button3Width

Description

Defines or returns the width of the Button 3 in pixels.
When the SameSize property is set to TRUE, all the buttons are specified the same width.

Button4Width

Description

Defines or returns the width of the Button 4 in pixels.
When the SameSize property is set to TRUE, all the buttons are specified the same width.

ButtonColor

Description

Defines or returns the color of the slider. LONG write-read access.

ButtonCommand

Description

Specifies the display in the message view.

Access in Runtime: Read and write

Syntax

Object.**ButtonCommand**[=Long]

Object

Required A "ScreenItem" object with the format "AlarmControl".

Long

Optional A value or a constant which specifies the display in the message view.

0x00000001; 1; alarm list·

0x00000002; 2; historical alarm list·(short-term)

0x00000004; 4; historical alarm list·(long-term)

0x00200000; 2097152; lock list·

0x00000008; 8; central annunciator acknowledgement·

0x00000010; 16; single acknowledgement·

0x00000020; 32; group acknowledgement·

0x00000040; 64; Autoscroll·

0x00000080; 128; selection dialog·

0x00000100; 256; block dialog·

0x00000200; 512; print alarm report·

0x00000800; 2048; emergency acknowledgement·

0x00001000; 4096; first alarm·

0x00002000; 8192; last alarm·

0x00004000; 16384; next alarm·

0x00008000; 32768; previous alarm·

0x00010000; 65536; Infotext dialog·

0x00020000; 131072; comment dialog·

0x00040000; 262144; Loop in interrupt·

0x00100000; 1048576; print current view·

0x00400000; 4194304; lock / unlock alarm·

0x00800000; 8388608; sort dialog·

0x01000000; 16777216; time base dialog·

0x02000000; 33554432; hit list

See also

AlarmControl (Page 5018)

CheckAlarmHigh

Description

TRUE, when the "AlarmHigh" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "AlarmHigh", "ColorAlarmHigh" and "TypeAlarmHigh" properties.

CheckAlarmLow

Description

TRUE, when the "AlarmLow" limit value is to be monitored. BOOLEAN write/read access.
The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "AlarmLow", "ColorAlarmLow" and "TypeAlarmLow" properties.

CheckLimitHigh4

Description

TRUE, when the "Reserve 4" upper limit value should be monitored. BOOLEAN write/read access.
The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "LimitHigh4", "ColorLimitHigh4" and "TypeLimitHigh4" properties.

CheckLimitHigh5

Description

TRUE, when the "Reserve 5" upper limit value should be monitored. BOOLEAN write/read access.
The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "LimitHigh5", "ColorLimitHigh5" and "TypeLimitHigh5" properties.

CheckLimitLow4

Description

TRUE, when the "Reserve 4" lower limit value should be monitored. BOOLEAN write/read access.
The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "LimitLow4", "ColorLimitLow4" and "TypeLimitLow4" properties.

CheckLimitLow5

Description

TRUE, when the "Reserve 5" lower limit value should be monitored. BOOLEAN write/read access.
The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "LimitLow5", "ColorLimitLow5" and "TypeLimitLow5" properties.

CheckToleranceHigh

Description

TRUE, when the "ToleranceHigh" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "ToleranceHigh", "ColorToleranceHigh" and "TypeToleranceHigh" properties.

CheckToleranceLow

Description

TRUE, when the "ToleranceLow" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "ToleranceLow", "ColorToleranceLow" and "TypeToleranceLow" properties.

CheckWarningHigh

Description

TRUE, when the "WarningHigh" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "WarningHigh", "ColorWarningHigh" and "TypeWarningHigh" properties.

CheckWarningLow

Description

TRUE, when the "WarningLow" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "WarningLow", "ColorWarningLow" and "TypeWarningLow" properties.

ClearOnNew

Description

TRUE, when the field entry is deleted as soon as the I/O field has the focus. BOOLEAN write-read access.

ColorAlarmHigh

Description

Defines or returns the bar color for the "AlarmHigh" limit value. LONG write/read access. The "CheckAlarmHigh" property must have been set to TRUE if the bar color should change on reaching the limit value.

ColorAlarmLow

Description

Defines or returns the bar color for the "AlarmLow" limit value. LONG write/read access. The "CheckAlarmLow" property must have been set to TRUE if the bar color should change on reaching the limit value.

ColorLimitHigh4

Description

Defines or returns the color for the "Reserve 4" upper limit value. LONG write/read access. The "CheckLimitHigh4" property must have been set to TRUE if the bar color should change on reaching the limit value.

ColorLimitHigh5

Description

Defines or returns the color for the "Reserve 5" upper limit value. LONG write/read access. The "CheckLimitHigh5" property must have been set to TRUE if the bar color should change on reaching the limit value.

ColorLimitLow4

Description

Defines or returns the color for the "Reserve 4" lower limit value. LONG write/read access. The "CheckLimitLow4" property must have been set to TRUE if the bar color should change on reaching the limit value.

ColorLimitLow5

Description

Defines or returns the color for the "Reserve 5" lower limit value. LONG write/read access. The "CheckLimitLow5" property must have been set to TRUE if the bar color should change on reaching the limit value.

ColorToleranceHigh

Description

Defines or returns the color for the "ToleranceHigh" upper limit value. LONG write/read access. The "CheckToleranceHigh" property must have been set to TRUE if the bar color should change on reaching the limit value.

ColorToleranceLow

Description

Defines or returns the color for the "ToleranceLow" lower limit value. LONG write/read access. The "CheckToleranceLow" property must have been set to TRUE if the bar color should change on reaching the limit value.

ColorWarningHigh

Description

Defines or returns the color for the "WarningHigh" upper limit value. LONG write/read access. The "CheckWarningHigh" property must have been set to TRUE if the bar color should change on reaching the limit value.

ColorWarningLow

Description

Defines or returns the color for the "WarningLow" lower limit value. LONG write/read access. The "CheckWarningLow" property must have been set to TRUE if the bar color should change on reaching the limit value.

CountValueColumns

Description

Returns the number of configured trends or column pairs (visible and invisible) of the window.

Access in Runtime: Read and write

Syntax

Object.**CountValueColumns**[=Int]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

Int

Optional A value or a constant that returns the number of configured trends or column pairs (visible and invisible) of the window.

See also

OnlineTableControl (Page 5114)

CurrentColumnIndex

Description

Specifies which settings can be assigned to a selected column pair. The property is evaluated by other properties. Valid values for the index have values from 0 to -1. The "CountValueColumns" property contains the number of column pairs that are to be shown.

Access in Runtime: Read and write

Syntax

Object.**CurrentColumnIndex**[=Int]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

Int

Optional A value or a constant that specifies which settings can be assigned to a particular column pair.

Comments

The "NumItems" property contains the number of trends / column pairs to be shown.

See also

OnlineTableControl (Page 5114)

DeleteEnable

Description

Specifies whether the recipe view can be cleared in Runtime.

Access in Runtime: Read and write

Syntax

Object.**DeleteEnable**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "UserArchiveControl".

BOOLEAN

Optional TRUE, if the recipe view can be cleared in Runtime.

DrawAlwaysInsideFrame

Description

Specifies whether the border line of the selected object should be illustrated with a line weight greater than 1 within the border or symmetrically to the border.

Access in Runtime: Read and write

Syntax

Object.**DrawAlwaysInsideFrame**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Rectangle".

BOOLEAN

Option TRUE, if the border line of the specified object will be illustrated with a line weight greater than 1 within the border.

See also

Rectangle (Page 5149)

DrawStylishButton

Description

Specifies whether the button will be displayed in the general Windows style.

Access in Runtime: Read and write

Syntax

Object.**DrawStylishButton**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Button".

BOOLEAN

Optional TRUE, if the button will be displayed in the general Windows style.

See also

Button (Page 5040)

EditAtOnce

Description

TRUE, if accessing the field with the <TAB> key permits input immediately and without further action. BOOLEAN write-read access.

Exponent

Description

TRUE, when the display of numbers should be with exponents (e.g."1.00e+000"). BOOLEAN write-read access.

FillColor

Description

Defines or returns the fill pattern color for the object.

Access during runtime: Read and write

Syntax

Object.**FillColor**[=LONG]

Object

Necessary. An object of the "ScreenItem" type with the format "Line", "Polyline", "Ellipse", "Circle", "EllipseSegment", "CircleSegment", "EllipticalArc", "CircularArc", "Rectangle", "Polygon", "TextField", "IOField", "SymbolicIOField", "DateTimeField", "Button", "Roundbutton", "Switch", "Bar", "StatusForce", "Clock", "Gauge", "Slider", "Checkbox".

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Enter the appropriate decimal value for each of the three RGB values.

Example:

RGB(200, 150, 100)

Example

The following example defines the fill color for "ScreenWindow1" to blue:

```
'VBS73
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
objScreen.FillStyle = 131075
objScreen.FillColor = RGB(0, 0, 255)
```

Filling

Description

TRUE, when the object can be filled by closed border lines (e.g. representing the fill level of a tank). BOOLEAN write-read access.
The fill level of the object is set by means of the "FillingIndex" property.

FillingIndex

Description

Defines the %age value (related to the height of the object) to which the object with closed border line is to be filled.
The fill level is represented by the current background color. The unfilled background is transparent.

FillStyle2

Description

Defines or returns the fill style of the bar.

FlashBackColor

Description

TRUE, when flashing of the background is activated. BOOLEAN write-read access

FlashBorderColor

Description

TRUE, when flashing of the object lines is activated. BOOLEAN write-read access.

FlashRate

Description

Defines or returns the flash frequency. Value range from 0 to 2.

0 = slow

1 = medium

2 = fast

FlashRateBackColor

Description

Defines or returns the flash frequency for the object background. Value range from 0 to 2.

0 = slow

1 = medium

2 = fast

FormatType

Description

Specifies the data type of the selected object.

Access in Runtime: Read and write

Syntax

Object.**FormatType**[=IOFieldDataFormat]

Object

Required A "ScreenItem" object with the format "IOField".

IOFieldDataFormat

hmiOutputFormatString (2): String

hmiOutputFormatDecimal (1): Decimal

hmiOutputFormatHexadecimal (3): Hexadecimal

hmiOutputFormatBinary (0): Binary

hmiOutputFormatDate (4): Date

hmiOutputFormatTime (5): Time

hmiOutputFormatDateTime (6): Date and time

See also

IOField (Page 5098)

GridBackColor

Description

Specifies the background color for the recipe view in Runtime.

Access in Runtime: Read and write

Syntax

Object.**GridBackColor**[=Color]

Object

Required A "ScreenItem" object with the format "UserArchiveControl".

Color

Optional A value or a constant that specifies the background color in the recipe view in Runtime.

GridForeColor

Description

Defines the font color for the recipe view.

Access in Runtime: Read and write

Syntax

Object.**GridForeColor**[=Color]

Object

Required A "ScreenItem" object with the format "UserArchiveControl".

Color

Optional A value or a constant that specifies the font color for the recipe view.

Hysteresis property

Description

TRUE, when the display should appear with hysteresis. BOOLEAN write-read access.

HysteresisRange

Description

Defines the hysteresis in % of the displayed value or returns it.
The Hysteresis property must be set to TRUE for the hysteresis to be calculated.

InsertEnable

Description

Specifies whether an entry can be made in the recipe view in Runtime.

Access in Runtime: Read and write

Syntax

Object.**InsertEnable**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "UserArchiveControl".

BOOLEAN

Optional TRUE, if an entry can be made in the recipe view in Runtime.

ItemBorderBackColor

Description

Defines or returns the background color for dividing lines in the selection list of the text list object. LONG write-read access. The background color is only visible with the property setting ItemBorderStyle > 0.

ItemBorderColor

Description

Defines or returns the color for deviding lines in the selection list of the text list object. LONG write-read access.

ItemBorderWidth

Description

Defines or returns the dividing line weight in pixels in the selection list of the text list object.

LeftComma property

Description

Defines or returns the number of digits to the left of the decimal point (0 to 20).

LimitHigh4

Description

Determines the upper limit value for "Reserve 4" or returns it.
The CheckLimitHigh4 property must be set to TRUE in order that the "Reserve 4" limit value can be monitored.
The type of the evaluation (in percent or absolute) is defined in the TypeLimitHigh4 property.

LimitHigh5

Description

Determines the upper limit value for "Reserve 5" or returns it.
The CheckLimitHigh5 property must be set to TRUE in order that the "Reserve 5" limit value can be monitored.
The type of the evaluation (in percent or absolute) is defined in the TypeLimitHigh5 property.

LimitLow4

Description

Determines the lower limit value for "Reserve 4" or returns it.
The CheckLimitLow4 property must be set to TRUE in order that the "Reserve 4" limit value can be monitored.
The type of the evaluation (in percent or absolute) is defined in the TypeLimitLow4 property.

LimitLow5

Description

Determines the lower limit value for "Reserve 5" or returns it.
The CheckLimitLow5 property must be set to TRUE in order that the "Reserve 5" limit value can be monitored.
The type of the evaluation (in percent or absolute) is defined in the TypeLimitLow5 property.

LimitMax

Description

Determines the upper limit value as an absolute value depending on the data format or returns it.
If the displayed value exceeds the upper limit value, it is displayed by a sequence of *** (not displayable).

LimitMin

Description

Determines the lower limit value as an absolute value depending on the data format or returns it.
If the displayed value exceeds the upper limit value, it is displayed by a sequence of *** (not displayable).

LockStatus

Description

TRUE, when a locked measuring point should be displayed. BOOLEAN write-read access.

LockText

Description

Defines the label of a button for a locked measuring point.
The LockStatus property must be set to TRUE for the label to be displayed.

LockTextColor

Description

Defines or returns the color of the button label for a locked measuring point. LONG write/read access.
The LockStatus property must be set to TRUE for the background color to be displayed.

LongStrokesBold

Description

TRUE, when the long sections of a scale should be displayed in bold face. BOOLEAN write-read access.

LongStrokesOnly

Description

TRUE, when only the long sections of a scale should be displayed . BOOLEAN write-read access.

LongStrokesSize

Description

Specifies whether only large scale marks are shown.

Access in runtime: Read and write

Syntax

Object.**ShowLargeTicksOnly**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE if only large scale marks are shown.

LongStrokesTextEach

Description

Returns the value which defines which sections of the scale displayed should be labeled (1 = every section, 2 = every second section, etc.). Read only access

LowerLimitColor(i)

Description

Specifies the color that identifies the tag values for the selected trend and which lie below the value of "LowerLimitValue(i)". Whether the information is evaluated depends on the property "LowerLimitEnabled(i)".

Access in Runtime: Read and write

Syntax

Object.**LowerLimitColor**(i)[=Color]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

Color

Optional A value or a constant which specifies the color which identifies tag values of the specified trend which are below the value of "LowerLimitValue(i)".

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

LowerLimitEnabled(i)

Description

Specifies whether the information from "LowerLimitColor(i)" is used to identify the tag values that lie beneath the value of "LowerLimitValue(i)".

Access in Runtime: Read and write

Syntax

Object.**LowerLimitEnabled**(i)[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

BOOLEAN

Optional TRUE, if the information from "LowerLimitColor(i)" is used to identify the tag values that lie beneath the value of "LowerLimitValue(i)".

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

LowerLimitValue(i)

Description

???Specifies whether the tag values which drop below the value of "LowerLimitValue(i)" are marked by the color specified in "LowerLimitColor(i)". Whether the specification is evaluated depends on the attribute "LowerLimitEnabled(i)".

Access in Runtime: Read and write

Syntax

Object.**LowerLimitValue**(i)[=Double]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

Double

Optional A value or a constant that determines if the tag values that fall short of the "LowerLimitValue(i)" are to be identified with the color specified in "LowerLimitColor(i)".

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

Marker property

Description

TRUE, when the limit values should be displayed as scale values. BOOLEAN write-read access.

MaximizeButton property

Description

TRUE, when the object can be maximized in Runtime. Read only access.

MCGUBackColorOff

Description

Defines or returns the background color for flash status "Off" in the case of the "Departed Unacknowledged" status. LONG write-read access.

MCGUBackColorOn

Description

Defines or returns the background color for flash status "On" in the case of the "Departed Unacknowledged" status. LONG write-read access.

MCGUBackFlash

Description

TRUE, when the background should flash when a message departs unacknowledged. BOOLEAN write-read access.

MCGUTextColorOff

Description

Defines or returns the color of the text for flash status "Off" for the "Outgoing unacknowledged" status. LONG write-read access.

MCGUTextColorOn

Description

Defines or returns the background color of the text for flash status "On" for the "Outgoing unacknowledged" status. LONG write-read access.

MCGUTextFlash

Description

TRUE, when the font should flash when a message departs unacknowledged. BOOLEAN write-read access.

MCKOBackColorOff

Description

Defines or returns the background color for flash status "Off" in the case of the "Arrived" status. LONG write-read access.

MCKOBackColorOn

Description

Defines or returns the background color for flash status "On" in the case of the "Arrived" status. LONG write-read access.

MCKOBackFlash

Description

TRUE, when the background should flash when a message arrives. BOOLEAN write-read access.

MCKOTextColorOff

Description

Defines or returns the color of the text for flash status "Off" for the "Incoming" status. LONG write-read access.

MCKOTextColorOn

Description

Defines or returns the background color of the text for flash status "On" for the "Incoming" status. LONG write-read access.

MCKOTextFlash

Description

TRUE, when the font should flash when a message arrives. BOOLEAN write-read access.

MCKQBackColorOff

Description

Defines or returns the background color for flash status "Off" in the case of the "Departed Acknowledged" status. LONG write-read access.

MCKQBackColorOn

Description

Defines or returns the background color for flash status "On" in the case of the "Departed Acknowledged" status. LONG write-read access.

MCKQBackFlash

Description

TRUE, when the background should flash when a message departs acknowledged. BOOLEAN write-read access.

MCKQTextColorOff

Description

Defines or returns the color of the text for flash status "Off" for the "Outgoing acknowledged" status. LONG write-read access.

MCKQTextColorOn

Description

Defines or returns the background color of the text for flash status "On" for the "Outgoing acknowledged" status. LONG write-read access.

MCKQTextFlash

Description

TRUE, when the font should flash when a message departs acknowledged. BOOLEAN write-read access.

MCText

Description

Defines or returns the label for the respective message class.

MessageClass

Description

Defines the respective limit violation (Alarm High, Alarm Low, Warning High, Warning Low, etc.) for which the "Display Text", "Incoming -", "Incoming acknowledged -", and "Outgoing unacknowledged -" settings have been configured.

NumberLines

Description

Defines or return the number of lines the text list object should contain. If the amount of configured text is larger than this value, the selection list receives a vertical scroll bar.

ObjectName

Description

Returns the name of the specified object.

Access in Runtime: Read

Syntax

Object.ObjectName

Object

Required A "ScreenItem" object. This property is a standard property of the ScreenItem object and is therefore available for all formats.

See also

Line (Page 5102)

Polyline (Page 5137)

Ellipse (Page 5065)

Circle (Page 5048)

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

EllipticalArc (Page 5070)

CircularArc (Page 5053)

Rectangle (Page 5149)

Polygon (Page 5134)

TextField (Page 5184)
IOField (Page 5098)
SymbolicIOField (Page 5170)
Button (Page 5040)
Switch (Page 5166)
GraphicView (Page 5091)
GraphicIOField (Page 5088)
Bar (Page 5033)
Clock (Page 5056)
Gauge (Page 5084)
Slider (Page 5159)
SymbolLibrary (Page 5175)
OnlineTrendControl (Page 5122)
FunctionTrendControl (Page 5077)
OnlineTableControl (Page 5114)
AlarmControl (Page 5018)
HTMLBrowser (Page 5096)
CheckBox (Page 5045)
OptionGroup (Page 5131)
WindowSlider (Page 5216)
Connector (Page 5058)
ScreenWindow (Page 5155)
DiskSpaceView (Page 5063)
ChannelDiagnose (Page 5044)
ScriptDiagnostics (Page 5157)
Group (Page 5094)

OnTop

Description

TRUE, when the object should remain in the foreground in Runtime. Read only access.

OperationMessage property

Description

TRUE, if an alarm should be output when an operation is performed. BOOLEAN write-read access.
The operation is sent to the alarm system and is logged. The alarm system can be used to output an alarm in an alarm line, for example.

Particularities for I/O field, text list, and slider

The reason for the operation can only be entered if the "OperationReport" property has been set to TRUE.

OperationReport property

Description

TRUE, if the reason for an operation should be recorded. BOOLEAN write/read access.
When the object is used or operated in Runtime, a dialog opens in which the operator can input the reason for the operation in the form of text. The operation is sent to the message system, and is archived.

Orientation

Description

Specifies the text direction (orientation) of the selected object.

Access in Runtime: Read and write

Syntax

Object.**Orientation**[=TextOrientation]

Object

Required An object of the "ScreenItem" type with the format "TextField", "IOField", "SymbolicIOField", "Button", "Switch", "Bar", "OnlineTrendControl", "FunctionTrendControl", "Checkbox", "OptionGroup" or "WindowSlider".

TextOrientation

hmiTextHorizontal (0): The text is shown horizontally.

hmiTextRotated90Degree (-1): The text is shown vertically and left justified.

hmiTextRotated270Degree (1): The text is shown vertically and right justified.

See also

TextField (Page 5184)

IOField (Page 5098)

SymbolicIOField (Page 5170)

Button (Page 5040)

Switch (Page 5166)

Bar (Page 5033)

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

CheckBox (Page 5045)

OptionGroup (Page 5131)

WindowSlider (Page 5216)

OutputFormat

Description

Returns the value for the representation of the output value. The representation is dependent on the data format. Read only access.

OutputValue

Description

Determines the default setting for the value to be displayed or returns it.
This value is used in Runtime when the associated tag cannot be connected or updated when a picture is started.

PersistentRTAuthorization

Description

Specifies the authorization for operators which is necessary to change the persistence setting in Runtime. The value to be entered corresponds to the number which has the desired authorization in the user administration. Whether the information is evaluated depends on the value of the property "AllowPersistance". This does not apply for the message view.

Access in Runtime: Read and write

Syntax

Object.**PersistentRTAuthorization**[=RtAuthorization]

Object

Required An object of the "ScreenItem" type with the format "AlarmControl", "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

RtAuthorization

Optional A value or a constant that specifies the authorization for operators to change the persistence setting.

See also

AlarmControl (Page 5018)

PersistentRuntime

Description

Specifies whether changed settings in the window will also be maintained after a screen change.

Access in Runtime: Read and write

Syntax

Object.**PersistentRuntime**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl" oder "OnlineTableControl".

BOOLEAN

Optional. TRUE, if changed settings in the window will also be maintained after a screen change.

Comments

The "AllowPersistance" property specifies whether the information will be evaluated.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

PersistentRuntimeAuthorization

Description

Specifies the access authorization to change the persistence setting in Runtime. The value to be entered corresponds to the number which has the desired authorization in the user administration.

Access in Runtime: Read and write

Syntax

Object.**PersistentRuntimeAuthorization**[=HmiRTAuthorization]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

HmiRTAuthorization

Optional A value or a constant that specifies the access authorization required to change the persistence setting in Runtime.

Comments

The "AllowPersistance" property specifies whether the information will be evaluated. This does not apply for the message view.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

PicDeactReferenced

Description

TRUE, when the picture assigned for the "Disable" status should be saved in the RoundButton object. Otherwise, only the associated object reference is saved. Read only access.

PicDeactTransparent

Description

Defines or returns which color of the bitmap object (.bmp, .dib) assigned to the "Disabled" status should be set to "transparent". LONG Write/Read Access.
The color is only set to "Transparent" if the value of the "PicDeactUseTransColor" property is "True".

PicDeactUseTransColor

Description

TRUE, when the transparent color defined by the "PicDeactTransparent" property for the "Disable" status should be used. BOOLEAN write-read access.

PicDownTransparent

Description

Defines or returns which color of the bitmap object (.bmp, .dib) assigned to the "On" status should be set to "transparent". LONG Write/Read Access.
The color is only set to "Transparent" if the value of the "PicDownUseTransColor" property is "True".

PicDownUseTransColor

Description

TRUE, when the transparent color defined by the "PicDownTransparent" property for the "On" status should be used. BOOLEAN write-read access.

PicTransColor

Description

Defines or returns which color of the assigned bitmap object (.bmp, .dib) should be set to "transparent". LONG Write/Read Access.
The color is only set to "Transparent" if the value of the "PicUseTransColor" property is "True".

PictureName

Description

Defines the path and file name of the background image in the process picture or returns it. LONG write-read access.

PictureUp

Description

Defines the picture to be displayed in the "Off" status or returns the picture name.
The picture (*.BMP or *.DIB) must be located in the "GraCS" folder of the current project so that it can be integrated.

PicUpTransparent

Description

Defines or returns which color of the bitmap object (.bmp, .dib) assigned to the "Off" status should be set to "transparent". LONG Write/Read Access.
The color is only set to "Transparent" if the value of the "PicUpUseTransColor" property is "True".

PicUpUseTransColor

Description

TRUE, when the transparent color defined by the "PicUpTransparent" property for "Off" status should be used. BOOLEAN write-read access.

PicUseTransColor

Description

TRUE, when the transparent color defined by the "PicDeactTransparent" property for the "Disable" status should be used. BOOLEAN write-read access.

Pinned

Description

Defines whether the display of the picture window in Runtime depends on the process picture in which the picture window was configured.

PointCount

Description

Defines or returns the number of corner points. Each corner point has position coordinates and is identified via an index.

PrintJobName

Description

Defines the print job triggered by the print function of the "Print" toolbar button. The recommended print job is set for the control by default.

Open the "Select Print Job" dialog using the selection button.

Syntax

Object.**PrintJobName**]

Object

Required An object of the "ScreenItem" type with the format "AlarmControl", "FunctionTrendControl", "OnlineTableControl", "OnlineTrendControl", "TrendRulerControl", "UserArchiveControl".

See also

AlarmControl (Page 5018)

PrintVisibleColumnsOnly

Description

Specifies whether only the current visible columns should be output in quick print.

Access in Runtime: Read and write

Syntax

Object.**PrintVisibleColumnsOnly**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "UserArchiveControl".

BOOLEAN

Optional TRUE, if only the current visible columns should be output in quick print.

Process

Description

Defines or returns presetting for the value to be displayed.
This value is used in Runtime when the associated tag cannot be connected or updated when a picture is started.

ReadOnly

Description

Specifies whether the Control has read and write access in Runtime.

Access in Runtime: Read and write

Syntax

Object.**ReadOnly**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "UserArchiveControl".

BOOLEAN

Optional TRUE, if the Control has read and write access in Runtime.

ReferenceRotationLeft

Description

Defines or returns the X-coordinate of the reference point about which the object should be rotated in Runtime.
The value of the x coordinate is relative to the object width. Enter the value in percent starting from the left edge of the rectangle enclosing the object.

ReferenceRotationTop

Description

Defines or returns the Y-coordinate of the reference point about which the object should be rotated in Runtime.

The value of the Y-coordinate is relative to the object height. Enter the value in percent starting from the top edge of the rectangle enclosing the object.

Relevant property

Description

TRUE, when the object will be taken into account when forming the group display. BOOLEAN write-read access.

RightComma property

Description

Defines or returns the number of decimal places (0 to 20).

RTPersistencePasswordLevel

Description

Displays the authorization for online configuration. You can edit the authorization using the selection button. Authorizations are configured in the user administration.

Syntax

Object.**RTPersistencePasswordLevel**

Object

Required. An object of the "ScreenItem" type with the formats "AlarmControl", "OnlineTrendControl", "TrendRulerControl", "UserArchiveControl" or OnlineTableControl" ("Runtime Professional) or "HMITrendView" (Runtime Advanced).

See also

AlarmControl (Page 5018)

UserArchiveControl (Page 5207)

SameSize

Description

TRUE, when all four buttons of a Group Display object have the same size. BOOLEAN write-read access.

ScaleTicks

Description

Defines the number of segments into which the bar will be subdivided by large tick marks of the scale:
0-100: Object can be divided into a maximum of 100 segments
= 0: The optimum number of segments is set automatically.

ScrollBars

Description

TRUE, when the object is equipped with a scroll bar in Runtime. Read only access.

ScrollPositionX

Description

Specifies the horizontal positioning of the scroll bar in a picture window with slider, or returns its value.

The picture is displayed in the picture window by positioning the horizontal and vertical scroll bars. If you wish to display the picture as a cutout where the scroll bars are located at the left and upper edge of the picture, use the properties "OffsetLeft" and "OffsetTop" as the origin of this cutout.

ScrollPositionY

Description

Specifies the vertical positioning of the scroll bar in a picture window with slider, or returns its value.

The picture is displayed in the picture window by positioning the horizontal and vertical scroll bars. If you wish to display the picture as a cutout where the scroll bars are located at the left and upper edge of the picture, use the properties "OffsetLeft" and "OffsetTop" as the origin of this cutout.

SelBGColor

Description

Defines or returns the background color of the selected entry in a text list object. LONG write-read access.

SelectionMode

Description

Defines whether and how a message line can be selected.

.

Access in Runtime: Read and write

Syntax

Object.**SelectionMode**[=Mode]

Object

Required A "ScreenItem" object with the format "MessageView".

Mode

0 - NoSelection: Prevents the selection of a message. Acknowledgement affects the oldest pending message.

1 - Cell: Enables the selection of fields in the message line. Acknowledgement affects the selected message.

2 - Line: Enables the selection of a message line. Acknowledgement affects the selected message.

SelectionMode

Description

Specifies whether and how the alarm line can be selected.

Access in Runtime: Read and write

Syntax

Object.**SelectionMode**[=AlarmViewAdvancedSelectionMode]

Object

Required An object of the "ScreenItem" type with the format "AlarmControl".

AlarmViewAdvancedSelectionMode

hmiAlarmViewAdvancedSelectionModeNone (0): Prevents the selection of an alarm. An acknowledgement always affects the oldest pending alarm.

hmiAlarmViewAdvancedSelectionModeCell (1): Enables the selection of fields in the alarm line. An acknowledgement always effects the selected alarm.

hmiAlarmViewAdvancedSelectionModeLine (2): Enables the selection of an alarm line. An acknowledgement always effects the selected alarm.

See also

AlarmControl (Page 5018)

SelTextColor

Description

Defines or returns the color of the text of the selected entry in the text list object. LONG write-read access.

ShortenCellText

Description

Specifies whether the content of the fields of an alarm line should be shortened if the column width is too small.

Access in Runtime: Read and write

Syntax

Object.**ShortenCellText**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "AlarmControl".

BOOLEAN

Optional TRUE, if the content of the fields of an alarm line should be shortened if the column width is too small.

See also

AlarmControl (Page 5018)

ShortenColumnHeaderText

Description

Specifies whether the content of the fields of a title line should be shortened if the column width is too small.

Access in Runtime: Read and write

Syntax

Object.**ShortenColumnHeaderText**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "AlarmControl".

BOOLEAN

Optional TRUE, if the content of the fields of a title line should be shortened if the column width is too small.

See also

AlarmControl (Page 5018)

ShowVerticalGridlines

Description

Specifies whether the columns of the message view are separated by vertical lines.

Access in Runtime: Read and write

Syntax

Object.**ShowVerticalGridlines**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "AlarmControl".

BOOLEAN

Optional TRUE, if the columns of the alarm window will be separated by vertical separation lines.

See also

AlarmControl (Page 5018)

ShowXValuesExponential(i)

Description

Specifies whether a value of the X coordinate is shown in exponential format. Whether the information is evaluated depends on the property "XAxisMode(i)".".

Access in Runtime: Read and write

Syntax

Object.**ShowXValuesExponential**(i)[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "FunctionTrendControl".

BOOLEAN

Optional TRUE, if a value of the X coordinate is shown in exponential format.

Comments

The parameter i indicates the number of the trend.

You use the function "Display value here" to view the X coordinate of the measurement value.

See also

FunctionTrendControl (Page 5077)

ShowYValuesExponential(i)

Description

Specifies whether a value of the Y coordinate is shown in exponential format. Whether the information is evaluated depends on the property "XAxisMode(i)". ".

Access in Runtime: Read and write

Syntax

Object.**ShowYValuesExponential**(i)[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "FunctionTrendControl".

BOOLEAN

Optional TRUE, if a value of the Y coordinate is shown in exponential format.

Comments

The parameter i indicates the number of the trend.

You use the function "Display value here" to view the Y coordinate of the measurement value.

See also

FunctionTrendControl (Page 5077)

SignificantMask property

Description

Is required in Runtime to display the active message class with the highest priority.
The value of the SignificantMask property represents an internal system output value does not require any specific configuration by the user. Updating is initiated in Runtime by clicking on the object.

SortByTimeEnable

Description

Specifies whether the sorting direction can be altered in Runtime.

Access in Runtime: Read and write

Syntax

Object.**SortByTimeEnable**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "AlarmControl".

BOOLEAN

Optional TRUE, if the sorting direction can be altered in Runtime.

See also

AlarmView (Page 5029)

SortOnColumnHeaderClick

Description

Specifies whether the alarm text blocks can be sorted via the column header of the alarm text block.

Access in Runtime: Read and write

Syntax

Object.**SortOnColumnHeaderClick**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "AlarmControl".

BOOLEAN

Optional TRUE, if the message blocks can be sorted by the column header of the message blocks.

See also

AlarmControl (Page 5018)

SortTimeAscending

Description

Determines whether the last message received is displayed on top (ascending sorting order) in the "MessageView" object.

Access during runtime: Read and Write

Syntax

Object.SortTimeAscending [= BOOLEAN]

Object

Required. An object of the "MessageView" type.

BOOLEAN

Optional. TRUE, if the last message received is displayed on top.

SortTimeEnable

Description

Determines whether the sorting of messages can be changed according to time in the "MessageView" object.

Access during runtime: Read and Write

Syntax

Object.SortTimeEnable [= BOOLEAN]

Object

Required. An object of the "MessageView" type.

BOOLEAN

Optional. TRUE, if the sorting can be changed by the operator on the device.

StartPointLeft

Description

Specifies the horizontal distance in pixels from the left edge of the screen.

Access in Runtime: Read and write

Syntax

Object.**StartPointLeft**[=Int]

Object

Required An object of the "ScreenItem type with the format "Line"".

Int

Optional A value that specifies the horizontal distance in pixels of the start point from the left edge of the screen.

See also

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

EllipticalArc (Page 5070)

CircularArc (Page 5053)

TagName

Description

The "Index" property references a trend. "TagName" defines the tag linked to this trend. It is specified in the form "Archivname\Variablenname" to display tags in a process value archive or "TasgName" to display an internal or external tag which is not stored in an archive.

TimeAxisShowGridLines(i)

Description

Specifies whether the trend window is shown with grid lines that run parallel to the x axis.

Access in Runtime: Read and write

Syntax

Object.**TimeAxisShowGridLines**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTrend".

BOOLEAN

Optional TRUE, if the trend window is shown with grid lines that run parallel to the x axis.

Comments

The distance between two grid lines can be altered with the "TimeAxisGridLineInterval(i)" property.

See also

OnlineTrendControl (Page 5122)

TimeColumnFormat(i)

Description

Specifies the format of the time columns for the selected column pair. The parameter i indicates the number of the column pair.

Access in Runtime: Read and write

Syntax

Object.**TimeColumnFormat**(i)[=TimeFormat]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

TimeFormat

(0): The information is carried out in the form hh:mm.

(1): The information is carried out in the form hh:mm:ss.

(2): The information is carried out in the form hh:mm:ss.ms.

(3): The information is carried out in the form hh:mm (full hours).

(4): The information is carried out in the form hh:mm:ss (full minutes).

(5): The information is carried out in the form hh:mm:ss:ms (full seconds).

(6): The information is carried out in the form dd_mm_yy_hh_mm.

(7): The information is carried out in the form dd_mm_yy_hh_mm (full hours).

(8): The information is carried out in the form dd_mm_yy.

(9): The information is carried out in the form dd_mm_yy_ (full days).

See also

OnlineTableControl (Page 5114)

TitleBackColor

Description

Specifies the background color of the table headers for the selected status. You open a color selection dialog box with the button.

Titleline

Description

TRUE, when the control has a title bar and it can be moved in Runtime. BOOLEAN write-read access.

ToleranceHigh property

Description

Defines or returns the limit value for "Tolerance high".
The type of the evaluation (in percent or absolute) is defined in the "TypeToleranceHigh" property.
The monitoring of the limit value is only valid if the "CheckToleranceHigh" property is set to "TRUE".

ToleranceLow property

Description

Defines or returns the limit value for "Tolerance low".
The type of the evaluation (in percent or absolute) is defined in the "TypeToleranceLow" property.
The monitoring of the limit value is only valid if the "CheckToleranceLow" property is set to "TRUE".

TopConnectedConnectionPointIndex

Description

Specifies or sets the index number of the top connecting point.

LONG write-read access.

TopConnectedObjectName

Description

Specifies or sets the object name of the object which is docked on at the bottom connecting point.

LONG write-read access.

Trend

Description

TRUE, when the tendency (rising or falling) of the measuring value being monitored should be displayed by a small arrow. BOOLEAN write-read access.

TrendTag

Description

Specifies which tag is linked with the selected trend.

You enter the value either in the form "archive name/tag name", to show tags of a process value archive, or in the form "tag name" to show an internal or an external tag that will now be saved in an archive.

Access in Runtime: Read and write

Syntax

Object.**TrendTag**[=HmiTag]

Object

Required A ""ScreenItem" object with the format "OnlineTrend".

HmiTag

Optional A value or a constant that specifies which tags are linked with the selected trend.

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

TypeAlarmHigh

Description

TRUE, when the upper limit value, at which an alarm is triggered, should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

TypeAlarmLow

Description

TRUE, when the lower limit value, at which an alarm is triggered, should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

TypeLimitHigh4

Description

TRUE, when the "Reserve 4" upper limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

TypeLimitHigh5

Description

TRUE, when the "Reserve 5" upper limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

TypeLimitLow4

Description

TRUE, when the "Reserve 4" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

TypeLimitLow5

Description

TRUE, when the "Reserve 5" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

TypeToleranceHigh

Description

TRUE, when the "Tolerance high" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

TypeToleranceLow

Description

TRUE, when the "Tolerance low" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

TypeWarningHigh

Description

TRUE, when the "Warning high" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

TypeWarningLow

Description

TRUE, when the "Warning low" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

UncertainStateColor(i)

Description

Specifies the color for the identification of values with an uncertain status.

Access in Runtime: Read and write

Syntax

Object.**UncertainStateColor**(i)[=Color]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

Color

Optional A value or a constant that specifies the color for the identification of values with an uncertain status.

Comments

The parameter i indicates the number of the trend.

If the start value of a value is not known after Runtime has been activated or a substitute value is used, then this value has an uncertain status.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

UncertainStateEnabled(i)

Description

Specifies if values with an uncertain status are identified with the color set up in "ReplacementColor(i)"".

If the start value of a value is not known after Runtime has been activated or a substitute value is being used, then this value is considered to have an uncertain status.

Access in Runtime: Read and write

Syntax

Object.**UncertainStateEnabled**(i)[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

BOOLEAN

Optional TRUE, if values with an uncertain status are identified with the color specified in ""ReplacementColor(i".

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

UpdateEnable

Description

Specifies whether the recipe view can be altered in Runtime.

Access in Runtime: Read and write

Syntax

Object.**UpdateEnable**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "UserArchiveControl".

BOOLEAN

Optional TRUE, if the recipe view can be altered in Runtime.

UpperLimitColor(i)

Description

Specifies the color that identifies the tag values that lie above the value of ""UpperLimitValue(i)". Whether the information is evaluated depends on the value of the property ""UpperLimitEnabled(i)".

Access in Runtime: Read and write

Syntax

Object.**UpperLimitColor**(i)[=Color]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

Color

Optional A value or a constant that identifies the tag values that lie above the value of ""UpperLimitValue(i)".

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

UpperLimitEnabled(i)

Description

Specifies whether the information from "UpperLimitColor(i)" is used to identify the tag values that lie above the value of "UpperLimitValue(i)".

Access in Runtime: Read and write

Syntax

Object.**UpperLimitEnabled**(i)[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

BOOLEAN

Optional TRUE, if the information from "UpperLimitColor(i)"" is used to identify the tag values that lie above the value of "UpperLimitValue(i)".

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

UpperLimitValue(i)

Description

Specifies whether the tag values that exceed the "UpperLimitValue(i)" are to be identified with the color specified in "UpperLimitColor(i)"". Whether the information is evaluated depends on the value of the property "UpperLimit(i)".".

Access in Runtime: Read and write

Syntax

Object.**UpperLimitValue**(i)[=Double]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

Double

Optional A value or a constant that specifies whether the tag values that exceed the ""UpperLimitValue(i)" value are to be identified with the color specified in UpperLimitColor(i)"".

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

UseAllServers

Description

Specifies whether the data to be shown in the alarm window can be requested from all servers in a distributed system. Alarm Logging must be activated on the respective servers.

Access in Runtime: Read-only

Syntax

Object.**UseAllServers**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "AlarmView".

BOOLEAN

Optional. TRUE, if the data to be shown in the alarm window can be requested from all servers in a distributed system.

See also

AlarmControl (Page 5018)

UserValue1 property

Description

Defines or returns a value.
The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.

UserValue2 property

Description

Defines or returns a value.
The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.

UserValue3 property

Description

Defines or returns a value.
The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.

UserValue4 property

Description

Defines or returns a value.
The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.

ValueAxisDecimalPrecision(i)

Description

Specifies the number of decimal places for the value range of the selected trend.

Access in Runtime: Read and write

Syntax

Object.**ValueAxisDecimalPrecision**(i)[=Int]

Object

Required A "ScreenItem" object with the formats "OnlineTrend" or "FunctionTrendView".

Int

Optional A value or a constant that specifies the number of decimal places for the value range of the selected trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ValueAxisShowGridLines(i)

Description

Specifies whether the trend window is shown with grid lines that run parallel to the y axis.

Access in Runtime: Read and write

Syntax

Object.**ValueAxisShowGridLines(**i)[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

BOOLEAN

Optional TRUE, if the trend window is shown with grid lines that run parallel to the y axis.

Comments

The distance between two grid lines can be altered with the "ValueAxisGridLineInterval(i)" property.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ValueColumnPrecision(i)

Description

Specifies the number of decimal places that are shown in this value column. The parameter i indicates the number of the column pair. A maximum of 16 decimal places can be shown.

Access in Runtime: Read and write

Syntax

Object.**ValueColumnPrecision**(i)[=Int]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

Int

Optional A value or a constant that specifies the number of decimal places that are shown in this value column.

See also

OnlineTableControl (Page 5114)

WarningHigh

Description

Defines or returns the upper limit value for "Warning High".
In order that the limit value is monitored, the "CheckWarningHigh" property must be set to TRUE.
The display on reaching the limit value and the type of evaluation are defined by means of the "ColorWarningHigh" and "TypeWarningHigh" properties.

WarningLow

Description

Defines or returns the lower limit value for "Warning Low".
In order that the limit value is monitored, the "CheckWarningLow" property must be set to TRUE.
The display on reaching the limit value and the type of evaluation are defined by means of the "ColorWarningLow" and "TypeWarningLow" properties.

StyleSettings

Description

Defines the style in which the object is displayed.

Access in runtime: Read and write

Syntax

Object.**StyleSettings**[=WinCCStyle]

Object

Required. An object of the "ScreenItem" type with the following format:

- Button
- RoundButton
- WindowsSlider

WinCCStyle

Optional. A value or a constant that specifies the style in which the object is displayed.

User Defined	Shows the object according to the respective settings.
Global	Shows the object in a globally defined design.
Windows Style	Shows the object in Windows style.

WindowBorder

Description

TRUE, when the window is displayed with borders in Runtime. Read only access.

XAxisBegin(i)

Description

Specifies the lower limit of the X axis of a trend that has been referenced with the property "CurrentCurveIndex". Whether the information is evaluated depends on the properties "XAxisAutorange(i)" and "ShareXAxis".

Access in Runtime: Read and write

Syntax

Object.**XAxisBegin**(i)[=String]

Object

Required A ""ScreenItem" object with the format ""FunctionTrendView".

String

Optional A value or a constant that specifies the lower limit of the X axis of a trend that has been referenced with the property "CurrentCurveIndex".

See also

FunctionTrendControl (Page 5077)

XAxisEnd(i)

Description

Specifies the lower limit of the X axis of a trend that has been referenced with the property "CurrentCurveIndex". Whether the information is evaluated depends on the properties "XAxisAutorange(i)"" and "ShareXAxis".

Access in Runtime: Read and write

Syntax

Object.**XAxisEnd**(i)[=String]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".

String

Optional A value or a constant that specifies the upper limit of the X axis of a trend that has been referenced with the property "CurrentCurveIndex".

See also

FunctionTrendControl (Page 5077)

XAxisShowGridLines(i)

Description

Specifies whether the trend window is shown with grid lines that run parallel to the X axis. The distance between two grid lines can be altered with the ""XAxisGridLineInterval(i)" property.

Access in Runtime: Read and write

Syntax

Object.**XAxisShowGridLines**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".

BOOLEAN

Optional TRUE, if the trend window is shown with grid lines that run parallel to the X axis.

See also

FunctionTrendControl (Page 5077)

2

Activate

Description

Determines whether the data to be displayed are requested by the alarm server.

Access in Runtime: Read and write

Syntax

Object.**Activate**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "AlarmControl".

BOOLEAN

Optional TRUE, if the data to be displayed are requested by the alarm server.

See also

AlarmControl (Page 5018)

AdjustBorder3DWithStyle

Description

Specifies whether the 3D border width will be displayed in the general Windows style.

Access in Runtime: Read and write

Syntax

Object.**AdjustBorder3DWithStyle**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Button".

BOOLEAN

Optional TRUE, if the 3D border width will be displayed in the general Windows style.

See also

Button (Page 5040)

AlarmFilter

Description

Specifies a SQL-Statement for selecting the alarms that will be shown in the alarm window.

Alarm list and hit list filtering and sorting settings

Access in Runtime: Read and write

Syntax

Object.**AlarmFilter**[=String]

Object

Required A "ScreenItem" object with the format "AlarmControl".

String

Optional A value or a constant that specifies a SQL-Statement for selecting the alarms that will be shown in the alarm window.

See also

AlarmControl (Page 5018)

AngleAlpha

Description

Defines or returns depth angle a for the 3D-effect of the "3DBarGraph" object. Value range in degrees from 0 to 90.

AngleBeta

Description

Defines or returns depth angle b for the 3D-effect of the "3DBarGraph" object. Value range in degrees from 0 to 90.

Axe property

Description

Defines or returns the position of the 3D bar in the coordinate system. Value range from 0 to 2.

0: The 3D-bar is displayed on the X-axis.

1: The 3D-bar is displayed on the Y-axis.

2: The 3D-bar is displayed on the Z-axis.

AxisSection

Description

Defines or returns the distance between two long axis sections. The information on the distance is given in scale units and is dependent on the minimum and maximum values configured.

Background

Description

TRUE, when the background of the 3D-bar graph object should be visible. BOOLEAN write-read access.

BarDepth

Description

Defines or returns the depth of the bar in pixels.

BarHeight property

Description

Defines or returns the height of the bar in pixels.

BaseX property

Description

Defines or returns the horizontal distance of the right bar edge to the left edge of the object field in pixels.

BaseY property

Description

Defines or returns the vertical distance of the bottom bar edge to the top edge of the objet field.

CloseButton

Description

TRUE, when the window is provided with a "Close" button. Read only access.

ColumnSizingEnable

Description

Specifies whether the width of the columns in the alarm window can be changed. The width of the columns can be changed only if the "AutoScroll" property is not enabled.

Access in Runtime: Read and write

Syntax

Object.**ColumnSizingEnable**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "AlarmControl".

BOOLEAN

Optional TRUE if the width of the columns in the alarm window can be changed.

See also

AlarmControl (Page 5018)

ForeFlashColorOff

Description

Defines or returns the color of the text for flash status "Off". LONG write-read access.

ForeFlashColorOn

Description

Defines or returns the color of the text for flash status "On". LONG write-read access.

InsertData(i)

Description

Specifies which data will be inserted for the active trend.

Access in Runtime: Read and write

Syntax

Object.**InsertData**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "FunctionTrendControi".

BOOLEAN

Optional TRUE, if "DataIndex(i)" is ignored and the data added to the rear of the data buffer.

FALSE, if the data is inserted at the position "DataIndex(i)" of the data buffer.

Comments

The trend view is redrawn in every operation with "InsertData(i)".

See also

FunctionTrendControl (Page 5077)

Layer00Checked

Description

TRUE, when limit 0 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer00Value and Layer00Color properties.

Layer00Color

Description

Defines or returns the color for limit 0. LONG write/read access.
When monitoring of the limit value is activated (Layer00Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer00Value

Description

Determines the value for "Limit 0" or returns it.
Monitoring only takes effect when the Layer00Checked property value is set to TRUE.

Layer01Checked

Description

TRUE, when limit 1 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer01Value and Layer01Color properties.

Layer01Color

Description

Defines or returns the color for limit 1. LONG write/read access.
When monitoring of the limit value is activated (Layer01Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer01Value

Description

Determines the value for "Limit 1" or returns it.
Monitoring only takes effect when the Layer01Checked property value is set to TRUE.

Layer02Checked

Description

TRUE, when limit 2 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer02Value and Layer02Color properties.

Layer02Color

Description

Defines or returns the color for limit 2. LONG write/read access.
When monitoring of the limit value is activated (Layer02Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer02Value

Description

Determines the value for "Limit 2" or returns it.
Monitoring only takes effect when the Layer02Checked property value is set to TRUE.

Layer03Checked

Description

TRUE, when limit 3 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer03Value and Layer03Color properties.

Layer03Color

Description

Defines or returns the color for limit 3. LONG write/read access.
When monitoring of the limit value is activated (Layer03Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer03Value

Description

Determines the value for "Limit 3" or returns it.
Monitoring only takes effect when the Layer03Checked property value is set to TRUE.

Layer04Checked

Description

TRUE, when limit 4 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer04Value and Layer04Color properties.

Layer04Color

Description

Defines or returns the color for limit 4. LONG write/read access.
When monitoring of the limit value is activated (Layer04Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer04Value

Description

Determines the value for "Limit 4" or returns it.
Monitoring only takes effect when the Layer04Checked property value is set to TRUE.

Layer05Checked

Description

TRUE, when limit 5 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer05Value and Layer05Color properties.

Layer05Color

Description

Defines or returns the color for limit 5. LONG write/read access.
When monitoring of the limit value is activated (Layer05Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer05Value

Description

Determines the value for "Limit 5" or returns it.
Monitoring only takes effect when the Layer05Checked property value is set to TRUE.

Layer06Checked

Description

TRUE, when limit 6 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer06Value and Layer06Color properties.

Layer06Color

Description

Defines or returns the color for limit 6. LONG write/read access.
When monitoring of the limit value is activated (Layer06Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer06Value

Description

Determines the value for "Limit 6" or returns it.
Monitoring only takes effect when the Layer06Checked property value is set to TRUE.

Layer07Checked

Description

TRUE, when limit 7 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer07Value and Layer07Color properties.

Layer07Color

Description

Defines or returns the color for limit 7. LONG write/read access.
When monitoring of the limit value is activated (Layer07Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer07Value

Description

Determines the value for "Limit 7" or returns it.
Monitoring only takes effect when the Layer07Checked property value is set to TRUE.

Layer08Checked

Description

TRUE, when limit 8 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer08Value and Layer08Color properties.

Layer08Color

Description

Defines or returns the color for limit 8. LONG write/read access.
When monitoring of the limit value is activated (Layer08Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer08Value

Description

Determines the value for "Limit 8" or returns it.
Monitoring only takes effect when the Layer08Checked property value is set to TRUE.

Layer09Checked property

Description

TRUE, when limit 9 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer09Value and Layer09Color properties.

Layer09Color

Description

Defines or returns the color for limit 9. LONG write/read access.
When monitoring of the limit value is activated (Layer09Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer09Value

Description

Determines the value for "Limit 9" or returns it.
Monitoring only takes effect when the Layer09Checked property value is set to TRUE.

Layer10Checked

Description

TRUE, when limit 10 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer10Value and Layer10Color properties.

Layer10Color

Description

Defines or returns the color for limit 10. LONG write/read access.
When monitoring of the limit value is activated (Layer10Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

Layer10Value

Description

Determines the value for "Limit 10" or returns it.
Monitoring only takes effect when the Layer10Checked property value is set to TRUE.

LightEffect property

Description

TRUE, when the light effect should be activated. BOOLEAN write-read access.

ListType property

Description

Returns the data type displayed in the case of a text list object. Read only access.

Value range from 0 to 2.

0 = decimal

1 = binary

2 = bit

Localizable

Description

Specifies whether a font should be assigned and can be formatted for each Runtime language.

Access in Runtime: Read and write

Syntax

Object.**Localizable**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl", "OnlineTableControl", "AlarmControl" or "UserArchiveControl".

BOOLEAN

Optional TRUE, if a font should be assigned and can be formatted for each Runtime language.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

AlarmControl (Page 5018)

Max property

Description

Defines or returns the absolute value in the case of a full value display. This value is displayed if the scale display is active.

PersistentRTCSAuthorization

Description

Specifies the operator authorization to be able to change the persistence setting in Runtime. The value to be entered corresponds to the number which has the desired authorization in the user administration. Whether the information is evaluated depends on the value of the property "AllowPersistance". This does not apply for the message view.

Access in Runtime: Read and write

Syntax

Object.**PersistentRTCSAuthorization**[=RtAuthorization]

Object

Required An object of the "ScreenItem" type with the format "AlarmControl", "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

RtAuthorization

Optional A value or a constant that specifies the operator authorization to change the persistence setting.

See also

AlarmControl (Page 5018)

PersistentToDownload

Description

Specifies whether the settings will be maintained until the next ES download.

Access in Runtime: Read and write

Syntax

Object.**PersistentToDownload**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

BOOLEAN

Optional TRUE, if the settings will be maintained until the next ES download.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

PersistentToDownloadAuthorization

Description

Specifies the access authorization required to change the settings until the next ES download.

Access in Runtime: Read and write

Syntax

Object.**PersistentToDownloadAuthorization**[=HmiRTAuthorization]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

HmiRTAuthorization

Optional A value or a constant that specifies the access authorization required to change the settings until the next ES download.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

PicDownReferenced

Description

TRUE, when the picture assigned for the "On" status is to be saved. Otherwise, only the associated object reference is saved. Read only access.

PicReferenced

Description

TRUE, when the assigned picture is references the object and is not saved in it. Read only access.

Position

Description

Defines the presetting for the position of the slider.

This value is used as the start value in Runtime.

To operate the process value linked to this attribute, it is necessary that the process value is also linked to the "Position" event. You will find the event "Position" in the "Event" tab, in the topic tree under SliderCtrl\Property Topics\Control Properties\Value.

PredefinedAngles

Description

Defines or returns the depth of the display of the 3DBarGraph object. Value range from 0 to 3.

0 = cavalier

1 = isometric

2 = axionometric

3 = freely defined

PrintConfiguration

Description

Specifies the log for the printout.

Access in Runtime: Read and write

Syntax

Object.**PrintConfiguration**[=HMIPrintConfiguration]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl", "OnlineTableControl", "AlarmControl" or "UserArchiveControl".

HMIPrintConfiguration

Optional A value or a constant which specifies the log for the printout.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

AlarmControl (Page 5018)

ProjectPath

Description

Contains the path and name of the associated project.

Access in Runtime: Read-only

Syntax

Object.**ProjectPath**[=String]

Object

Required A "ScreenItem" object with the format "AlarmControl".

String

Optional A value or a constant that specifies which path and name the associated project will contain.

See also

AlarmControl (Page 5018)

RowSizingEnable

Description

Specifies whether the row height can be changed.

The "RowSizingEnable" property is then only disabled if both properties "RowSizingEnable" and "AdaptFontSizeToLineHeight" have the value FALSE.

Access in Runtime: Read and write

Syntax

Object.**RowSizingEnable**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "AlarmControl".

BOOLEAN

Optional TRUE, if the row height can be changed.

See also

AlarmControl (Page 5018)

Scrollable

Description

Specifies whether the DXF screen display supports the Scroll functions in Runtime mode.

Access in Runtime: Read and write

Syntax

Object.**Scrollable**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "DXFView".

BOOLEAN

Optional TRUE, if the DXF screen display supports the Scroll functions in Runtime mode.

ShowMessagesAtDate

Description

Determines at which time the alarms will be displayed.

Note

This function does not filter out alarms, it positions the display range of the alarm list as of the specified date.

Access during runtime: Read and Write

Syntax

Object.ShowMessagesAtDate [= DATE]

Object

Required. An object of the "MessageView" type.

Remarks

To assign SmartTags to the "ShowMessagesAtDate" property, you must formulate the assignment as follows:

```
'Example 1
HmiRuntime.Screens("Screen_1").ScreenItems("Alarm
view_1").ShowMessagesAtDate = SmartTags("intern_1").Value
'Example 2
HmiRuntime.Screens("Screen_1").ScreenItems("Alarm
view_1").ShowMessagesAtDate = CDate(SmartTags("intern_1"))
SmartTag "intern_1" must be of the "DateTime" type.
```

See also

AlarmView (Page 5029)

Sort

Description

Specifies the sorting criteria in SQL syntax for the database.

Access in Runtime: Read and write

Syntax

Object.**Sort**[=String]

Object

Required An object of the "ScreenItem" type with the format "UserArchiveControl".

String

Optional A value or a constant that specifies the sort criteria in SQL syntax for the database.

TableFocusOnButtonCommand

Description

Specifies whether by clicking on a Button in a script in Runtime the table of the Controls will be activated.

Access in Runtime: Read and write

Syntax

Object.**TableFocusOnButtonCommand**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "AlarmView".

BOOLEAN

Optional TRUE, if by clicking on a Button in a script in Runtime the table of the Controls will be activated.

See also

AlarmControl (Page 5018)

TimeRangeBase(i)

Description

Specifies the unit for determining the time interval of the selected trend.

Access in Runtime: Read and write

Syntax

Object.**TimeRangeBase**(i)[=TagLoggingTimeUnit]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

TagLoggingTimeUnit

(1): The time intervals are set up in milliseconds.

hmiCurveTimeRangeBase500ms (500): The time intervals are set up in 500 ms.

hmiCurveTimeRangeBaseSecond (1000): The time intervals are set up in seconds.

hmiCurveTimeRangeBaseMinute (60000): The time intervals are set up in minutes.

hmiCurveTimeRangeBaseHour (3600000): The time intervals are set up in hours.

hmiCurveTimeRangeBaseDay (86400000): The time intervals are set up in days.

Comments

The time range to be displayed for the trend i is achieved by multiplying the values "TimeRangeBase(i)" and "TimeRangeFactor(i)", whereby the value of "TimeRangeBase(i)" is interpreted in milliseconds. The "TimeRangeBase(i" and "TimeRangeFactor(i)" properties are only evaluated if the "TimeRange" property has the value "-1".

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

TimeRangeFactor(i)

Description

Specifies the factor for determining the time interval of the selected trend.

Access in Runtime: Read and write

Syntax

Object.**TimeRangeFactor**(i)[=Int]

Object

Required A "ScreenItem" object with the formats "OnlineTrend", "FunctionTrendView" or "TableView".

Int

Optional A value or a constant that specifies the factor for determining the time interval of the selected trend.

Comments

The time range to be displayed for the trend i is achieved by multiplying the values ""TimeRangeBase(i and "TimeRangeFactor(i)", whereby the value of "TimeRangeBase(i)" is interpreted in milliseconds. The "TimeRangeBase(i" and "TimeRangeFactor(i)" " properties are only evaluated if the property "TimeRange" has the value "-1".

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

UseTimeRange(i)

Description

Specifies how the time range that is to be shown for the selected trend will be defined.

Access in Runtime: Read and write

Syntax

Object.**UseTimeRange**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the formats "OnlineTrend","FunctionTrendView" or "TableView".

BOOLEAN

Optional TRUE, if the time range that is to be shown is defined by means of a start time ("TimeAxisBeginTime(i)") and an end time ("TimeAxisEndTime").

FALSE,, if the time range that is to be shown is defined by means of a start time ("TimeAxisBeginTime(i)") and an end time "TimeRangeBase(i)" and "TimeRangeFactor(i)".

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

ValidateFormatPattern

Description

Specifies that the "format pattern" field can be checked for specific characters.

Access in Runtime: Read and write

Syntax

Object.**ValidateFormatPattern**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "IOField".

BOOLEAN

Optional TRUE, if the "format pattern" field can be checked for specific characters.

See also

IOField (Page 5098)

ValueAxisBegin(i)

Description

Specifies the lower limit of the value range for the selected trend that is to be displayed. Whether the information is evaluated depends on the properties "Autorange" and "ShareValueAxis".

Access in Runtime: Read and write

Syntax

Object.**ValueAxisBegin**(i)[=Double]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

Double

Optional A value or a constant that specifies the lower limit of the value range of the selected trend that is to be shown.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ValueAxisEnd(i)

Description

Specifies the upper limit of the value range of the selected trend that is to be displayed. Whether the information is evaluated depends on the properties "Autorange" and "ShareValueAxis".

Access in Runtime: Read and write

Syntax

Object.**ValueAxisEnd**(i)[=Double]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

Double

Optional A value or a constant that specifies the upper limit of the value range of the selected trend that is to be shown.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ZeroPointValue

Description

Defines the value of the zero point of the scale indicator.

Defines or returns the absolute value for the zero point.

Zoomable

Description

Specifies whether the DXF screen display supports the zoom functions in Runtime mode.

Access in Runtime: Read and write

Syntax

Object.**Zoomable**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "DXFView".

BOOLEAN

Optional TRUE, if the DXF screen display supports the zoom functions in Runtime mode.

3

Active

Description

Specifies whether the data that is to be shown is requested by the log server.

Access in Runtime: Read and write

Syntax

Object.**Active**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl," or "OnlineTableControl".

BOOLEAN

Optional TRUE, if the data that is to be shown is requested by the log server.

Comments

The screen opening times are reduced if you do not set this attribute. You can change the value dynamically.

To differentiate between the "Activate" property and the "Activate" methods, the property is addressed via "Object".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

AdaptFontSizeToLineHeight

Description

Specifies whether the font size will be adapted to the line height.

Access in Runtime: Read and write

Syntax

Object.**AdaptFontSizeToLineHeight**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "AlarmControl".

BOOLEAN

Optional TRUE, if the font size will be adapted to the line height.

See also

AlarmControl (Page 5018)

AddAssignment

Description

Adds an entry.

AdjustRulerWindow

Description

Specifies whether the ruler window is adapted every time the trend view is displayed.

Access in Runtime: Read and write

Syntax

Object.**AdjustRulerWindow**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTrendControl".

BOOLEAN

Optional TRUE, if the ruler window is adapted every time the trend view is displayed.

Comments

TRUE, if the ruler window will be moved and then shown and hidden, it will be displayed at the original position in the original size.

See also

OnlineTrendControl (Page 5122)

AlarmHigh

Description

Defines the top limit value at which an alarm should be triggered or returned.
The type of evaluation (percentage or absolute) is specified with the "TypeAlarmHigh" property.

Access in Runtime: Write and read

Syntax

Object.AlarmHigh[=REAL]

Object

Required A "ScreenItem" object with the format "Bar".

REAL

Value for the upper limit

See also

Bar (Page 5033)

Alignment

Description

Determines the layout of the scale (left/right or top/bottom) depending on the position of the bar object or returns it.

Access in Runtime: Read and write

Syntax

Object.**Alignment**[=ScalePosition]

Object

Required An object of the "ScreenItem" type with the format "Bar".

ScalePosition

1: Right or bottom .

0: Left or top.

See also

Bar (Page 5033)

AlignmentLeft

Description

Defines or returns the horizontal alignment of the text. Value range from 0 to 2.

Access in Runtime: Read and write

Syntax

Object.**AlignmentLeft**[=HorizontalAlignment]

Object

Required An object of the "ScreenItem" type with the format "TextField", "IOField", "SymbolicIOField", "Button", "RoundButton", "CheckBox", "OptionGroup", "MultilineEdit", "ComboBox", "ListBox".

HorizontalAlignment

0: Alignment left

1: Alignment centered

2: Alignment right

AllowPersistence

Description

Specifies whether the persistence is adjustable.

Access in Runtime: Read and write

Syntax

Object.**AllowPersistence**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl", "FunctionTrendControl" or "OnlineTableControl".

BOOLEAN

Optional TRUE, if the persistence is adjustable.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

BackColor2

Description

Defines or returns the bar color for the display of the current value. LONG write-read access.

BackFillStyleReadOnlySpecial

Description

Specifies that the "Background fill style" can only be read.

Access in Runtime: Read and write

Syntax

Object.**BackFillStyleReadOnlySpecial**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "IOField".

BOOLEAN

Optional TRUE, if the "Background fill style" can only be read.

See also

IOField (Page 5098)

BarStartValue

Description

Specifies the value of the zero point of the scale.

Access in Runtime: Read and write

Syntax

Object.**BarStartValue**[=Double]

Object

Required A "ScreenItem" object with the format "Bar".

Double

Optional A value or a constant that specifies the value of the zero point for the scale.

See also

Bar (Page 5033)

BarWidth

Description

Defines or returns the width of the bar in pixels.

BeginTime(i)

Description

Specifies the start time of the time range to be shown in the trend view. The property "TagProviderType(i)" must have the value -1.

Access in Runtime: Read and write

Syntax

Object.**BeginTime**(i)[=Time]

Object

Required An object of the "ScreenItem" type with the format "FunctionTrendControl" oder "OnlineTableControl".

Time

Optional A value or a constant which determines the start time of the time range to be shown in the trend view.

Comments

The parameter i indicates the number of the trend.

If "BeginTime(i)" is changed, illegal combinations with different attributes for the data connection could arise. This means you have to prevent an immediate acceptance of the change with "FreezeProviderConnections" before changing "BeginTime(i)".

See also

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

BorderColorBottom

Description

Defines or returns the border color for the bottom/right part of the object. LONG write-read access.

BorderColorTop

Description

Defines or returns the border color for the top/left part of the object. LONG write-read access.

CentrePoint

Description

Specifies the position of the center point.

Access in Runtime: Read and write

Syntax

Object.**CentrePoint**[=Point]

Object

Required A "ScreenItem" object with the formats "EllipseSegment", "CircleSegment", "EllipticalArc" or "CircularArc".

Point

Optional A value that specifies the position of the center point.

See also

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

EllipticalArc (Page 5070)

CircularArc (Page 5053)

CentrePointLeft

Description

Specifies the horizontal distance in pixels of the center point from the left edge of the screen.

Access in Runtime: Read and write

Syntax

Object.**CentrePointLeft**[=Int]

Object

Required A "ScreenItem" object with the formats "EllipseSegment", "CircleSegment", "EllipticalArc" or "CircularArc".

Int

Optional A value that specifies the horizontal distance in pixels of the center point from the left edge of the screen.

See also

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

EllipticalArc (Page 5070)

CircularArc (Page 5053)

CentrePointTop

Description

Specifies the vertical distance in pixels of the center point from the upper edge of the screen.

Access in Runtime: Read and write

Syntax

Object.**CentrePointTop**[=Int]

Object

Required A "ScreenItem" object with the formats "EllipseSegment", "CircleSegment", "EllipticalArc" or "CircularArc".

Int

Optional A value that specifies the vertical distance in pixels of the center point from the upper edge of the screen.

See also

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

EllipticalArc (Page 5070)

CircularArc (Page 5053)

ColorBottom

Description

Defines or returns the color for the bottom/right stop of the slider object. LONG write-read access.

ColorChangeType

Description

TRUE, if the change of color should occur segment by segment in the case of a color change (e.g. on reaching a limit value). If set to FALSE, it defines the change of color for the entire bar. BOOLEAN write-read access.

ColorTop

Description

Defines or returns the color for the top/left stop of the slider object. LONG write-read access.

ColumnColor(i)

Description

Determines the color of the i column pair. The parameter i indicates the number of the column pair.

Access in Runtime: Read and write

Syntax

Object.**ColumnColor(i)**[=Color]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

Color

Optional A value or a constant that specifies the color of the i column pair.

See also

OnlineTableControl (Page 5114)

ColumnDisplayName(i)

Description

Specifies the name for the selected column pair. The parameter i indicates the number of the column pair.

Access in Runtime: Read and write

Syntax

Object.**ColumnDisplayName(i)**[=String]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

String

Optional A value or a constant that specifies the name for the selected column pair.

See also

OnlineTableControl (Page 5114)

ColumnUpdateEnabled(i)

Description

Specifies whether the selected column pair will be shown statically or dynamically.

Access in Runtime: Read and write

Syntax

Object.**ColumnUpdateEnabled(i)**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

BOOLEAN

Optional TRUE if the column pair is to be shown statically.

Comments

The "CurrentColumnIndex" property references the column pair that is currently active.

See also

OnlineTableControl (Page 5114)

Command

Description

Specifies whether there is a forced update of the values displayed in the control.

Access in Runtime: Read and write

Syntax

Object.**Command**[=String]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "OnlineTableControl".

String

Optional A value or a constant which specifies whether there is a forced update of the values displayed in the control.

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

Comment

Description

Reads or sets the Alarm object comment.

CommonTimeAxisColor

Description

Specifies the color of the mutual X axis.

Access in Runtime: Read and write

Syntax

Object.**CommonTimeAxisColor**[=Color]

Object

Required A "ScreenItem" object with the format "OnlineTrendControl".

Color

Optional A value or a constant that specifies the color of the common x axis.

See also

OnlineTrendControl (Page 5122)

ConfigureTimeAxis(i)

Description

Specifies whether a common time axis is used for all the trends in the trend view.

Access in Runtime: Read and write

Syntax

Object.**ConfigureTimeAxis(i)**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTrendControl".

BOOLEAN

Optional TRUE, if a common axis is used for all the trends in the trend view.

See also

OnlineTrendControl (Page 5122)

CurrentCurveIndex

Description

Specifies the index for the active trends.

Access in Runtime: Read and write

Syntax

Object.**CurrentCurveIndex**[=Int]

Object

Required An object of the "ScreenItem" type with the format "FunctionTrendControl" or "OnlineTrendControl".

Int

Optional A value or a constant that specifies the index of the trends that are currently active.

Comments

The "CurrentCurveIndex" property is evaluated by other properties so that their settings can be assigned to a specific trend. Valid values range from 0 to -1. The "CurvesCount" property contains the number of trends to be shown.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

CurveColor(i)

Description

Specifies the color of the selected trend.

Access in Runtime: Read and write

Syntax

Object.**CurveColor(i)**[=Color]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

Color

Optional A value or a constant that specifies the color of the selected trend.

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

CurveLineWidth(i)

Description

Specifies the line weight of the selected trend.

Access in Runtime: Read and write

Syntax

Object.**CurveLineWidth(i)**[=Int]

Object

Required A "ScreenItem" object with the format "OnlineTrendControl".

Int

Optional A value or a constant that specifies the line weight of the selected trend. Value range from 0 to 10.

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

CurvesCount

Description

Returns the number of configured trends or column pairs (visible and invisible) of the window.

Access in Runtime: Read and write

Syntax

Object.**CurvesCount**[=Int]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

Int

Optional A value or a constant that returns the number of configured trends or column pairs (visible and invisible) of the window.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

CurveUpdateEnabled(i)

Description

Specifies whether the selected trend will be shown statically or dynamically.

Access in Runtime: Read and write

Syntax

Object.**CurveUpdateEnabled(i)**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTrendControl".

BOOLEAN

Optional TRUE if the trend is to be shown statically.

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

DataIndex(i)

Description

Returns the current index for the data of the current trend.

Access in Runtime: Read and write

Syntax

Object.**DataIndex(i)**[=Int]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

Int

Optional A value or a constant that specifies the current index of the data for the current trend.

See also

FunctionTrendControl (Page 5077)

DataLogTag

Description

Specifies which tag is linked with the selected trend.

Access in Runtime: Read and write

Syntax

Object.**DataLogTag**[=HmiLoggingTag]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "OnlineTableControl".

HmiLoggingTag

Optional A value or a constant that specifies which tags are linked to the selected trend.

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

DataX(i)

Description

Inserts one single data record and must be set prior to the call of "InsertData(i)".

Access in Runtime: Read and write

Syntax

Object.**DataX(i)**[=object]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

object

Optional A value or a constant that inserts one single data record.

See also

FunctionTrendControl (Page 5077)

DataXY(i)

Description

Inserts several data records as Array with value pairs and must be set before calling "InsertData(i)".

The data in the Array are applied if "DataX(i)" is a VT_EMPTY type. Otherwise, the single value pair resulting from "DataX(i)" and "DataY(i)" will be used in the attribute "InsertData(i)".

Access in Runtime: Read and write

Syntax

Object.**DataXY**(i)[=object]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".

object

Optional A value or a constant which inserts several data records as Array with value pairs.

See also

FunctionTrendControl (Page 5077)

DataY(i)

Description

Inserts one single data record and must be set prior to the call of "InsertData(i)".

Access in Runtime: Read and write

Syntax

Object.**DataY**(i)[=Double]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

Double

Optional A value or a constant that inserts one single data record.

See also

FunctionTrendControl (Page 5077)

DeleteData(i)

Description

Specifies which data from the data buffer of the active trend will be deleted.

TRUE:

Access in Runtime: Read and write

Syntax

Object.**DeleteData**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

BOOLEAN

Optional TRUE, if all the data of the trend is to be deleted.

FALSE, if the value pair at position "DataIndex(i)" is to be deleted.

See also

FunctionTrendControl (Page 5077)

Direction

Description

Defines or returns the bar direction or the position of the slider object.

Access in Runtime: Read and write

Syntax

Object.**Direction** [=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "Bar" or "Slider".

BOOLEAN

0 = top

1 = bottom

2 = left

3 = right

DisplayName(i)

Description

Specifies the name for the selected trend.

Access in Runtime: Read and write

Syntax

Object.**DisplayName**(i)[=String]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

String

Optional A value or a constant that specifies the name for the selected trend.

Comments

The parameter i indicates the number of the trend.

See also

FunctionTrendControl (Page 5077)

DrawEnhancedHTMLBrowser

Description

Specifies the style of the HTML-Browsers.

Access in Runtime: Read and write

Syntax

Object.**DrawEnhancedHTMLBrowser**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "HTMLBrowser".

BOOLEAN

Optional TRUE, if an extended HTML-Browser will be displayed.

See also

HTMLBrowser (Page 5096)

Editable(i)

Description

Specifies whether a column pair can be edited. The parameter i indicates the number of the column pair.

Access in Runtime: Read and write

Syntax

Object.**Editable**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

BOOLEAN

Optional TRUE, if the column pair can be edited. The column pair is referenced by the parameter i.

See also

OnlineTableControl (Page 5114)

EditEnabled

Description

Specifies whether edit mode is enabled for a cell, if the property "Editable(i)" has the value TRUE.

Access in Runtime: Read and write

Syntax

Object.**EditEnabled**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

BOOLEAN

Optional TRUE, if the edit mode is enabled for a cell if the property "Editable(i)" has the value TRUE.

See also

OnlineTableControl (Page 5114)

EndPoint

Description

Specifies the position of the end point.

Access in Runtime:

Syntax

Object.**EndPoint**[=Point]

Object

Required A "ScreenItem" object with the formats "EllipseSegment", "CircleSegment", "EllipticalArc" or "CircularArc".

Point

Optional A value that specifies the position of the end point.

See also

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

EllipticalArc (Page 5070)

CircularArc (Page 5053)

EndPointLeft

Description

Specifies the horizontal distance in pixels of the end point from the left edge of the screen.

Access in Runtime: Read and write

Syntax

Object.**EndPointLeft**[=Int]

Object

Required A "ScreenItem" object with the formats "EllipseSegment", "CircleSegment", "EllipticalArc" or "CircularArc".

Int

Optional A value that specifies the horizontal distance in pixels of the end point from the left edge of the screen.

See also

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

EllipticalArc (Page 5070)

CircularArc (Page 5053)

EndPointTop

Description

Specifies the vertical distance in pixels of the end point from the upper edge of the screen.

Access in Runtime: Read and write

Syntax

Object.**EndPointTop**[=Int]

Object

Required A "ScreenItem" object with the formats "EllipseSegment", "CircleSegment", "EllipticalArc" or "CircularArc".

Int

Optional A value that specifies the vertical distance in pixels of the end point from the upper edge of the screen.

See also

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

EllipticalArc (Page 5070)

CircularArc (Page 5053)

EndTime(i)

Description

Specifies the end time of the time range to be shown in the trend view. The property "TagProviderType(i)" must have the value -1.

Access in Runtime: Read and write

Syntax

Object.**EndTime**(i)[=Time]

Object

Required An object of the "ScreenItem" type with the format "FunctionTrendControl" or "OnlineTabelControl".

Time

Optional A value or a constant that specifies the end time of the time range that is to be shown in the trend window.

Comments

The parameter i indicates the number of the trend.

If "EndTime(i)" is changed, illegal combinations with different attributes for the data connection could arise. This means you have to prevent an immediate acceptance of the change with "FreezeProviderConnections" before changing "EndTime(i)".

See also

FunctionTrendControl (Page 5077)

OnlineTableControl (Page 5114)

ExportXML

ExportXML

Only used internally.

The attribute can be assigned dynamic properties by means of the name **ExportXML**.

ExtendedOperation

Description

TRUE, when the slider regulator is set at the respective end value (minimum/maximum value). This is done by clicking the mouse in an area outside the current regulator setting. BOOLEAN write-read access.

ExtraSpaceForLabelDisplay

Description

Specifies the extra space required for the label.

Access in Runtime: Read and write

Syntax

Object.**ExtraSpaceForLabelDisplay**[=Int]

Object

Required A "ScreenItem" object with the format "Bar".

Int

Optional A value that specifies the extra space required for the label.

See also

Bar (Page 5033)

FlashFlashPicture

Description

TRUE, when flashing of the flash picture is activated. BOOLEAN write-read access.

FlashForeColor

Description

TRUE, when flashing of the text is activated. BOOLEAN write-read access.

FlashPicReferenced

Description

TRUE, when the assigned flash picture is to be saved. Otherwise, only the associated object reference is saved. Read only access.

FlashPicture

Description

Returns the flash picture. Read-only access.
The graphic (*.BMP or *.DIB) must be located in the "GraCS" folder of the current project so that it can be integrated.
In this context, the "FlashPicReferenced" property defines whether the flash picture should be saved or referenced together with the object.

FlashRateBorderColor

Description

Defines or returns the flash frequency for the lines of the object. Value range from 0 to 2.

0 = slow

1 = medium

2 = fast

FlashRateFlashPic

Description

Defines or returns the flash frequency for the flash picture. Value range from 0 to 2.

0 = slow

1 = medium

2 = fast

FlashRateForeColor

Description

Defines or returns the flash frequency for the object label. Value range from 0 to 2.

0 = slow

1 = medium

2 = fast

FormatPatternReadOnlySpecial

Description

Specifies that the "format pattern" field can only be read.

Access in Runtime: Read and write

Syntax

Object.**FormatPatternReadOnlySpecial**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "IOField".

BOOLEAN

Optional TRUE, if the "format pattern" field can only be read.

See also

IOField (Page 5098)

FreezeProviderConnections

Description

Specifies whether the properties for the data connection ("TagProviderType(i)", "Source.."...) can be changed without the change becoming effective immediately. If you change from, for example, "XDataLogTag(i)" illegal combinations with "YDataLogTag(i)" could arise.

This means attributes must have the value TRUE before changing the data connection "FreezeProviderConnections". After changing all the properties for the data connection, "FreezeProviderConnection" will be set to FALSE and the changes will become effective.

Access in Runtime: Read and write

Syntax

Object.**FreezeProviderConnections**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

BOOLEAN

Optional TRUE, if the properties for the data connection ("TagProviderType(i)", "Source.."...) can be changed without the change becoming effective immediately.

See also

FunctionTrendControl (Page 5077)

GetSelectionText

Description

Returns the selected text of the "MultiLineEdit" object.

HideTagNames

Description

Specifies whether the tag name is shown in the f/t() trend view.

Access in Runtime: Read and write

Syntax

Object.**HideTagNames**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTrendControl".

BOOLEAN

Optional TRUE, if the tag names are displayed.

See also

OnlineTrendControl (Page 5122)

LanguageSwitch

Description

Returns the value which defines where the language dependent assigned texts are stored. Read only access.

TRUE, if the texts are managed in the project texts editor. Translations into other languages are made in the project texts editor.

FALSE, when the texts are managed directly in the object. Translations into other languages can be made by means of the export/import function.

LeaveMarginForBorder

Description

Specifies whether an additional margin will be left for borders.

Access in Runtime: Read and write

Syntax

Object**.LeaveMarginForBorder**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Bar".

BOOLEAN

Optional TRUE, if an additional margin is enabled for borders.

See also

Bar (Page 5033)

LeaveMarginForMarkers

Description

Specifies whether an additional margin will be left for markers.

Access in Runtime: Read and write

Syntax

Object.**LeaveMarginForMarkers**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Bar".

BOOLEAN

Optional TRUE, if an additional margin is enabled for markers.

See also

Bar (Page 5033)

Min

Description

Defines or returns the absolute value in the case of the smallest value display. This value is displayed if the scale display is active.

NumberOfValues(i)

Description

Specifies the number of values that are shown in the trend window. The property "TagProviderType(i)"" must have the value -1. Whether the information is evaluated depends on the property "UseMeasurePoints(i)".".

Access in Runtime: Read and write

Syntax

Object.**NumberOfValues**(i)[=Int]

Object

Required An object of the "ScreenItem" type with the format "FunctionTrendControl".

Int

Optional A value or a constant that specifies the number of values that are shown in the trend window.

Comments

Number of value pairs for trend i (from tag provider)

See also

FunctionTrendControl (Page 5077)

Parent

Description

Returns the screen in which the specified object is embedded.

Access in Runtime: Read

Syntax

Object.Parent

Object

Required A "ScreenItem" object. This property is a standard property of the ScreenItem object and is therefore available for all formats.

Application example

The following example writes the name of the root screen in "BaseScreenName" tag:

```
'VBS_Example_Parent
Dim objScrItem, BaseScreenName
Set objScrItem = HMIRuntime.Screens(1).ScreenItems(1)
BaseScreenName = "Name of BaseScreen: " & objScrItem.Parent.ObjectName
```

See also

Line (Page 5102)

Polyline (Page 5137)

Ellipse (Page 5065)

Circle (Page 5048)

EllipseSegment (Page 5067)

CircleSegment (Page 5051)

EllipticalArc (Page 5070)
CircularArc (Page 5053)
Rectangle (Page 5149)
Polygon (Page 5134)
TextField (Page 5184)
IOField (Page 5098)
SymbolicIOField (Page 5170)
Button (Page 5040)
Switch (Page 5166)
GraphicView (Page 5091)
GraphicIOField (Page 5088)
Bar (Page 5033)
Clock (Page 5056)
Gauge (Page 5084)
Slider (Page 5159)
SymbolLibrary (Page 5175)
OnlineTrendControl (Page 5122)
FunctionTrendControl (Page 5077)
OnlineTableControl (Page 5114)
AlarmControl (Page 5018)
HTMLBrowser (Page 5096)
CheckBox (Page 5045)
OptionGroup (Page 5131)
WindowSlider (Page 5216)
Connector (Page 5058)
ScreenWindow (Page 5155)
DiskSpaceView (Page 5063)
ChannelDiagnose (Page 5044)
ScriptDiagnostics (Page 5157)
Group (Page 5094)

PicUpReferenced

Description

TRUE, when the picture assigned for the "Off" status should be saved in the object. Otherwise, only the associated object reference is saved. Read only access.

RemoveAllAssignment

Description

Deletes all entries.

RemoveAssignment

Description

Removes an entry.

RowNumber

Description

Specifies the row number of the Row object of a Table Control.

See also

AlarmControl (Page 5018)

RulerPrecision(i)

Description

Specifies the number of decimal places with which a measurement value of the specified trend will be displayed if it is measured via the function "Display value here".

Access in Runtime: Read and write

Syntax

Object.**RulerPrecision**(i)[=Int]

Object

Required A "ScreenItem" object with the format "OnlineTrendControl".

Int

Optional A value or a constant that specifies the number of decimal places with which a measurement value of the specified trend will be displayed if it is measured via the function "Display value here".

See also

OnlineTrendControl (Page 5122)

RulerXPrecision(i)

Description

Specifies the number of decimal places with which a measurement value of the X coordinate will be displayed. Whether the information is evaluated depends on the value of the property "XAxisMode(i)".

Access in Runtime: Read and write

Syntax

Object.**RulerXPrecision**(i)[=Int]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

Int

Optional A value or a constant that specifies the number of decimal places with which a measurement value of the X coordinate will be displayed.

Comments

The parameter i indicates the number of the trend.

You use the function "Display value here" to view the X coordinate of the measurement value.

See also

FunctionTrendControl (Page 5077)

RulerYPrecision(i)

Description

Specifies the number of decimal places with which a measurement value of the Y coordinate will be displayed. Whether the information is evaluated depends on the value of the property "XAxisMode(i)".".

Access in Runtime: Read and write

Syntax

Object.**RulerYPrecision**(i)[=Int]

Object

Required An object of the "ScreenItem" type with the format "FunctionTrendControl".

Int

Optional A value or a constant that specifies the number of decimal places with which a measurement value of the Y coordinate will be displayed.

Comments

The parameter i indicates the number of the trend.

You use the function "Display value here" to view the Y coordinate of the measurement value.

See also

FunctionTrendControl (Page 5077)

SetAllAssignment

Description

Sets all entries.

SetpointTrendArchiveStartId(i)

Description

Specifies the starting position from which the values of the setpoint trend will be read from the recipe. Whether the information is evaluated depends on the value of the property "ShowSetpointTrend(i)".

Access in Runtime: Read and write

Syntax

Object.**SetpointTrendArchiveStartId**(i)[=Int]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

Int

Optional A value or a constant that specifies the starting position from which the values of the setpoint trend will be read from the recipe.

Comments

The parameter i indicates the number of the trend.

See also

FunctionTrendControl (Page 5077)

SetpointTrendColor(i)

Description

Specifies the color of a setpoint trend. Whether the information is evaluated depends on the value of the property "ShowSetpointTrend(i)".

Access in Runtime: Read and write

Syntax

Object.**SetpointTrendColor**(i)[=Color]

Object

Required An object of the "ScreenItem" type with the format "FunctionTrendControl".

Color

Optional A value or a constant that specifies the color of a setpoint trend.

Comments

The parameter i indicates the number of the trend.

See also

FunctionTrendControl (Page 5077)

SetpointTrendNumberOfValues(i)

Description

Specifies the number of value pairs of a setpoint trend. Whether the information is evaluated depends on the value of the property "ShowSetpointTrend(i)".

Access in Runtime: Read and write

Syntax

Object.**SetpointTrendNumberOfValues**(i)[=Int]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

Int

Optional A value or a constant that specifies the number of value pairs of a setpoint trend.

Comments

The parameter i indicates the number of the trend.

See also

FunctionTrendControl (Page 5077)

SetProps

Description

Sets the value of the property.

SetSelection

Description

Sets a list of the lines which are selected.

ShareTimeAxis

Description

Specifies whether the trends of a trend window are displayed with a shared X axis.

Access in Runtime: Read and write

Syntax

Object.**ShareTimeAxis**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTrendControl".

BOOLEAN

Optional. TRUE, if the trends of the trend window are displayed with a shared X axis.

See also

OnlineTrendControl (Page 5122)

ShareTimeColumn

Description

Specifies whether a shared time column will be used in the table window.

Access in Runtime: Read and write

Syntax

Object.**ShareTimeColumn**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

BOOLEAN

Optional TRUE, if a shared time column will be used in the table window.

See also

OnlineTableControl (Page 5114)

ShareValueAxis

Description

Specifies whether the trends of a trend window are displayed with a shared Y axis.

Access in Runtime: Read and write

Syntax

Object.**ShareValueAxis**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTrendControl".

BOOLEAN

Optional TRUE, if the trends of the trend window are displayed with a shared Y axis.

See also

OnlineTrendControl (Page 5122)

ShareXAxis

Description

Specifies whether the trends of a trend window are displayed with a shared X axis.

Access in Runtime: Read and write

Syntax

Object.**ShareXAxis**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

BOOLEAN

Optional TRUE, if the trends of the trend window are displayed with a shared X axis.

See also

FunctionTrendControl (Page 5077)

ShareYAxis

Description

Specifies whether the trends of a trend window are displayed with a shared Y axis.

Access in Runtime: Read and write

Syntax

Object.**ShareYAxis**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "FunctionTrendControl".

BOOLEAN

Optional TRUE, if the trends of the trend window are displayed with a shared Y axis.

See also

FunctionTrendControl (Page 5077)

ShowBorder

Description

Specifies whether the window will be shown with a border.

Access in Runtime: Read and write

Syntax

Object.**ShowBorder[**=BOOLEAN]

Object

Required A "ScreenItem" object with the format "UserArchiveControl".

BOOLEAN

Optional TRUE if the window is to be shown with a border.

ShowColumn(i)

Description

Specifies whether a column pair that is referenced via the "CurrentColumnIndex" property is shown.

Access in Runtime: Read and write

Syntax

Object.**ShowColumn**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTableControl".

BOOLEAN

Optional TRUE, if a column pair that is referenced via the "CurrentColumnIndex" property is shown.

See also

OnlineTableControl (Page 5114)

ShowCurve(i)

Description

Specifies whether a trend that is referenced via the "CurrentCurveIndex" property is shown.

Access in Runtime: Read and write

Syntax

Object.**ShowCurve**(i)[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

BOOLEAN

Optional TRUE, if a trend that is referenced via the "CurrentCurveIndex" property is shown.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ShowInputControls

Description

Establishes whether additional fields exist for entering the address and password.

Syntax

Object.**ShowInputControls**

Object

Required A "ScreenItem" object with the format "SmartClientView".

ShowMainFrame

Description

Specifies whether the entire bar object is provided with a border.

Access in Runtime: Read and write

Syntax

Object.**ShowMainFrame**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Bar".

BOOLEAN

Option TRUE, if the entire bar object is shown with a border.

See also

Bar (Page 5033)

ShowOverflowIndicator

Description

Specifies whether an overflow indicator is shown when the process value exceeds or falls short of the limit value.

Access in Runtime: Read and write

Syntax

Object.**ShowOverflowIndicator**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format ""Bar".".

BOOLEAN

TRUE, if an overflow indicator is shown when the process value exceeds or falls short of the limit value.

See also

Bar (Page 5033)

ShowSetpointTrend(i)

Description

Specifies whether a setpoint trend that belongs to a trend that is referenced via "CurrentCurveIndex" should be shown.

Access in Runtime: Read and write

Syntax

Object.**ShowSetpointTrend**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "FunctionTrendControl".

BOOLEAN

Optional TRUE, if a setpoint trend that belongs to a trend that is referenced via CurrentCurveIndex" should be shown.

See also

FunctionTrendControl (Page 5077)

SmallChange

Description

Defines how many steps the controller can be moved with one mouse click or returns the value.

StartPoint

Description

Specifies the beginning of the object or returns it.

StatusbarShowArchiveName

Description

Specifies whether the archive name will be displayed in the status bar.

Access in Runtime: Read and write

Syntax

Object.**StatusbarShowArchiveName**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "UserArchiveControl".

BOOLEAN

Optional TRUE, if the archive name will be displayed in the status bar.

StatusbarShowColumn

Description

Specifies whether the current number of selected data record columns will be displayed.

Access in Runtime: Read and write

Syntax

Object.**StatusbarShowColumn**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "UserArchiveControl".

BOOLEAN

Optional TRUE, specifies whether the current number of selected data record columns will be displayed.

StatusbarShowRecord

Description

Specifies whether the field coordinates of the selected data record will be displayed in the status bar.

Access in Runtime: Read and write

Syntax

Object.**StatusbarShowRecord**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "UserArchiveControl".

BOOLEAN

Optional TRUE, if the field coordinates of the selected data record will be displayed in the status bar.

StatusbarShowRow

Description

Specifies whether the current number of selected data record rows will be displayed.

Access in Runtime: Read and write

Syntax

Object.**StatusbarShowRow**[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "UserArchiveControl".

BOOLEAN

Optional TRUE, if the current number of selected data record rows will be displayed.

StatusbarShowText

Description

Specifies whether the current status of the database will be displayed in the status bar.

Access in Runtime: Read and write

Syntax

Object.**StatusbarShowText**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "UserArchiveControl".

BOOLEAN

Optional TRUE, specifies whether the current status of the database will be displayed in the status bar.

SwapDimensionsWithOrientation

Description

Specifies whether the values for the height and width of the bar should be automatically replaced if the bar alignment is changed.

Access in Runtime: Read and write

Syntax

Object.**SwapDimensionsWithOrientation**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Bar".

BOOLEAN

Optional TRUE, if the values for the height and width of the bar should be automatically replaced if the bar alignment is changed.

See also

Bar (Page 5033)

TimeAxisShowLargeIncrements(i)

Description

Specifies whether the time axis is scaled with long marking lengths.

Access in Runtime: Read and write

Syntax

Object.**TimeAxisShowLargeIncrements**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTrend".

BOOLEAN

Optional TRUE, if the time axis is scaled with long marking lengths.

Comments

The distance of two long marking lines can be altered with the ""TimeAxisLargeIncrementSize(i) property.

See also

OnlineTrendControl (Page 5122)

TimeAxisShowSmallIncrements(i)

Description

Specifies whether the time axis is scaled with short marking lengths.

Access in Runtime: Read and write

Syntax

Object.**TimeAxisShowSmallIncrements**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "OnlineTrend".

BOOLEAN

Optional TRUE, if the time axis is scaled with short marking lengths.

Comments

The distance of two short marking lines can be altered with the "TimeAxisSmallIncrementSize(i)" property.

See also

OnlineTrendControl (Page 5122)

TimeJumpColor(i)

Description

Specifies the color that is identified in the time jump available in the log. Whether the information is evaluated depends on the property "TimeJumpEnabled(i)".".

Access in Runtime: Read and write

Syntax

Object.**TimeJumpColor**(i)[=Color]

Object

Required A "ScreenItem"" object with the formats ""OnlineTrend" or ""TableView".".

Color

Optional A value or a constant that specifies the color that identifies the time jump available in the log.

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TimeJumpEnabled(i)

Description

Specifies if the time jumps available in the log are identified in the color set up in "TimeJumpColor(i)".

Access in Runtime: Read and write

Syntax

Object.**TimeJumpEnabled**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the formats "OnlineTrend" or ""TableView".

BOOLEAN

Optional TRUE, if the time jumps available in the log are identified in the color specified in "TimeJumpColor(i)".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TimeOverlapColor(i)

Description

Specifies the color that is identified in the time overlap available in the log. Whether the information is evaluated depends on the attribute "TimeOverlapEnabled(i)".".

Access in Runtime: Read and write

Syntax

Object.**TimeOverlapColor**(i)[=Color]

Object

Required A "ScreenItem" object with the formats "OnlineTrend" or ""TableView".

Color

Optional A value or a constant that specifies the color that identifies the time overlap available in the log.

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TimeOverlapEnabled(i)

Description

Specifies if the time overlaps available in the log are identified in the color set up in "TimeOverlapColor(i)".

Access in Runtime: Read and write

Syntax

Object.**TimeOverlapEnabled**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the formats "OnlineTrend"" or "TableView".

BOOLEAN

Optional TRUE, if the time jumps overlaps available in the log are identified in the color specified in ""TimeOverlapColor(i)"".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

Type

Description

Returns the type of the specified object as STRING. Name diagram: Hmi<ObjectName>, e.g. HmiCircle for the "circle" screen object.

Access in Runtime: Read

Syntax

Object.Type

Object

Required A "ScreenItem" object.

UnselBGColor property

Description

Defines or returns the background color of entries in the text list object which are not selected. LONG write-read access.

UnselTextColor property

Description

Defines or returns the color of the text for entries in the text list object which are not selected. LONG write-read access.

UpdateCycle property

Description

Returns the type and frequency of updating the picture window in Runtime. Read only access.

UseBarBorderConstraints

Description

Specifies whether the bar border is restricted to specific values.

Access in Runtime: Read and write

Syntax

Object.**UseBarBorderConstraints**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Bar".

BOOLEAN

Optional. TRUE, if the bar border is restricted to specific values.

See also

Bar (Page 5033)

UseEffectiveProcessValue

Description

Specifies whether an actual percentage made up of maximum value, minimum value, average of the last 15 values and hysteresis will be used.

Access in Runtime: Read and write

Syntax

Object.**UseEffectiveProcessValue**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Bar".

BOOLEAN

Optional TRUE, if an actual percentage made up of maximum value, minimum value, average of the last 15 values and hysteresis will be used.

See also

Bar (Page 5033)

UseGDI

Description

Specifies whether the ellipse is shown via GDI or GDI+.

Access in Runtime: Read and write

Syntax

Object.**UseGDI**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Ellipse".

BOOLEAN

Optional TRUE, if the ellipse is shown via GDI.

See also

Ellipse (Page 5065)

UseMeasurePoints(i)

Description

Specifies whether the time range for the selected trend will be determined by measuring points or the end time. Whether the information is evaluated depends on the properties "TimeAxisEndTime", "TimeAxisBeginTime(i)".

Define time range for curve i through measure points (TRUE) or end time (FALSE) (for time based tag providers)

Access in Runtime: Read and write

Syntax

Object.**UseMeasurePoints**(i)[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl"or "FunctionTrendControl".

BOOLEAN

Optional TRUE, if the time range for the selected trend will be determined by measuring points.

Comments

The parameter i indicates the number of the trend.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

UseMultipleLimits

Description

Specifies whether one or more pairs of limits are shown as a selection or a line.

Access in Runtime: Read and write

Syntax

Object.**UseMultipleLimits**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format ""Bar".

BOOLEAN

Optional TRUE, if one or more pairs of limits are displayed as a selection or a line.

See also

Bar (Page 5033)

UseScaleConstraints

Description

Specifies whether the distance between two large marking lengths of the scale are calculated from the minimum and maximum values.

Access in Runtime: Read and write

Syntax

Object.**UseScaleConstraints**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Bar".

BOOLEAN

Optional TRUE, if the distance between two large marking lengths of the scale are calculated from the minimum and maximum values.

See also

Bar (Page 5033)

UseScaledBarBorder

Description

Specifies whether the rectangle that surrounds the object is shown dependent of the scale or as a default.

Access in Runtime: Read and write

Syntax

Object.**UseScaledBarBorder**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Bar".".

BOOLEAN

Optional TRUE, if the rectangle that surrounds the object is shown dependent on the scale.

See also

Bar (Page 5033)

UseSeparateDiagrams

Description

Specifies whether the trends are shown staggered.

Access in Runtime: Read and write

Syntax

Object.**UseSeparateDiagrams**[=BOOLEAN]

Object

Required A "ScreenItem" object with the formats "OnlineTrend" or "FunctionTrendView".

BOOLEAN

Optional TRUE, if the trends are shown staggered.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

UseSimplePresicionOffset

Description

Specifies how the field length will be calculated for the labeling of the scale.

Access in Runtime: Read and write

Syntax

Object.**UseSimplePresicionOffset**[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "Bar".

BOOLEAN

Optional TRUE, if the field length will be calculated for the labeling of the scale.

See also

Bar (Page 5033)

UserArchiveNumberOfValues(i)

Description

Specifies the number of values that are loaded from the recipe. The "TagProviderType(i)"" property must have the value -2.

Access in Runtime: Read and write

Syntax

Object.**UserArchiveNumberOfValues**(i)[=Int]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".

Int

Optional A value or a constant that specifies the number of values that are loaded from the recipe.

Comments

The parameter i indicates the number of the trend.

If "UserArchiveNumberOfValues(i)"" is changed, illegal combinations with different attributes for the data connection could arise. This means you have to prevent immediate acceptance of the change with ""FreezeProviderConnections"" before changing ""UserArchiveNumberOfValues(i)"".

See also

FunctionTrendControl (Page 5077)

UserArchiveStartId(i)

Description

Specifies the data record starting from which the values for the selected trend are loaded from the recipe. The "TagProviderType(i)"" property must have the value -2.

Access in Runtime: Read and write

Syntax

Object.**UserArchiveStartId**(i)[=Int]

Object

Required An object of the "ScreenItem" type with the format "FunctionTrendControl".

Int

Optional A value or a constant that specifies the data record starting from which the values for the selected trend are loaded from the recipe.

Comments

The parameter i indicates the number of the trend.

If "UserArchiveStartId(i)" is changed, illegal combinations with different attributes for the data connection could arise. This means you have to prevent immediate acceptance of the change with ""FreezeProviderConnections" before changing ""UserArchiveStartId(i)".

See also

FunctionTrendControl (Page 5077)

UserData

Description

Specifies the value that is passed to the VB script when a user-defined menu entry or toolbar button is executed.

Use the "Data" field in the "Menus and toolbars" editor to pass a parameter to the procedure.

Access in Runtime: Read and write

Syntax

Object.**UserData**[=String]

Object

Required A "Item"-type object.

String

Optional. A value or constant that is passed to the VB script when a user-defined menu entry or toolbar button is executed.

ValueAxisGridLineInterval(i)

Description

Specifies the distance between two grid lines. Whether the information is evaluated depends on the properties "ValueAxisShowGridLines(i)" and "TimeAxisShowGridLines(i)".".

Access in Runtime: Read and write

Syntax

Object.**ValueAxisGridLineInterval**(i)[=Double]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

Double

Optional A value or a constant that specifies the distance between two grid lines.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ValueAxisLargeIncrementSize(i)

Description

Specifies the distance between two long marking lengths of the scale. Whether the information is evaluated depends on the properties "ValueAxisShowLargeIncrements(i)" and ""TimeAxisShowLargeIncrements(i)".".

Access in Runtime: Read and write

Syntax

Object.**ValueAxisLargeIncrementSize**(i)[=Double]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

Double

Optional A value or a constant that specifies the distance between two marking lengths of the scale.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ValueAxisShowLargeIncrements(i)

Description

Specifies whether the value axis is scaled with long marking lengths.

Access in Runtime: Read and write

Syntax

Object.**ValueAxisShowLargeIncrements**(i)[=BOOLEAN]

Object

Required An object of the "ScreenItem" with the format "OnlineTrendControl" or "FunctionTrendControl".

BOOLEAN

Optional TRUE, if the value axis is scaled with long marking lengths.

Comments

The distance of two long marking lines can be altered with the "ValueAxisLargeIncrementSize(i)"" property.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ValueAxisShowSmallIncrements(i)

Description

Specifies whether the value axis is scaled with short marking lengths.

Access in Runtime: Read and write

Syntax

Object.**ValueAxisShowSmallIncrements**(i)[=BOOLEAN]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

BOOLEAN

Optional TRUE, if the value axis is scaled with short marking lengths.

Comments

The distance between two short marking lines can be altered with the "ValueAxisSmallIncrementSize(i)" property.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ValueAxisSmallIncrementSize(i)

Description

Specifies the distance between two short marking lengths of the scale. Whether the information is evaluated depends on the properties ""ValueAxisShowSmallIncrements(i)")" and ""TimeAxisShowSmallIncrements(i)".".

Access in Runtime: Read and write

Syntax

Object.**ValueAxisSmallIncrementSize**(i)[=Double]

Object

Required An object of the "ScreenItem" type with the format "OnlineTrendControl" or "FunctionTrendControl".

Double

Optional A value or a constant that specifies the distance between two short marking lengths of the scale.

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

XAxisDecimalPrecision(i)

Description

Specifies the number of decimal places with which the scaling value of the X axis will be displayed.

Access in Runtime: Read and write

Syntax

Object.**XAxisDecimalPrecision**(i)[=Int]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".

Int

Optional A value or a constant that specifies the number of decimal places with which the scaling value for the X axis will be displayed.

See also

FunctionTrendControl (Page 5077)

XAxisGridLineInterval(i)

Description

Specifies the distance between two grid lines of the X axis. Whether the information is evaluated depends on the value of the property ""XAxisShowGridLines(i)".".

Access in Runtime: Read and write

Syntax

Object.**XAxisGridLineInterval**(i)[=Double]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".".

Double

Optional A value or a constant that specifies the distance between two grid lines of the X axis.

See also

FunctionTrendControl (Page 5077)

XAxisLargeIncrementSize(i)

Description

Specifies the distance of two long marking lengths of the X axis. Whether the information is evaluated depends on the property "XAxisShowLargeIncrements(i)".".

Access in Runtime: Read and write

Syntax

Object.**XAxisLargeIncrementSize**(i)[=Double]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".

Double

Optional A value or a constant that specifies the distance of two long marking lengths of the scaling of the X axis.

See also

FunctionTrendControl (Page 5077)

XAxisMode(i)

Description

Specifies whether a mutual time axis will be used in the trend window for all trends.

Specifies time units used by tag logging controls.

Access in Runtime: Read and write

Syntax

Object.**XAxisMode**(i)[=TrendViewTimeAxisMode]

Object

Required A "ScreenItem" object with the format "FunctionTrendContol".

TrendViewTimeAxisMode

(1): The time axis will be scaled.

(2): No time range has been specified. Only a random sample will be displayed.

(3): The time range will be established via tags.

(4): The time range will be defined by constant values.

Comments

Specifies whether a mutual X axis will be used in the trend window for all trends.

See also

FunctionTrendControl (Page 5077)

XAxisShowLargeIncrements(i)

Description

Specifies whether the X axis is scaled with long marking lengths. The distance of two long marking lines can be altered with the "XAxisLargeIncrementSize(i)"" property.

Access in Runtime: Read and write

Syntax

Object.**XAxisShowLargeIncrements**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format ""FunctionTrendView".

BOOLEAN

Optional TRUE, if the X axis is scaled with long marking lengths.

See also

FunctionTrendControl (Page 5077)

XAxisShowSmallIncrements(i)

Description

Specifies whether the X axis is scaled with short marking lengths. The distance of two short marking lines can be altered with the "XAxisSmallIncrementSize(i)" property.

Access in Runtime: Read and write

Syntax

Object.**XAxisShowSmallIncrements**(i)[=BOOLEAN]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".

BOOLEAN

Optional TRUE, if the X axis is scaled with short marking lengths.

See also

FunctionTrendControl (Page 5077)

XAxisSmallIncrementSize(i)

Description

Specifies the distance of two short marking lengths of the X axis. Whether the information is evaluated depends on the property "XAxisShowSmallIncrements(i).".

Access in Runtime: Read and write

Syntax

Object.**XAxisSmallIncrementSize**(i)[=Double]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".

Double

Optional A value or a constant that specifies the distance of two short marking lengths of the X axis scaling.

See also

FunctionTrendControl (Page 5077)

XDataLogTag(i)

Description

Specifies the tag that will be shown along the X axis. The property "TagProviderType(i)" must have the value -1.

Access in Runtime: Read and write

Syntax

Object.**XDataLogTag**(i)[=HmiLoggingTag]

Object

Required A ""ScreenItem" object with the format "FunctionTrendView".

HmiLoggingTag

Optional A value or a constant that specifies which tag will be shown along the X axis.

Comments

If "XOnlineTag(i)"" is changed, illegal combinations with different attributes for the data connection could arise. This means you have to prevent mmediate acceptance of the change with "FreezeProviderConnections before changing "XOnlineTag(i).

See also

FunctionTrendControl (Page 5077)

XOnlineTag(i)

Description

Specifies the tag that will be shown along the X axis. The property "TagProviderType(i)"" must have the value -1.

Access in Runtime: Read and write

Syntax

Object.**XOnlineTag**(i)[=HmiTag]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".

HmiTag

Optional A value or a constant that specifies which tag will be shown along the X axis.

Comments

The parameter i indicates the number of the trend.

If "XOnlineTag(i)" is changed, illegal combinations with different attributes for the data connection could arise. This means you have to prevent an immediate acceptance of the change with "FreezeProviderConnections" before changing "XOnlineTag(i)".

See also

FunctionTrendControl (Page 5077)

YDataLogTag(i)

Description

Specifies the tag that will be shown along the Y axis. The property "TagProviderType(i)"" must have the value -1.

Access in Runtime: Read and write

Syntax

Object.**YDataLogTag**(i)[=HmiLoggingTag]

Object

Required A "ScreenItem" object with the format "FunctionTrendView"".

HmiLoggingTag

Optional A value or a constant that specifies which tag will be shown along the Y axis.

Comments

If "YOnlineTag(i)"" is changed, illegal combinations with different attributes for the data connection could arise. This means you have to prevent immediate acceptance of the change with "FreezeProviderConnections" before changing "YOnlineTag(i)"".

See also

FunctionTrendControl (Page 5077)

YOnlineTag(i)

Description

Specifies the tag that will be shown along the Y axis. The property "TagProviderType(i)" must have the value -1.

Access in Runtime: Read and write

Syntax

Object.**YOnlineTag**(i)[=HmiTag]

Object

Required A "ScreenItem" object with the format "FunctionTrendView".

HmiTag

Optional A value or a constant that specifies which tag will be shown along the Y axis.

Comments

The parameter i indicates the number of the trend.

If "YOnlineTag(i)"" is changed, illegal combinations with different attributes for the data connection could arise. This means you have to prevent immediate acceptance of the change with "FreezeProviderConnections" before changing "YOnlineTag(i)".

See also

FunctionTrendControl (Page 5077)

Methods

Methods A-G

Activate

Description

Activates the permanent window or the root screen.

To activate a not selected screen, use the "BaseScreenName" property.

It only makes sense to use the activate method with the following operable screen objects. An error message is output at screen objects which cannot be operated, for example rectangles.

- I/O field
- Switch
- Symbol library
- Trend view
- f(x) trend view
- HTML browser
- Slider
- Graphic I/O field
- Symbolic I/O field
- Button
- Alarm view
- User view
- Recipe view
- Sm@rtClient View
- Status/Force

Syntax

Expression.Activate

Expression

Necessary. An output which returns an object of the "Screen" or "ScreenItem" type.

Parameters

--

See also

ScreenItem (Page 5003)

Screen (Page 5000)

ChannelDiagnose (Page 5044)

CheckBox (Page 5045)

Circle (Page 5048)

CircleSegment (Page 5051)

CircularArc (Page 5053)

Clock (Page 5056)
Connector (Page 5058)
DateTimeField (Page 5061)
DiskSpaceView (Page 5063)
Ellipse (Page 5065)
EllipseSegment (Page 5067)
EllipticalArc (Page 5070)
Gauge (Page 5084)
GraphicIOField (Page 5088)
GraphicView (Page 5091)
HTMLBrowser (Page 5096)
IOField (Page 5098)
Rectangle (Page 5149)
ScriptDiagnostics (Page 5157)
Switch (Page 5166)
SymbolicIOField (Page 5170)
SymbolLibrary (Page 5175)
TextField (Page 5184)
TrendView (Page 5195)
TubeArcObject (Page 5198)
TubeDoubleTeeObject (Page 5200)
TubePolyline (Page 5202)
TubeTeeObject (Page 5205)
UserView (Page 5214)
WindowSlider (Page 5216)
StatusForce (Page 5164)
SmartClientView (Page 5162)
Slider (Page 5159)
ScreenWindow (Page 5155)
RoundButton (Page 5151)
Polyline (Page 5137)
Polygon (Page 5134)
OptionGroup (Page 5131)
MultiLineEdit (Page 5109)
MediaPlayer (Page 5107)

Listbox (Page 5104)

Line (Page 5102)

Bar (Page 5033)

Button (Page 5040)

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

FunctionTrendControl (Page 5077)

AlarmView (Page 5029)

AlarmControl (Page 5018)

ActivateDynamic

Description

Dynamically activates a trigger and the specified cycle for a property at runtime. This requires a VB script at the property as well as a trigger set to "On demand". Every time the trigger is activated a different activation cycle can be used.

Syntax

```
Expression.ActivateDynamic (ByVAl bstrPropertyName As String, ByVal
bstrCycleName As String)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

Parameter	Description
bstrPropertyName	Name of property to which trigger relates.
bstrCycleName	Name of activation cycle, e.g. "CycleTime1s".

See also

ChannelDiagnose (Page 5044)

CheckBox (Page 5045)

Circle (Page 5048)

CircleSegment (Page 5051)

CircularArc (Page 5053)

Clock (Page 5056)
Connector (Page 5058)
DiskSpaceView (Page 5063)
Ellipse (Page 5065)
EllipseSegment (Page 5067)
EllipticalArc (Page 5070)
Gauge (Page 5084)
GraphicIOField (Page 5088)
GraphicView (Page 5091)
HTMLBrowser (Page 5096)
IOField (Page 5098)
Rectangle (Page 5149)
ScriptDiagnostics (Page 5157)
SymbolicIOField (Page 5170)
SymbolLibrary (Page 5175)
TextField (Page 5184)
TubeArcObject (Page 5198)
TubeDoubleTeeObject (Page 5200)
TubePolyline (Page 5202)
TubeTeeObject (Page 5205)
UserView (Page 5214)
WindowSlider (Page 5216)
Slider (Page 5159)
ScreenWindow (Page 5155)
RoundButton (Page 5151)
Polyline (Page 5137)
Polygon (Page 5134)
OptionGroup (Page 5131)
MultiLineEdit (Page 5109)
MediaPlayer (Page 5107)
Listbox (Page 5104)
Line (Page 5102)
Bar (Page 5033)
Button (Page 5040)
OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

Add

Description of TagSet Object

Adds a tag to the list. A tag may be added to the tag object by using name or reference.

syntax

```
Expression.Add [Tag]
```

Expression

Necessary. An expression which returns an object of type "TagSet".

Parameters

VARIANT

Parameters	Description
Tag	Name of a WinCC tag or reference to a tag object to be added to the list.

Example:

In the following example, a TagSet object is generated and a tag is added.

```
'VBS170
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
```

Tag objects may also be added as follows.

```
'VBS171
Dim Tag
Set Tag = HMIRuntime.Tags("Motor2")
Dim group2
Set group2 = HMIRuntime.Tags.CreateTagSet
group2.Add Tag
```

Description of DataSet Object

Adds a value or object reference to the list.

Note

The Data Set Object does not support classes.
Objects of type Screen, Screens, ScreenItem, ScreenItems, Tag and TagSet cannot be included in the DataSet list.
For object references it must be ascertained that objects are multiread-enabled.

syntax

```
Expression.Add [vtName], [vtUserData]
```

Expression

Necessary. An expression which returns an object of type "DataSet".

Parameters

VARIANT

Parameters	Description
vtName	Name by which value or tag are to be added to list.
vtUserData	Value to be added to list.

Example:

In this example, a value is included in the DataSet list.

```
'VBS172
HMIRuntime.DataSet.Add "Motor1",23
```

See also

DataSet (list) (Page 4990)

OnlineTrendControl (Page 5122)

AttachDB method

Description

Executes the "Connect backup" key function of the control.

Syntax

```
Ausdruck.AttachDB()
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

CalculateStatistic

Description

Executes the "calculate statistics" key function of the f(t) trend view.

Syntax

```
Ausdruck.CalculateStatistic()
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

CopyRows

Description

Executes the "Copy lines" key function of the control.

Syntax

```
Ausdruck.CopyRows()
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

--

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

UserArchiveControl (Page 5207)

AlarmControl (Page 5018)

Create

Description

Creates a new Alarm object.

Syntax

```
Expression.Create (VARIANT vtApplication)
```

Expression

Required An expression which returns an object of type "Alarm".

Parameters

```
VARIANT
```

Parameters	Description
vtApplication	Name of alarm object (optional)

See also

OnlineTrendControl (Page 5122)

CreateTagSet

Description

Creates a new TagSet object. This object may be used for optimized multi-tag access.

syntax

```
Expression.CreateTagSet()
```

Expression

Necessary. An expression which returns an object of type "TagSet".

Parameters

VARIANT

Example:

The following example shows how to create a TagSet object.

```
'VBS168
'Build a Reference to the TagSet Object
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
```

See also

OnlineTrendControl (Page 5122)

CutRows

Description

Executes the "Cut rows" key function of the recipe view.

Syntax

```
Ausdruck.CutRows()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

UserArchiveControl (Page 5207)

DeactivateDynamic

Description

Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/ during runtime.

Syntax

```
Ausdruck.DeactivateDynamic(ByVal bstrPropertyName As String)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

String

Parameters	Description
bstrPropertyName	Name of property to which trigger relates.

See also

ChannelDiagnose (Page 5044)

CheckBox (Page 5045)

Circle (Page 5048)

CircleSegment (Page 5051)
CircularArc (Page 5053)
Clock (Page 5056)
Connector (Page 5058)
DiskSpaceView (Page 5063)
Ellipse (Page 5065)
EllipseSegment (Page 5067)
EllipticalArc (Page 5070)
Gauge (Page 5084)
GraphicIOField (Page 5088)
GraphicView (Page 5091)
HTMLBrowser (Page 5096)
IOField (Page 5098)
Rectangle (Page 5149)
ScriptDiagnostics (Page 5157)
SymbolicIOField (Page 5170)
SymbolLibrary (Page 5175)
TextField (Page 5184)
TubeArcObject (Page 5198)
TubeDoubleTeeObject (Page 5200)
TubePolyline (Page 5202)
TubeTeeObject (Page 5205)
UserView (Page 5214)
WindowSlider (Page 5216)
Slider (Page 5159)
ScreenWindow (Page 5155)
RoundButton (Page 5151)
Polyline (Page 5137)
Polygon (Page 5134)
OptionGroup (Page 5131)
MultiLineEdit (Page 5109)
MediaPlayer (Page 5107)
Listbox (Page 5104)
Line (Page 5102)
Bar (Page 5033)

Button (Page 5040)

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

DeleteRows

Description

Executes the "Delete rows" key function of the recipe view.

Syntax

```
Ausdruck.DeleteRows()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

UserArchiveControl (Page 5207)

DetachDB

Description

Executes the "Disconnect backup" key function of the control.

Syntax

```
Ausdruck.DetachDB()
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

Edit

Description

Executes the "Edit" key function of the table view.

Syntax

```
Ausdruck.Edit()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

AlarmControl (Page 5018)

Export

Description

Executes the "Export archive" or "Export data" key function of the control.

Syntax

```
Ausdruck.Export()
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

GetColumn

Description

Returns the column object of the recipe view designated by name or index as "ICCAxUAColumn" type.

Syntax

```
Ausdruck.GetColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of the column of the recipe view

Example

```
'VBS312
Dim ctrl
Dim objColumn
Set ctrl = ScreenItems("RecipeControl")
Set objColumn = ctrl.GetColumn("Field1")
objColumn.Length = 30
Set objColumn = ctrl.GetColumn(3)
objColumn.Align = 2
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "Column" listing, for example, you write "objColumn.Align" instead of "objColumn.ColumnAlign".

See also

OnlineTrendControl (Page 5122)

UserArchiveControl (Page 5207)

GetColumnCollection

Description

Returns the list of all column objects of the recipe view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS313
Dim ctrl
Dim coll
Dim field
Set ctrl = ScreenItems("RecipeControl")
Set coll = ctrl.GetColumnCollection
HMIRuntime.Trace "Number of fields:" & coll.Count & vbCrLf
For Each field In coll
 HMIRuntime.Trace field.Name & vbCrLf
 HMIRuntime.Trace field.Type & vbCrLf
 HMIRuntime.Trace field.Length & vbCrLf
 HMIRuntime.Trace field.Caption & vbCrLf
Next
```

See also

OnlineTrendControl (Page 5122)

UserArchiveControl (Page 5207)

GetHistlistColumnCollection

Description

Returns the list of all column objects of the message view hit list as "ICCAxCollection" type.

Syntax

```
Expression.GetHitlisteColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS315
Dim ctrl
Dim coll
Dim hitlistcol
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetHitlistColumnCollection
HMIRuntime.Trace "Number of hitlist columns:" & coll.Count & vbCrLf
For Each hitlistcol In coll
 HMIRuntime.Trace hitlistcol.Index & vbCrLf
 HMIRuntime.Trace hitlistcol.Name & vbCrLf
 HMIRuntime.Trace hitlistcol.Sort & vbCrLf
 HMIRuntime.Trace hitlistcol.SortIndex & vbCrLf
Next
```

See also

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

GetHitlistColumn

Description

Returns the column object of the message view hit list designated by name or index as "ICCAxMessageColumn" type.

Syntax

```
Expression.GetHitlistColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of hitlist column

Example

```
'VBS314
Dim ctrl
Dim objHitlistColumn
Set ctrl = ScreenItems("AlarmControl")
Set objHitlistColumn = ctrl.GetHitlistColumn("Date")
objHitlistColumn.Sort = 2
Set objHitlistColumn = ctrl.GetHitlistColumn("AverageComeGo")
objHitlistColumn.Visible = FALSE
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "HitlistColumn" listing, for example, you write "objHitlistColumn.Visible" instead of "objHitlistColumn.HitlistColumnVisible".

See also

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

GetMessageBlock

Description

Returns the message block of the message view designated by name or index as "ICCAxMessageBlock" type.

Syntax

```
Expression.GetMessageBlock(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of message block.

Example

```
'VBS316
Dim ctrl
Dim objMsgBlock
Set ctrl = ScreenItems("AlarmControl")
Set objMsgBlock = ctrl.GetMessageBlock("Date")
objMsgBlock.Align = 2
Set objMsgBlock = ctrl.GetMessageBlock("Number")
objMsgBlock.LeadingZeros = 4
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "MessageBlock" listing, for example, you write "objMsgBlock.Align" instead of "objMsgBlock.MessageBlockAlign".

See also

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

GetMessageBlockCollection

Description

Returns the list of all message block objects of the message view as "ICCAxCollection" type.

Syntax

```
Expression.GetMessageBlockCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS317
Dim ctrl
Dim coll
Dim msgblock
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetMessageBlockCollection
For Each msgblock In coll
 msgblock.Align = 1
 msgblock.Length = 12
 msgblock.Selected = TRUE
Next
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "MessageBlock" listing, for example, you write "msgblock.Align" instead of "msgblock.MessageBlockAlign".

See also

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

GetMessageColumn

Description

Returns the column object of the message view designated by name or index as "ICCAxMessageColumn" type.

Syntax

```
Expression.GetMessageColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of column in message list.

Example

```
'VBS318
Dim ctrl
Dim objMessColumn
Set ctrl = ScreenItems("AlarmControl")
Set objMessColumn = ctrl.GetMessageColumn("Date")
objMessColumn.Visible = FALSE
Set objMessColumn = ctrl.GetMessageColumn("Number")
objMessColumn.Sort = 1
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "MessageColumn" listing, for example, you write "objMessColumn.Visible" instead of "objMessColumn.MessageColumnVisible".

See also

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

GetOperatorMessage

Description

Returns the operator message object of the message view designated by name or index as "ICCAxOperatorMessage" type.

Syntax

```
Expression.GetOperatorMessage(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of operator message

Example

```
'VBS320
Dim ctrl
Dim objOpMess
Set ctrl = ScreenItems("AlarmControl")
Set objOpMess = ctrl.GetOperatorMessage(0)
objOpMess.Source1 = "Number"
objOpMess.SourceType1 = 1
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "OperatorMessage" listing, for example, you write "objOpMess.Source1" instead of "objOpMess.OperatorMessageSource1".

See also

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

GetMessageColumnCollection

Description

Returns the list of all column objects of the message view as "ICCAxCollection" type.

Syntax

```
Expression.GetMessageColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS319
Dim ctrl
Dim coll
Dim msgcol
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetMessageColumnCollection
HMIRuntime.Trace "Number of message columns:" & coll.Count & vbCrLf
For Each msgcol In coll
 HMIRuntime.Trace msgcol.Index & vbCrLf
 HMIRuntime.Trace msgcol.Name & vbCrLf
 HMIRuntime.Trace msgcol.Sort & vbCrLf
 HMIRuntime.Trace msgcol.SortIndex & vbCrLf
Next
```

See also

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

GetOperatorMessageCollection

Description

Returns the list of all operator message objects of the message view as "ICCAxCollection" type.

Syntax

```
Expression.GetOperatorMessageCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS321
Dim ctrl
Dim coll
Dim opmsg
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetOperatorMessageCollection
For Each opmsg In coll
 HMIRuntime.Trace opmsg.Index & vbCrLf
 HMIRuntime.Trace opmsg.Name & vbCrLf
 HMIRuntime.Trace opmsg.Number & vbCrLf
 HMIRuntime.Trace opmsg.Selected & vbCrLf
Next
```

See also

OnlineTrendControl (Page 5122)

AlarmControl (Page 5018)

GetRow

Description

Returns the row object defined by its row number in the table-based controls as "ICCAxDataRow" type.

Syntax

```
Expression.GetRow(ByVal IRow As Long)
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

Long

Parameters	Description
IRow	Number of the desired line of the control.

Example

```
'VBS356
Dim coll
Dim ctrl
Dim lIndex
Dim lCellIndex
Set ctrl = ScreenItems("UAControl")
Set coll = ctrl.GetRowCollection
'enumerate and trace out row numbers
For lIndex = 1 To coll.Count
 HMIRuntime.trace "Row: " & (ctrl.GetRow(lIndex).RowNumber) & "  "
 'enumerate and trace out column titles and cell texts
 For lCellIndex = 1 To ctrl.GetRow(lIndex).CellCount
  HMIRuntime.trace ctrl.GetRow(0).CellText(lCellIndex) & "  "
  HMIRuntime.trace ctrl.GetRow(lIndex).CellText(lCellIndex) & "  "
 Next
 HMIRuntime.trace vbNewLine
Next
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "Row" listing, for example, you write "objRow.CellCount" instead of "objRow.RowCellCount".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

AlarmControl (Page 5018)

GetRowCollection

Description

Returns the list of all row objects of the table-based controls type "ICCAxDataRowCollection".

Syntax

```
Expression.GetRowCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Properties of the ICCAxDataRowCollection

The ICCAxDataRowCollection refers to runtime data. The data is read-only. It is not possible to add and edit the data.

The following properties are available for the ICCAxDataRowCollection:

- Count - Determines the number of rows in the collection.
- Item - Access to an individual row within the collection via the row number. Numbering runs from 1 to Count. A Row object is returned.

Example

```
'VBS357
Dim ctrl
Dim coll
Dim lIndex
Dim lCellIndex
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetRowCollection
HMIRuntime.Trace "Number of message rows:" & coll.Count & vbCrLf
'enumerate and trace out row numbers
For lIndex = 1 To coll.Count
 HMIRuntime.Trace "Row: " & (ctrl.GetRow(lIndex).RowNumber) & "  "
 'enumerate and trace out column titles and cell texts
 For lCellIndex = 1 To ctrl.GetRow(lIndex).CellCount
  HMIRuntime.Trace ctrl.GetMessageColumn(lCellIndex -1).Name & " "
  HMIRuntime.Trace ctrl.GetRow(lIndex).CellText(lCellIndex) & "  "
 Next
 HMIRuntime.Trace vbNewLine
Next
```

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

AlarmControl (Page 5018)

GetRulerBlock

Description

Returns the block object of the evaluation table designated by name or index as "ICCAxRulerBlock" type.

Syntax

```
Expression.GetRulerBlock(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of the block in the evaluation table

Example

```
'VBS322
Dim ctrl
Dim objRulerBlock
Set ctrl = ScreenItems("RulerControl")
Set objRulerBlock = ctrl.GetRulerBlock(0)
objRulerBlock.Caption = "RulerBlock1"
Set objRulerBlock = ctrl.GetRulerBlock("Name")
objRulerBlock.Length = 10
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "RulerBlock" listing, for example, you write "objRulerBlock.Caption" instead of "objRulerBlock.BlockCaption".

See also

OnlineTrendControl (Page 5122)

TrendRulerControl (Page 5187)

GetRulerBlockCollection

Description

Returns the list of all block objects of the evaluation table as "ICCAxCollection" type.

Syntax

```
Expression.GetRulerBlockCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS323
Dim ctrl
Dim coll
Dim rulerblock
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetRulerBlockCollection
For Each rulerblock In coll
 rulerblock.Align = 1
 rulerblock.Length = 12
Next
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "RulerBlock" listing, for example, you write "rulerblock.Align" instead of "rulerblock.RulerBlockAlign".

See also

OnlineTrendControl (Page 5122)

TrendRulerControl (Page 5187)

GetRulerColumn

Description

Returns the column object of the evaluation table designated by name or index as "ICCAxRulerColumn" type.

Syntax

```
Expression.GetRulerColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of the column of the evaluation

Example

```
'VBS324
Dim ctrl
Dim objRulercol
Set ctrl = ScreenItems("RulerControl")
Set objRulercol = ctrl.GetRulerColumn("Name")
objRulercol.Sort = 0
Set objRulercol = ctrl.GetRulerColumn("ValueY")
objRulercol.Visible = FALSE
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "RulerColumn" listing, for example, you write "objRulercol.Visible" instead of "objRulercol.ColumnVisible".

See also

OnlineTrendControl (Page 5122)

TrendRulerControl (Page 5187)

GetRulerColumnCollection

Description

Returns the list of all column objects of the evaluation table as "ICCAxCollection" type.

Syntax

```
Expression.GetRulerColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS325
Dim ctrl
Dim coll
Dim rulercol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetRulerColumnCollection
HMIRuntime.Trace "Number of ruler columns:" & coll.Count & vbCrLf
For Each rulercol In coll
 HMIRuntime.Trace rulercol.Index & vbCrLf
 HMIRuntime.Trace rulercol.Name & vbCrLf
 HMIRuntime.Trace rulercol.Sort & vbCrLf
 HMIRuntime.Trace rulercol.SortIndex & vbCrLf
Next
```

See also

OnlineTrendControl (Page 5122)

TrendRulerControl (Page 5187)

GetRulerData

Description

Returns the value of the called trend at the ruler position.

Syntax

```
Expression.GetRulerData(ByVal RulerIndex As Long, pvValue As
Variant, Optional pvTimeStamp As Variant, Optional pvFlags As
Varian) Long
```

Expression

Necessary. An expression which returns an object of the "Trend" type.

Parameters

Parameters	Description
RulerIndex	0 =Ruler
pvValue	Value of X axis
pvTimeStamp	Time or value of the Y axis
pvFlags	Qualitycode

Example

```
'VBS326
Dim ctrl
Dim objTrend
Dim objIOField1
Dim objIOField2
Dim rulvalue
Dim rultime
Set ctrl = ScreenItems( "Control1" )
Set objTrend = ctrl.GetTrend( "Trend 1" )
Set objIOField1 = ScreenItems( "I/O Field1" )
Set objIOField2 = ScreenItems( "I/O Field2" )
objTrend.GetRulerData 0, rulvalue, rultime
objIOField1.OutputValue = rulvalue
objIOField2.OutputValue = rultime
```

See also

OnlineTrendControl (Page 5122)

GetSelectedRow

Description

Returns the selected row object of a table-based control as "ICCAxDataRow" type.

Syntax

```
Expression.GetSelectedRow()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Example

```
'VBS358
Dim ctrl
 Dim lCellIndex
 Dim lCellCount
 Dim headingRow
 Dim selectedRow
 Set ctrl = ScreenItems("TableControl")
 Set headingRow = ctrl.GetRow(0)
 Set selectedRow = ctrl.GetSelectedRow
 lCellCount = headingRow.CellCount
 'enumerate and trace out column titles and cell texts
 For lCellIndex = 1 To lCellCount
  HMIRuntime.trace headingRow.CellText(lCellIndex) & ":  "
  HMIRuntime.trace selectedRow.CellText(lCellIndex)
    HMIRuntime.trace vbNewLine
 Next
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "Row" listing, for example, you write "objRow.CellCount" instead of "objRow.RowCellCount".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

AlarmControl (Page 5018)

GetSelectedRows

Description

Returns the selected row objects of a table-based control as type "ICCAxDataRow" for multiple selection.

Syntax

```
Expression.GetSelectedRows()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Example

```
'VBS359
Dim ctrl
 Dim lCellIndex
 Dim lCellCount
 Dim lRowIndex
 Dim lRowCount
 Dim headingRow
 Dim selectedRow
 Dim selectedRows
 Set ctrl = ScreenItems("TableControl")
 Set headingRow = ctrl.GetRow(0)
 Set selectedRows = ctrl.GetSelectedRows
 lCellCount = headingRow.CellCount
 lRowCount = selectedRows.Count
 'enumerate selected rows
 For lRowIndex = 1 To lRowCount
  Set selectedRow = selectedRows(lRowIndex)
  HMIRuntime.Trace "Row number: " & CStr(lRowIndex) & vbNewLine
  'enumerate and trace out column titles and cell texts
  For lCellIndex = 1 To lCellCount
   HMIRuntime.trace headingRow.CellText(lCellIndex) & ":  "
   HMIRuntime.trace selectedRow.CellText(lCellIndex)
     HMIRuntime.trace vbNewLine
  Next
 Next
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "Row" listing, for example, you write "objRow.CellCount" instead of "objRow.RowCellCount".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

AlarmControl (Page 5018)

GetStatisticAreaColumn

Description

Returns the column object of the statistic area window of the evaluation window designated by name or index as "ICCAxRulerColumn" type.

Syntax

```
Ausdruck.GetStatisticAreaColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of column of statistics area window.

Example

```
'VBS327
Dim ctrl
Dim objStatAreaCol
Set ctrl = ScreenItems("RulerControl")
Set objStatAreaCol = ctrl.GetStatisticAreaColumn("DatasourceY")
objStatAreaCol.Visible = FALSE
Set objStatAreaCol = ctrl.GetStatisticAreaColumn("ValueY(LL)")
objStatAreaCol.Sort = 1
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "StatisticAreaColumn" listing, for example, you write "objStatAreaCol.Visible" instead of "objStatAreaCol.ColumnVisible".

See also

OnlineTrendControl (Page 5122)

TrendRulerControl (Page 5187)

GetStatisticAreaColumnCollection

Description

Returns the list of all column objects of the statistic area window of the evaluation table as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetStatisticAreaColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS328
Dim ctrl
Dim coll
Dim statcol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetStatisticAreaColumnCollection
HMIRuntime.Trace "Number of statistic Area columns:" & coll.Count & vbCrLf
For Each statcol In coll
 HMIRuntime.Trace statcol.Index & vbCrLf
 HMIRuntime.Trace statcol.Name & vbCrLf
 HMIRuntime.Trace statcol.Sort & vbCrLf
 HMIRuntime.Trace statcol.SortIndex & vbCrLf
Next
```

See also

OnlineTrendControl (Page 5122)

TrendRulerControl (Page 5187)

GetStatisticResultColumn

Description

Returns the column object of the statistic window of the evaluation window designated by name or index as "ICCAxRulerColumn" type.

Syntax

```
Ausdruck.GetStatisticResultColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of column of statistics window.

Example

```
'VBS329
Dim ctrl
Dim objStatResCol
Set ctrl = ScreenItems("RulerControl")
Set objStatResCol = ctrl.GetStatisticResultColumn("MaxValue")
objStatResCol.Visible = FALSE
Set objStatResCol = ctrl.GetStatisticResultColumn("Average")
objStatResCol.Sort = 2
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "StatisticResultColumn" listing, for example, you write "objStatResCol.Visible" instead of "objStatResCol.ColumnVisible".

See also

OnlineTrendControl (Page 5122)

TrendRulerControl (Page 5187)

GetStatisticResultColumnCollection

Description

Returns the list of all column objects of the statistic window of the evaluation table as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetStatisticResultColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS330
Dim ctrl
Dim coll
Dim statcol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetStatisticResultColumnCollection
HMIRuntime.Trace "Number of statistic result columns:" & coll.Count & vbCrLf
For Each statcol In coll
 HMIRuntime.Trace statcol.Index & vbCrLf
 HMIRuntime.Trace statcol.Name & vbCrLf
 HMIRuntime.Trace statcol.Sort & vbCrLf
 HMIRuntime.Trace statcol.SortIndex & vbCrLf
Next
```

See also

OnlineTrendControl (Page 5122)

TrendRulerControl (Page 5187)

GetStatusbarElement

Description

Returns the element of the control status bar designated as name or index as type "ICCAxStatusbarElement".

Syntax

```
Ausdruck.GetStatusbarElement(ByVal vIndex As Variant)
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of status bar element.

Example

```
'VBS331
Dim ctrl
Dim objStatusBar
Set ctrl = ScreenItems( "Control1" )
Set objStatusBar = ctrl.GetStatusbarElement(1)
objStatusBar.Visible = FALSE
Set objStatusBar = ctrl.GetStatusbarElement(3)
objStatusBar.Width = 10
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "StatusbarElement" listing, for example, you write "objStatusBar.Visible" instead of "objStatusBar.StatusbarElementVisible".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

GetStatusbarElementCollection

Description

Returns the list of all status bar elements of the control as type "ICCAxCollection".

Syntax

```
Ausdruck.GetStatusbarElementCollection()
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS332
Dim ctrl
Dim coll
Dim statelement
Set ctrl = ScreenItems.Item("Control1")
Set coll = ctrl.GetStatusbarElementCollection
HMIRuntime.Trace "Number of statusbar elements:" & coll.Count & vbCrLf
For Each statelement In coll
 HMIRuntime.Trace statelement.Name & vbCrLf
 HMIRuntime.Trace statelement.Width & vbCrLf
 HMIRuntime.Trace statelement.Text & vbCrLf
Next
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "StatusbarElement" listing, for example, you write "statelement.Name" instead of "statelement.StatusbarElementName".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

GetTimeAxis

Description

Returns the time object that is specified by name or index for the f(t) trend view as "ICCAxTimeAxis" type.

Syntax

```
Expression.GetTimeAxis(ByVal vIndex As Variant)
```

Expression

Necessary. Expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of time axis.

Example

```
'VBS333
Dim ctrl
Dim objTimeAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTimeAxis = ctrl.GetTimeAxis(1)
objTimeAxis.Visible = FALSE
Set objTimeAxis = ctrl.GetTimeAxis("axis 2")
objTimeAxis.Label = "Time axis 2"
objTimeAxis.DateFormat = "dd.MM.yy"
objTimeAxis.TimeFormat = "HH:mm:ss.ms"
objTimeAxis.RangeType = 2
'The format to be used for date and time entries depends on the
'regional settings and language options in the operating system.
objTimeAxis.BeginTime = "06.04.2010 9:33:18"
'objTimeAxis.BeginTime = "04/06/2010 9:33:18"
objTimeAxis.MeasurePoints = 100
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "TimeAxis" listing, for example, you write "objTimeAx.Visible" instead of "objTimeAx.TimeAxisVisible".

See also

OnlineTrendControl (Page 5122)

GetTimeAxisCollection

Description

Returns a list of all time objects of the f(t) trend view as "ICCAxCollection" type.

Syntax

```
Expression.GetTimeAxisCollection()
```

Expression

Necessary. Expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for ICCAxCollection:

- Count
- Item

The following functions are available for ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS334
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis1
Dim objTimeAxis2
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis1 = ctrl.GetTimeAxisCollection.AddItem("TimeAxis2010")
Set objTimeAxis2 = ctrl.GetTimeAxisCollection.AddItem("TimeAxis2011")
objTimeAxis1.TrendWindow = objTrendWnd.Name
objTimeAxis1.Label = "2010"
objTimeAxis1.RangeType = 1
'The format to be used for date and time entries depends on the
'regional settings and language options in the operating system.
objTimeAxis1.BeginTime = "01.01.2010 0:00:00"
'objTimeAxis1.BeginTime = "01/01/2010 0:00:00"
objTimeAxis1.EndTime = "31.12.2010 11:59:59"
'objTimeAxis1.EndTime = "12/31/2010 11:59:59"
objTimeAxis2.TrendWindow = objTrendWnd.Name
objTimeAxis2.Label = "2011"
objTimeAxis2.RangeType = 1
objTimeAxis2.BeginTime = "01.01.2011 0:00:00"
'objTimeAxis2.BeginTime = "01/01/2011 0:00:00"
objTimeAxis2.EndTime = "31.12.2011 11:59:59"
'objTimeAxis2.EndTime = "12/31/2011 11:59:59"
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis1.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend2")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis2.Name
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "TimeAxis" listing, for example, you write "objTimeAxis1.Label" instead of "objTimeAxis1.TimeAxisLabel".

See also

OnlineTrendControl (Page 5122)

GetTimeColumn

Description

Returns the time column object of the table view designated by name or index as "ICCAxTime Column" type.

Syntax

```
Ausdruck.GetTimeColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of time column.

Example

```
'VBS335
Dim ctrl
Dim objTimeCol
Set ctrl = ScreenItems("TableControl")
Set objTimeCol = ctrl.GetTimeColumn("Timecolumn1")
objTimeCol.ShowDate = FALSE
Set objTimeCol = ctrl.GetTimeColumn("Timecolumn2")
objTimeCol.Visible = FALSE
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "TimeColumn" listing, for example, you write "objTimeColumn.ShowDate" instead of "objTimeColumn.TimeColumnShowDate".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

GetTimeColumnCollection

Description

Returns the list of all time column objects of the table view as "ICCAxCollection" type.

Syntax

```
Expression.GetTimeColumnCollection()
```

Expression

Necessary. Expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for ICCAxCollection:

- Count
- Item

The following functions are available for ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS336
Dim ctrl
Dim objTimeCol1
Dim objTimeCol2
Dim coll
Dim timecol
Set ctrl = ScreenItems("TableControl")
Set objTimeCol1 = ctrl.GetTimeColumnCollection.AddItem("TimeColumn2010")
Set objTimeCol2 = ctrl.GetTimeColumnCollection.AddItem("TimeColumn2011")
objTimeCol1.Caption = "2010"
objTimeCol1.RangeType = 1
'The format to be used for date and time entries depends on the
'regional settings and language options in the operating system.
objTimeCol1.BeginTime = "01.01.2010 0:00:00"
'objTimeCol1.BeginTime = "01/01/2010 0:00:00"
objTimeCol1.EndTime = "31.12.2010 11:59:59"
'objTimeCol1.EndTime = "12/31/2010 11:59:59"
objTimeCol2.Caption = "2011"
objTimeCol2.RangeType = 0
objTimeCol2.BeginTime = "01.01.2011 0:00:00"
'objTimeCol2.BeginTime = "01/01/2011 0:00:00"
objTimeCol2.TimeRangeFactor = 1
objTimeCol2.TimeRangeBase = 3600000
Set coll = ctrl.GetTimeColumnCollection
For Each timecol In coll
 timecol.Align = 1
 timecol.Length = 12
 timecol.BackColor = RGB(240,240,0)
 timecol.ForeColor = RGB(130,160,255)
Next
```

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

GetToolbarButton

Description

Returns the button function designated by name or index on the control toolbar as "ICCAxToolbarButton" type.

Syntax

```
Ausdruck.GetToolbarButton(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of toolbar button function.

Example

```
'VBS337
Dim ctrl
Set ctrl = ScreenItems( "Control1" )
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "ToolbarButton" listing, for example, you write "toolbu.Index" instead of "toolbu.ToolbarButtonIndex".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

GetToolbarButtonCollection

Description

Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetToolbarButtonCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following methods are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS338
Dim ctrl
Dim coll
Dim toolbu
Set ctrl = ScreenItems( "Control1" )
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
 HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
 HMIRuntime.Trace "Authorization: " & toolbu.PasswordLevel & vbCrLf
Next
```

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

GetTrend

Description

Returns the f(t) or f(x) trend view designated by name or index as "ICCAxTrend" or "ICCAxFunctionTrend" type.

Syntax

```
Ausdruck.GetTrend(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of curve.

Example for Runtime Professional

```
'VBS339
Dim ctrl
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrend = ctrl.GetTrend( "Trend 1" )
objTrend.PointStyle = 1
objTrend.LineWidth = 4
Set objTrend = ctrl.GetTrend(2)
objTrend.Provider = 1
objTrend.TagName = "Archive\ArchiveTag2"
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "Trend" listing, for example, you write "objTrend.PointStyle" instead of "objTrend.TrendPointStyle".

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

GetTrendCollection

Description

Returns the list of all trends of the f(t) or f(x) trend view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetTrendCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example for Runtime Professional

```
'VBS340
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis
Dim objValAxis
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = ctrl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValAxis = ctrl.GetValueAxisCollection.AddItem("myValueAxis")
objTimeAxis.TrendWindow = objTrendWnd.Name
objValAxis.TrendWindow = objTrendWnd.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.Provider = 1
objTrend.TagName = "Archive\ArchiveTag1"
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValAxis.Name
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "Trend" listing, for example, you write "objTrend.TagName" instead of "objTrend.TrendTagName".

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

GetTrendWindow

Description

Returns the trend view object of the f(t) trend view or the f(x) trend view designated by name or index as "ICCAxTrendWindow" type.

Syntax

```
Ausdruck.GetTrendWindow(ByVal vIndex As Variant)
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of curve window.

Example for Runtime Professional

```
'VBS341
Dim ctrl
Dim objTrendWnd
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindow(1)
objTrendWnd.Visible = FALSE
Set objTrendWnd = ctrl.GetTrendWindow("trend window 2")
objTrendWnd.VerticalGrid = TRUE
objTrendWnd.FineGrid = TRUE
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "TrendWindow" listing, for example, you write "objTrendWnd.Visible" instead of "objTrendWnd.TrendWindowVisible".

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

GetTrendWindowCollection

Description

Returns the list of all trend window objects of the f(t) trend display or the f(x) trend display as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetTrendWindowCollection()
```

Expression

Required. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example for Runtime Professional

```
'VBS342
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis
Dim objValAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = ctrl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValAxis = ctrl.GetValueAxisCollection.AddItem("myValueAxis")
objTimeAxis.TrendWindow = objTrendWnd.Name
objValAxis.TrendWindow = objTrendWnd.Name
```

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

GetValueAxis

Description

Returns the value axis object of the f(t) trend view designated by name or index as "ICCAxValueAxis" type.

Syntax

```
Ausdruck.GetValueAxis(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of the value axis

Example

```
'VBS343
Dim ctrl
Dim objValAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objValAxis = ctrl.GetValueAxis(1)
objValAxis.Visible = FALSE
Set objValAxis = ctrl.GetValueAxis("axis 2")
objValAxis.Label = "Value axis 2"
objValAxis.ScalingType = 0
objValAxis.Precisions = 2
objValAxis.AutoRange = TRUE
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "ValueAxis" listing, for example, you write "objValueAx.Visible" instead of "objValueAx.ValueAxisVisible".

See also

OnlineTrendControl (Page 5122)

GetValueAxisCollection

Description

Returns the list of all value axis objects of the f(t) trend view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetValueAxisCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS344
Dim ctrl
Dim objTrendWnd
Dim objValAxis1
Dim objValAxis2
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objValAxis1 = ctrl.GetValueAxisCollection.AddItem("myValueAxis1")
Set objValAxis2 = ctrl.GetValueAxisCollection.AddItem("myValueAxis2")
objValAxis1.TrendWindow = objTrendWnd.Name
objValAxis1.Label = "Value1"
objValAxis2.TrendWindow = objTrendWnd.Name
objValAxis2.inTrendColor = TRUE
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.ValueAxis = objValAxis1.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend2")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.ValueAxis = objValAxis2.Name
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "ValueAxis" listing, for example, you write "objValueAxis1.Label" instead of "objValueAxis1.ValueAxisLabel".

See also

OnlineTrendControl (Page 5122)

GetValueColumn

Description

Returns the value column object defined by name or index for the tabular view as "ICCAxValueColumn" type.

Syntax

```
Expression.GetValueColumn(ByVal vIndex As Variant)
```

Expression

Necessary. Expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of the value column of the f(t) trend view.

Example

```
'VBS345
Dim ctrl
Dim objValueColumn
Set ctrl = ScreenItems("TableControl")
Set objValueColumn = ctrl.GetValueColumn("Valuecolumn1")
objValueColumn.Precisions = 4
Set objValueColumn = ctrl.GetValueColumn(2)
objValueColumn.ExponentialFormat = TRUE
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "ValueColumn" listing, for example, you write "objValueColumn.Precisions" instead of "objValueColumn.ValueColumnPrecisions".

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

GetValueColumnCollection

Description

Returns the list of all value column objects of the f(t) trend view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetValueColulmnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS346
Dim ctrl
Dim objValCol1
Dim objValCol2
Dim coll
Dim valcol
Set ctrl = ScreenItems("TableControl")
Set objValCol1 = ctrl.GetValueColumnCollection.AddItem("ValueColumn1")
Set objValCol2 = ctrl.GetValueColumnCollection.AddItem("ValueColumn2")
objValCol1.Caption = "Value Archive"
objValCol1.Provider = 1
objValCol1.TagName = "ProcessValueArchive\arch1"
objValCol1.TimeColumn = "TimeColumn1"
objValCol2.Caption = "Value Tag"
objValCol2.Provider = 2
objValCol2.TagName = "tagxx"
objValCol2.TimeColumn = "TimeColumn2"
Set coll = ctrl.GetValueColumnCollection
For Each valcol In coll
 valcol.Align = 2
 valcol.Length = 10
 valcol.AutoPrecisions = TRUE
Next
```

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

GetXAxis

Description

Returns the X axis object of the f(x) trend view designated by name or index as "ICCAxValueAxis" type.

Syntax

```
Ausdruck.GetXAxis(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of X axis.

Example

```
'VBS347
Dim ctrl
Dim objXAx
Set ctrl = ScreenItems("FunctionTrendControl")
Set objXAx = ctrl.GetXAxis(1)
objXAx.Visible = FALSE
Set objXAx = ctrl.GetXAxis("axis 2")
objXAx.Label = "X axis 2"
objXAx.ScalingType = 0
objXAx.Precisions = 2
objXAx.Color = RGB(109,109,109)
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "XAxis" listing, for example, you write "objXAx.Visible" instead of "objXAx.XAxisVisible".

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

GetXAxisCollection

Description

Returns the list of all X axis objects of the f(x) trend view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetXAxisCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS348
Dim ctrl
Dim objXAxis1
Dim objXAxis2
Dim coll
Dim axes
Set ctrl = ScreenItems("FunctionTrendControl")
Set objXAxis1 = ctrl.GetXAxisCollection.AddItem("myXAxis1")
objXAxis1.Label = "temperature"
Set objXAxis2 = ctrl.GetXAxisCollection.AddItem("myXAxis2")
objXAxis2.Label = "pressure"
Set coll = ctrl.GetXAxisCollection
HMIRuntime.Trace "Number of XAxis:" & coll.Count & vbCrLf
For Each axes In coll
 HMIRuntime.Trace axes.Name & vbCrLf
 HMIRuntime.Trace axes.Label & vbCrLf
Next
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "XAxis" listing, for example, you write "objXAxis1.Label" instead of "objXAxis1.XAxisLabel".

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

GetYAxis

Description

Returns the Y axis object of the f(x) trend view designated by name or index as "ICCAxValueAxis" type.

Syntax

```
Ausdruck.GetYAxis(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of Y axis.

Example

```
'VBS349
Dim ctrl
Dim objYAx
Set ctrl = ScreenItems("FunctionTrendControl")
Set objYAx = ctrl.GetYAxis(1)
objYAx.Visible = FALSE
Set objYAx = ctrl.GetYAxis("axis 2")
objYAx.Label = "Y axis 2"
objYAx.Align = 0
objYAx.Precisions = 3
objYAx.EndValue = 90.000
objYAx.BeginValue = 10.000
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "YAxis" listing, for example, you write "objYAx.Visible" instead of "objYAx.YAxisVisible".

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

GetYAxisCollection

Description

Returns the list of all Y axis objects of the f(x) trend view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetYAxisCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS350
Dim ctrl
Dim objYAxis1
Dim objYAxis2
Dim coll
Dim axes
Set ctrl = ScreenItems("FunctionTrendControl")
Set objYAxis1 = ctrl.GetXAxisCollection.AddItem("myYAxis1")
objYAxis1.Label = "temperature"
Set objYAxis2 = ctrl.GetXAxisCollection.AddItem("myYAxis2")
objYAxis2.Label = "pressure"
Set coll = ctrl.GetYAxisCollection
HMIRuntime.Trace "Number of YAxis:" & coll.Count & vbCrLf
For Each axes In coll
 HMIRuntime.Trace axes.Name & vbCrLf
 HMIRuntime.Trace axes.Label & vbCrLf
Next
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "YAxis" listing, for example, you write "objYAxis1.Label" instead of "objYAxis1.YAxisLabel".

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

Methods H-R

HideAlarm

Description

Executes the "Hide alarm" button function of the alarm view.

Syntax

```
Expression.HideAlarm()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

InsertData

Description

Adds data to the called trend.

Syntax

```
Expression.InsertData(dblAxisX As Variant, dblAxisY As Variant)
```

Expression

Necessary. An expression which returns an object of the "Trend" type.

Parameters

Parameters	Description
dblAxisX	Value of X axis
dblAxisY	Value of Y axis

Example

```
'VBS300
Dim lngFactor
Dim dblAxisX
Dim dblAxisY
Dim objTrendControl
Dim objTrend
Set objTrendControl = ScreenItems("Control1")
Set objTrend = objTrendControl.GetTrend("Trend 1")
For lngFactor = -100 To 100
dblAxisX = CDbl(lngFactor * 0.02)
dblAxisY = CDbl(dblAxisX * dblAxisX + 2 * dblAxisX + 1)
objTrend.InsertData dblAxisX, dblAxisY
Next
```

Item

Description

Returns an element from a list.

Syntax

Expression.Item(Index)

Expression

Necessary. An expression which returns a list.

Parameters

Index

The name or the index number of an element of the list:

- ScreenItems-Listing: Use an object name, e.g. "HmiRuntime.Screens(1).ScreenItems("Circle")", or the index number.
- Screens-Listing: Use either the name or the index number.
- SmartTags-Listing: You can only use the tag name as index in the SmartTags list. A counting of all tags is not possible.

If the transferred value does not correspond with an element in the list, an error occurs. The return value has the value "Nothing".

```
On Error Resume Next
Dim screen
Set screen = HmiRuntime.Screens("Screen_1")
If (screen is Nothing)
then...
Else...
End If
```

Best practice for optimizing auto-completion support is to use combined addressing by means of screen name and object name, e.g. `"HmiRuntime.Screens("Screen").ScreenItems("Circle")"`.

Example

The item method is the default method for lists. Therefore, the results are the same for the following two examples:

```
'VBS_Example_Item
HMIRuntime.Screens.Item(1)
HMIRuntime.Screens(1)
```

Both instructions reference the respective base screen.

See also

ScreenItems (list) (Page 5005)

ScreenItem (Page 5003)

LockAlarm

Description

Executes the "Disable alarm" button function of the alarm view.

Syntax

```
Expression.LockAlarm()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

--

See also

AlarmControl (Page 5018)

LoopInAlarm

Description

Executes the "Loop in alarm" button function of the alarm view.

Syntax

```
Expression.LoopInAlarm()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

--

See also

AlarmControl (Page 5018)

MoveAxis

Description

Executes the "Move axes area" button function of the f(t) and f(x) trend views.

Syntax

```
Expression.MoveAxis()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

MoveToFirst

Description

Executes the "First line" button function of the control.

Syntax

```
Expression.MoveToFirst()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

UserArchiveControl (Page 5207)

MoveToFirstLine

Description

Executes the "First alarm" button function of the alarm view.

Syntax

```
Ausdruck.MoveToFirstLine()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

MoveToFirstPage

Description

Executes the "First page" button function of the alarm view.

Syntax

```
Ausdruck.MoveToFirstPage()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

MoveToLast

Description

Executes the "last data record" button function of the control.

Syntax

```
Ausdruck.MoveToLast()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

UserArchiveControl (Page 5207)

MoveToLastLine

Description

Executes the "Last alarm" button function of the alarm view.

Syntax

```
Ausdruck.MoveToLastLine()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

AlarmControl (Page 5018)

MoveToLastPage

Description

Executes the "Last page" button function of the alarm view.

Syntax

```
Ausdruck.MoveToLastPage()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

MoveToNext

Description

Executes the "next data record" button function of the control.

Syntax

```
Ausdruck.MoveToNext()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

UserArchiveControl (Page 5207)

MoveToNextLine

Description

Executes the "Next alarm" button function of the alarm view.

Syntax

```
Ausdruck.MoveToNextLine()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

MoveToNextPage

Description

Executes the "Next page" button function of the alarm view.

Syntax

```
Ausdruck.MoveToNextPage()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

MoveToPrevious

Description

Executes the "previous data record" button function of the control.

Syntax

```
Ausdruck.MoveToPrevious()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

UserArchiveControl (Page 5207)

MoveToPreviousLine

Description

Executes the "Previous alarm" button function of the alarm view.

Syntax

```
Ausdruck.MoveToPreviousLine()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

--

See also

AlarmControl (Page 5018)

MoveToPreviousPage

Description

Executes the "Previous page" button function of the alarm view.

syntax

```
Ausdruck.MoveToPreviousPage()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

NextColumn

Description

Executes the "Next column" button function of the table view.

Syntax

```
Ausdruck.NextColumn()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

OnlineTableControl (Page 5114)

NextTrend

Description

Executes the "Next trend" button function of the f(t) and f(x) trend views.

Syntax

```
Ausdruck.NextTrend()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

OneToOneView

Description

Executes the "Original view" button function of the f(t) and f(x) trend views.

Syntax

```
Ausdruck.OneToOneView()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

PasteRows

Description

Executes the "Insert rows" button function of the recipe view.

Syntax

```
Expression.PasteRows()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

UserArchiveControl (Page 5207)

PreviousColumn

Description

Executes the "Previous column" button function of the table view.

Syntax

```
Ausdruck.PreviousColumn()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

OnlineTableControl (Page 5114)

PreviousTrend

Description

Executes the "Previous trend" button function of the f(t) and f(x) trend views.

Syntax

```
Ausdruck.PreviousTrend()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

Print

Description

Executes the "Print" button function of the control.

Syntax

```
Ausdruck.Print()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

QuitHorn

Description

Executes the "Acknowledge central alarm generator" button function of the alarm view.

Syntax

```
Ausdruck.QuitHorn()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

QuitSelected

Description

Executes the "Single acknowledgment" button function of the alarm view.

Syntax

```
Ausdruck.QuitSelected()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

QuitVisible

Description

Executes the "Group acknowledgment" button function of the alarm view.

Syntax

```
Ausdruck.QuitVisible()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ReadTags

Description

Executes the "Read tags" button function of the recipe view.

Syntax

```
Ausdruck.ReadTags()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

UserArchiveControl (Page 5207)

Read

Description of Tag Object

Reads out the status of a tag (tag object) shortly after the moment it was called. At the same time, the tag object is provided with the values read. Upon reading a tag, its value, quality code and time stamp are determined. The "LastError" property can be used to determine whether the call was successful.

The "Name" and "Tagprefix" properties are not changed by this.

If the value of the tag is read successfully, the properties of the tag object are assigned the following values:

Property	Allocation
Value	Tag values
Name	Tag name (unchanged)
QualityCode	Quality level
Timestamp	Current tag time stamp
LastError	0
ErrorDescription	" "

If the value of the tag is not read successfully, the properties of the tag object are assigned the following values:

Property	Allocation
Value	VT_Empty
Name	Tag name (unchanged)
QualityCode	Bad Out of Service
Timestamp	0
LastError	Read operation error codes
ErrorDescription	Error description on LastError

syntax

```
Expression.Read([Readmode])
```

Expression

Necessary. An expression which returns a tag object. The return value of the Read method is the value of the tag read out.

Parameters

The optional "Readmode" parameter enables the distinction between two types of reading:

Parameters	Description
0	The tag value is read from the process image (cache). 0 is the default value.
1	The value of a tag is read directly from AS or channel (direct).

If the "Readmode" parameter is omitted, the value is read from the process image by default. The return value of the Read method is the tag value read out as VARIANT.

Reading From the Process Image

When reading from the process image, the tag is logged on and, from that moment, polled cyclically from the PLC. The login cycle is dependent on the configured trigger. The value is read from the tag image by WinCC. For Close Picture, the tag actions are ended again. The call is characterized by the following:

- The value is read by WinCC from the tag image.
- The call is faster in comparison to direct reading (except with the first call: The first call basically takes longer because the value from the PLC must be read out and logged on.)
- The duration of the call is not dependent on the bus load or AS.

Behavior in actions with a tag trigger

All of the tags contained in the tag trigger are already known with Open Picture and are registered with the defined monitoring time. Since all tags are requested at once, the best possible optimization can be targeted from the channel. If a tag, contained in the trigger, is requested with Read during an action, the value already exists and is transferred directly to the call. If a tag is requested which is not contained in the trigger, the behavior is the same as with a standard trigger.

Behavior in actions with a cyclic trigger

tags are registered with half of the cycle time with the first call. For every other call, the value is present.

Behavior in event-driven actions

The tag is registered in the "upon change" mode with the first call. Process tags that are registered in the "upon change" mode correspond with a cyclic read job with a cycle time of 1s.

If an event (e.g. mouse click) requests a value asynchronously, the tag is transferred to the tag image. The tag is requested cyclically from the AS as of this point in time and therefore increases the basic load. To bypass this increase in the basic load, the value can also be read synchronously. The synchronous call causes a one-off increase in the communication load but the tag is not transferred to the tag image.

Direct reading

In the case of direct reading, the current value is returned. The tag is not registered cyclically, the value is requested from the AS one time only. Direct reading has the following properties:

- The value is read explicitly from the AS.
- The call takes longer compared to reading from the process image.
- The duration of the call is dependent on the bus load and AS, amongst other things.

Example

Reading a tag directly from AS or channel

```
'VBS100
Dim objTag
Dim vntValue
Set objTag = HMIRuntime.Tags("Tagname")
vntValue = objTag.Read(1)    'Read direct
MsgBox vntValue
```

Reading a tag from the process image

```
'VBS101
Dim objTag
Dim vntValue
Set objTag = HMIRuntime.Tags("Tagname")
vntValue = objTag.Read    'Read from cache
MsgBox vntValue
```

Description of TagSet Object

The TagSet object offers the option of reading several tags in one call.

Functionality here is mostly identical with that of a tag object. In the following, only deviations thereof are described.

Expression

Necessary. An expression which returns an object of type "TagSet".

Reading From the Process Image

The TagSet object offers the advantage of requesting several tags in one read command. The tags are registered in the process image as a group, improving performance in the process.

Direct reading

Since one call may process several read commands, performance is enhanced in comparison to single calls.

Example

The following example shows how tags are included in the TagSet list, how tag values are imported and subsequently read.

```
'VBS174
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.Read
HMIRuntime.Trace "Motor1: " & group("Motor1").Value & vbNewLine
HMIRuntime.Trace "Motor2: " & group("Motor2").Value & vbNewLine
```

If the optional parameter "Readmode" is set to 1, the process tags are not registered but read directly from AS or channel.

```
group.Read 1
```

Refresh

Description

Drawing all visible pictures again.

syntax

```
Expression.Refresh
```

Expression

Necessary. An expression which returns a "Screens" or "Screen" type object.

Parameter

--

Examples

The first example forces all visible pictures to be drawn again:

```
'VBS149
HMIRuntime.Screens.Refresh
```

The second example forces the basic picture to be immediately redrawn:

```
'VBS150
HMIRuntime.Screens(1).Refresh
```

See also

Screen (Page 5000)

HMIRuntime (Page 4993)

Remove

Description of TagSet Object

Removes a tag from the TagSet list. The tag may be removed by name or reference to a tag object.

Syntax

```
Expression.Remove [Tag]
```

Expression

Required An expression that returns an object of type "TagSet".

Parameters

VARIANT

Parameters	Description
Tag	Name of a WinCC tag or reference to a tag object to be removed from the list.

Example

The following example shows how several tags are included in the TagSet list, and how to remove a tag again.

```
'VBS175
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.Remove "Motor1"
```

Description of DataSet Object

Deletes the element specified in parameter "Name" from a list.

Syntax

```
Expression.Remove [Name]
```

Expression

Required An expression which returns an object of type "DataSet".

Parameters

VARIANT

Parameters	Description
Name	Name of the object to be removed from the list.

Example

The example shows how to remove the object "motor1" from the list.

```
'VBS166
HMIRuntime.DataSet.Remove("motor1")
```

Description of objects Logging, AlarmLogs, DataLogs

The method deletes previously swapped-in log segments from the Runtime project.

Log segments that were deleted using the "Remove" method are removed from the "Common logging" folder of the project.

The call may require a somewhat longer time period, depending on the log data. This may block the processing of subsequent scripts. Blockage of actions within the picture may be avoided if you start the call in a Global Scripting action, such as starting the action through a trigger tag.

The process of disconnecting and clearing logs creates a CPU load. This will affect performance.

Note

Calling up the "Remove" method is presently only possible at the server. There is an example, however, which shows how the method may be started by the client from a server.

Syntax

Objects Logging, AlarmLogs

```
Expression.Remove [TimeFrom] [TimeTo] [TimeOut] [ServerPrefix]
```

Expression

Required An expression that returns an object of type "Logging" or "AlarmLogs".

Object DataLogs

```
Expression.Remove [TimeFrom] [TimeTo] [TimeOut] [Type] [ServerPrefix]
```

Expression

Required An expression that returns an object of type "DataLogs".

Parameters

TimeFrom

Point in time, from which the logs are to be cleared.
When specifying the time information, a short form is also possible. This is described in the "Time format" section.

TimeTo

Time up to which log segments are to be cleared.
When specifying the time information, a short form is also possible. This is described in the "Time Format" section.

Timeout

Timeout in milliseconds.
If you enter "-1" as a value, the wait will be infinite. If you enter a value of "0", there will be no wait.

Type

Type of log.
The parameter can (optionally) be used only to delete log segments of the tag logging.
The following values can be entered:

Assigned value	Type	Description
1	hmiDataLogFast	Tag Logging Fast data
2	hmiDataLogSlow	Tag Logging Slow data
3	hmiDataLogAll	Tag Logging Fast and Slow data

ServerPrefix

Reserved for future versions.

Return value

If an error occurred during deletion of the log segments, the method will return an error alarm. Additional information may be found under the subject heading "Error alarms from database area".

Time format

The format for specifying time information is defined as follows: YYYY-MM-DD hh:mm:ss, where YYYY represents the year, MM the month, DD the day, hh the hour, mm the minute and ss the second. For example, the time of 2 minutes and one second past 11 o'clock on July 26, 2004 is displayed as follows: 2004-07-26 11:02:01.

For parameters "TimeFrom" and "TimeTo" the statement of data and time is also possible in short form. Not all format fields must be filled in this case. The short form means that the information on date and time may be lacking one or several parameters, beginning with the value for seconds. For example, the time may be specified in the "YYYY-MM" or "YYYY-MM-DD hh" format. Using the statement "TimeFrom" = "2004-09" and "TimeTo" = "2004-10-04" all log segments between September 2004 up to and including October 4.

Example

In the following example, log segments within a certain time period that were swapped in (again) after the fact are removed and the return value is output as a trace.

```
'VBS182
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Remove("2004-08-22","2004-09-22",-1) &
vbNewLine
```

In the following example, all log segments that were swapped-in (again) after the fact are removed and the return value is output as a trace.

```
'VBS183
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Remove("","",-1) & vbNewLine
```

See also

Logging (Page 4998)

DataSet (list) (Page 4990)

DataLogs (list) (Page 4989)

AlarmLogs (list) (Page 4986)

RemoveAll

Description of TagSet Object

Deletes all tags from a TagSet list.

syntax

```
Expression.RemoveAll
```

Expression

Necessary. An expression which returns an object of type "TagSet".

Parameters

--

Example:

The following example shows how several tags are included in the TagSet list, and how to remove all tags again.

```
'VBS176
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.RemoveAll
```

Description of DataSet Object

Deletes all values or object references from a DataSet list.

syntax

```
Expression.RemoveAll
```

Expression

Necessary. An expression which returns an object of type "DataSet".

Parameters

--

Example:

The example shows how all objects are removed from the list.

```
'VBS167
HMIRuntime.DataSet.RemoveAll
```

See also

DataSet (list) (Page 4990)

Restore

Description of objects Logging, AlarmLogs, DataLogs

The method adds swapped-out log segments to the Runtime project.

Upon swapping-in, the log segments are copied to the "Common logging" folder of the project. Therefore the appropriate storage capacity must be available.

The call may require a somewhat longer time period, depending on the log data. This may block the processing of subsequent scripts. Blockage of actions within the picture may be avoided if you start the call in a Global Scripting action, such as starting the action through a trigger tag.

The connecting/copying of the logs generates a CPU load because the SQL server experiences additional load, especially if signature checking is activated. Copying of log segments will slow down hard disk access.

If signature checking is activated, an error message is returned if an unsigned or modified log is to be swapped in. There is always only one error alarm returned, even if several errors occurred during a swap-in process. Additionally, a WinCC system alarm is generated for each log segment. An entry is added to the Windows event viewer in the "Application" section. This provides the opportunity to check which log segments are creating the error.

- With an unsigned log, the return value "0x8004720F" is returned. The event viewer contains the entry "Validation of database <db_name> failed! No signature found!".
  The log is swapped in.
- With a changed log, the return value "0x80047207" is returned. The event viewer contains the entry "Validation of database <db_name> failed!".
  The log is not swapped in.

Note

Calling up the "Restore" method is presently only possible at the server. There is an example, however, which shows how the method may be started by the client from a server.

Syntax

Objects Logging, AlarmLogs

```
Expression.Restore [SourcePath] [TimeFrom] [TimeTo] [TimeOut]
[ServerPrefix]
```

Expression

Required An expression that returns an object of type "Logging" or "AlarmLogs".

Object DataLogs

```
Expression.Restore [SourcePath] [TimeFrom] [TimeTo] [TimeOut] [Type]
[ServerPrefix]
```

Expression

Required An expression that returns an object of type "DataLogs".

Parameters

SourcePath

Path to log data.

TimeFrom

Point in time, from which the logs are to be swapped in.
When specifying the time information, a short form is also possible. This is described in the "Time format" section.

TimeTo

Time up to which log segments are to be swapped in.
When specifying the time information, a short form is also possible. This is described in the "Time Format" section.

Timeout

Timeout in milliseconds.
If you enter "-1" as a value, the wait will be infinite. If you enter a value of "0", there will be no wait.

Type

Type of log.
The parameter can (optionally) be used only to store log segments of the tag logging.
The following values can be entered:

Assigned value	Type	Description
1	hmiDataLogFast	Tag Logging Fast data
2	hmiDataLogSlow	Tag Logging Slow data
3	hmiDataLogAll	Tag Logging Fast and Slow data

ServerPrefix

Reserved for future versions.

Return value

If an error occurred while swapping in log segments, the method will return an error message. Additional information may be found under the subject heading "Error alarms from database area".

Time format

The format for specifying time information is defined as follows: YYYY-MM-DD hh:mm:ss, where YYYY represents the year, MM the month, DD the day, hh the hour, mm the minute and ss the second. For example, the time of 2 minutes and one second past 11 o'clock on July 26, 2004 is displayed as follows: 2004-07-26 11:02:01.

For parameters "TimeFrom" and "TimeTo" the statement of data and time is also possible in short form. Not all format fields must be filled in this case. The short form means that the information on date and time may be lacking one or several parameters, beginning with the value for seconds. For example, the statement may be in the form of "YYYY-MM" or "YYYY-MM-DD hh". Using the statement "TimeFrom" = "2004-09" and "TimeTo" = "2004-10-04" all log segments between September 2004 up to and including October 4.

Example

In the following example, all log segments since the start of the specified time period are swapped in again, and the return value is output as a trace.

```
'VBS184
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Restore("D:\Folder","2004-09-14","",-1) &
vbNewLine
```

In the following example, all Tag Logging Slow log segments in the specified time period are swapped in again, and the return value is output as a trace.

```
'VBS185
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.DataLogs.Restore("D:\Folder","2004-09-14
12:30:05","2004-09-20 18:30",-1,2) & vbNewLine
```

In the following example, all Alarm Logging log segments up to the specified time period are swapped in again, and the return value is output as a trace.

```
'VBS186
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.AlarmLogs.Remove("","2004-09-20",-1) &
vbNewLine
```

See also

Logging (Page 4998)

DataLogs (list) (Page 4989)

AlarmLogs (list) (Page 4986)

Methods S-Z

SelectAll

Description

Selects all rows in a table-based control.

Syntax

```
Expression.SelectAll()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

AlarmControl (Page 5018)

SelectedStatisticArea

Description

Executes the "Set statistics range" button function of the table view.

Syntax

```
Ausdruck.SelectedStatisticArea()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

OnlineTableControl (Page 5114)

SelectRow

Description

Selects a specific row in a table-based control.

Syntax

```
Expression.SelectRow(ByVal IRow As Long, Optional bExtendSelection
As Boolean)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

Parameters	Description
IRow	Number of the row to be selected.
bExtendSelection	Indicates as an option whether the current selection will be extended. Is only relevant if multiple selections are possible.

Example

- Row 1 is currently selected. If SelectRow(2, True) is called, then row 1 and row 2 will be selected.
- Row 1 is currently selected. If SelectRow(2, False) or SelectRow(2) is called without an optional parameter, then only row 2 will be selected.

See also

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

AlarmControl (Page 5018)

ServerExport

Description

Executes the "Export log" button function of the recipe view.

Syntax

```
Ausdruck.ServerExport()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

UserArchiveControl (Page 5207)

ServerImport

Description

Executes the "Import log" button function of the recipe view.

Syntax

```
Ausdruck.ServerImport()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

UserArchiveControl (Page 5207)

ShowColumnSelection

Description

Executes the "Select columns" button function of the table view.

Syntax

```
Ausdruck.ShowColumnSelection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

--

See also

OnlineTableControl (Page 5114)

ShowComment

Description

Executes the "Comment dialog" button function of the alarm view.

Syntax

```
Ausdruck.ShowComment()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

See also

AlarmControl (Page 5018)

ShowDisplayOptionsDialog

Description

Executes the "Display options dialog" button function of the alarm view.

Syntax

```
Ausdruck.ShowDisplayOptionsDialog()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowEmergencyQuitDialog

Description

Executes the "Single acknowledgment" button function of the alarm view.

Syntax

```
Ausdruck.ShowEmergencyQuitDialog()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowHelp

Description

Executes the "Help" button function of the control.

Syntax

```
Ausdruck.ShowHelp()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

ShowHideList

Description

Executes the "List of alarm to hide" button function of the alarm view.

Syntax

```
Ausdruck.ShowHideList()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowHitList

Description

Executes the "Hitlist" button function of the alarm view.

Syntax

```
Ausdruck.ShowHitList()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowInfoText

Description

Executes the "About dialog" button function of the alarm view.

Syntax

```
Ausdruck.ShowInfoText()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowLockDialog

Description

Executes the "Lock dialog" button function of the alarm view.

Syntax

```
Ausdruck.ShowLockDialog()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowLockList

Description

Executes the "Lock list" button function of the alarm view.

Syntax

```
Ausdruck.ShowLockList()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowLongTermArchiveList

Description

Executes the "Historical alarm list (long-term)" button function of the alarm view.

Syntax

```
Ausdruck.ShowLongTermArchiveList()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowMessageList

Description

Executes the "Alarm list" button function of the alarm view.

Syntax

```
Ausdruck.ShowMessageList()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowPercentageAxis

Description

Executes the "Relative axis" button function of the f(t) trend view.

Syntax

```
Ausdruck.ShowPercentageAxis()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

OnlineTrendControl (Page 5122)

ShowPropertyDialog

Description

Executes the "Configuration dialog" button function of the control.

Syntax

```
Ausdruck.ShowPropertyDialog()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

FunctionTrendControl (Page 5077)

AlarmControl (Page 5018)

ShowSelectArchive

Description

Executes the "Select data connection" button function of the recipe view.

Syntax

```
Ausdruck.ShowSelectArchive()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

UserArchiveControl (Page 5207)

ShowSelection

Description

Executes the "Selection dialog" button function of the recipe view.

Syntax

```
Ausdruck.ShowSelection ()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

UserArchiveControl (Page 5207)

ShowSelectionDialog

Description

Executes the "Selection dialog" button function of the alarm view.

Syntax

```
Ausdruck.ShowSelectionDialog()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

--

See also

AlarmControl (Page 5018)

ShowSelectTimeBase

Description

Executes the "Timebase dialog" button function of the recipe view.

Syntax

```
Ausdruck.ShowSelectTimeBase()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

--

See also

UserArchiveControl (Page 5207)

ShowShortTermArchiveList

Description

Executes the "Historical alarm list (short-term)" button function of the alarm view.

Syntax

```
Ausdruck.ShowShortTermArchiveList()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowSort

Description

Executes the "Sorting dialog" button function of the recipe view.

Syntax

```
Ausdruck.ShowSort()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

UserArchiveControl (Page 5207)

ShowSortDialog

Description

Executes the "Sorting dialog" button function of the alarm view.

Syntax

```
Ausdruck.ShowSortDialog()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

--

See also

AlarmControl (Page 5018)

ShowTagSelection

Description

Executes the "Select data connection" button function of the control.

Syntax

```
Ausdruck.ShowTagSelection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

FunctionTrendControl (Page 5077)

ShowTimebaseDialog

Description

Executes the "Timebase dialog" button function of the alarm view.

Syntax

```
Ausdruck.ShowTimebaseDialog()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

ShowTimeSelection

Description

Executes the "Select time range" button function of the control.

Syntax

```
Ausdruck.ShowTimeSelection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

FunctionTrendControl (Page 5077)

ShowTrendSelection

Description

Executes the "Select trends" button function of the f(t) and f(x) trend views.

Syntax

```
Ausdruck.ShowTrendSelection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

-- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

FunctionTrendControl (Page 5077)

StartStopUpdate

Description

Executes the "Start" or "Stop" button function of the control.

Syntax

```
Ausdruck.StartStopUpdate()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

OnlineTableControl (Page 5114)

FunctionTrendControl (Page 5077)

Stop

Description

Closes WinCC Runtime.

Syntax

Expression.Stop

Expression

Required An expression which returns an "HMIRuntime" type object.

Parameters

--

See also

HMIRuntime (Page 4993)

Trace

Description

Returns a user-defined text through the operating system channel for debug alarms.

The methods HMIRuntime.Trace works only in a PC-based environment. The text transferred as parameter can be displayed using the diagnostics tools "GSC Diagnostics" or "ApDiag". You can use the "ShowSystemAlarm" system function if you need to run a trace without using external tools.

Syntax

Expression.Trace"STRING"

Expression

Required An expression which returns an "HMIRuntime" type object.

Parameters

STRING

The text which is issued as a debug alarm. The transferred text can be displayed using the diagnostics tools "GSC Diagnostics" or "ApDiag". You can use the "ShowSystemAlarm" system function if you need to run a trace without using external tools.

Example

In the following example a debug alarm is issued:

```
'VBS example trace
HMIRuntime.Trace "Customized error message"
```

See also

HMIRuntime (Page 4993)

UnhideAlarm

Description

Executes the "Unhide alarm" button function of the alarm view.

Syntax

```
Ausdruck.UnhideAlarm()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

--

See also

AlarmControl (Page 5018)

UnlockAlarm

Description

Executes the "Unlock alarm" button function of the alarm view.

Syntax

```
Ausdruck.UnlockAlarm()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

AlarmControl (Page 5018)

UnselectAll

Description

Removes all selections from the cells of a table-based control.

Syntax

```
Expression.UnselectAll()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

AlarmControl (Page 5018)

UnselectRow

Description

Removes the selections from a specific cell of a table-based control.

Syntax

```
Expression.UnselectRow(ByVal IRow As Long)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

Long

Parameters	Description
IRow	Number of the row to be selected.

See also

OnlineTableControl (Page 5114)

TrendRulerControl (Page 5187)

UserArchiveControl (Page 5207)

AlarmControl (Page 5018)

Write

Description of Tag Object

Writes a value to a tag. The "LastError" property can be used to determine whether the call was successful.

If the value of the tag is set successfully, the properties of the tag object are assigned the following values:

Property	Allocation
Value	Tag values set by the user (unchanged)
Name	Tag name (unchanged)
QualityCode	Bad Out of Service
Timestamp	0
LastError	0
ErrorDescription	" "

If the value of the tag is not set successfully, the properties of the tag object are assigned the following values:

Property	Allocation
Value	Tag values set by the user (unchanged)
Name	Tag name (unchanged)
QualityCode	Bad Out of Service
Timestamp	0
LastError	Write operation error codes
ErrorDescription	Error description on LastError

syntax

```
Expression.Write [Value],[Writemode]
```

Expression

Necessary. An expression which returns a tag object.

Parameters

The value to be written can be transferred directly to the method as a parameter. If the parameter is not specified, the value in the "Value" property is used. The "Writemode" option parameter can be used to select whether the tag value should be written synchronously or asynchronously. If the "Writemode" parameter is not used, writing is performed asynchronously as its default value.

During the writing process, no information is supplied on the status of the tags.

The "Value" property contains the value which was set before or during the writing operation, therefore is may not correspond to the real current value of the tag. If the data on the tag should be updated, use the Read method.

Parameters	Description
Value (optional)	The tag value is specified. The specified value overwrites the value in the "Value" property in the tag object. The tag value is not specified. The tag receives the current value from the "Value" property of the tag object.
Writemode (optional)	0 or empty: The tag value is written asynchronously. 0 is the default value. 1: The tag value is written synchronously.

On asynchronous writing, it is written immediately into the tag image. The user does not receive any feedback if the value has been written in the programmable controller, too.

In the case of synchronous writing (direct to the PLC), the writing operation actually occurs when the PLC is ready to operate. The use receives a check-back message if the writing operation was not successful.

Example

Asynchronous writing

```
'VBS104
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Value = 5
objTag.Write
MsgBox objTag.Value
```

or

```
'VBS105
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Write 5
MsgBox objTag.Value
```

Synchronous writing

```
'VBS106
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Value = 5
objTag.Write ,1
MsgBox objTag.Value
```

or

```
'VBS107
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Write 5, 1
MsgBox objTag.Value
```

Description of TagSet Object

The TagSet object offers the option of writing several tags in one call.

Functionality here is mostly identical with that of a tag object. In the following, only deviations thereof are described.

Expression

Necessary. An expression which returns an object of type "TagSet".

Parameters

In order to write different values, the "Value" property of individual tag objects must be set, and write must be called thereafter without the "Value" parameter. Since the write commands are grouped into one call, it results in improved performance compared to single calls.

In a TagSet object, it is not possible to pass on a value using the "Write" method. Individual values must be set using the "Value" property of the individual tag objects.

Example

The following example shows how tags are included in the TagSet list, how tag values are set and subsequently written.

```
'VBS173
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Wert1"
group.Add "Wert2"
group("Wert1").Value = 3
group("Wert2").Value = 9
group.Write
```

If you set the optional parameter "Writemode" equal to 1, the process tags are written synchronously (directly to AS).

```
group.Write 1
```

WriteTags

Description

Executes the "Write tags" button function of the recipe view.

Syntax

```
Expression.WriteTags()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

UserArchiveControl (Page 5207)

ZoomArea

Description

Executes the "Zoom area" button function of the f(t) and f(x) trend views.

Syntax

```
Ausdruck.ZoomArea()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ZoomInOut

Description

Executes the "Zoom +/-" button function of the f(t) and f(x) trend views.

Syntax

```
Ausdruck.ZoomInOut()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

ZoomInOutTime

Description

Executes the "Zoom time axis +/-" button function of the f(t) trend view.

Syntax

```
Ausdruck.ZoomInOutTime()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

OnlineTrendControl (Page 5122)

ZoomInOutValues

Description

Executes the "Zoom value axis +/-" button function of the f(t) trend view.

Syntax

```
Ausdruck.ZoomInOutValues()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameter

- -

See also

OnlineTrendControl (Page 5122)

ZoomInOutX

Description

Executes the "Zoom X axis +/-" button function of the f(x) trend view.

Syntax

```
Ausdruck.ZoomInOutX()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

FunctionTrendControl (Page 5077)

ZoomInOutY

Description

Executes the "Zoom Y axis +/-" button function of the f(x) trend view.

Syntax

```
Ausdruck.ZoomInOutY()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

FunctionTrendControl (Page 5077)

ZoomMove

Description

Executes the "Move trend area" button function of the f(t) and f(x) trend views.

Syntax

```
Ausdruck.ZoomMove()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

OnlineTrendControl (Page 5122)

FunctionTrendControl (Page 5077)

12.9 Mobile Panels

12.9.1 Field of application of the Mobile Panel Wireless

Introduction

The Mobile Panel Wireless enables you to move through the plant without restrictions and with or without security technology. You can operate and monitor moving machine parts or the entire plant depending on the situation. You can plan, configure, simulate and extend wireless networks quickly and reliably.

The Mobile Panels Wireless are based on standardized WLAN technology and are specially tailored to the requirements of automation.

WLAN area

A WLAN area is the area of the plant in which the HMI device communicates with the PLC over a wireless local area network. The wireless coverage of the network has a sufficient signal strength in this area.

The WLAN of a plant is formed by the radio area of one or more Access Points.

Access Point

The Access Point serves as a gateway between the wireless and hard-wired network. The HMI device communicates with an Access Point via WLAN. You can operate the different machines or plants without annoying cables. The HMI device is connected to a network in which it communicates with a PLC via the access point.

Effective range

Safety-related operator input is only possible in a limited part of a WLAN area upstream of a machine or plant known as the effective range. One exception to this rule is the emergency stop, which works in the entire WLAN area.

The configured effective ranges are stored in the HMI device. The HMI device can only log on to a machine within the effective range. This also requires there no other HMI device is logged on in the same effective range.

To ensure clear logon and machine operator inputs, one effective range must not overlap another effective range.

Tags

Effective ranges are formed by the spatial assignment of tags to a machine or plant. Tags create a geometrical space which is relevant for logon to a machine. After logon you operate the machine using acknowledgement buttons.

A distinction is made between two types of tags:

- Transponder
- RFID tag

Transponder

After successful logon to a machine, the spanned geometrical area will be monitored. The distance between the HMI device and the transponder can only be measured if the two devices are in the reception range of each other. When you leave the effective range, safety-related operator input of the machine with acknowledgment buttons is prevented.

Zones in the transponder system

Zones are configurable objects within a transponder system. A zone is defined by the maximum distance from one or more transponders. In a hard-wired Mobile Panel, the length of the connection cable defines a zone around a connection box.

Zones are operating areas in which safety-related operator input is not possible. To enter a zone, you have to configure a screen change to the correct process picture.

RFID tag

After successful logon to a machine, the spanned geometrical area is not monitored. Monitoring therefore calls for an additional organizational measure, for example: a light barrier or a protective fence. The protection area is separated from the plant by a security system. The protection area is the area in the plant in which you operate one or more machines in fail-safe operation. One or more RFID Tags are distributed within protection zones. The protection area is not a configurable object.

12.9.2 How does the transponder system work?

Overview

To give a basic understanding of all this, the following section describes the interaction between:

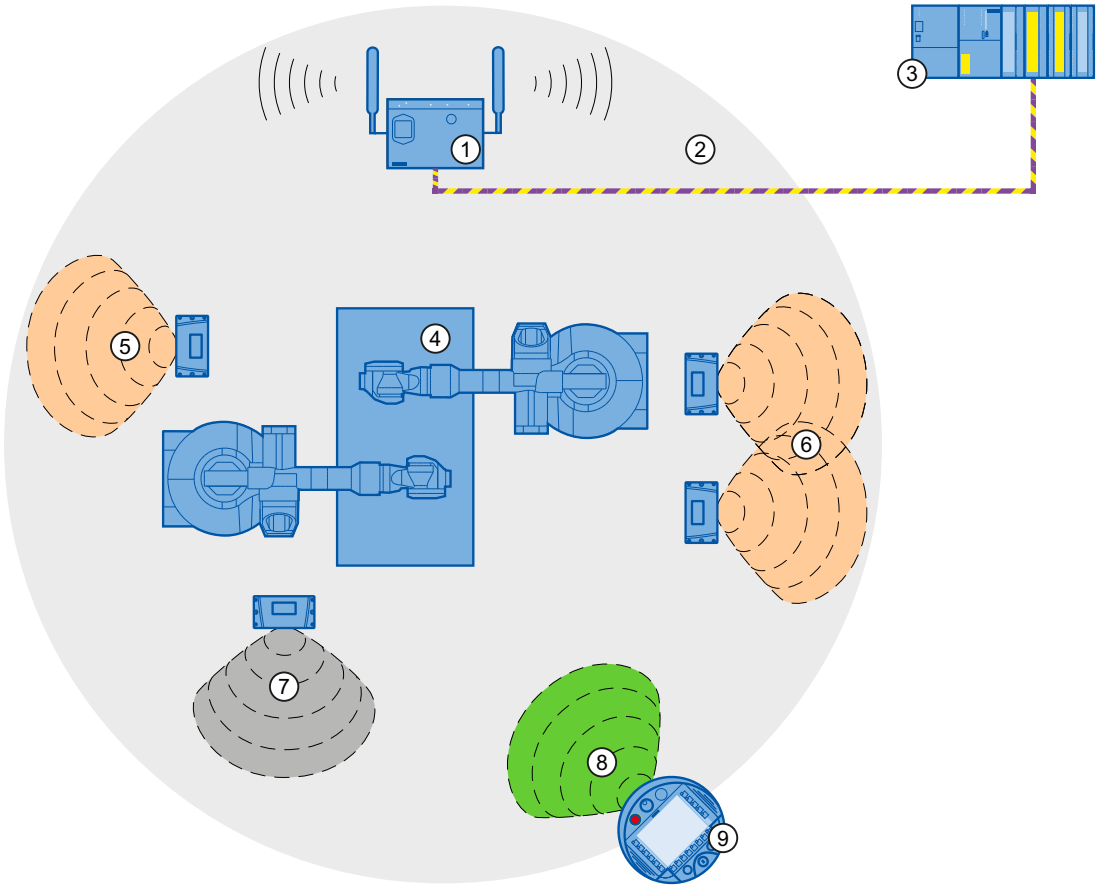
- HMI device
- Zone

- Effective range
- Transponder

Note

You should always assign administrator rights or set up encryption to protect the project containing a Mobile Panel Wireless for fail-safe operation against unauthorized access.

Principle of operation in the transponder system



1 Access Point
2 WLAN of the Access Point
3 PLC
4 Plant
5 Effective range "Robot 1" with a transponder
6 Effective range "Robot 2" with two transponders
7 Zone "Zone 1" with a transponder
8 Radio area of the HMI device
9 HMI device

Example:

- A screen change is configured for "Zone 1".
- Transponder1 is assigned to the "Robot 2" effective range.
- The operator enters the radio cone of the transponder to receive the transponder ID.

- To log onto the machine "Robot 2", the operator presses the object "Effective range - Name".
- The "Effective range - Logon" dialog box opens. The operator enters the effective range ID and confirms with "OK". The dialog is closed.
- If the logon was successful, then the acknowledgement buttons will be active.
- The HMI device measures the distance from the transponder which is less than the configured distance of the effective range.

Result:

The operator is logged onto the machine "Robot 2". The operating element "Effective range - Name" is green and shows the name of the effective range "Robot 2".

If the HMI device is logged onto the effective range, then the following will apply:

- The operator may not leave the effective range without logging off. If the operator leaves the effective range without logging off for more than 30 seconds, a local rampdown will occur.
- No other HMI device can log onto the machine.

To log off from the machine, the operator presses the operating element "Effective range - Name". A logoff dialog is displayed. The operator confirms the dialog.

After successful logoff, another operator can log on to the machine "Robot2".

When the operator enters "Zone 1", a screen change will take place.

12.9.3 How does the RFID system work?

Overview

RFID stands for radio frequency identification. RFID allows wireless identification of objects without contact or visual sighting.

For a basic understanding, the following section describes the interaction between:

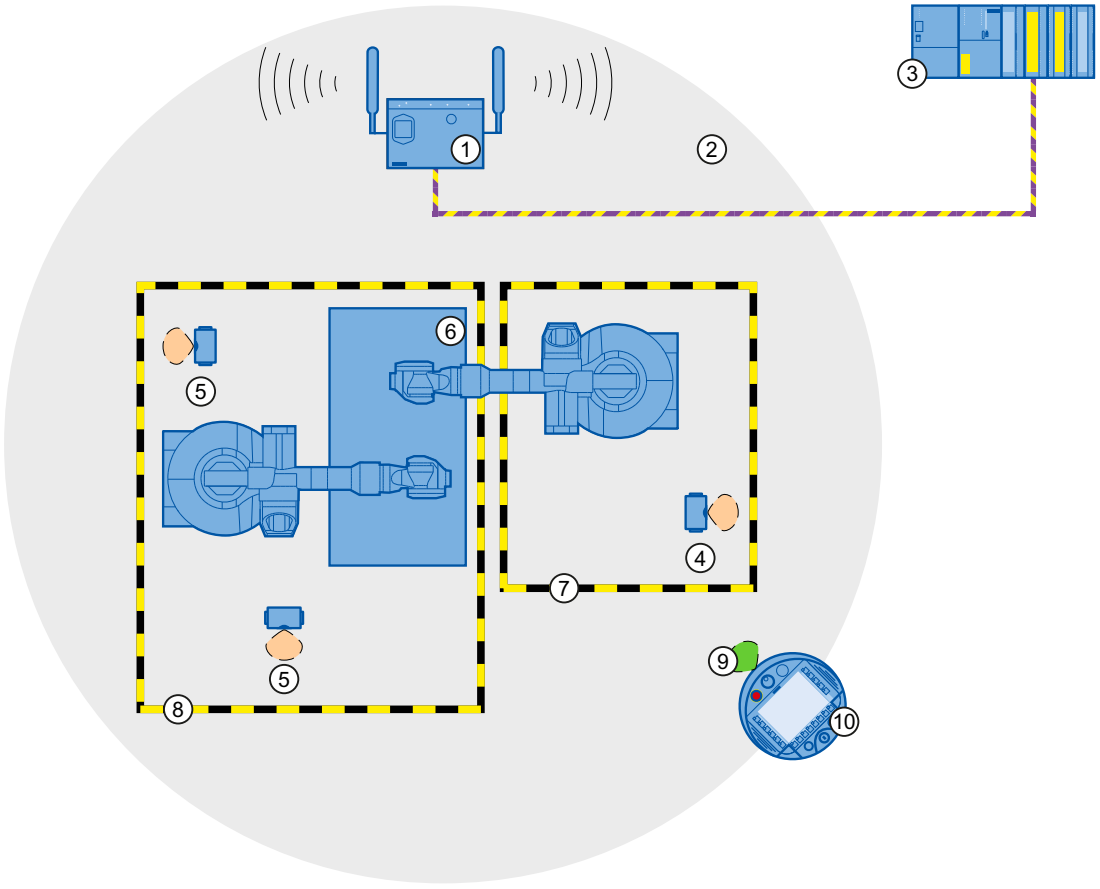
- HMI device
- Effective range (RFID)
- RFID tag

Note

RFID Tags in the plant area

You may only affix RFID Tags in restricted plant areas that are protected by additional protective measures, for example, protective railings.

Principle of operation in the RFID system



① Access Point
② WLAN of the Access Point
③ PLC
④ RFID tag with effective range (RFID) for logon to "Robot 1"
⑤ RFID tag with effective range (RFID) for logon to "Robot 2"
⑥ Plant
⑦ Protective area limit of the machine "Robot1"
⑧ Protective area limit of the machine "Robot2"
⑨ Radio area of the HMI device
⑩ HMI device

Example:

- "RFID Tag1" is assigned to the effective range (RFID) "Robot1".
- An operator enters the restricted plant area. The operator manually switches to the screen with the configured operating element "Effective range - Name (RFID)".

- At "RFID Tag1", he presses the "Scan" button.
  During the logon operation, the button is yellow and displays the text "Scanning…".
- The "Effective range - Logon" dialog box opens. The operator enters the effective range (RFID) ID and confirms it with "OK". The dialog is closed.
- The "Confirmation of logon" dialog box opens. The operator confirms logon with the acknowledgment button.
- If logon was successful, the acknowledgement buttons are active.

Result:

The operator is logged on to the machine "Robot 1" and operates the machine within the protection area in fail-safe operation. The operating element "Effective range - Name (RFID)" is green and shows the name of the effective range (RFID) "Robot1".

If the HMI device is logged on to the machine, the following applies:

- The user may not leave the protection area without logging off from the machine. If the operator leaves the protection area without logging off, a local rampdown occurs.
- No other HMI device can log on to this machine.

To log off from the machine, the operator presses the operating element "Effective range - Name (RFID)". A logoff dialog is displayed. The operator confirms the dialog.

After successful logoff, another operator can log on to the machine "Robot 1".

12.9.4 Configuring the Mobile Panel V2 for fail-safe operation (up to V12)

12.9.4.1 Configuration overview

Introduction

Additional steps in the TIA Portal are necessary before the configuration of the Mobile Panels V2 for fail-safe operation. This affects the following HMI devices:

- Mobile Panel 277F IWLAN V2
- Mobile Panel 277F IWLAN (RFID Tag)

Procedures overview

1. Creating PLC for fail-safe operation
2. Installing a general station description (GSD)
3. Creating a module from the GSD file
4. Creating a connection between a module from the GSD file and the PLC
5. Creating Mobile Panel V2 for fail-safe operation

See also

Creating Mobile Panel (Page 5956)

Creating a module from the GSD file (Page 5952)

Installing a general station description (GSD) (Page 5952)

Creating PLC for fail-safe operation (Page 5951)

Creating a connection between a module and the PLC (Page 5955)

12.9.4.2 Creating PLC for fail-safe operation

Requirement

A project is open

Procedure

1. Click "Add new device" in the project navigation.
2. Click "PLC" in the "Add new device" dialog.
3. Select a PLC for fail-safe operation, for example, CPU-315F-2PN/DP. The "Devices & Networks" editor opens.
4. Open the device view.
5. Select the device and click on "Device overview".
6. Select the device.
7. Click "Properties > PROFINET interface [X2] >Ethernet addresses > Interface connected with > Add new subnet" in the inspector window.
8. Enter the IP address under "IP protocol > Set IP address in the project".
9. Enter a time for "Properties > PROFINET interface [X2] > F parameter > Set default F monitoring time for F I/O of this interface".
10. Specify the properties for fail-safe operation in "Properties > Fail-safe"
    - "F-capability activated"
    - "Basis for PROFIsafe addresses"
11. Specify further properties for fail-safe operation as usual, for example, protection level.

Result

You have created a PLC in the project and made the settings for fail-safe operation.

See also

Configuration overview (Page 5948)

12.9.4.3 Installing a general station description (GSD)

Introduction

A GSD file (device master data file) contains all the IO device properties. The language used to describe the GSD files is GSDML (Generic Station Description Markup Language). Install a GSD file to expand the hardware catalog.

Requirement

- The "Devices & Networks" editor is open.
- The "Network view" is open.
- The GSD file is available on your PC.

The GSD file can be found online:

GSD file of Mobile Panel V2 for fail-safe operation (http://support.automation.siemens.com/WW/view/de/19241467)

Procedure

1. Click "Options > Install general station description (GSD)".
2. In the "Install general station description" dialog, choose the directory in which the GSD files are located.
3. Select the GSD file from the list.
4. Click the "Install" button.
5. To create a log file for the installation, click on the "Save log file" button.

Result

You have installed the GSD file. You can find the new devices installed by means of the GSD files in the hardware catalog under "Additional field devices > PROFINET". Any problems that occur during the installation can be traced using the log file.

See also

Configuration overview (Page 5948)

12.9.4.4 Creating a module from the GSD file

Requirements

- A PLC for fail-safe operation is created in the project e.g. CPU-315F-2PN/DP.
- The GSD file is installed.
- The "Network view" is opened in the "Devices & Networks" editor.

Procedure

1. Open the hardware catalog.
2. Open the directory "Additional field devices > PROFINET-IO > HMI > Siemens AG > SIMATIC HMI > 277".
3. Now open the folder of the Mobile Panel V2 for fail-safe operation whose device type you also have as hardware:
   - Mobile Panel 277F IWLAN V2 or
   - Mobile Panel 277F IWLAN V2 (RFID tag)
4. For Mobile Panel 277F IWLAN V2, select the module with the order number 6AV6645-0EC01-0AX1.

5. For Mobile Panel 277F IWLAN V2 (RFID tag), select the module with the order number 6AV6645-0EF01-0AX1.

Note

The order number of the module in the GSD file has to match the order number of the Mobile Panel V2 in the software.

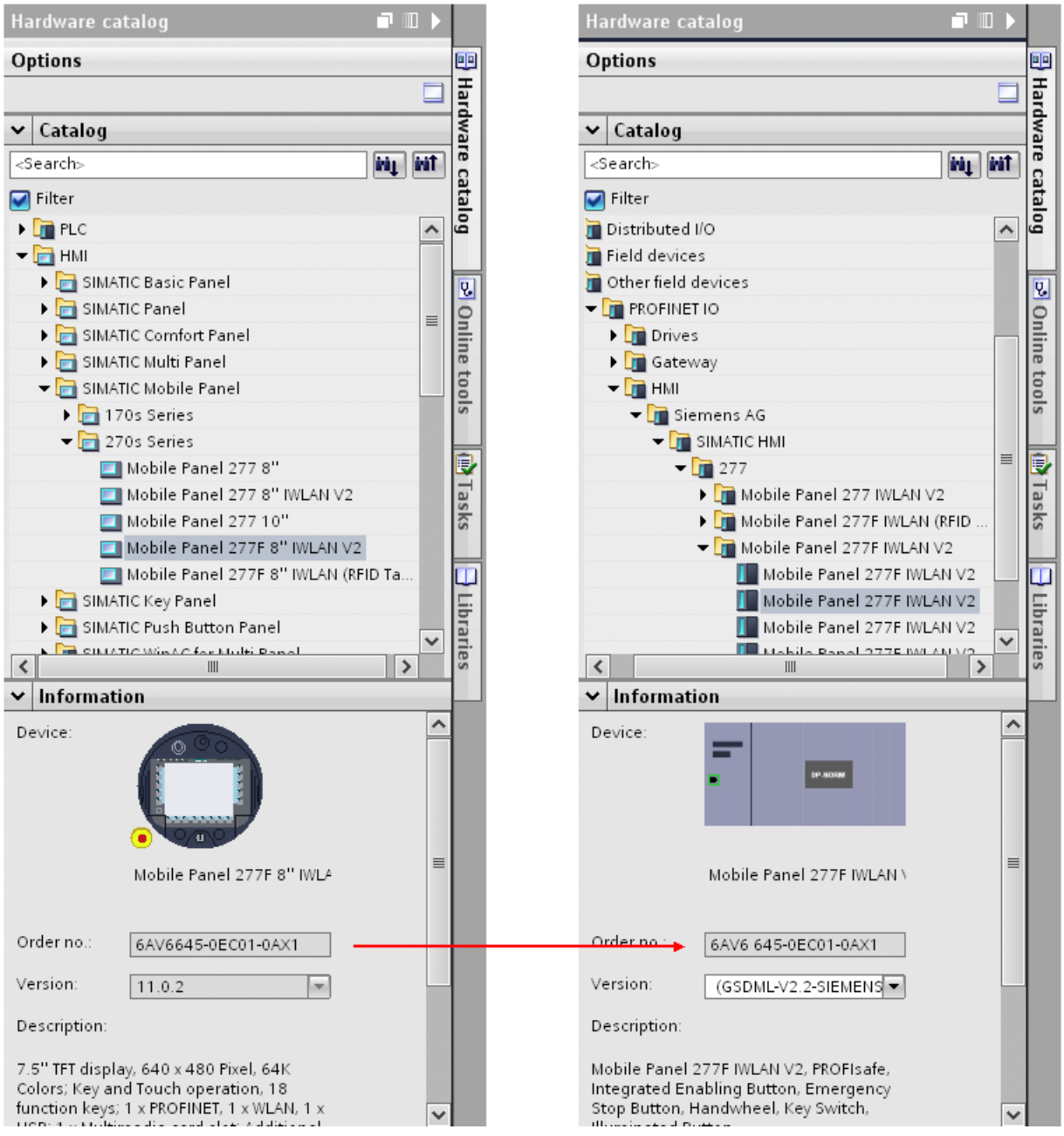


Figure 12-5 Left: Mobile Panel in the software. Right: Module in the GSD file

6. Drag-and-drop the desired module into the network view.

Result

You have created a module from the GSD file in your project.

See also

PROFIsafe address (Page 5977)

Configuration overview (Page 5948)

12.9.4.5 Creating a connection between a module and the PLC

Requirements

- A PLC for fail-safe operation is created in the project e.g. CPU-315F-2PN/DP.
- The GSD file is installed.
- A module from the GSD file has been created.
- The "Devices & Networks" editor is open.

Procedure

1. Open the "Network view".
2. Select the module and right-click on "Not assigned". The shortcut menu is opened.
3. Select "Assign new IO controller". A dialog box opens.
4. Select the IO controller and confirm with OK. A connection is established.
5. Open the device view
6. Click on "Mobile277Failsafe_IO_1" in the device overview.
7. Specify the properties for fail-safe operation in the inspector window under "Properties > PROFIsafe":
   - "F-block ID"
   - "F-Source-Add"
   - "F-Dest-Add"
8. Click on the module in the device view.
9. Assign a name under "Properties > General > Name" in the inspector window.
10. Assign an IP address under "PROFINET interface [X1] > Ethernet addresses > IP protocol > Set IP address in the project".

Result

The module from the GSD file is connected to the PLC. The module adopts the settings made in the PLC for PROFIsafe and PROFINET.

See also

Configuration overview (Page 5948)

12.9.4.6 Creating Mobile Panel

Requirement

- The inspector window is open.
- A PLC for fail-safe operation is created in the project, for example, CPU-315F-2PN/DP.
- The GSD device is created in the project.

Inserting Mobile Panel into the project.

1. Click "Add new device" in the project navigation.
2. Click on "HMI" in the "Add new device" dialog.
3. Select a Mobile Panel V2 for fail-safe operation, for example, Mobile Panel 277F IWLAN V2. The device is inserted into the project.

Note

Select the corresponding Mobile Panel for the module already installed from the GSD file.

Defining properties

1. Open the "Devices and Networks" editor.
2. Open the "Device view".
3. Select the Mobile Panel in the work area.
4. Open "Properties > General" in the inspector window.
5. To facilitate assignment in the project, enter the name of the associated module under "Name"
6. Enter the same IP address as in the module under "Set IP address in the project".
7. Activate "Set IP address using a different method".

Result

Entering the same IP address as in the device enables error-free communication. Alternatively, you can set the IP address on the Mobile Panel after loading the project.

See also

Configuration overview (Page 5948)

12.9.5 Configuring KTP Mobile Panels for fail-safe operation

12.9.5.1 Creating KTP Mobile Panel for fail-safe operation

Introduction

Configure fail-safe operation in the "Devices & Networks" editor for the following KTP Mobile Panels:

- KTP400F Mobile Panel
- KTP700F Mobile Panel
- KTP900F Mobile Panel

Requirement

- The Inspector window is open.
- A PLC for fail-safe operation is created in the project, for example, CPU-1517F-2PN/DP.

Inserting a mobile panel into the project

1. Click "Add new device" in the project tree.
2. Click on "HMI" in the "Add new device" dialog.
3. Select a KTP Mobile Panel for fail-safe operation, for example, Mobile Panel KTP700F.
   The device is inserted into the project.

Configuring fail-safe operation for the mobile panel

1. Open the network view in the "Devices & Networks" editor.
2. Select the Mobile Panel in the work area.
3. In the Inspector window, select "Properties > General > Operating mode > IO device".
4. Assign the fail-safe PLC under "Assigned IO controller".
   PROFINET is activated and the transfer areas of the I-device communication are displayed.
5. In the Inspector window, click "Activate PROFIsafe" under "Properties > General" > PROFIsafe".
   PROFIsafe is activated and the F-parameters are displayed under "Properties > General> PROFIsafe parameters > F-parameters".
6. To enter a new F-monitoring time, select "Manual assignment of F-monitoring time".
7. Enter a F-monitoring time suitable for your HMI device and plant.

Result

The fail-safe operation of the KTP Mobile Panel is configured in WinCC. Now create the corresponding F-blocks in STEP 7 and interconnect them in the controller. Once the project is loaded, set the PROFIsafe address on the mobile panel.

You can find additional information on configuration and programming of fail-safe blocks in the manual "SIMATIC Safety - Configuration and Programming" of the STEP 7 documentation and in the device manual of your mobile panel.

12.9.5.2 Configuring the termination of the PROFIsafe connection

Procedure

Before you disconnect your Mobile Panel from a connection box, you have to terminate the connection to PROFIsafe.

Use the "Disconnect PROFIsafe" system function to close the PROFIsafe connection. The system function forwards your call to the fail-safe PROFINET module.

Requirements

- A mobile panel has been created for fail-safe operation, for example, KTP700F Mobile Panel.
- A screen has been created and opened.

Procedure

To configure the PROFIsafe the connection and disconnection, follow these steps:

- Configure the "Disconnect PROFIsafe" system function on a button, for example, or use it in a script.

Result

When you release the configured trigger to terminate the PROFIsafe connection in Runtime, a dialog opens.

12.9.6 Basics

12.9.6.1 Zones

Introduction

The section below applies to the following Mobile Panels:

- Mobile Panels Wireless, for example, Mobile Panel 277 IWLAN V2 and Mobile Panel 277F IWLAN V2
- KTP Mobile Panels, wired, e.g., Mobile Panel KTP700

A "Zones" work area is only visible for these HMI devices.

Note

For more information refer to the operating instructions for the HMI device:

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

Open

Depending on the HMI device selected, open the work area in the project window by double-clicking on "Zones" or "Zones and effective ranges".

Work area

For Mobile Panels Wireless, the "Zones" work area displays the zones that have been set up and the transponders assigned to them.

For wired Mobile Panels, the "Zones" work area shows the zones that have been set up and the associated box ID of the connection box.

Inspector window

When a zone is selected, edit the name and display name under "General".

For Mobile Panels Wireless, configure the limit of the zone around the transponder. The zone of a transponder has the events "Upon entry" and "Upon exit"

With wired Mobile Panels enter the box ID of the connection box. The zone of a connection box includes the event "Connected".

Configure the system function ActivateScreen" to the events.

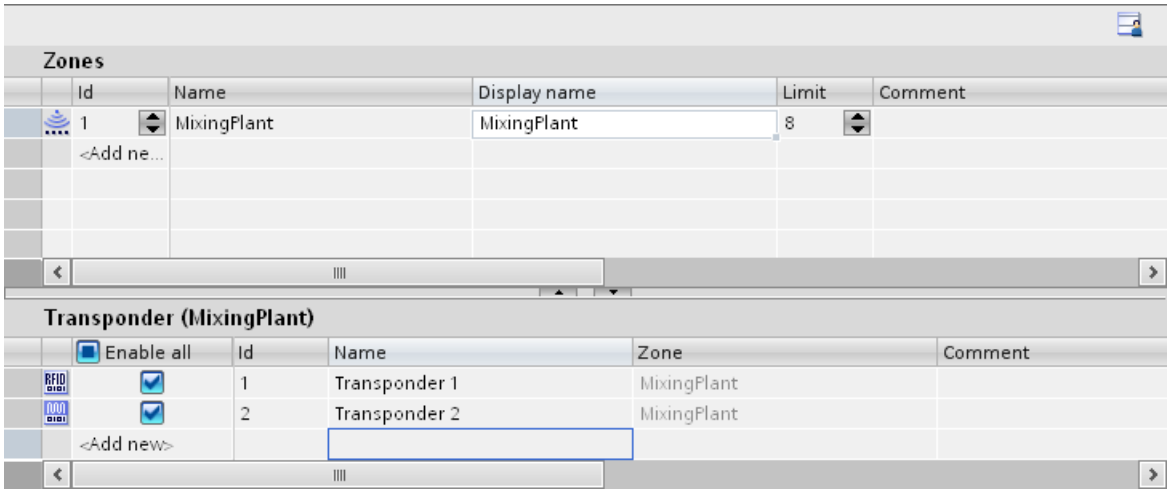
12.9.6.2 Zones work area at transponders

Introduction

The "Zones" work area displays the zones and the transponders assigned to them in table format. You create a list of transponders and assign certain transponders to a zone. The limit of the zone is defined by the maximum distance from the transponders.

Principle

The work area consists of the "Zones" and "Transponders" tables.



If you select a zone in the "Zones" table, the "Transponder" table displays the following:

- Transponder enabled: The transponder is assigned to the selected zone.
- Transponder deactivated: The transponder is not yet assigned to any zone.
- Transponder not available: The transponder has already been assigned to another zone. To undo the assignment, switch to the relevant zone and deactivate the transponder.

The zone and transponder IDs are initially assigned automatically. They can be changed, however.

Please note the following:

- A maximum of 254 zones may be configured.
- The zone ID must be unique and fall within the range from 1 - 254.
- You can initially configure transponders without assigning them to a zone.
- There may be no more than 255 transponders assigned to a zone.
- The transponder ID must be unique and fall within the range from 1 - 65534.

The transponder IDs are set on the transponder.

12.9.6.3 Zones work area on connection boxes

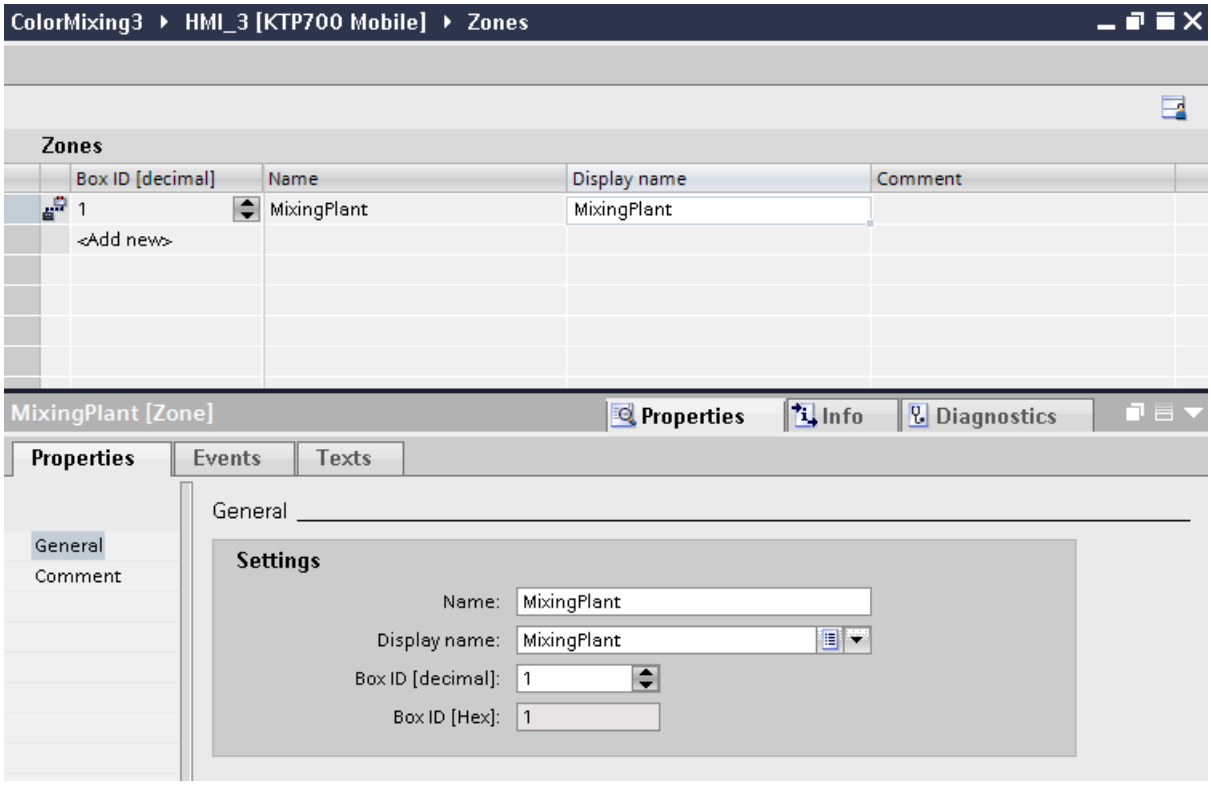
Introduction

The "Zones" work area displays the box IDs of the connection boxes and their zones in table format.

You create a list of box IDs of the connection boxes and assign specific connection boxes to a zone. The limit of the zone is defined by the length of the connecting cable to your Mobile Panel.

Principle

The work area consists of the "Zones" table.



The box ID is set on the connection box and can be read out from the HMI device.

12.9.6.4 Effective ranges

Introduction

The following section applies only to the Mobile Panels Wireless that support fail-safe operation, e.g. Mobile Panel 277F IWLAN V2. The "Effective ranges" work area is only visible for these HMI devices.

You set up effective ranges in order to control safety related operations. An effective range is defined by the maximum distance from one or more transponders.

Note

For more information refer to the operating instructions for the HMI device:

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

Open

Open the work area in the project window by double-clicking on "Zones & effective ranges". Click the "Effective ranges" tab.

Work area

The "Effective ranges" work area displays the effective ranges that have been set up and the transponders assigned to them.

Inspector window

When an effective range is selected, you can edit the names, display names and the limits of the effective range under the "General" category.

When you select a transponder, you can see both the effective range and the zone that are assigned to that transponder. You can only assign a transponder to one effective range so that the effective ranges do not overlap.

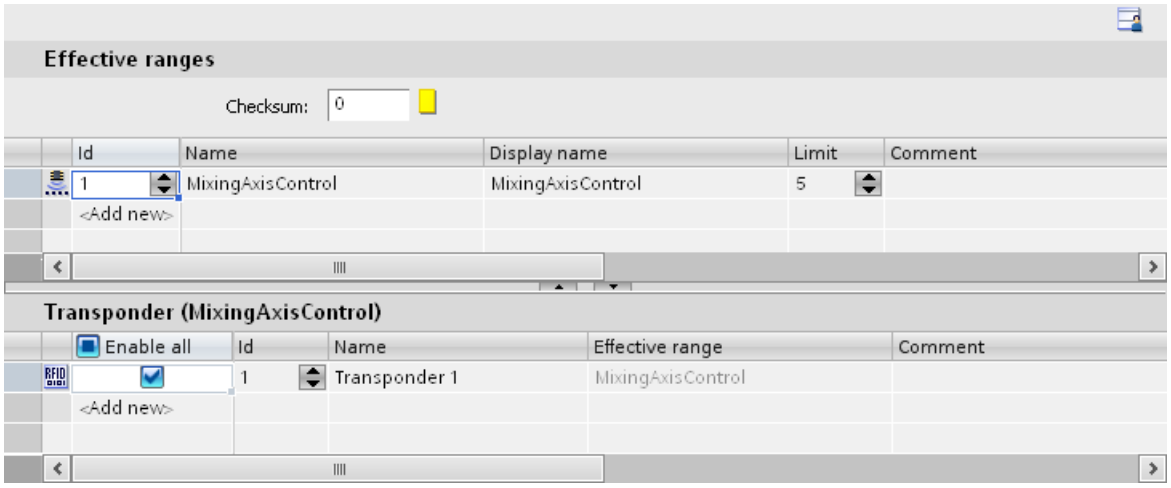
12.9.6.5 Effective ranges working area

Introduction

The "Effective ranges" work area displays the effective ranges and their transponders in table format. You create a list of transponders and assign certain transponders to an effective range. The limit of the effective range is defined by the maximum distance from the transponders.

Principle

The work area consists of the "Effective ranges" and "Transponders" tables.



The HMI device calculates a checksum from the local data so that the configured effective ranges and transponders reliably match those locally on the machine. The project cannot be started on the HMI device unless the local checksum agrees with the checksum stored in the "Effective ranges" editor.

When you select an effective range from the "Effective ranges" table, the "Transponder" table displays the following:

- Transponder enabled: The transponder is assigned to the selected effective range.
- Transponder deactivated: The transponder is not yet assigned to any effective range.
- Transponder not available: The transponder has already been assigned to another effective range. To undo the assignment, switch to the relevant effective range and deactivate the transponder.
- The zone assigned to the transponder is displayed in addition to the effective range.

The effective range and transponder IDs are initially assigned automatically. They can be changed, however.

Please note the following:

- A maximum of 127 effective ranges may be configured.
- The effective range ID must be unique and fall within the range from 1 - 127.
- The display name of an effective range must not be the same as its ID.
- You can initially configure transponders without assigning them to an effective range.
- There may be no more than 127 transponders assigned to an effective range.
- The transponder ID must be unique and fall within the range from 1 - 65534.

The transponder IDs are set on the transponder.

Effective range in Runtime

To log on to the effective range in Runtime, only the display name of the effective range appears to the operator in the Runtime language. The operator reads the effective range ID in the plant and enters it on the HMI device. This ensures that the right machine operated. The acknowledgment buttons are activated after successful logon.

12.9.6.6 Effective ranges (RFID)

Introduction

The following section only applies to the Mobile Panel 277F IWLAN (RFID tag) which supports fail-safe operation with RFID. The "Effective ranges (RFID)" work area is only visible with these HMI devices.

To monitor safety-related operator inputs, you set up effective ranges (RFID).

Note

For more information, refer to the operating instructions of the HMI device:

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

Open

Open the work area in the project window by double-clicking on "Effective ranges (RFID)".

Work area

The "Effective ranges (RFID)" work area displays the effective ranges that have been set up and the RFID Tags assigned to them.

Inspector window

When an effective range (RFID) is selected, you can edit the name, display name and ID of the effective range under the "General" category.

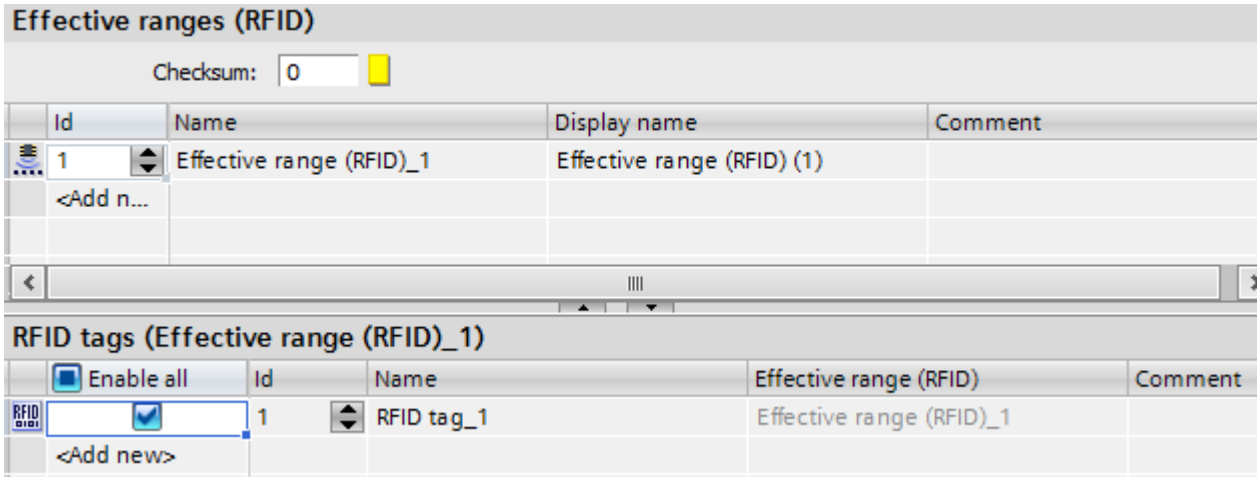
12.9.6.7 Work area for effective ranges (RFID)

Introduction

The "Effective ranges (RFID)" work area displays the effective ranges (RFID) and their RFID Tags in table format. You create a list of RFID Tags and assign specific RFID Tags to an effective range.

Principle

The work area consists of the "Effective ranges (RFID)" and "RFID Tags" tables.



The HMI device calculates a checksum from the local data so that the configured effective ranges (RFID) and RFID Tags reliably match those locally on the machine. The project cannot be started on the HMI device unless the local checksum agrees with the checksum stored in the "Effective ranges (RFID)" editor.

When you select an effective range (RFID) in the "Effective ranges (RFID)" table, the "RFID Tags" table displays the following:

- RFID Tag enabled: The RFID Tag is assigned to the selected effective range (RFID).
- RFID Tag disabled: The RFID Tag is not yet assigned to any effective range (RFID).
- RFID Tag not available: The RFID Tag has already been assigned to another effective range (RFID). To undo the assignment, switch to the relevant effective range (RFID) and deactivate the RFID Tag.

The IDs of the effective ranges (RFID) and RFID Tags are initially assigned automatically. They can be changed, however.

Please note the following:

- A maximum of 127 effective ranges (RFID) may be configured.
- The effective range (RFID) ID must be unique and fall within the range from 1 - 127.
- The display name of an effective range (RFID) must not be the same as its ID.
- You can initially configure RFID Tags without assigning them to an effective range (RFID).
- A maximum of 127 RFID Tags is available to assign to an effective range (RFID).
- The RFID Tag ID must be unique and fall within the range 1-65534.

The RFID Tag ID is written to the RFID Tag by the HMI device.

Effective range in Runtime

To log on to the effective range (RFID) in Runtime, only the display name of the effective range (RFID) is displayed to the operator in the Runtime language. The operator reads the effective range (RFID) ID in the plant and enters it on the HMI device. This ensures that the right machine operated. The acknowledgment buttons are activated after successful logon.

12.9.7 Working with zones on transponders

12.9.7.1 Configuring a zone in the transponder system

Validity

The next section applies to all Mobile Panels Wireless, e.g. Mobile Panel 277 IWLAN.

Introduction

You will have to set up a zone in order to carry out plant-specific operator control and monitoring tasks.

Requirements

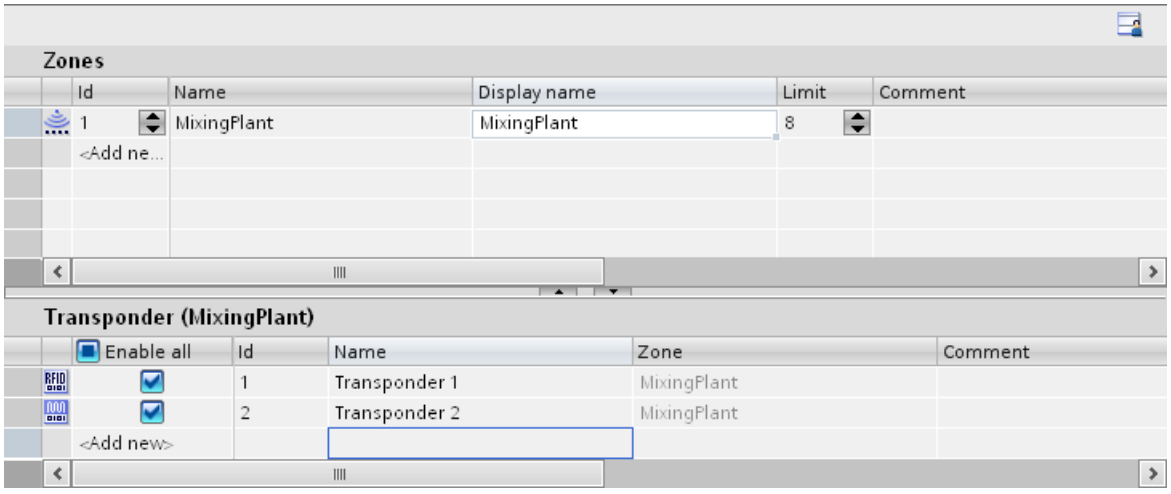
- A suitable editing language, e.g. English, must have been set in the "Project languages" editor.
- The "Zones" work area is open.

Procedure

1. Double-click "Add..." in the "Zones" table.
2. Enter the "MixingPlant" zone in the "Name" field.
3. Enter "MixingSystem" as the "Display name" for the zone at runtime. Enter zone "1" in the "ID" field.
4. Enter "8" meters as the zone "Limit".
5. Double-click "Add..." in the "Transponders" table.
6. Enter "Transponder 1" as the "Name" of the transponder.
7. Enter transponder "1" in the "ID" field.
   Then set transponder IDs on the transponder.
8. Enable the assignment of "Transponder 1".
   "Transponder 1" is assigned to the "MixingPlant" zone.
9. Repeat steps 6 to -9 accordingly for "Transponder 2".
   "Transponder 2" is assigned to the "MixingPlant" zone.

Result

The "MixingPlant" zone is configured as two conical areas arranged at a distance of 8 meters, regardless of the effective ranges: one area around "Transponder 1" and one around "Transponder 2".



The display name designates the zone in the set editing language. An overview of the display names can be found in the "Project texts" editor.

12.9.7.2 Displaying the screen on entering a zone

Introduction

Configure a screen change when entering and leaving the zone.

Requirements

- A Mobile Panel which supports zones has been created, for example, Mobile Panel 277 IWLAN.
- The "MixingPlant" zone is configured.
- The "Plant_1" screen has been created and opened.
- The "MixingPlant_1" screen has been created and opened.
- The "Zones" work area is open.

Procedure

1. Select the "MixingPlant" zone from the "Zones" table.
2. Click "Properties> Events" in the Inspector window.
3. Click the event "On entry".

4. Select the "ActivateScreen" system function in the "function list".
5. Select "MixingPlant_1" as the "Screen name".
6. Click the event "On exit".
7. Select the "ActivateScreen" system function in the function list.
8. Select "Plant_1" as the "Screen name".

Result

The "MixingPlant_1" screen is displayed on the HMI device when you enter the "MixingPlant" zone. When you exit the zone, the "Plant_1" screen appears once more.

12.9.7.3 Displaying an object in relation to the zone

Validity

The following example is independent of the WLAN and applies to all Mobile Panels.

Introduction

You configure an I/O field that is only visible within a specific zone.

Requirements

- A Mobile Panel which supports zones has been created, for example, Mobile Panel 277 IWLAN.
- The "Zone_id" variable of the "Int" data type has been created.
- A "Plant_1_Overview" screen has been created.

Procedure

1. Open the "Runtime Settings > General" editor.
2. Select the "Zone_id" tag as the "ID Zone/ Effective range".
3. Open the screen "Plant_1_Overview".
4. Drag and drop the "I/O box" object from the toolbox into the screen.
5. Click "Properties > Animations" in the Inspector window.
6. Click "Layout > Visibility."
7. Select "Zone_id" as the "tag".
8. Select "from" 1 "to" 1 as the "Range".
9. Enable "Visible".

Result

The IO field is only visible at runtime under the following conditions:

- The hard-wired Mobile Panel is connected to a connection box with ID = 1.
- The Mobile Panel Wireless is located in a zone with ID = 1, for example, the "MixingPlant" zone.

12.9.8 Working with zones on connection boxes

12.9.8.1 Configuring the zone of a connection box

Validity

The following section applies to all KTP Mobile Panels, for example, Mobile Panel KTP700.

Introduction

You will have to set up a zone in order to carry out plant-specific operator control and monitoring tasks.

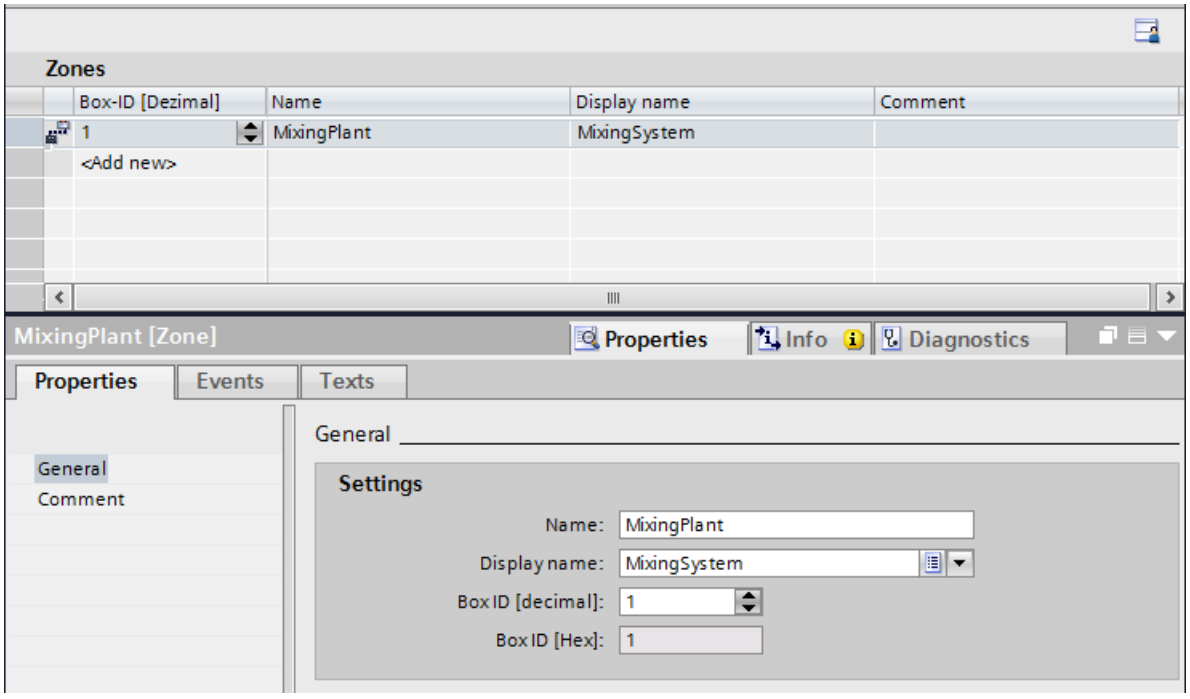
Requirements

- A suitable editing language, e.g. English, must have been set in the "Project languages" editor.
- The "Zones" work area is open.

Procedure

1. Double-click "Add..." in the "Zones" table.
2. Enter the box ID of your connection box for "ID".
3. Enter the "MixingPlant" zone in the "Name" field.

4. Enter "MixingSystem" as the "Display name" for the zone at runtime.



5. Repeat these steps for all required connection boxes in the project.

Result

The "MixingPlant" zone is configured. The range of the zone is defined by the length of the connecting cable of the Mobile Panel.

The display name designates the zone in the set editing language. An overview of the display names can be found in the "Project texts" editor.

12.9.8.2 Configuring a screen for connection to connection box

Introduction

You configure the screen that is displayed when a Mobile Panel is connected to a connection box.

Requirements

- A Mobile Panel which supports zones has been created, for example, Mobile Panel KTP700.
- The "MixingPlant" zone is configured.
- The "Plant_1" screen has been created and opened.

- The "MixingPlant_1" screen has been created and opened.
- The "Zones" work area is open.

Procedure

1. Select the "MixingPlant" zone from the "Zones" table.
2. Click "Properties> Events" in the Inspector window.
3. Click on the "Connected" event.
4. Select the "ActivateScreen" system function in the "function list".
5. Select "MixingPlant_1" as "Screen name".

Result

When you connect your Mobile Panel to the connection box of the "MixingPlant" zone, the "MixingPlant_1" screen appears on the display of the HMI device.

12.9.9 Working with effective ranges

12.9.9.1 Overview

Introduction

The following configuration guide describes the steps needed to set up an effective range for fail-safe operation on a Mobile Panel Wireless.

Procedures overview

1. Configuring the effective range:
   You configure the "MixingAxisControl" effective range as a conical area around "Transponder1 " at a distance of 5 meters.
2. Configuring the name of the effective range:
   To ensure logon to the effective range, configure the object "Effective range - Name".
3. Configuring additional effective range objects:
   Configure additional objects for displaying the position and signal strength within an effective range.
4. Set parameters for loading:
   - PROFIsafe communication
   - WLAN network
   - Power management
   - Interface

5. Configure the data channel
6. Configure network operation
7. Set the transponder
8. Commissioning effective ranges
9. Switching on and Testing the HMI device
10. Start loading manually
11. Acknowledging the effective range at the plant:
    You acknowledge the effective ranges and their transponders.
12. Determine the checksum
13. Load the project once again with checksum:
    Enter the checksum you have determined in the project and load the project once again.
14. Test effective range

Note

For detailed information on items 5 to 14, refer to the operating instructions of the HMI device.

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

12.9.9.2 Configure effective range

Introduction

You set up an effective range in order to control safety related operator input upstream of a machine or plant.

Requirements

- A suitable editing language, e.g. English, must have been set in the "Project languages" editor.
- The "Effective ranges" work area is open.

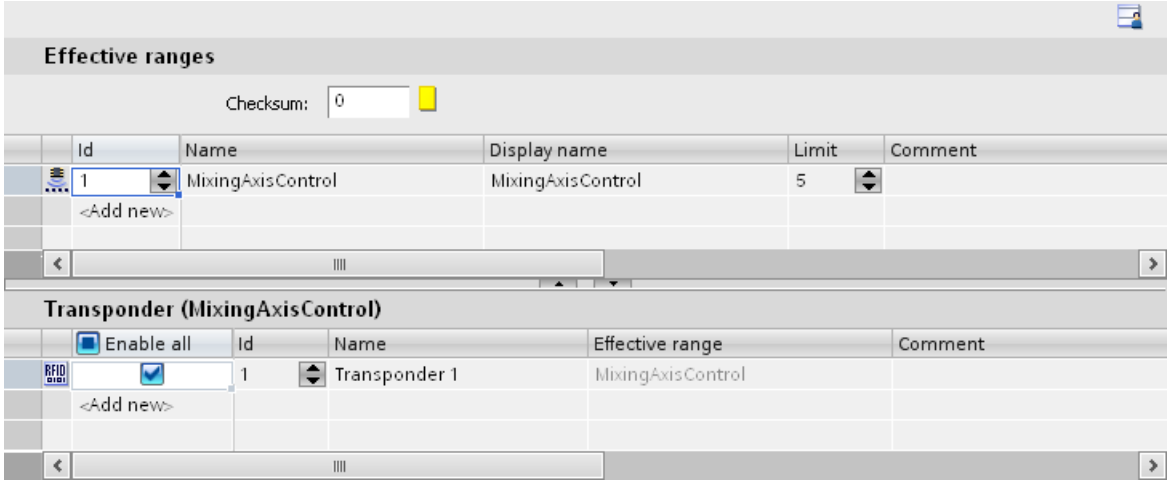
Procedure

1. Double-click "Add..." in the "Effective Ranges" table.
2. Enter "MixingAxisControl" as the "Name" of the effective range.
3. Enter "Mixing_axis_PLC" as the "Display name" for the effective range at runtime.
4. Enter effective range "1" in the "ID" field.
   Make sure that the effective range ID matches the ID on the machine.
5. Enter "5" meters as the "Limit" of the effective range.

6. Double-click "Add..." in the "Transponders" table.
7. Enter "Transponder 1" as the "Name" of the transponder.
8. Enter transponder "1" in the "ID" field.
   Then set transponder IDs on the transponder.
9. Enable the assignment of "Transponder 1".
   "Transponder 1" is assigned to the "MixingAxisControl" effective range.

Result

You configure the "MixingAxisControl" effective range as a conical area around "Transponder 1" at a distance of 5 meters, regardless of the zones.



The display name designates the effective range in the set editing language. An overview of the display names can be found in the "Project texts" editor.

12.9.9.3 Configuring the effective range name

Introduction

Configure at least the "Effective range - Name" object so that an operator can log on and off a machine using an effective range. If you do not configure an effective range object for a Mobile Panel Wireless for fail-safe operation, a warning appears during generation.

Requirements

A screen has been created and opened.

Procedure

1. Select the "Effective range name" object from the "Elements" category in the toolbox.
2. Drag and drop the "Effective range - Name" object from the toolbox into the screen.

Result

If the operator clicks an object during runtime, he initiates logon or logoff of the HMI device to a machine that is assigned to an effective range.

12.9.9.4 Configuring additional Mobile Wireless objects

Introduction

You will need to configure other objects in order to display further information about the effective range during testing, for example.

Requirements

A screen has been created and opened.

Procedure

1. Select the "Effective range - Signal" object from the "Elements" category in the toolbox.
2. Drag and drop the "Effective range - Signal" object from the toolbox into the screen.
3. Repeat steps 1 and 2 with the "WLAN - Reception" object.

Result

The "Effective range - Signal" object indicates how accurately the Mobile Panel Wireless is within an effective range. The "WLAN - Reception Quality" object displays the signal strength of the WLAN wireless connection.

12.9.10 Working with effective ranges (RFID)

12.9.10.1 Overview

Validity

The next section applies only to Mobile Panels Wireless with RFID, for example, Mobile Panel 277F IWLAN (RFID Tag).

Introduction

The following configuration guide describes the steps required to set up an effective range (RFID) for fail-safe operation on a Mobile Panel Wireless with RFID.

Procedures overview

1. Configure effective range (RFID):
   You configure the effective range (RFID) "MixingAxisControl".
2. Configuring the name of the effective range (RFID):
   To ensure logon to the effective range, configure the object "Effective range - Name (RFID)".
3. Set parameters for loading
4. Configure the data channel
5. Configure network operation
6. Install RFID Tag in the plant
7. Turn on the HMI device and enable loading mode
8. Loading a project
9. Write the suitable IDs to the RFID Tags in the plant
10. Determine the checksum
11. Load the project once again with checksum:
    Enter the checksum you have determined in the project and load the project once again.
12. Test effective range (RFID)

Note

For detailed information on items 4 to 12, refer to the operating instructions of the HMI device.

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

12.9.10.2 Configuring effective range (RFID)

Introduction

You set up an effective range (RFID) to control safety-related operator input upstream of a machine or plant.

Requirements

- A suitable editing language, e.g. English, must have been set in the "Project languages" editor.
- The "Effective ranges (RFID)" work area is open.

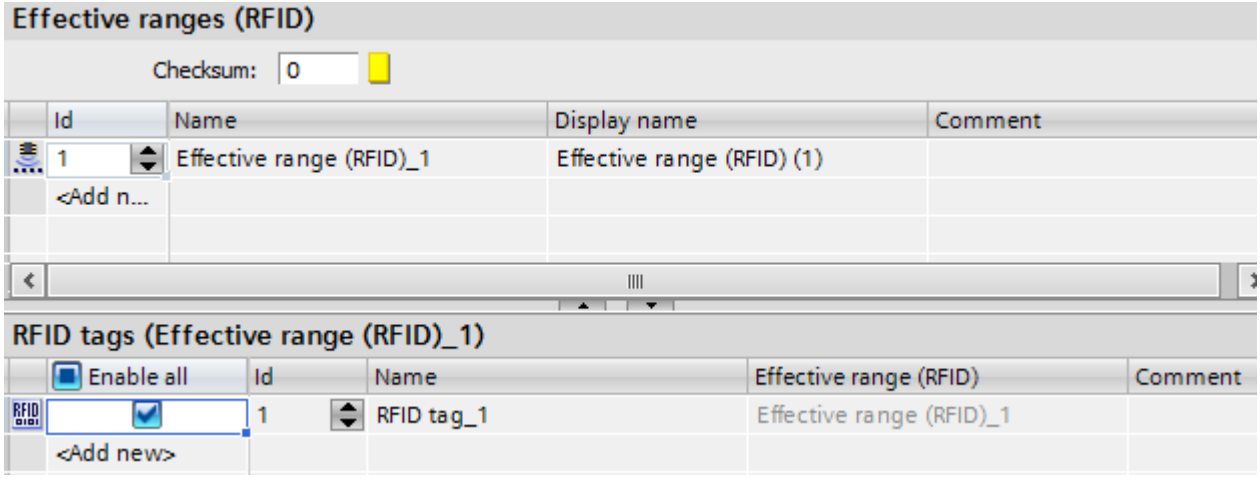
Procedure

1. Double-click "Add..." in the "Effective Ranges (RFID)" table.
2. Enter "MixingAxisControl" as the "Name" of the effective range (RFID).
3. Enter "MixingAxisControl" in the "Display name" field of the effective range (RFID).

4. Enter "1" in the "ID" field of the effective range (RFID).
   Make sure that the effective range (RFID)-ID matches the ID on the machine.
5. Double-click "Add..." in the "RFID Tag table.
6. Enter "RFID Tag 1" as the "Name" of the RFID Tag.
7. Enter "1" as the "ID" of the RFID Tag. Then set the RFID Tag-ID for the RFID Tag.
8. Enable the assignment of "RFID Tag 1".
   "RFID Tag 1" is assigned to the effective range (RFID) "MixingAxisControl".

Result

The effective range (RFID) "MixingAxisControl" is configured.



The display name describes the effective range (RFID) in the set editing language. An overview of the display names can be found in the "Project texts" editor.

12.9.10.3 Configuring the effective range name (RFID)

Introduction

Configure the "Effective range - Name" object so that an operator can log on and off a machine using an effective range (RFID). If you do not configure the "Effective range-Name" object for a Mobile Panel Wireless RFID with (RFID), a warning will appear during generation.

Requirements

A screen has been created and opened.

Procedure

1. Select the "Effective range - Name (RFID)" object from "<Elements" in the toolbox.
2. Drag and drop the "Effective range - Name (RFID)" object from the toolbox into the screen.

Result

If the operator clicks an object during runtime, he will initiate logon or logoff of the HMI device to a machine which has been assigned to an effective range (RFID).

12.9.11 Reference

12.9.11.1 PROFIsafe address

Validity

The PROFIsafe address is only available for fail-safe operation for the Mobile Panel Wireless.

Note

Additional information on fail-safe operation can be found in the operating instructions for the HMI device:

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

PROFIsafe address

The PROFIsafe address used for fail-safe I/Os in PROFIsafe communication is unique on the entire network and on all stations. On the Mobile Panel Wireless, you create a default value in the project and load it to the HMI device. You can also store a PROFIsafe address on the HMI device.

The PROFIsafe address fall within the range from 1 to 65534. The two PROFIsafe addresses (in the Engineering System and on the HMI device) are checked for formal validity.

The PROFIsafe address is loaded as follows at the start of Runtime:

- You programmed a valid PROFIsafe address in the Control Panel.
  The HMI loads the PROFIsafe address set in the Control Panel.
- You did not program a valid PROFIsafe address in the Control Panel.
  The HMI loads the PROFIsafe address set in the project.

  Note

  The invalid address 65.535 is programmed by default in the Control Panel of the HMI device. The HMI loads the address set in the project.

See also

Power Management (Page 5978)

Zone ID / connection point ID (Page 5978)

12.9.11.2 Power Management

Validity

In the "Runtime Settings > Power Management" editor, configure the energy-saving mode of a Mobile Panel.

Power Management

The Mobile Panel Wireless can save energy in the following ways:

- Reduce brightness - minimal saving
- Switch off display - significant saving

You can also indicate after how long without operator input the power management function is enabled. The communication link is maintained. The time for "Switch monitor off" is longer than the time for "Reduce brightness".

See also

PROFIsafe address (Page 5975)

12.9.11.3 Zone ID / connection point ID

Validity

In the "Runtime Settings > General" configure an internal tag which, depending on the value, shows specific information on the display of the HMI device.

Note

Mobile Panels Wireless with RFID

For Mobile Panels Wireless with RFID, the area "Zone ID / Connection point ID" is not available.

Zone ID / connection point ID

This field is used to configure an internal tag that contains the following value at runtime in relation to the type of HMI device:

- The ID of the connection box or effective range to which the hard-wired Mobile Panel is connected.
- The ID of the zone in which the Mobile Panel Wireless is located. If the value is set to 255, then the HMI device is not in any zone.

See also

PROFIsafe address (Page 5975)

12.10 Planning tasks

12.10.1 Field of application of the Scheduler

Definition

You can use the Scheduler to configure tasks to run independent of the screen in the background. You create tasks by linking system functions or scripts to a trigger. The linked functions will be called when the triggering event occurs.

Example of an application

The Scheduler is used to execute event-controlled tasks automatically. For example, you use a task to automate the following:

- Regular swap out of log data
- Printout of an alarm report when an alarm buffer overflow occurs
- Printout of a report at shift end
- Monitoring a tag
- Monitoring of a user change

Note

The availability of the listed examples is determined by the HMI device.

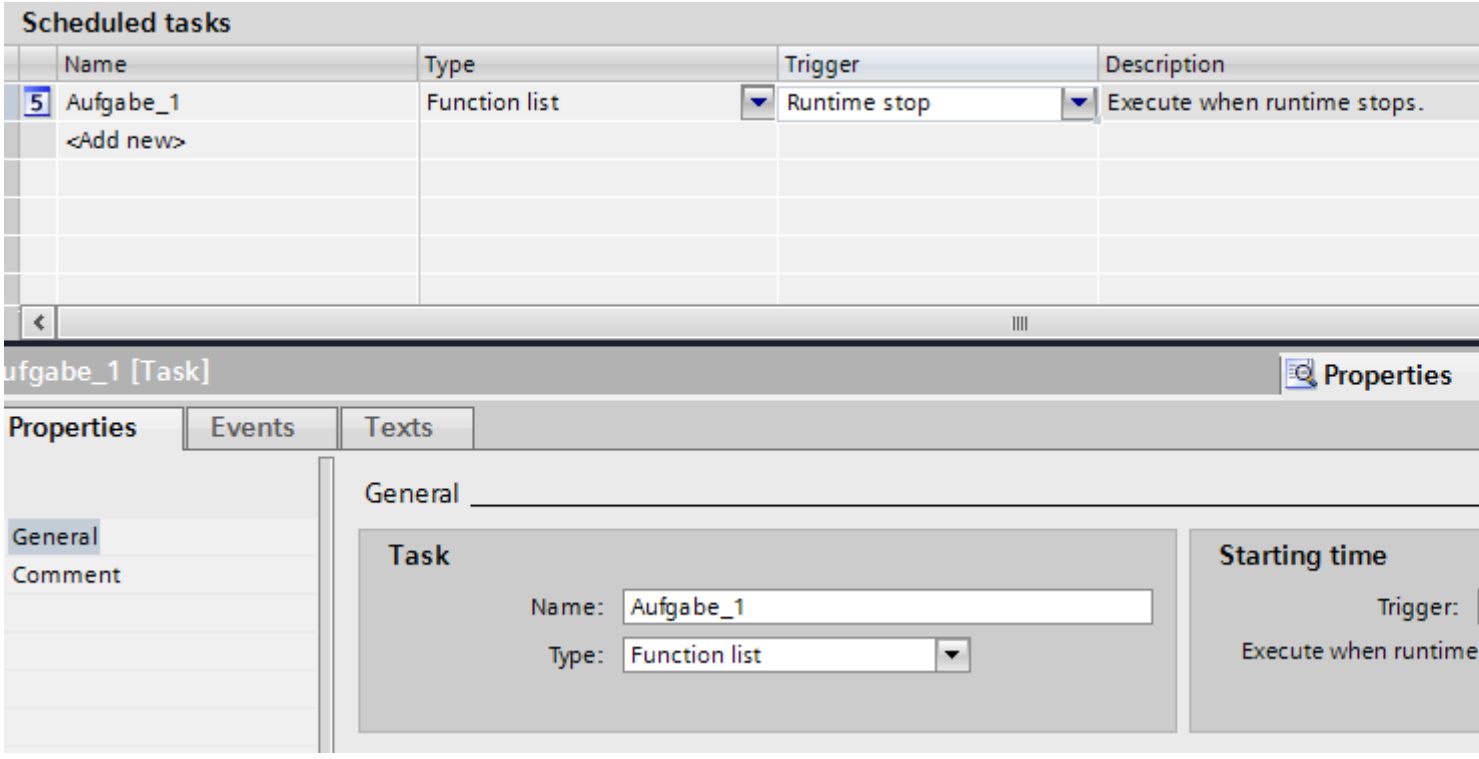
See also

Work area of the "Scheduler" editor (Page 5980)

12.10.2 Working with tasks and triggers

Introduction

A task consists of a trigger and a task type.



Starting a task

Controlled by a trigger, the Scheduler starts the task linked to the trigger.

12.10.3 Basics

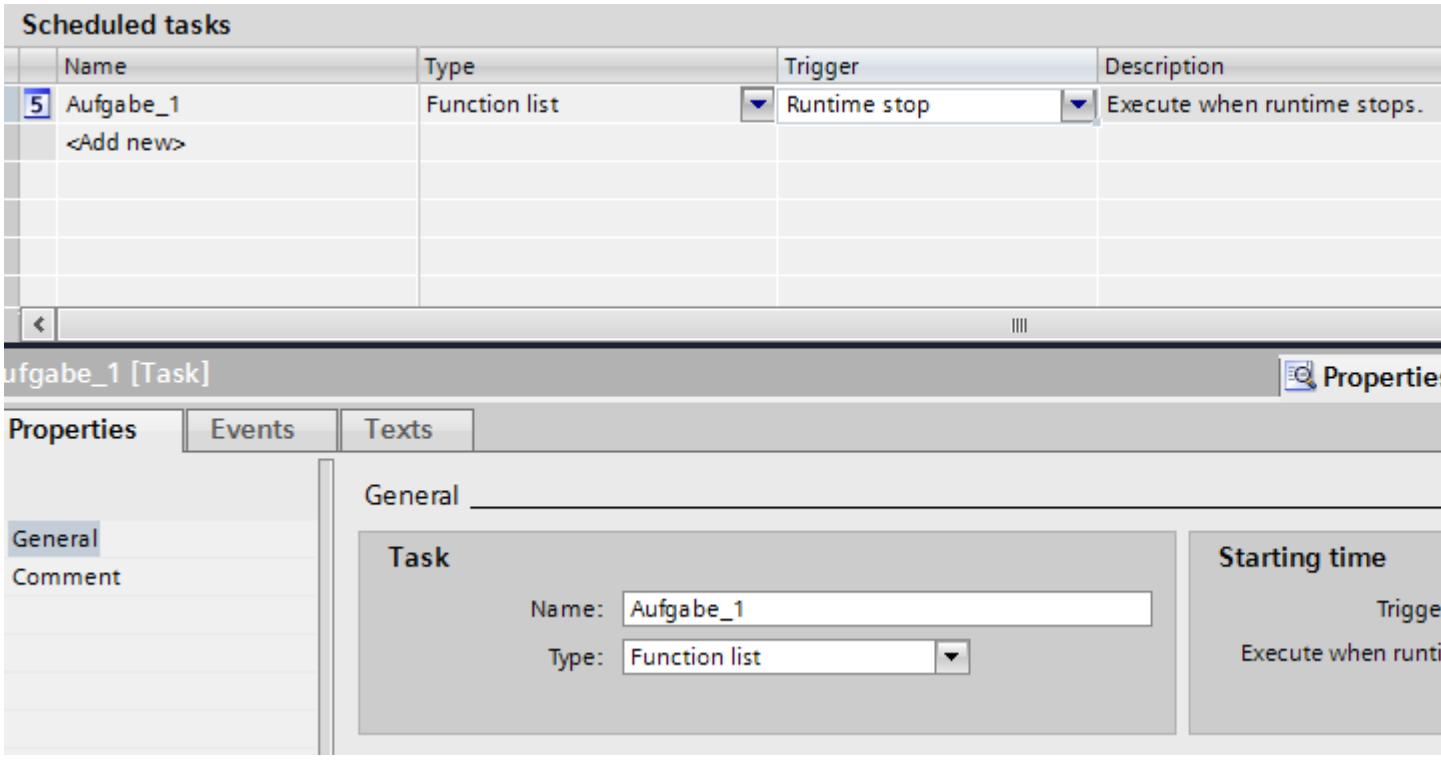
12.10.3.1 Work area of the "Scheduler" editor

Introduction

Double-click on "Scheduler" to open it in the project view. The work area shows the scheduled tasks, which consist of the trigger and the task type, for example, the function list.

Structure

The work area consists of the table of jobs.



The table of tasks shows specified tasks with their properties, such as triggers. You select a task type and a trigger. You assign a name and a comment to the task. The description provides a written summary of the task including the timing for the task.

Inspector window

The "Properties" tab of the Inspector window is split into two parts.

The "Job" area lists the name of the job and the job type. The "Starting time" area shows the trigger. The area is different depending on the trigger selected.

In the "Events" tab use the function list with system functions that will be executed in the task.

Note

You can obtain more detailed information about the elements of the user interface using the tooltips. To do so, move the mouse pointer to the relevant object or press <F1> if the object has already been selected.

See also

Field of application of the Scheduler (Page 5977)

Triggers (Page 5983)

12.10.3.2 Function list

Function list

A trigger starts the function list. The function list is executed line by line. Each line contains a system function. You can configure exactly one function list for each task.

Note

The choice of configurable system functions in a function list depends on the selected trigger and the HMI device.

12.10.3.3 Function list

Function list

A trigger starts the function list. The function list is executed line by line. Each line contains a system function or a local script. You can configure exactly one function list for each task.

Note

The choice of configurable system functions in a function list depends on the selected trigger and the HMI device.

Processing of system functions and scripts

System functions and scripts in a function list are processed in Runtime sequentially from top to bottom.

Auto-Hotspot

Use a script with loops, conditional statements and abort conditions to program non-sequential and conditional procedures.

12.10.3.4 Triggers

Triggers

Introduction

A trigger is linked to a task and forms the triggering event which will call this task. The task is executed when the trigger occurs.

Event trigger

When a task is linked to a system event, the task will be triggered by the event. System events include, for example, Runtime stop, screen change, user change, etc.

Each system event can only be configured once for each HMI device.

Deactivating job

If you do not need a certain job temporarily, deactivate the job in the Engineering System. You also use the trigger "Deactivated" to make a previously configured system event available once again.

Example: Task "A" is planned with the system event "Shutdown". This system event is then no longer available for another task "B". Select "Disabled" as the trigger for task "A" to make the "Runtime stop" system event available again.

Note

The available triggers depend on the HMI device.

See also

Work area of the "Scheduler" editor (Page 5978)

Triggers

Introduction

A trigger is linked to a task and forms the triggering event which will call this task. The task is executed when the trigger occurs.

Types of triggers

The Scheduler recognizes different types of triggers:

Acyclic triggers

They consist of a specification for the date and time of day. Tasks with an acyclic trigger are performed once by the Scheduler on the specified date and at the specified time.

Cyclic triggers

Tasks linked to a cyclic trigger occur regularly at a specific point in time.

- Standard cycle: With a standard cycle, the beginning of the first interval coincides with the start of Runtime. The length of the interval is determined by the cycle.
- Trigger with defined starting time: You specify a start time and a time interval. You can use cyclic triggers to perform tasks on a daily or weekly basis, for example. For example, you can select a "Weekly" trigger to specify a week as the interval. You specify the beginning of the interval with the day of the week and the time of day.

Note

User-defined cycles

You have defined your own cycles. You can only use cycles in the Scheduler whose maximum value amounts to one hour.

Event trigger

When a task is linked to a system event, the task will be triggered by the event. In contrast to cyclic triggers, system events usually occur acyclically. A Runtime stop is a system event, for example.

Each system event can only be configured once for each HMI device.

Deactivating job

If you do not need a certain job temporarily, deactivate the job in the Engineering System. You also use the trigger "Deactivated" to make a previously configured system event available once again.

Example: Task "A" is planned with the "User change" system event. This system event is then no longer available for another task "B". Select "Disabled" as the trigger for task "A" to make the "User change" system event available again.

Note

The available triggers depend on the HMI device.

Timers for cyclic triggers

Timers for cyclic and acyclic triggers

You have the option to change the start time of a task dynamically in Runtime. You can select an HMI variable as the timer. The value of the tag determines the start time for the task during Runtime.

Note

An HMI tag must be of the type "DateTime". A PLC tag must be of the type "Date and Time" or "DTL".

As long as the operator does not change the tag after Runtime starts, the start time configured in the task is valid. As soon as the operator enters a change to the tag, the current value of the tag is always the start time. To reset the configured time as the start time, the operator has to enter the configured time again.

12.10.4 Planning jobs

12.10.4.1 Planning tasks with acyclic triggers

Introduction

You are planning a one-time Runtime stop for maintenance work. The task starts a function which requires Runtime to stop.

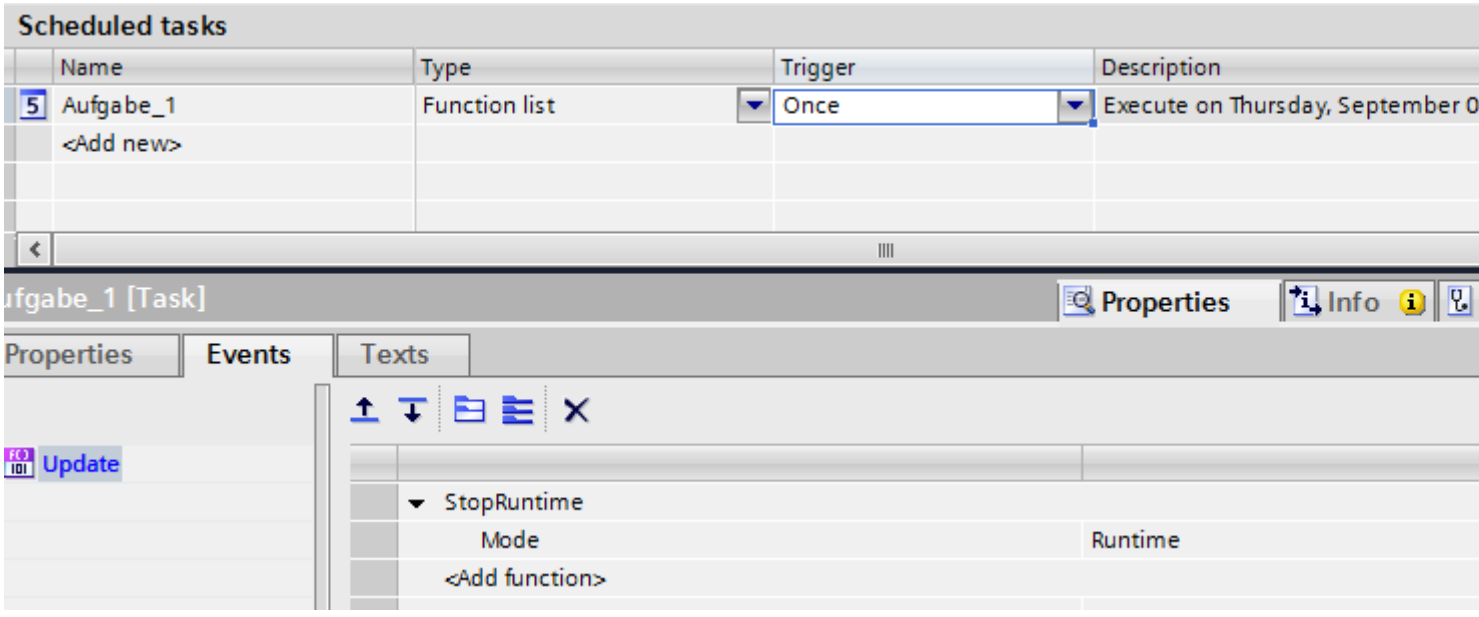
Requirements

- The "Scheduler" editor is open.
- The Inspector window is open.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Maintenance at end of year" as the "Name."
3. Select "Once" as the "Trigger."
4. Select "31.12.2008" in the "Properties > Properties > General > on" field of the Inspector window.
5. Enter the time "18:00" in the "at" field.
6. Click "Properties > Events" in the Inspector window.

7. Select the "StopRuntime" system function in the function list.
8. Select "Runtime" as the "Mode".



Result

The task is executed once. Runtime will be stopped at 18:00 on December 31, 2008.

Note

"StopRuntime" event on RT Professional

Please note that the "StopRuntime" event means runtime is in the process of ending and that some functionalities are therefore no longer available.

12.10.4.2 Planning tasks with cyclic triggers

Introduction

You plan a job that provides a weekly printout of a report.

Requirements

- A "Weekly report" report has been created.
- The "Scheduler" editor is open.
- The Inspector window is open.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Weekly report printout for end of shift" as the "Name."
3. Select "Weekly" as the "Event".
4. Open the "Properties" tab in the Inspector window.
5. Select "Friday" in the "on day" field of the "Starting point" area.
6. Enter the time "18:00" in the "at" field in the "Starting point" area.
7. Click in the "Events" tab of the "Function list" on the entry "No function."
8. Select "Print/PrintReport" as the function.
9. Select "Weekly report" report.

Result

The task is executed weekly. The report "Weekly report" will be printed every Friday at 18:00 hours.

12.10.4.3 Planning tasks with event triggers

Introduction

You plan a task that generates a screen change when the user changes.

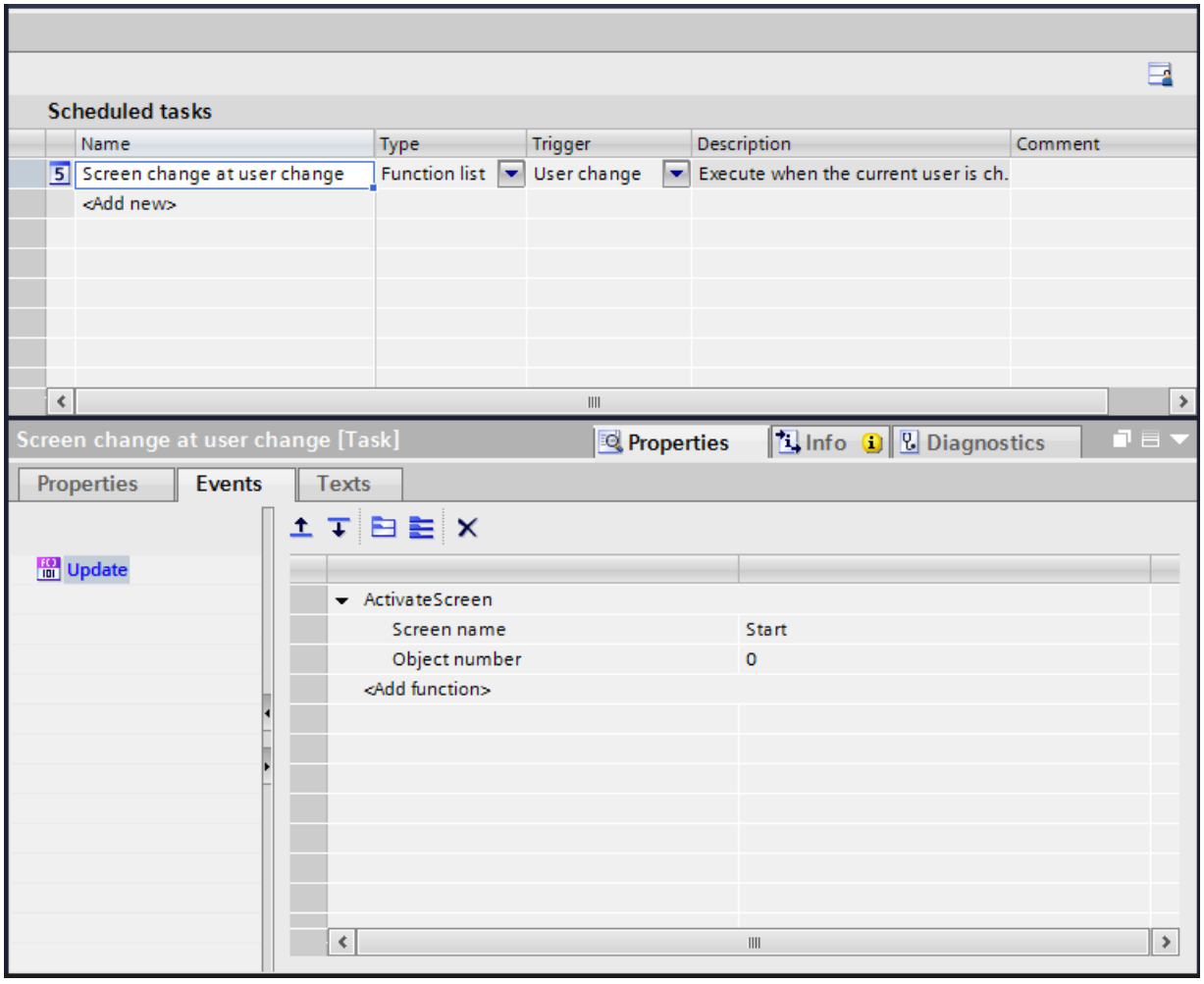
Requirements

- The "Scheduler" work area is open.
- You have created the "Start" screen.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Screen change at user change" as the "Name."
3. Select "User change" as the "Trigger."
4. In the Inspector window, select "Properties > Events".

5. Select the "Screen/ActivateScreen" system function in the function list.
6. Select the "Start" screen in the screen name field.



Result

The task is executed with the "User change" event. When a new user logs on successfully, the "Start" screen is called up.

12.10.4.4 Administer task

Changing the designation

1. In the work area, double-click the field in the "Name" column.
2. Change the name of the task.
3. Confirm your entry with <Return>.

As an alternative, you can change the designation in the "Job > Name" box of the Inspector window.

Changing triggers

1. In the work area, mark the field in the "Trigger" column.
2. Open the drop-down list using the button and select the trigger you require.

You can alternatively change the trigger in the "Starting time" area of the Inspector window.

Deleting a task

1. Select one or more lines for the tasks to be deleted.
2. Open the shortcut menu with the right button and select the menu command "Delete".

As an alternative, delete one or more tasks by using the <Del> button.

12.10.5 Examples

12.10.5.1 Example: Terminating Runtime every day

Task

You plan a job that terminates Runtime every day at the end of work.

Requirements

- The "Scheduler" work area is open.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Runtime stop at end of work" as the "Name."
3. Select "Daily" as the "Event."
4. Open the "Properties" tab in the Inspector window.
5. Enter the time "21:00" in the "at" field in the "Starting point" area.
6. Open the "Events" tab in the Inspector window.
7. Click the "No function" entry in the "Function list".
8. Select "System functions/System/StopRuntime" as the function.

Result

Runtime is terminated every day at 21:00.

12.10.5.2 Example: Update user following change of user

Task

Configure an I/O field which displays the logged on user. Configure a task which updates the I/O field when the logged on user changes.

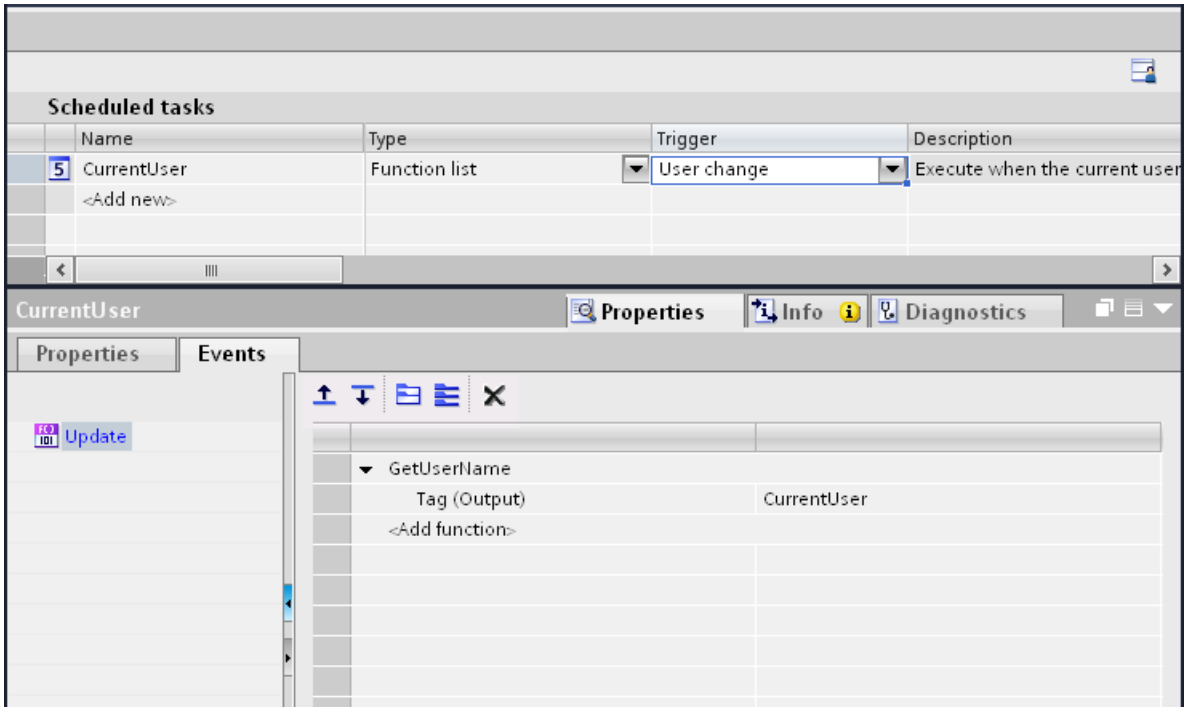
Requirements

- A "CurrentUser" tag of the "String" type is created.
- A screen has been created and opened.
- An I/O field is created in the screen.

Procedure

1. Click on the "I/O field" object.
2. In the Inspector window, select "Properties > Properties > General":
   - Select "String" as the "Display format."
   - Select "CurrentUser" as the "Variable."
   - Select "Output" as the mode.
3. Change to the work area of the Scheduler.
4. Click "Add..." in the table of the task area.
5. Enter "CurrentUser" as the "Name".
6. Select "User change" as the "Trigger."
7. In the Inspector window, select "Properties > Events".

8. Select the system function "ReadUserName" from the "User Management" group in the function list.
9. Select "CurrentUser" as the "Variable."



Result

When a new user logs on successfully, the "ReadUserName" function is called up. The "CurrentUser" tag is updated and displayed in the I/O field of the newly logged on user.

If a user does not log on successfully, the logged on user is logged off. The I/O field continues to display the user previously logged on until a new user logs on successfully.

12.10.5.3 Example: Changing the starting point of a job in Runtime

Example: Changing the starting point of a task in Runtime

Introduction

A cyclic trigger, for example "Starting daily at 18:00," is configured in the Engineering System. The time "18:00" is fixed as a constant. To change the start time in Runtime, select a tag as the "Timer." The value of the tag determines the start time of the task during Runtime.

Task

You plan a task that swaps out log data daily and has the "LogTime" tag as the timer. You configure a "LogTime" tag in an input field.

Requirements

- The "DataLogSource" source log has been created as a data log.
- The "DataLogDestination" destination log has been created as a data log.
- A "ChangeLogTime" screen has been created.

Procedures overview

1. Creating a timer tag: You create a tag with which you change the starting point of the log data swap in Runtime.
2. Configuring a Date/time field: The operator changes the starting point of the task by using the Date/Time field in Runtime.
3. Configure the task with a timer tag: You create a task whose start time can be changed dynamically in Runtime.

Result

If nothing is entered in the "LogDataTimeField" input field, then the task starts daily at 18:00 by default. The log data are swapped out. If an operator enters 19:00 in the "LogDataTimeField" input field, then the log data are swapped out at 19:00. Provided the value of the tag is not changed again before the task starts.

See also

Example: 2. Configuring a Date/time field (Page 5993)

Example: 1. Configuring a tag for Runtime

Task

In the following example you configure the "LogTime" variable.

Procedure

1. Open the "HMI tags" editor.
2. Click "Add..." in the table of the task area.
3. Enter "LogTime" as the "Name" of the variable.
4. Select "Internal variable" in the "Connection" column.
5. Select "DateTime" in the "Date type" column.

Result

A variable of the "DateTime" type has been created.

Example: 2. Configuring a Date/time field

Task

In the following example you configure a Date/time field. To change the starting point of the task in Runtime, link the "LogTime" variable to the Date/time field.

Requirements

- The "LogTime" variable of the "DateTime" type has been created.
- The "ChangeLogTime" screen is open.

Procedure

1. In the toolbox view, drag-and-drop a "Date/time field" from the "Basic objects" category to the screen.
2. In the Inspector window, click "Properties >Properties > General".
3. Disable the "System time."
4. Click "Variable" and select "LogTime."
5. Disable "Display date" and enable "Display time."
6. Select "Input/output" as the "Mode."
7. Click "Properties > Miscellaneous."
8. Enter "LogDataTimeField" as the "Name".

Result

The operator enters a time using the created date/time field.

See also

Example: Changing the starting point of a task in Runtime (Page 5989)

Example: 3. Configuring a swap to archive with a timer variable

Task

In the following example you configure a task with a timer tag with which you change the starting point of the task in Runtime.

Requirements

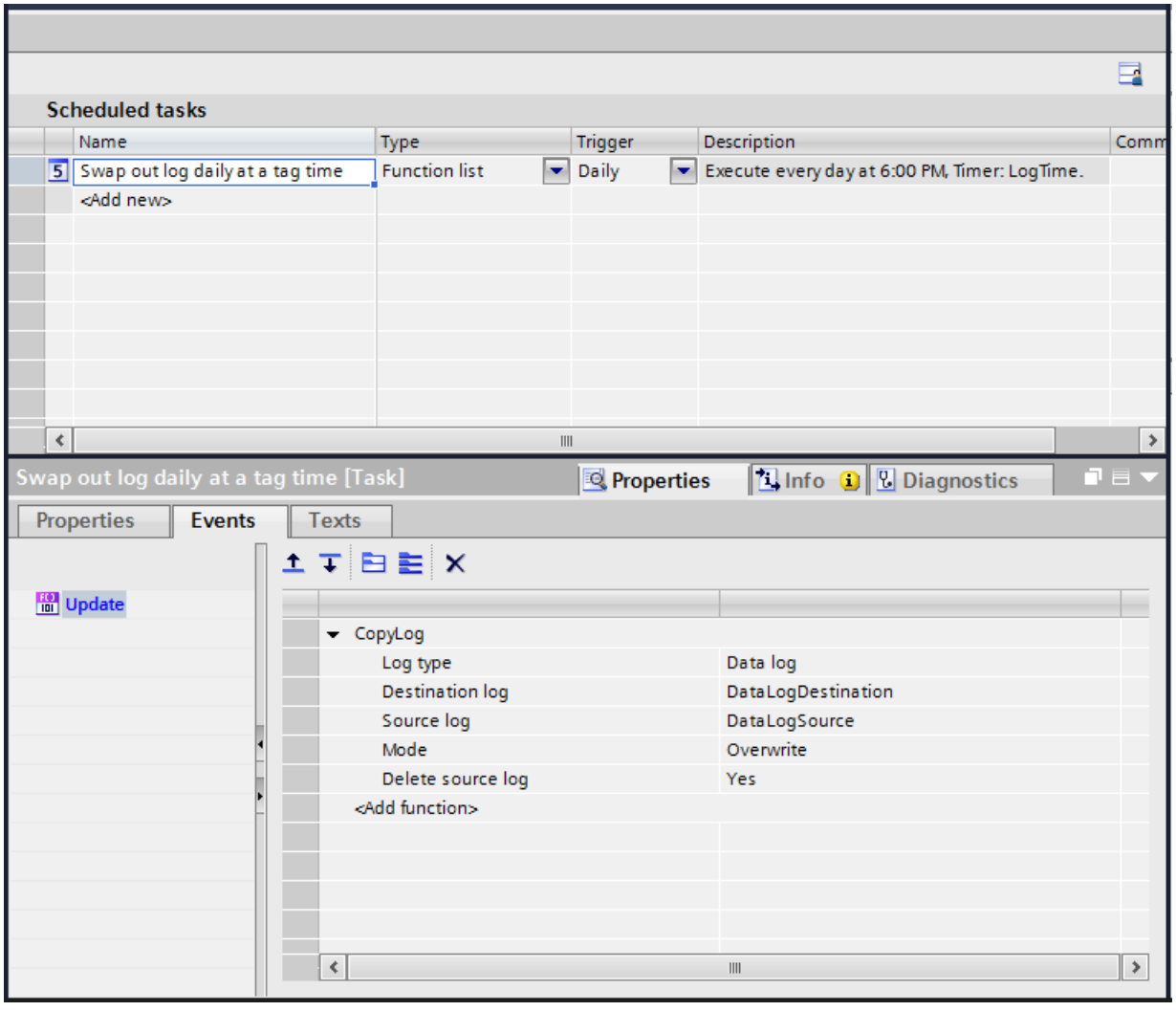
- The "LogTime" tag has been created.
- The "DataLogSource" source log has been created as a data log.

- The "DataLogDestination" destination log has been created as a data log.
- The Scheduler work area is open.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Swap out log daily at a tag time" as the "Name".
3. Select "Daily" as the "Trigger".
4. In the Inspector window "Properties >Properties > General > at" enter the time "18:00".
5. Select the "LogTime" tag as the "Standard timer".
6. In the Inspector window, open "Properties > Events".

7. Select the "System functions/Logs/CopyLog" system function in the function list.
8. Select these settings:
   - Select "Data log" as the "Log type".
   - Select "DataLogDestination" as the "Destination log".
   - Select "DataLogSource" as the "Source log".
   - Select "Overwrite" as the "Mode".
   - Select "Yes" for "Delete source log".



Result

If nothing is entered in the "LogDataTimeField" input field, then the task starts daily at 18:00 by default. The log data are swapped out. If an operator enters 19:00 in the "LogDataTimeField" input field, then the log data are swapped out at 19:00. Provided the value of the tag is not changed again before the task starts.

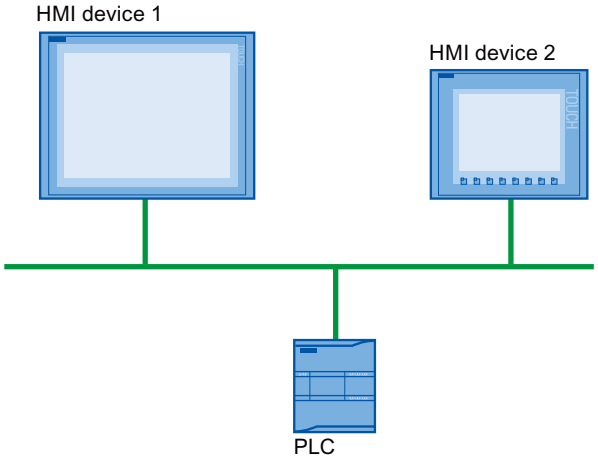
12.11 Communicating with PLCs

12.11.1 Basics of communication

12.11.1.1 Communication between devices

Communication

The data exchange between two devices is known as communication. The devices can be interconnected directly or via a network. The interconnected devices in communication are referred to as communication partners.



Data transferred between the communication partners may serve different purposes:

- Display processes
- Operate processes
- Output alarms
- Archive process values and alarms
- Document process values and alarms
- Administer process parameters and machine parameters

Communication partners

Communication between the following devices is described in more detail in this section:

- PLC
  The PLC controls a process by means of a user program.
- HMI device
  You use the HMI device to operate and monitor the process.

Basic information for all communication

The basis for all types of communication is a network configuration. In a network configuration, you specify the connection that exists between the configured devices.

With the network configuration, you also ensure the necessary prerequisites for communication, in other words:

- Every device in a network is assigned a unique address.
- The devices carry out communication with consistent transmission characteristics.

Automation system

The following characteristics describe an automation system:

- The PLC and HMI device are interconnected
- The network between the PLC and HMI device is configured

Communication between HMI devices

The HTTP protocol is available for communication between HMI devices.

For more detailed information, refer to the documentation on the SIMATIC HMI HTTP protocol.

Communication via a uniform and vendor-neutral interface

With OPC (Openess Productivity Collaboration), WinCC has a uniform and manufacturer-neutral software interface. This interface enables standardized data exchange between industrial, office, and manufacturing applications.

For more detailed information, refer to the documentation for OPC.

12.11.1.2 Devices and networks in the automation system

Introduction

To set up an automation system, you must configure, parameterize, and interconnect the individual devices.

You insert PLCs and HMI devices into the project in the same way. Likewise, you configure the two devices in the same way.

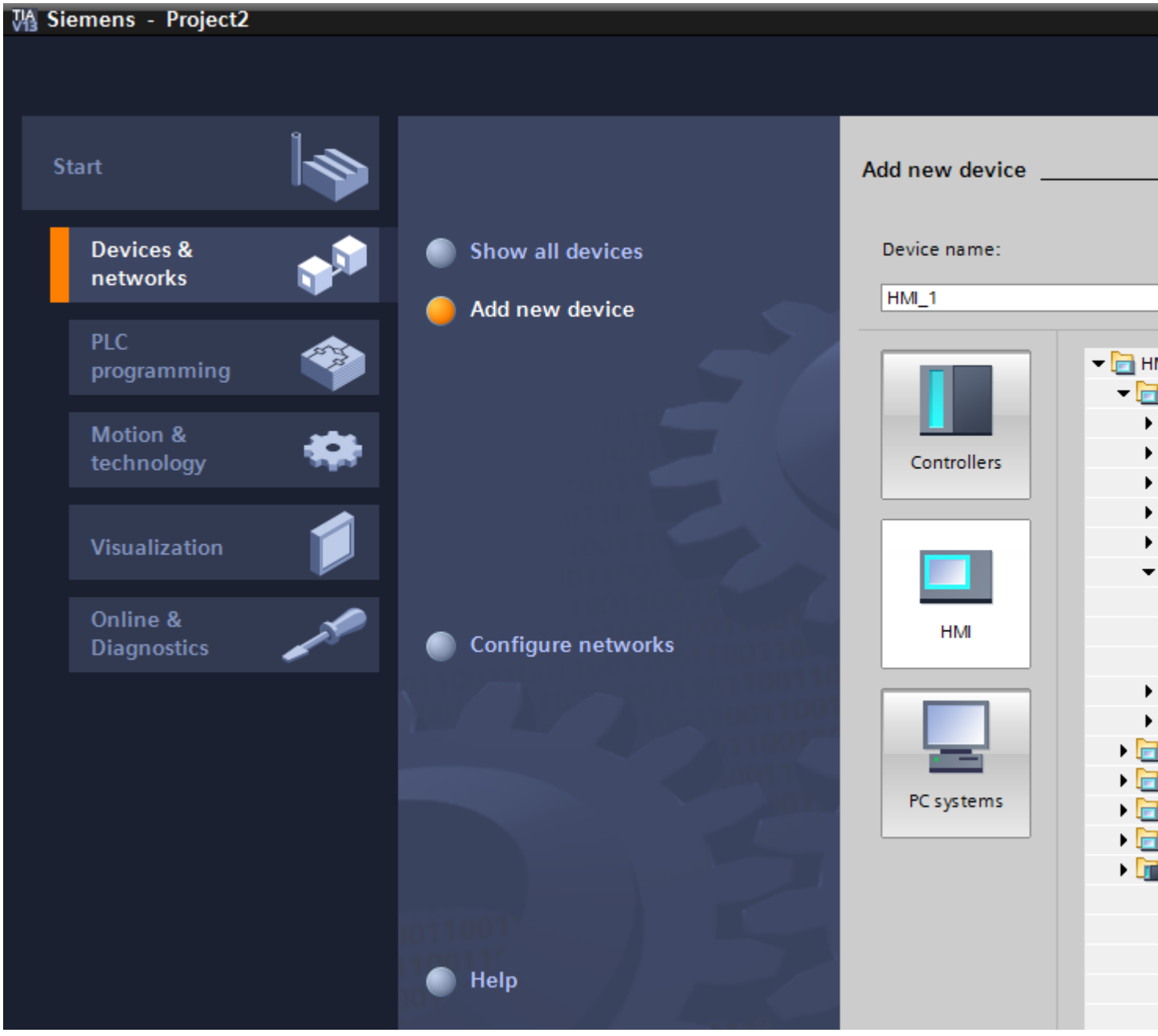
Automation system setup:

1. Insert PLC into the project.
2. Insert HMI device into the project.
3. Network the devices together.
4. Interconnect the devices.

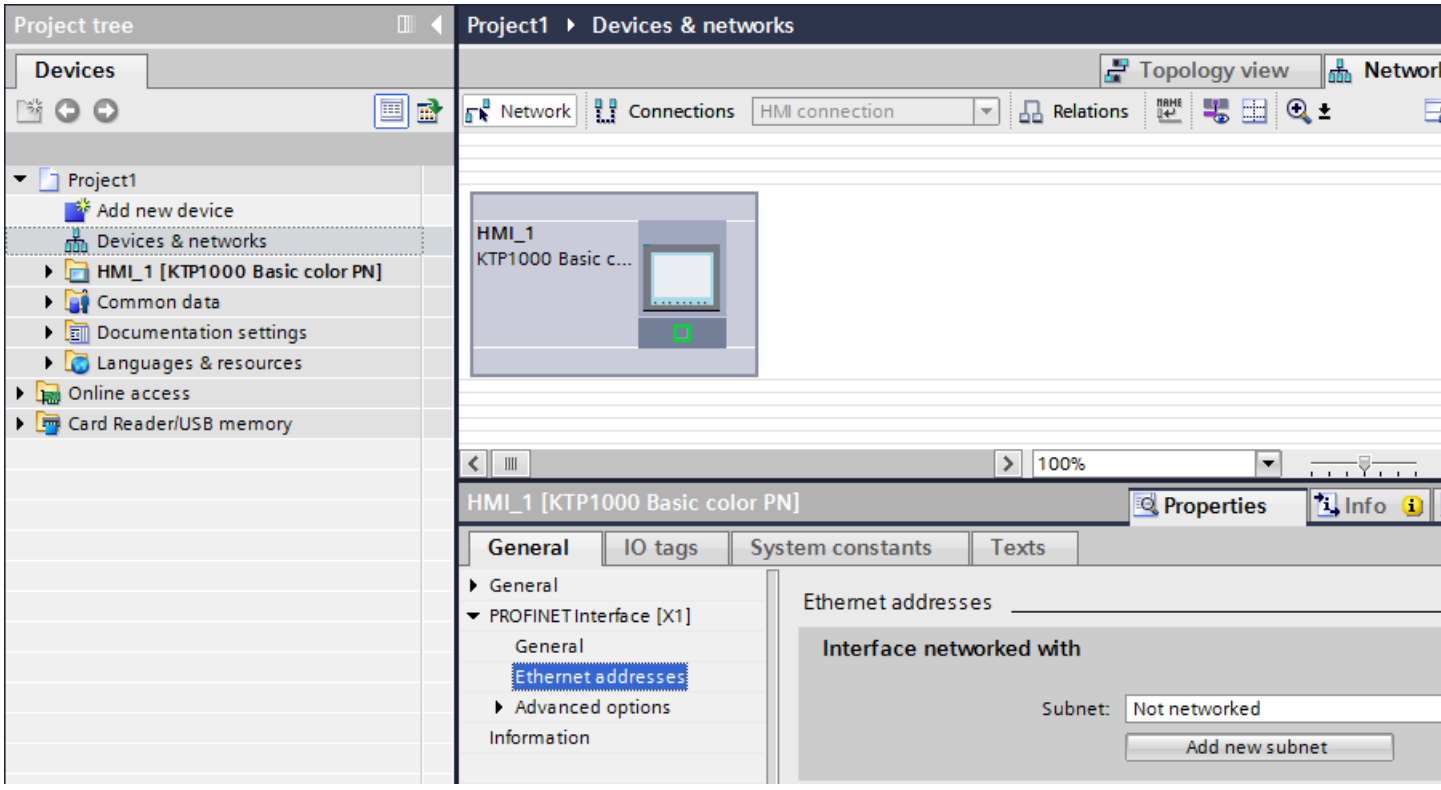
Inserting devices

If you have created a project, you can add a device in the portal view or project view.

- Portal view

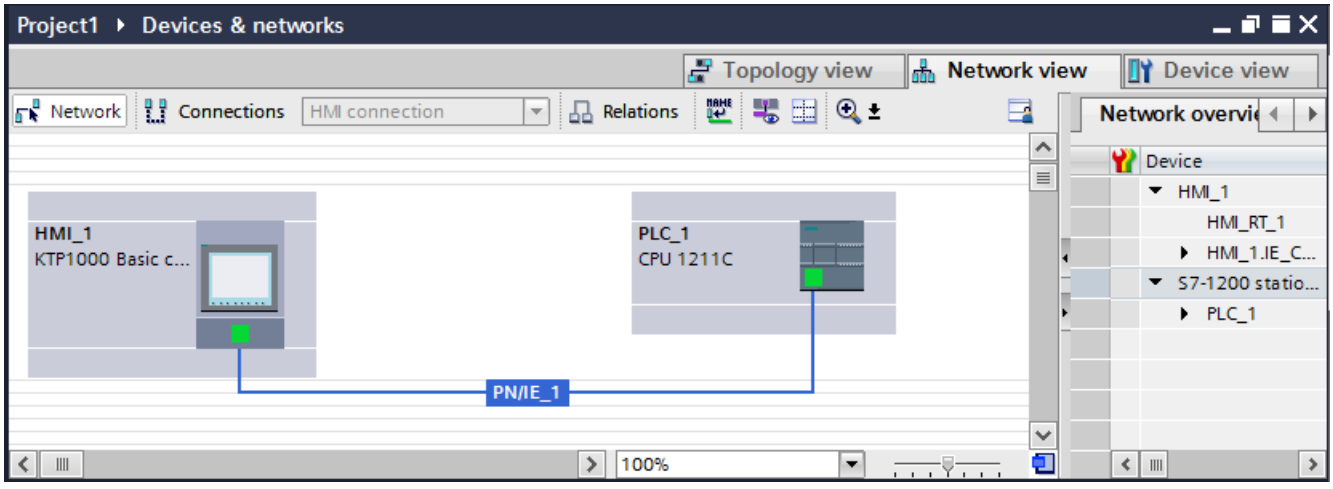


- Project view



Networking devices

You can network the interfaces of the communication-capable devices conveniently in the "Devices & Networks" editor. In the networking step, you configure the physical device connections.

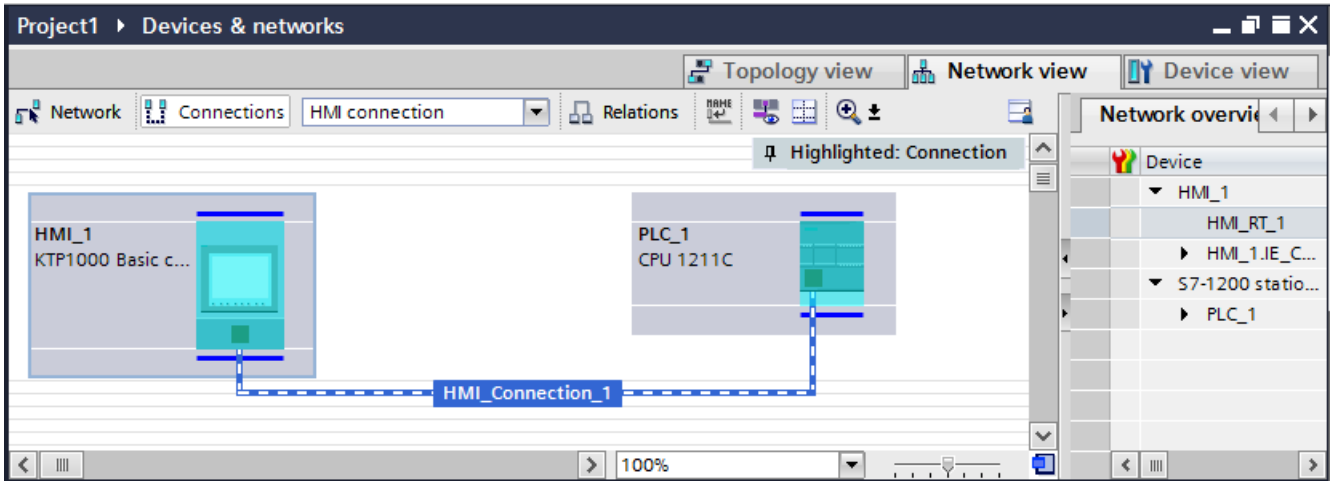


The tabular network overview supplements the graphical network view with the following additional functions:

- You obtain detailed information on the structure and parameter settings of the devices.
- Using the "Subnet" column, you can connect communication-capable components to subnets that have been created.

Connecting devices

After you network the devices together, you configure the connection. You configure the "HMI connection" connection type for communication with the HMI device.



12.11.1.3 Data exchange using tags

Communication using tags

Process values are forwarded in runtime using tags. Process values are data which is stored in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. Define external tags for processing the process values in WinCC.

WinCC works with two types of tag:

- External tags
- Internal tags

Working with tags

See the chapter "Working with tags" for further information about configuring tags.

12.11.1.4 Data exchange using area pointers

Communication using area pointers

Area pointers are parameter fields. WinCC receives information from these parameter fields in Runtime during the course of the project. This information contains data on the location and size of data areas in the PLC.

During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

The area pointers are managed centrally in the "Connections" editor. Area pointers are used to exchange data from specific user data areas.

You use the following area pointers in WinCC:

- Data record
- Date/time
- Coordination
- Job mailbox
- Date/time PLC
- Project ID
- Screen number

The availability of the various area pointers is determined by the HMI device used.

12.11.1.5 Communication drivers

Communication drivers

A communication driver is a software component that establishes a connection between a PLC and an HMI device. The communication driver thus enables the assignment of process values to HMI tags.

The interface as well as the profile and transmission speed can be chosen, depending on the HMI device used and the connected communication partner.

12.11.2 Editors for communication

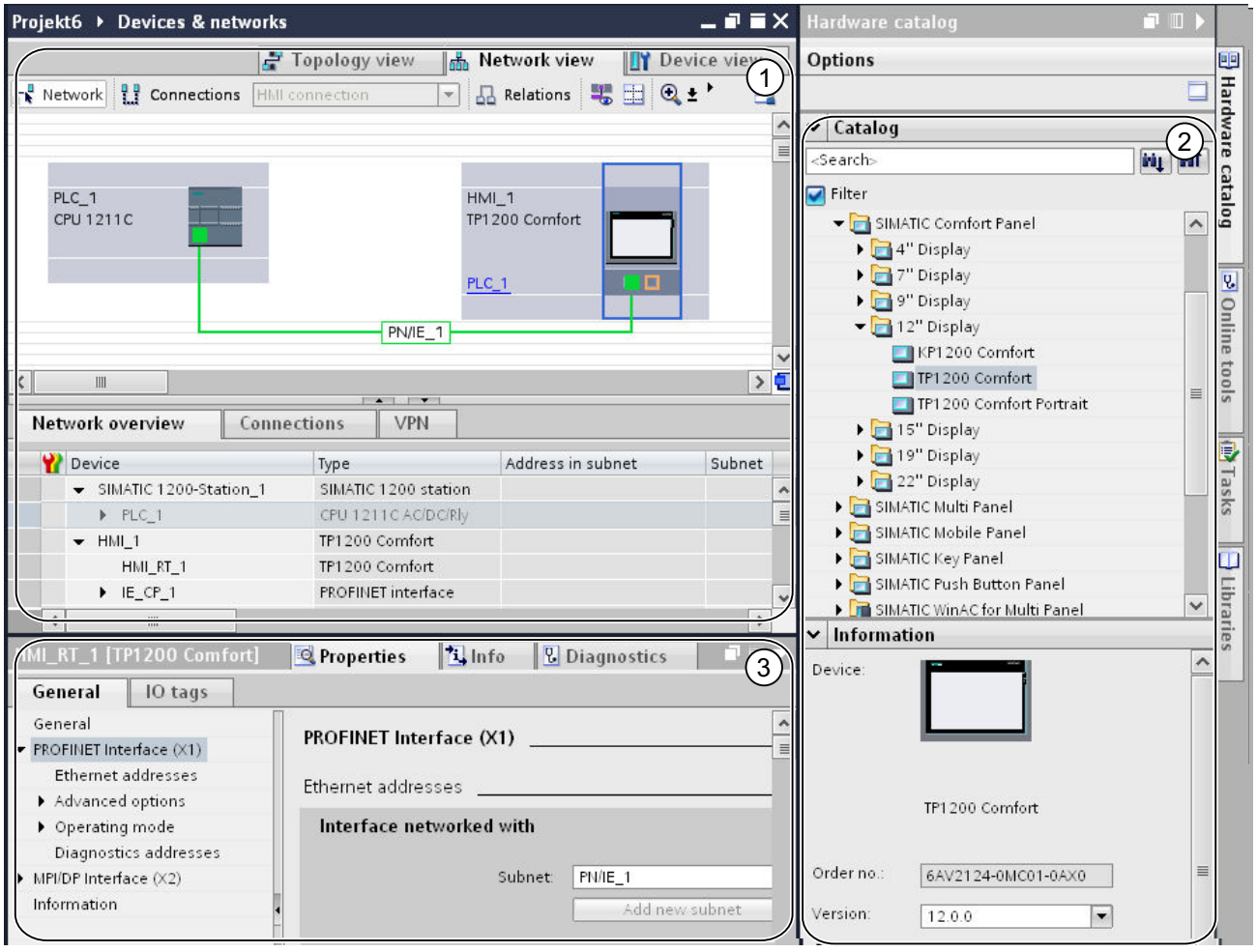
12.11.2.1 "Devices & networks" editor

Function of the hardware and network editor

The "Devices & networks" editor is the development environment for networking, configuring and assigning parameters to devices and modules.

Configuration

The "Devices & networks" editor consists of the following components:



1 Device view, network view, topology view
2 Hardware catalog
3 Inspector window

The "Devices & networks" editor provides you with three different views of your project. You can switch between these three views at any time depending on whether you want to produce and edit individual devices and modules, entire networks and device configurations or the topological structure of your project.

The inspector window contains information on the object currently marked. Here you can change the settings for the object marked.

Drag the devices and modules you need for your automation system from the hardware catalog to the network, device or topology view.

12.11.2.2 Network view

Introduction

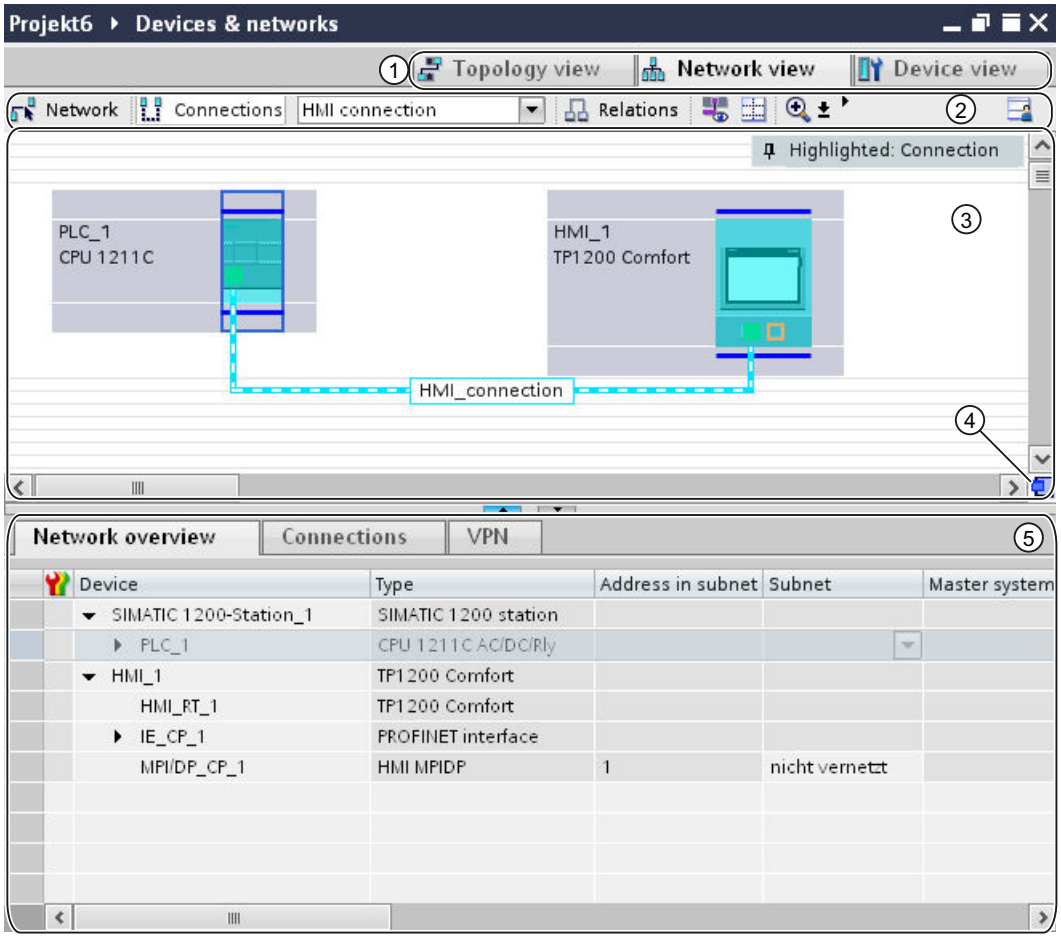
The network view is a working area of the hardware and network editor.

You undertake the following tasks in network view:

- Configuring and assign device parameters
- Networking devices with one another

Structure

The following diagram shows the components of the network view:



1 Changeover switch: network view/device view/topology view

2 Toolbar of network view

3 Graphic area of network view

4 Overview navigation

5 Table area of network view

You can use your mouse to change the spacing between the graphic and table areas of the network view.

To do this, click on the upper edge of the table view and expand or contract this by moving the mouse with the mouse button held down.

You can use the two small arrow keys to minimize, maximize or select the latest table structure of the table view with just one click.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Mode to network devices.
	Mode to create connections. You can select the connection type using the adjacent drop-down list.
	Mode to create relations.
	Show interface addresses.
	Adjust the zoom setting. You can select the zoom setting or enter it directly in the adjacent drop-down list. You can also zoom in or zoom out the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
	Show page breaks Enables the page break preview. Dashed lines will be displayed at the positions where the pages will break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the network view displays any network-related devices, networks, connections and relations. In this area, you add devices from the hardware catalog, connect them with each other via their interfaces and configure the communication settings.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

The table area of the network view includes various tables for the devices, connections and communication settings present:

- Network overview
- Connections
- I/O communication

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

12.11.2.3 Network data

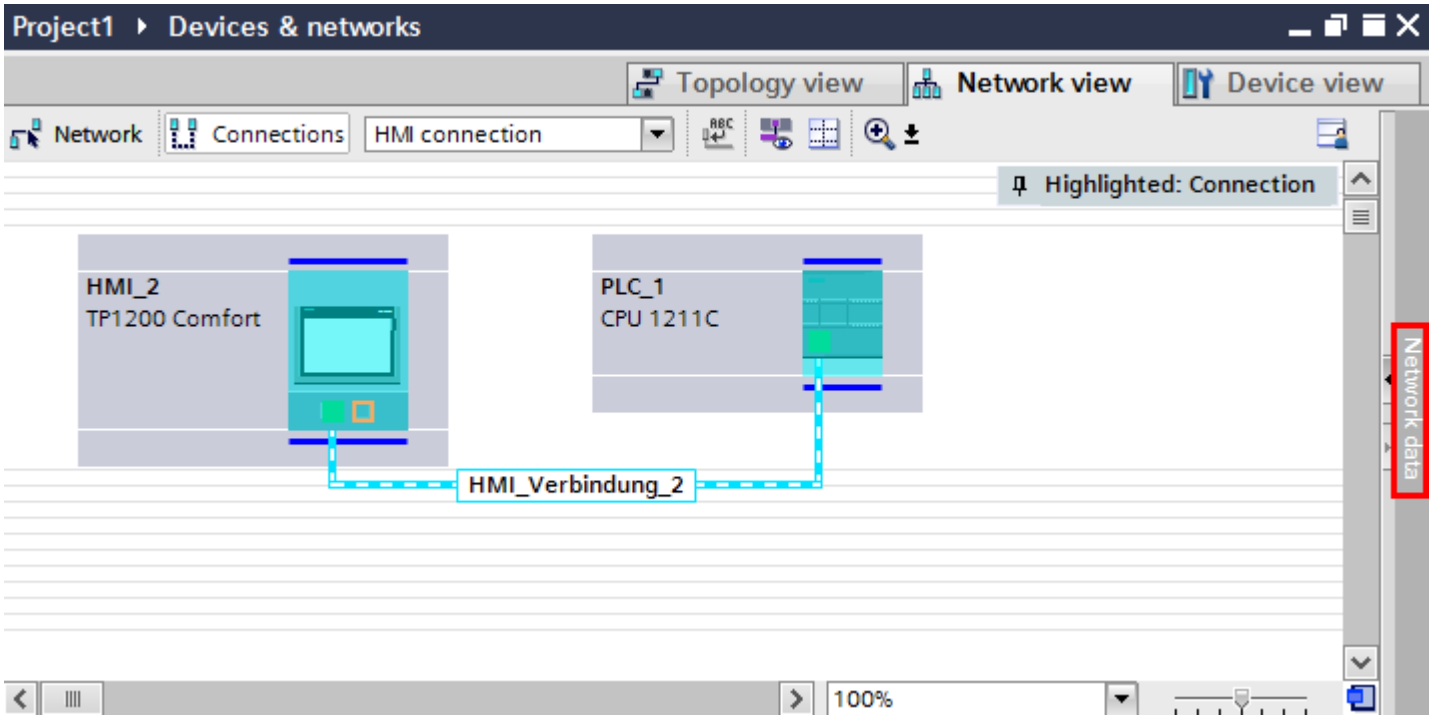
Introduction

The "Network view" editor also gives you a tabular view of the "Network data" in addition to the graphic network view.

You have the following selections in the "Network data" editor:

- Network overview
- Connections
- VPN
- I/O communication

You open the "Network data" below the graphic network overview.



Basic functions

The network data are displayed in tabular form and support the following basic functions for editing a table:

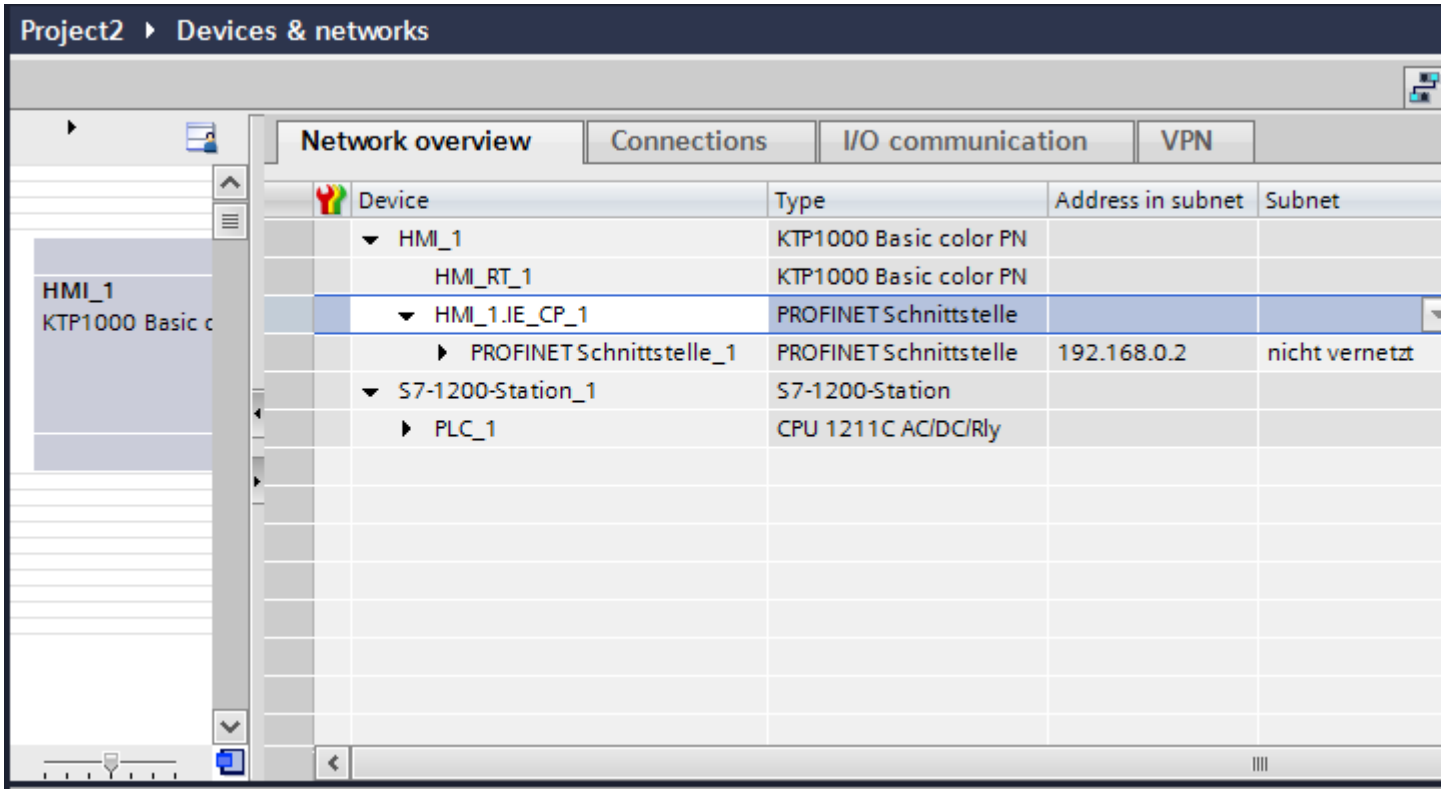
- Displaying and hiding table columns
  Note: The columns of relevance to configuration cannot be hidden.
- Optimizing column width
- Sorting table
- Displaying the meaning of a column, a row or cell using tooltips.

Network overview

The tabular network overview adds the following functions to the graphic network view:

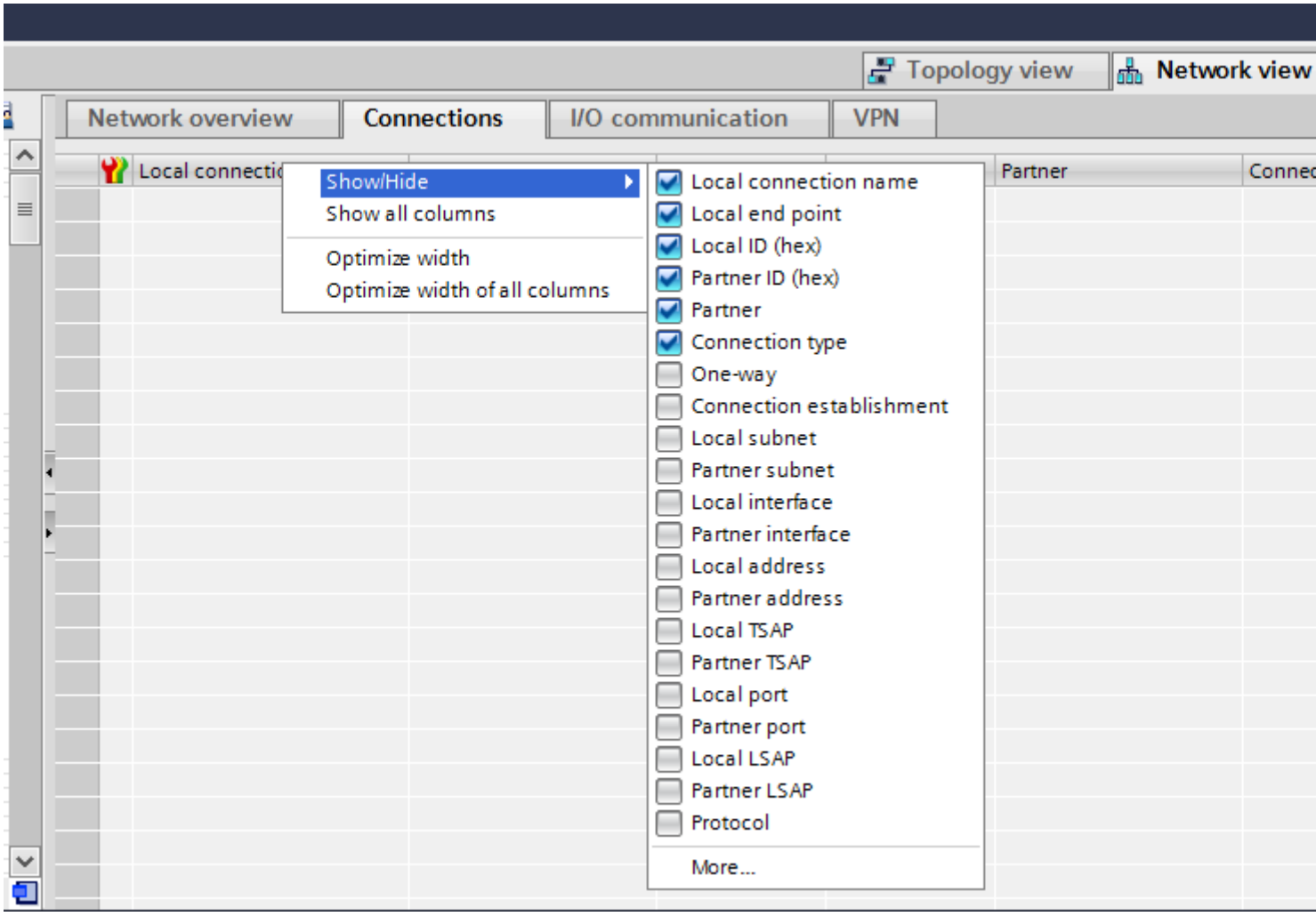
- You obtain detailed information on the structure and parameter settings of the devices.
- Using the "Subnet" column, you can connect components capable of communication with created subnets.

The



Connections

You will find additional network data under "Connections".



12.11.2.4 Diagnostics of online connections

Diagnostics of online connections

You can read out the diagnostics data of existing connections in the TIA Portal.

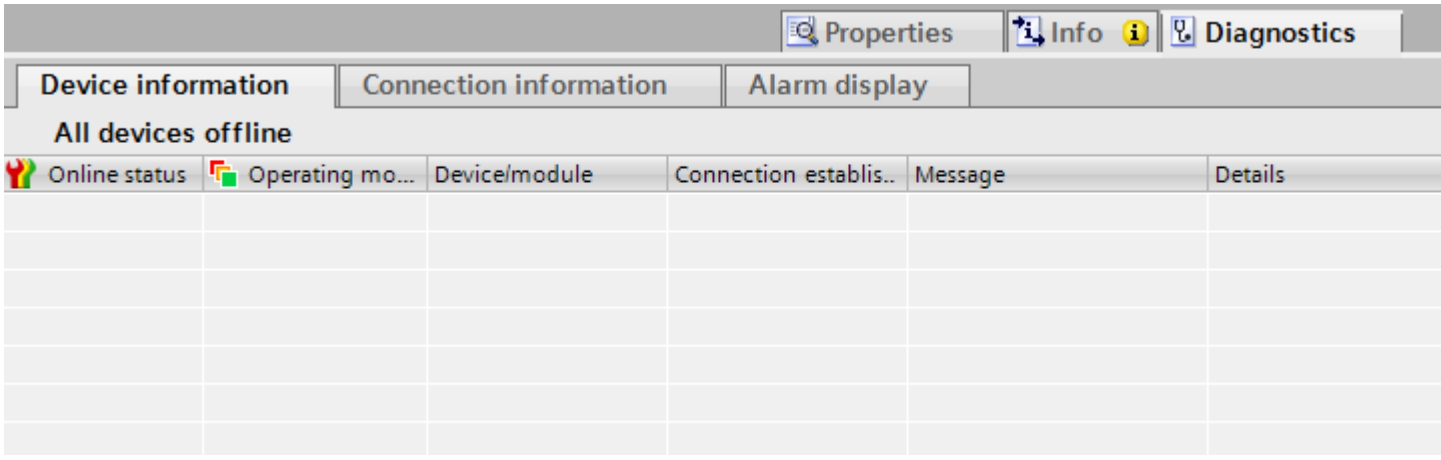
The "Diagnostics" function shows the connection data in tabular form in the Inspector window.

Requirements

- Devices must be in "Online" mode.

Device information

The diagnostics data of all devices in "Online" mode is displayed in the "Diagnostics > Device information" Inspector window.



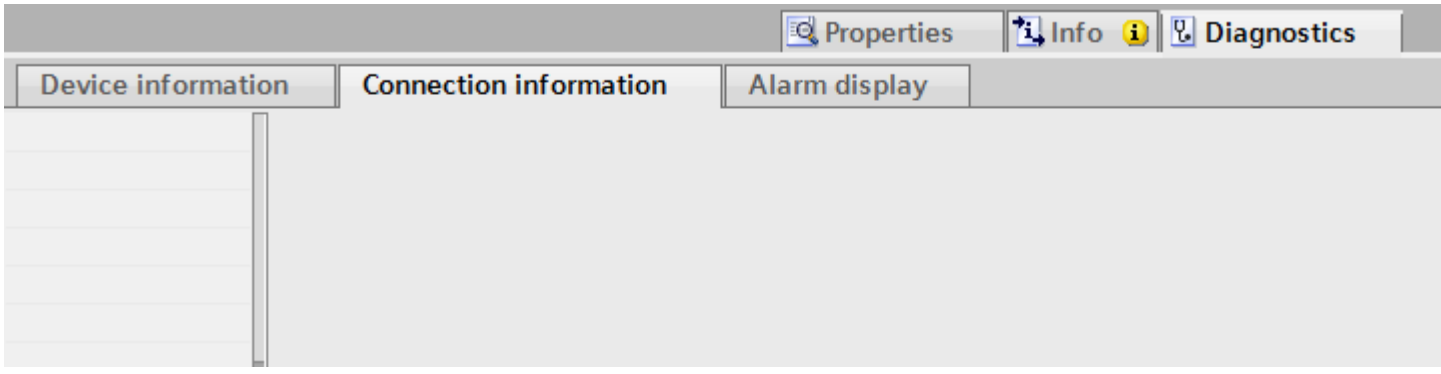
The following data is displayed:

- Online status
- Operating mode
- Device / module
- Alarm
- Details
- Help

Connection information

Use the "Connection information" function to display the diagnostics data of the connection selected in the "Devices&Networks" editor.

A graphic displays the communication partners of the connection and by which communication driver they are connected with each other.



The following data is displayed:

- End point
- Interface
- Subnet
- Address
- TSAP
- Number of HMI resources

12.11.2.5 Device view

Introduction

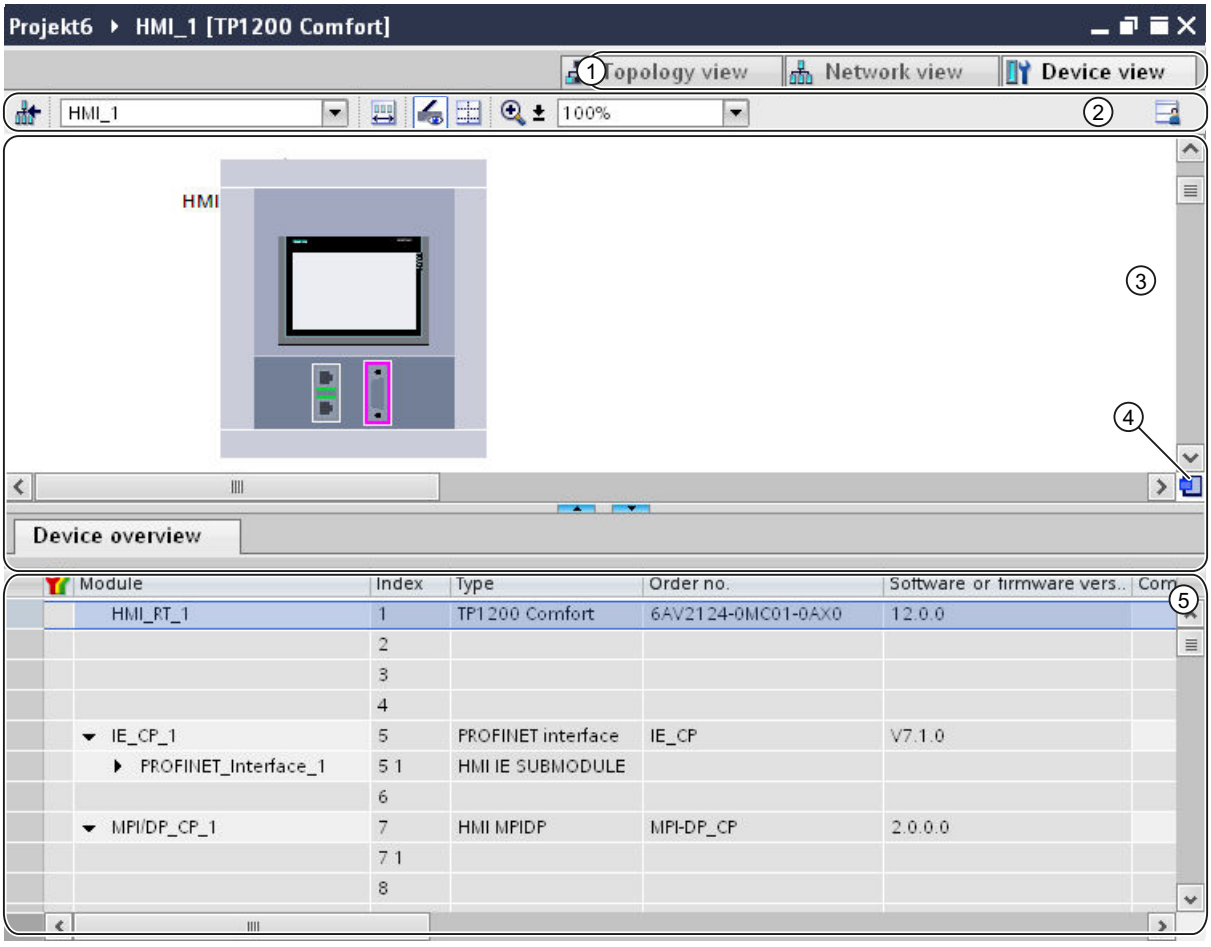
The device view is a working area of the hardware and network editor.

You undertake the following tasks in the device view:

- Configuring and assign device parameters
- Configuring and assign module parameters

Structure

The following diagram shows the components of the device view:



1 Changeover switch: network view/device view/topology view

2 Toolbar of device view

3 Graphic area of the device view

4 Overview navigation

5 Table area of device view

You can use your mouse to change the spacing between the graphic and table areas of the device view.

To do this, click on the upper edge of the table view and expand or contract this by moving the mouse with the mouse button held down. You can use the two small arrow keys to minimize, maximize or select the latest table structure of the table view with just one click.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Switches to the network view. Note: The device view can switch between the existing devices using the drop-down list.
	Show the area of unplugged modules.
	Show module labels.
	Adjust the zoom setting. Select the zoom setting or enter it directly in the adjacent drop-down list. You can use the Zoom button to zoom in or out incrementally or to drag a frame around an area to be enlarged. You can read the address descriptions of the I/O channels of signal modules at a zoom level setting of 200% or higher.
	Show page breaks Enables the page break preview. Dashed lines will be displayed at the positions where the pages will break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the device view displays hardware components and if necessary the associated modules that are assigned to each other via one or more racks. In the case of devices with racks, you have the option of installing additional hardware objects from the hardware catalog into the slots on the racks.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

The table area of the device view gives you an overview of the modules used and the most important technical and organizational data.

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

12.11.2.6 Topology view

Introduction

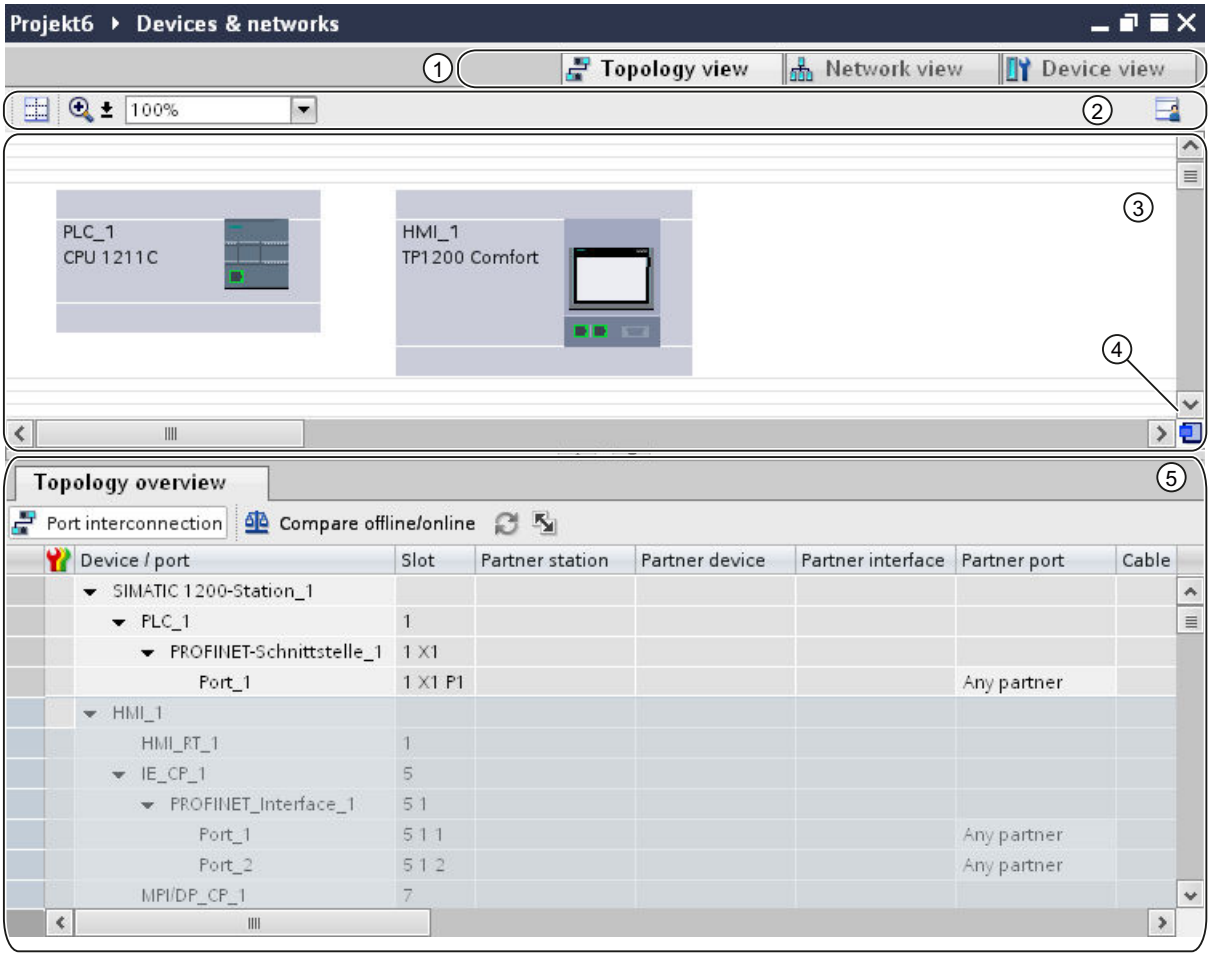
The topology view is a working area of the hardware and network editor.

You undertake the following tasks in topology view:

- Displaying the Ethernet topology
- Configuring the Ethernet topology
- Identify and minimize differences between the desired and actual topology

Structure

The following figure provides an overview of the topology view.



1 Changeover switch: device view / network view / topology view

2 Topology view toolbar

3 Graphic area of the topology view

4 Overview navigation

5 Table area of the topology view

You can use your mouse to change the spacing between the graphic and table areas of the topology view.

To do this, click on the upper edge of the table view and expand or contract this by moving the mouse with the mouse button held down. You can use the two small arrow keys to minimize, maximize or select the latest table structure of the table view with just one click.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Adjust the zoom setting. You can select the zoom setting via the adjacent drop-down list or enter it directly. You can also zoom in or zoom out the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
	Show page breaks Enables the page break preview. Dashed lines will be displayed at the positions where the pages will break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the topology view displays Ethernet modules with their appropriate ports and port connections. Here you can add additional hardware objects with Ethernet interfaces.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

This displays the Ethernet or PROFINET modules with their appropriate ports and port connections in a table. This table corresponds to the network overview table in the network view.

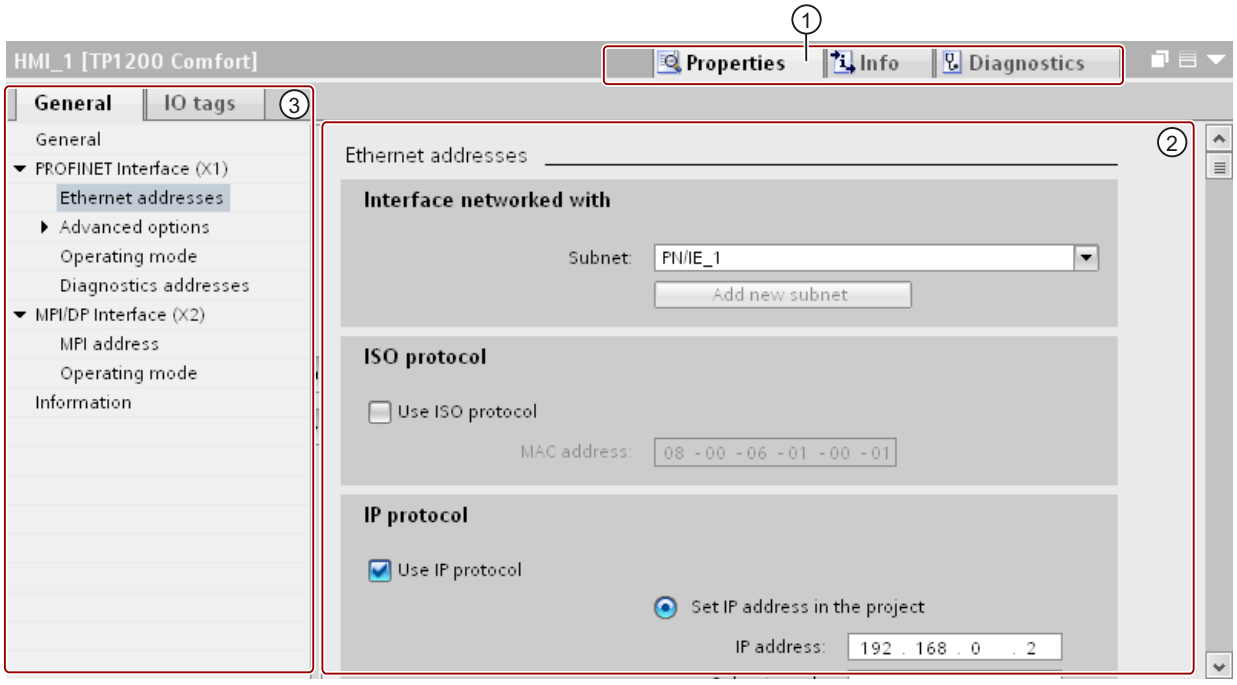
You can use the shortcut menu of the title bar of the table to adjust the tabular display.

12.11.2.7 Inspector window

The properties and parameters shown for the object selected can be edited in the inspector window.

Structure

The inspector window consists of the following components:



1	Switch between various information and work areas
2	Navigation between various pieces of information and parameters
3	Display showing the selected information and parameters

Function

The information and parameters in the inspector window are split into different types of information:

- Properties
- Info
- Diagnostics

To display the corresponding information and parameters, click in the area you want. The "Properties" area is the most important one for configuring an automation system. This area is displayed by default.

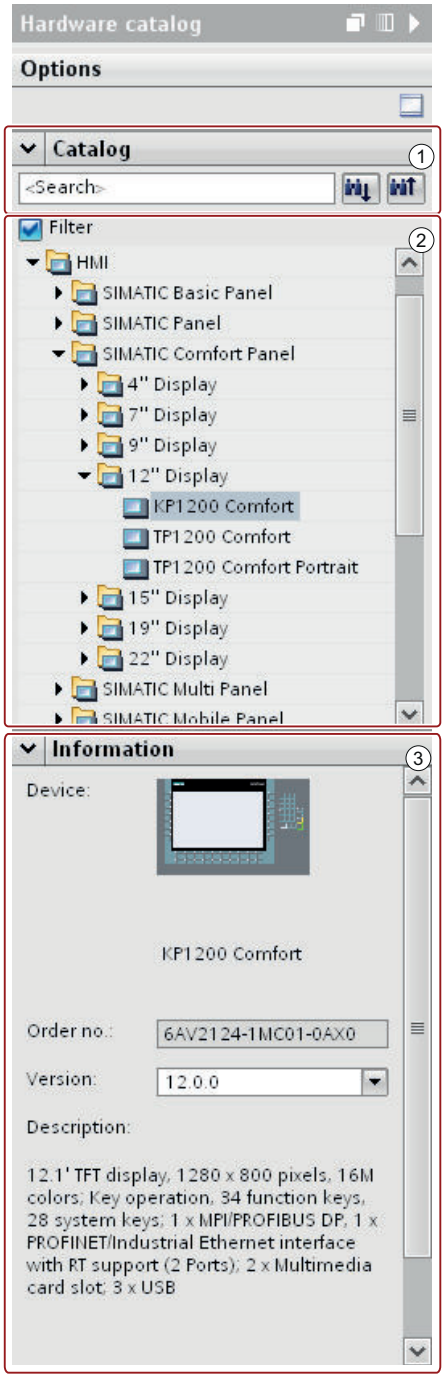
The left pane of the inspector window is used for area navigation. Information and parameters are arranged there in groups. If you click on the arrow symbol to the left of the group name, you can expand the group if sub-groups are available. If you select a group or sub-group, the corresponding information and parameters are displayed in the right pane of the inspector window and can be edited there too.

12.11.2.8 Hardware catalog

The "Hardware catalog" task card gives you easy access to a wide range of hardware components.

Structure

The "Hardware catalog" task card consists of the following panes:



1 "Catalog" pane, search and filter function

2 "Catalog" pane, component selection

3 "Information" pane

Search and filter function

The search and filter functions of the "Catalog" pane make it easy to search for particular hardware components. You can limit the display of the hardware components to certain criteria using the filter function. For example, you can limit the display to objects that you can also place within the current context or which contain certain functions.

Objects that can be used in the current context include, for example, interconnectable objects in the network view or only modules compatible with the device in the device view.

Component selection

The component selection in the "Catalog" pane contains the installed hardware components in a tree structure. You can move the devices or modules you want from the catalog to the graphic work area of the device or network view.

Installed hardware components without a license are grayed out. You cannot use non-licensed hardware components.

Hardware components belonging to various components groups thematically are partially implemented as linked objects. When you click on such linked hardware components, a catalog tree opens in which you can find the appropriate hardware components.

Information

The "Information" pane contains detailed information on the object selected from the catalog:

- Schematic representation
- Name
- Order number
- Version number
- Description

12.11.2.9 Information on hardware components

In the hardware catalog, you can display information on selected hardware components in the "Information" pane. You can also display further information on the selected hardware components using the shortcut menu.

Access to further information

If you select a hardware object in the hardware catalog and open the shortcut menu, you not only have the "Copy" function available but also three options for accessing information on Service & Support:

- Information regarding product support
- FAQs
- Manuals

The required information is displayed in the work area of the hardware and network editor.

Note

You can only access Service & Support when you are connected to the Internet and the function is enabled. By default, the function is disabled.

To enable the function, refer to the instructions in the section "Enabling product support".

Information regarding product support

Here, you have access to general information on hardware and software components. The order number of the selected hardware object is entered automatically in the search mask. You can, however, also search for other hardware and software components.

FAQs

Here, you have access to "Frequently Asked Questions" (FAQs). You can view various entries on hardware and software questions. Using a detailed search mask, you can filter the required topics.

Manuals

Here, you have access to the manuals of the various hardware components. This is particularly useful if the configuration, addressing or parameter assignment you are planning requires more detailed knowledge of the hardware you are using.

12.11.3 Networks and connections

12.11.3.1 SIMATIC communication networks

Communication networks

Overview

Communication networks are a central component of modern automation solutions. Industrial networks have to fulfill special requirements, for example:

- Coupling of automation systems as well as simple sensors, actuators, and computers.
- The information has to be correct and has to be transferred at the right moment.
- Robust against electromagnetic disturbances, mechanical stresses and soiling
- Flexible adaptation to the production requirements

Industrial networks belong to the LANs (Local Area Networks) and allow communication within a limited area.

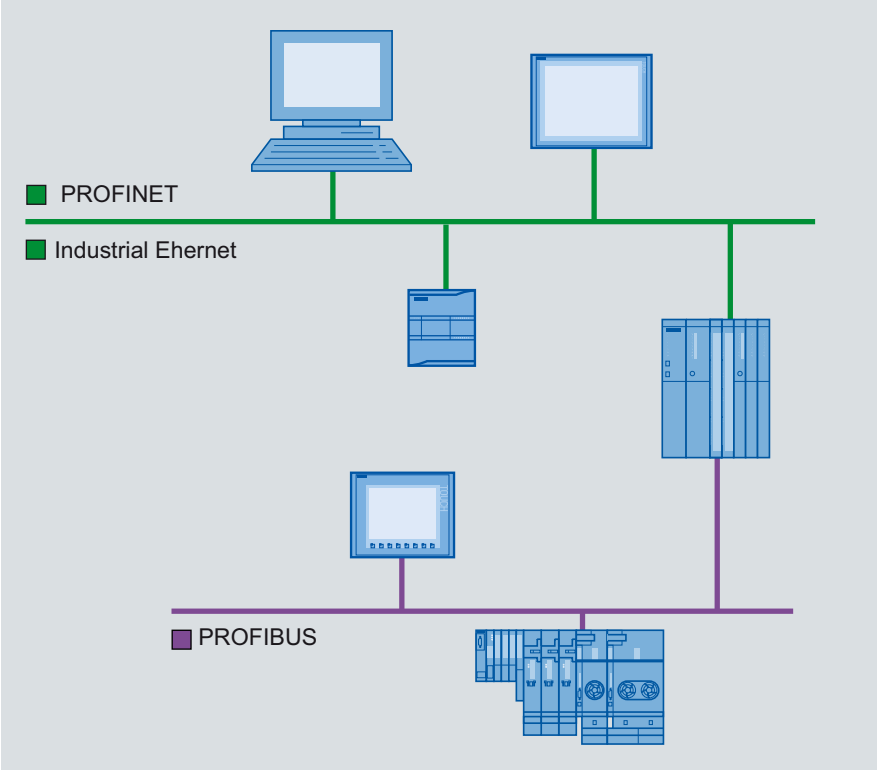
Industrial networks fulfill the following communication functions:

- Process and field communication of the automation systems including sensors and actuators
- Data communication between automation systems
- IT communication for integrating the modern information technology

Overview of the networks

This section examines the following networks:

- **Industrial Ethernet**
  The industrial network standard for all levels
- **PROFINET**
  The open Industrial Ethernet standard for automation
- **PROFIBUS**
  The international standard for the field area and market leader at the field busses
- **MPI**
  The integrated interface of the SIMATIC products
- **PPI**
  The integrated interface specially for the S7-200



PROFINET and Ethernet

Industrial Ethernet

Industrial Ethernet, which is based on IEEE 802.3, enables you to connect your automation system to your office networks. Industrial Ethernet provides IT services that you can use to access production data from the office environment.

Ethernet network

An Ethernet network allows you to interconnect all devices that are connected to the network via an integrated Ethernet interface or a communication module. This enables connection of multiple HMI devices to one SIMATIC S7 PLC and multiple SIMATIC S7 PLCs to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used. Additional information is available in the documentation for the respective HMI device.

PROFINET

PROFINET is an open standard for industrial automation defined by IEEE 61158 and based on Industrial Ethernet. PROFINET makes use of IT standards all the way to the field level and enables plant-wide engineering.

With PROFINET, you can realize high-performance automation solutions for applications with stringent real-time requirements.

PROFIBUS

PROFIBUS DP

PROFIBUS DP (distributed I/O) is used to connect the following devices:

- PLCs, PCs, HMI devices
- Distributed I/O devices, e.g., SIMATIC ET 200
- Valves
- Drives

PROFIBUS DP's fast response times make it ideally suited for the manufacturing industry.

Its basic functionality includes cyclic exchange of process data between the master and PROFIBUS DP slaves, as well as diagnostics.

PROFIBUS network

You can connect an HMI device within the PROFIBUS network to any SIMATIC S7 module that has an integrated PROFIBUS or PROFIBUS DP interface. You can thereby connect multiple HMI devices to one SIMATIC S7 PLC and multiple SIMATIC S7 PLCs to one HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used. Additional information is available in the documentation for the respective HMI device.

You configure the SIMATIC S7-200 PLC as a passive device in the network. You connect the SIMATIC S7-200 using the DP connector or a PROFIBUS communication module.

MPI

MPI

MPI (Multi-Point Interface) is the integrated interface for SIMATIC products:

- PLCs
- HMI devices
- Programming device/PC

Small subnets with the following characteristics are set up with MPI:

- Short distances
- Few devices
- Small data quantities

MPI network

You connect the HMI device to the MPI interface of the SIMATIC S7 PLC. This enables connection of multiple HMI devices to one SIMATIC S7 PLC and multiple SIMATIC S7 PLCs to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used. Additional information is available in the documentation for the respective HMI device.

Network architectures

MPI is based on the PROFIBUS standard (IEC 61158 and EN 50170) and supports the following bus topologies:

- Line
- Star
- Tree

An MPI subnet contains a maximum of 127 devices and consists of multiple segments. Each segment contains a maximum of 32 devices and is limited by terminating resistors. Repeaters are used to connect segments. The maximum cable length without a repeater is 50 m.

PPI

Introduction

PPI (point-to-point interface) is an integrated interface that was developed specially for the SIMATIC S7-200. A PPI network typically connects S7-200 PLCs. However, other SIMATIC PLCs (e.g., S7-300 and S7-400) or HMI devices can communicate with a SIMATIC S7-200 in the PPI network.

PPI network

A PPI connection is a point-to-point connection. The HMI device is the master. The SIMATIC S7-200 is the slave.

You can connect a maximum of one SIMATIC S7-200 to an HMI device. You use the serial connector of the CPU to connect the HMI device. You can connect multiple HMI devices to one SIMATIC S7-200. From the perspective of the SIMATIC S7-200, only one connection at a time is possible.

Note

The PPI network can contain a maximum of four masters in addition to the HMI device. For performance reasons, do not configure more than four devices at a time as a master in the PPI network.

Network architectures

PPI is based on the PROFIBUS standard (IEC 61158 and EN 50170) and supports the following bus topologies:

- Line
- Star

Multi-master networks with a maximum of 32 masters are set up with PPI:

- An unlimited number of masters can communicate with each slave.
- A slave can be assigned to multiple masters.

The RS 485 repeater can be used to extend the PPI network. Modems can also be connected to the PPI network.

12.11.3.2 Configuring networks and connections

Networking devices

Introduction

The "Devices & Networks" editor is provided for configuring connections. You can network devices in the editor. You can also configure and assign parameters to devices and interfaces. You then configure the required connections between the networked devices.

In the "Devices & Networks" editor you configure HMI connections with the PLCs:

- SIMATIC S7 1500
- SIMATIC S7 1200
- SIMATIC S7 300
- SIMATIC S7 400

You configure the HMI connections to other PLCs in the "Connections" editor of the respective HMI device.

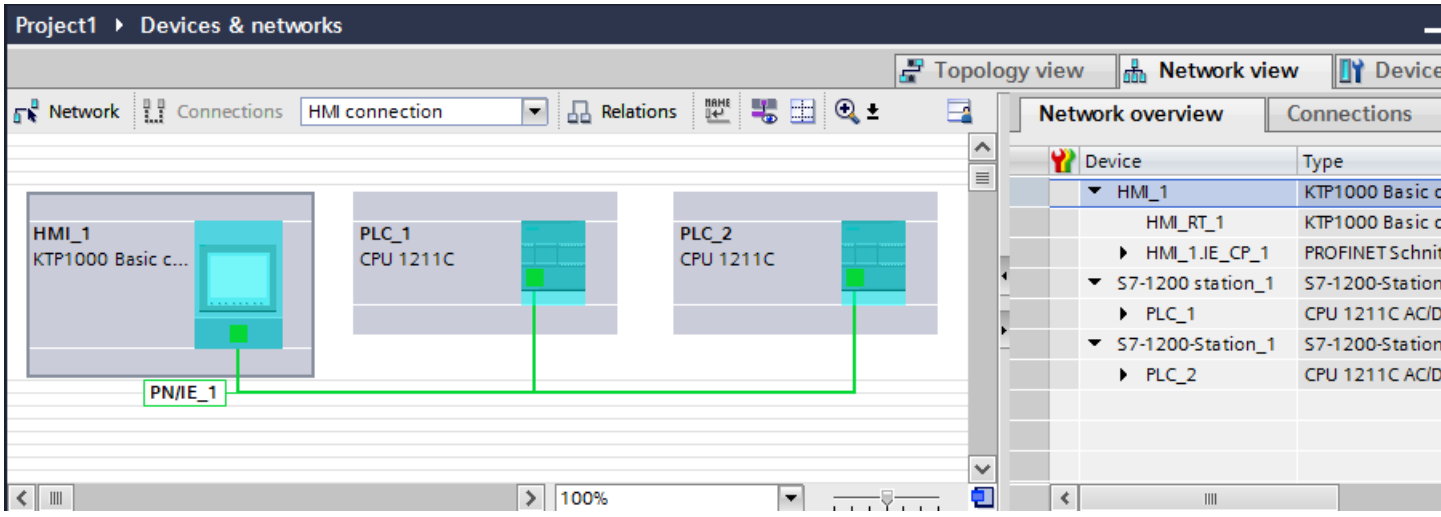
Networking devices

The network view of the "Devices & Networks" editor includes a graphical area and a tabular area. You can use the graphical area to network the devices in the project with drag-and-drop. The tabular area provides an overview of the devices and their components.

You can network the following PLCs together with HMI devices in the "Devices & Networks" area.

- SIMATIC S7 1500
- SIMATIC S7 1200
- SIMATIC S7 300
- SIMATIC S7 400

All other PLCs are available in the TIA Portal and are configured "not integrated". You configure "not integrated" connections in the "Connections" editor of the HMI device.



With the networking step, you configure the physical connection of the communication partners. The networking of devices is depicted by lines that are colored according to the interface.

Configuring an integrated connection in the "Devices & Networks" editor

Introduction

You configure an HMI connection between the HMI device and a SIMATIC S7 PLC in the "Devices & Networks" editor. This HMI connection is the direct connection between the communication partners that you have created in a project.

Integrated connections

Connections of devices within a project are referred to as integrated connections. In the case of integrated connections, you can directly configure addresses of PLC tags.

Note

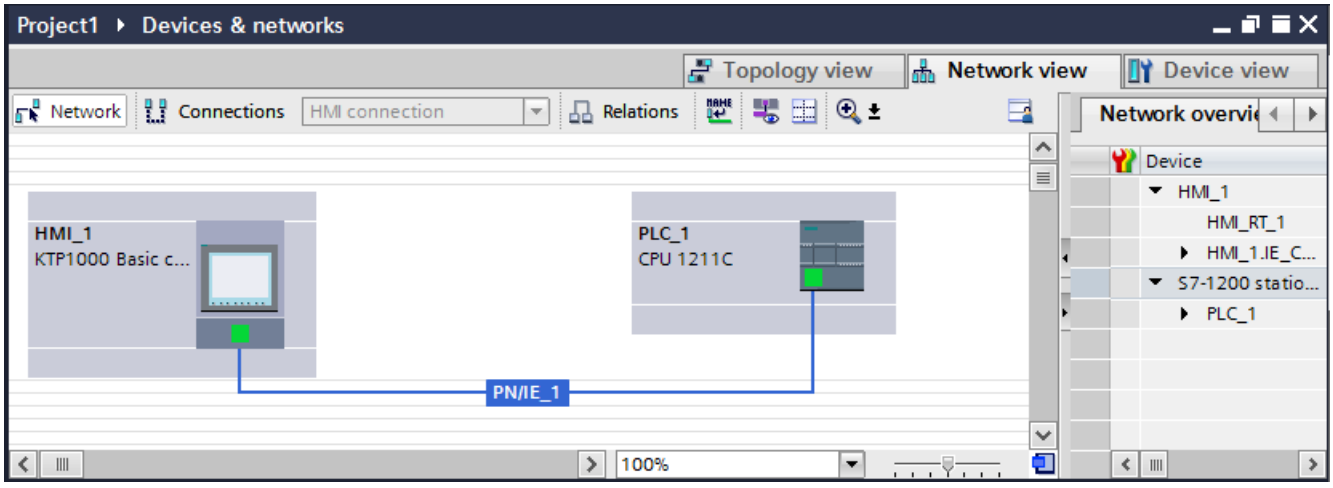
An HMI connection can be configured in the "Devices & Networks" editor for the following PLCs only:

- SIMATIC S7 1500
- SIMATIC S7 1200
- SIMATIC S7 300
- SIMATIC S7 400

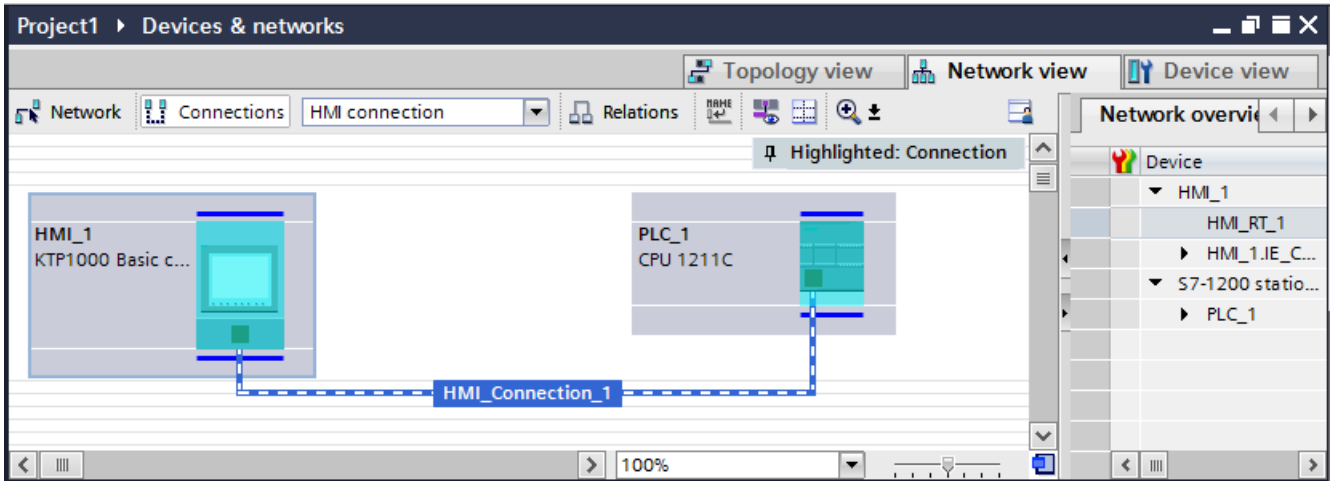
You configure the HMI connections to all other PLCs in the "Connections" editor of the HMI device.

Configuring an HMI connection in the "Devices & Networks" editor

1. Insert an HMI device and a SIMATIC S7 PLC in your project.



2. Switch to "Connections" mode.
3. Select the "HMI connection" connection type.
4. Use a drag-and-drop operation to interconnect the two PROFINET interfaces.



5. Change the IP address and subnet mask address parameters according to the requirements of your project.

Special considerations of the "Devices & Networks" editor

Introduction

If you are configuring or have already configured networks or HMI connections, the "Devices & Networks" editor supports you with the following functions:

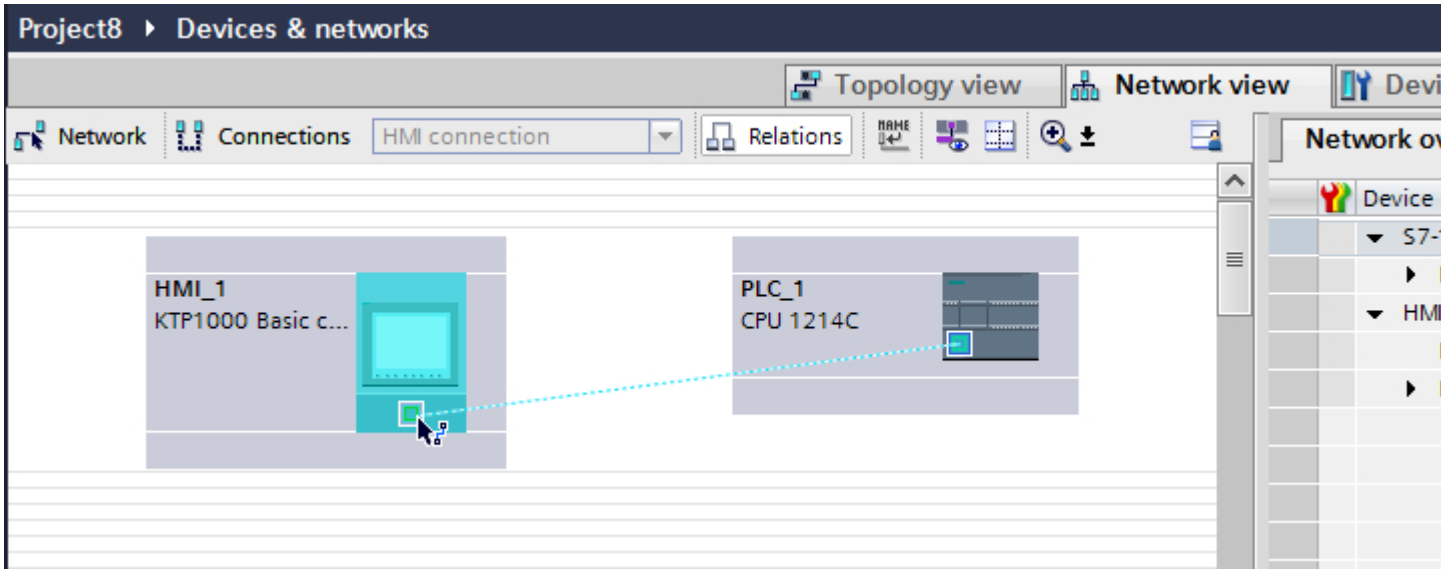
- Highlighting of communication partners
- Highlighting of HMI connections
- Automatic creation of subnets

Highlighting of communication partners

All communication partners for which an HMI connection is possible are highlighted in turquoise if you have selected the "HMI connection" type.

Starting from the interface of a device create an HMI connection to the device of another device using a drag-and-drop operation. During the drag-and-drop operation all potential communication partners are highlighted in turquoise.

Use the ESC key to stop connecting interfaces using a drag-and-drop operation.



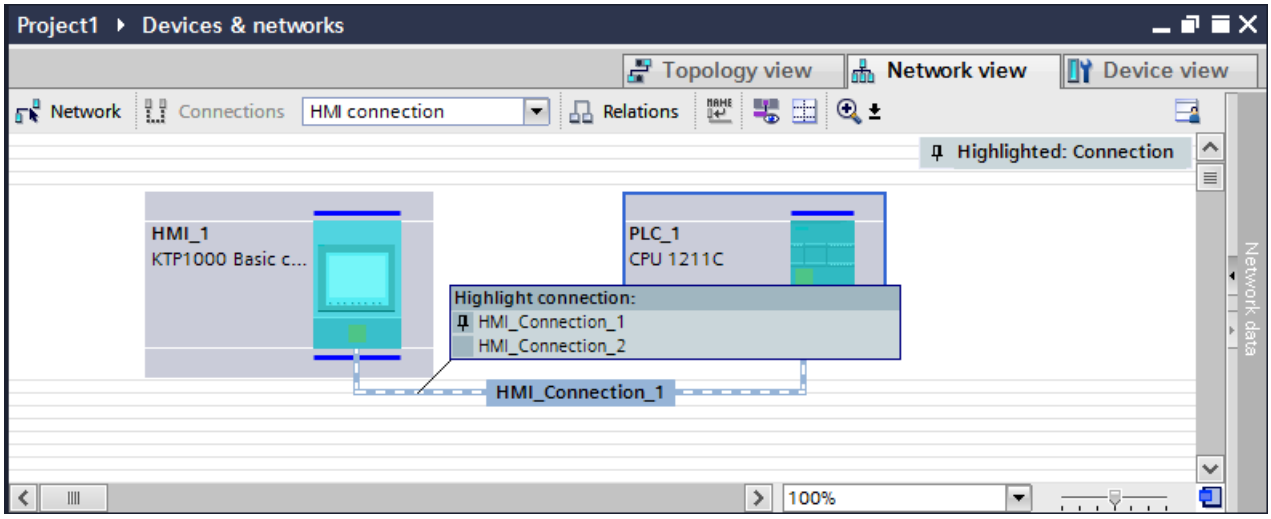
When the mouse pointer is moved over the interface of a device, the following icons indicate whether a connection is possible:

A connection is possible.

A connection is not possible.

Highlighting of HMI connections

A turquoise highlighting of the connection indicates that a HMI connection was created. If several HMI connections are created, you can select one of the already created HMI connections in a dialog.



Then you can configure the parameters of the selected HMI connection and the communication partners in the inspector window.

Subnets

Subnetworks are automatically created and used under the following conditions:

- If both communication partners are not already interconnected in different networks
- If a free interface is available to both communication partners.
- If a subnetwork already exists then that subnetwork is automatically used for the HMI connection.

Configuring a non-integrated connection in the "Connections" editor

Introduction

You use the "Connections" editor of the HMI device to configure a connection between an HMI device and a PLC that cannot be configured in the "Devices & Networks" editor.

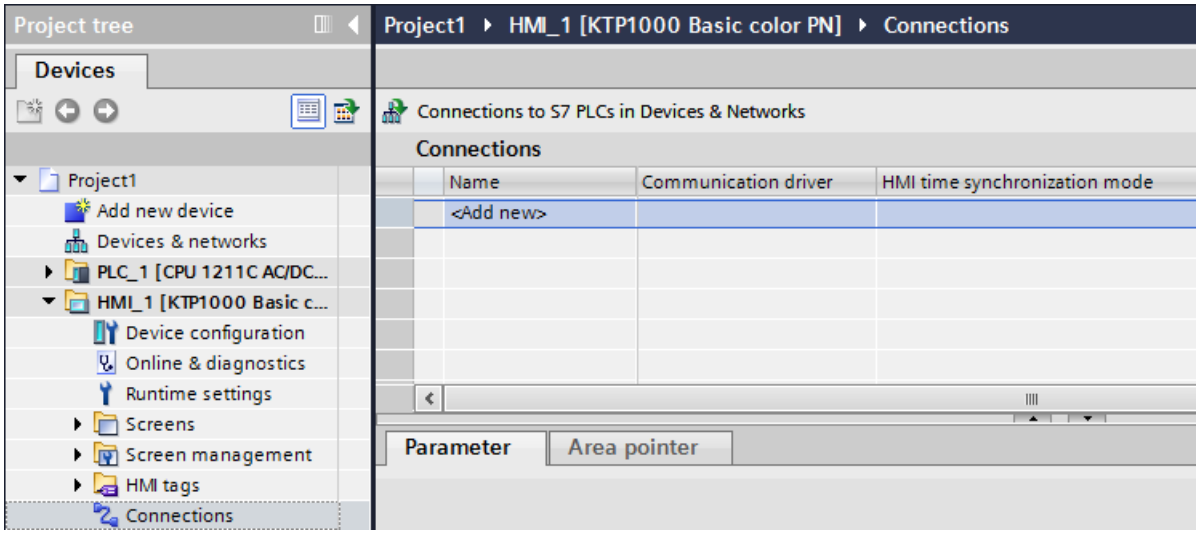
These connections are referred to as non-integrated connections.

Requirements

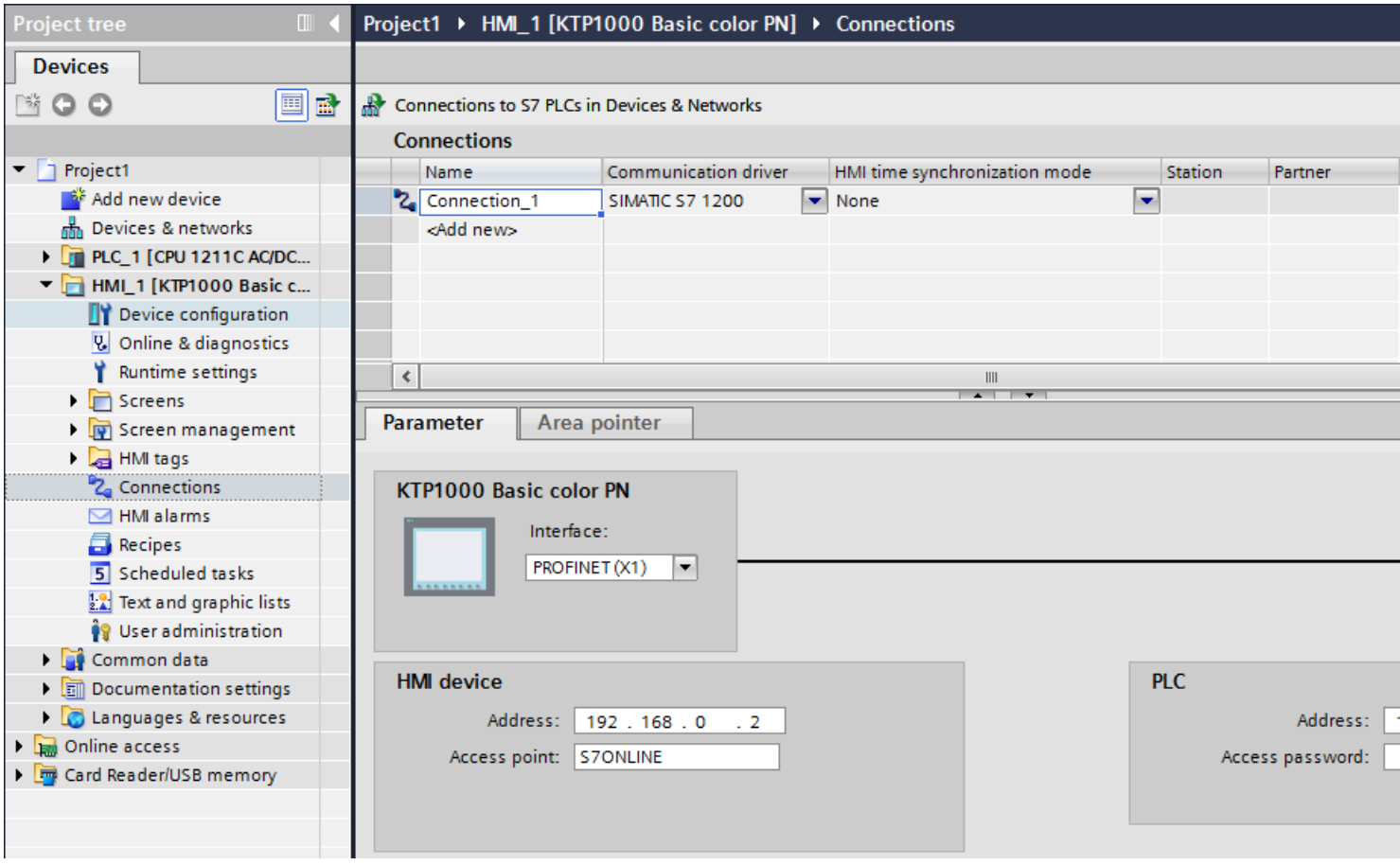
- A project is open.
- An HMI device has been created.

Configuring a connection in the "Connections" editor

1. Open the "Connections" editor of the HMI device.
2. Create a new connection.

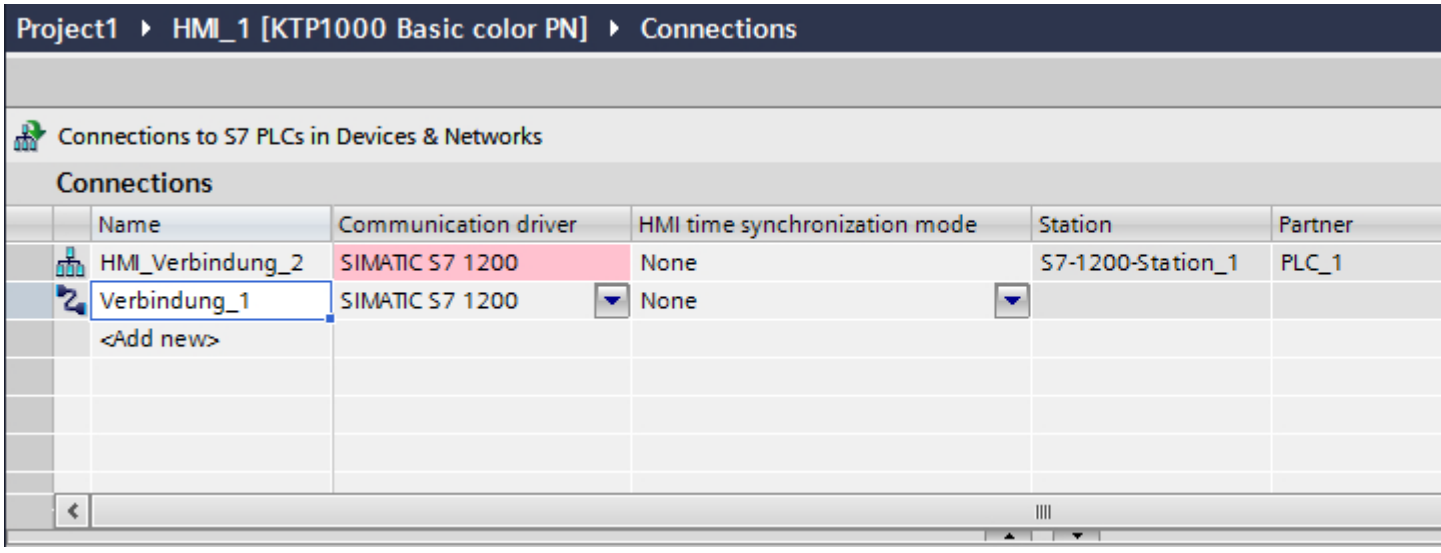


3. Select the communication driver.
4. Set the connection parameters.



Integrated connections in the "Connections" editor

If you have already configured the integrated connections of the HMI device in the "Devices & Networks" editor, they are also displayed in the "Connections" editor.



Meaning of the icons used:

- Integrated connection
- Non-integrated connection

Configuring a routed connection in the "Devices & Networks" editor

Introduction

You can configure a routed HMI connection to a PLC in another subnet in the "Devices & Networks" editor.

Note

A routed HMI connection can only be configured for the following PLCs:

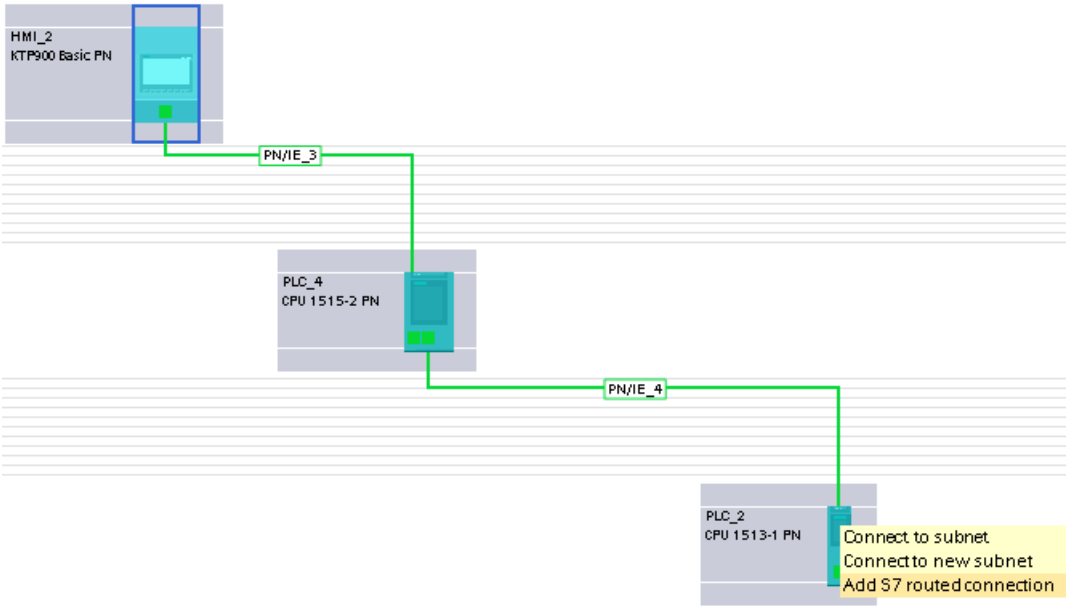
- SIMATIC S7 1500
- SIMATIC S7 1200

Requirement

- An HMI device has been created.
- PLCs have been set up in different networks
- The network view in the "Devices & Networks" editor is open.

Configuring a routed HMI connection

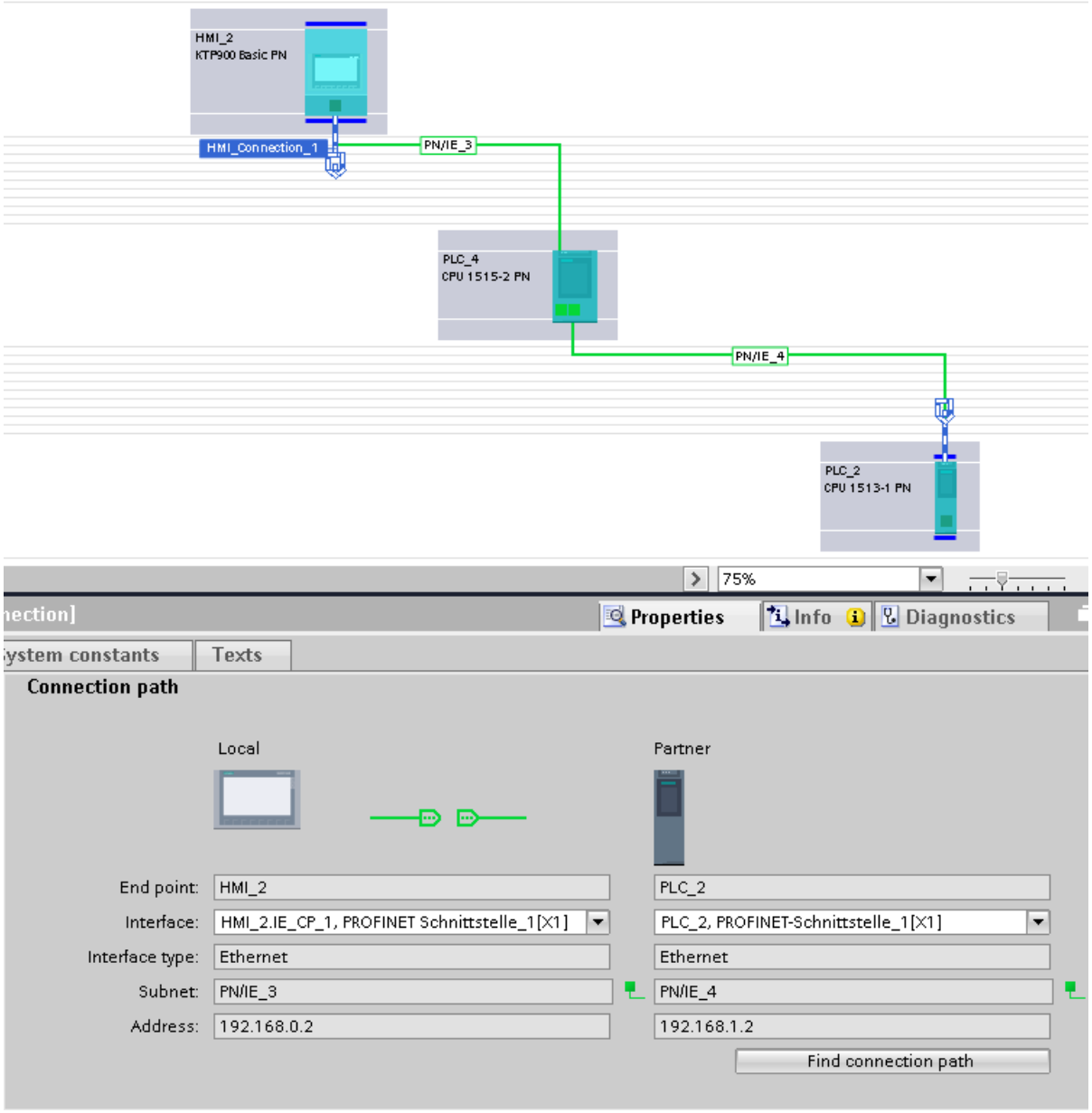
1. Switch to "Connections" mode.
2. Select the "HMI connection" connection type.
3. Drag-and-drop the HMI device to the PLC in the other subnet.
   A dialog opens.



4. Select "Add routed connection".

Result

The routed HMI connection has now been created. If you change the HMI device type or the configuration of the PLC, you will need to adapt the routed HMI connection again.



12.11.4 Data exchange

12.11.4.1 Data exchange using tags

Basics of tags

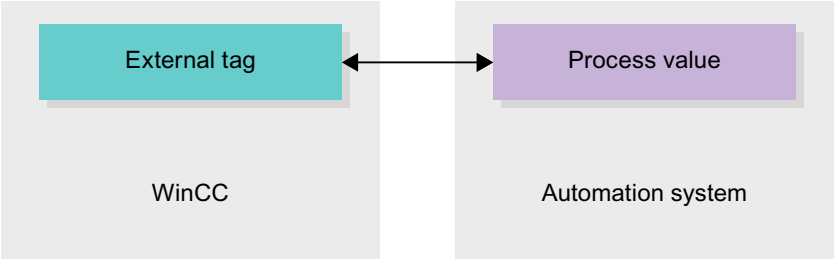
Introduction

Process values are forwarded in runtime using tags. Process values are data which is stored in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. Define external tags for processing the process values in WinCC.

WinCC works with two types of tag:

- External tags
- Internal tags

The external tags form the link between WinCC and the automation systems. The values of external tags correspond to the process values from the memory of an automation system. The value of an external tag is determined by reading the process value from the memory of the automation system. It is also possible to rewrite a process value in the memory of the automation system.



Internal tags do not have a process link and only convey values within the WinCC. The tag values are only available as long as runtime is running.

Tags in WinCC

For external tags, the properties of the tag are used to define the connection that the WinCC uses to communicate with the automation system and form of data exchange.

Tags that are not supplied with values by the process - the internal tags - are not connected to the automation system. In the tag's "Connection" property, this is identified by the "Internal tag" entry.

You can create tags in different tag tables for greater clarity. You then directly access the individual tag tables in the "HMI tags" node in the project tree. The tags from all tag tables can be displayed with the help of the table "Show all tags".

With structures you bundle a number of different tags that form one logical unit. Structures are project-associated data and are available for all HMI devices of the project. You use the "Types" editor in the project library to create and edit a structure.

Overview of HMI tag tables

Introduction

HMI tag tables contain the definitions of the HMI tags that apply across all devices. A tag table is created automatically for each HMI device created in the project.

In the project tree there is an "HMI tags" folder for each HMI device. The following tables can be contained in this folder:

- Default tag table
- User-defined tag tables
- Table of all tags

In the project tree you can create additional tag tables in the HMI tags folder and use these to sort and group tags and constants. You can move tags to a different tag table using a drag-and-drop operation or with the help of the "Tag table" field. Activate the "Tags table" field using the shortcut menu of the column headings.

Default tag table

There is one default tag table for each HMI device of the project. It cannot be deleted or moved. The default tag table contains HMI tags and, depending on the HMI device, also system tags. You can declare all HMI tags in the standard tags table or, as necessary, additional user-defined tables of tags.

User-defined tag tables

You can create multiple user-defined tag tables for each HMI device in order to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. To group tag tables, create additional subfolders in the HMI tags folder.

All tags

The "All tags" table shows an overview of all HMI tags and system tags of the HMI device in question. This table cannot be deleted, renamed or moved. This table also contains the "Tags table" column, which indicates the tag table of where a tag is included. Using the "Tags table" field, the assignment of a tag to a tags table can be changed.

With devices for Runtime Professional, the table "All tags" contain an additional tab "System tags". The system tags are created by the system and used for internal management of the project. The names of the system tags begin with the "@" character. System tags cannot be deleted or renamed. You can evaluate the value of a system tag, but cannot modify it.

Additional tables

The following tables are also available in an HMI tag table:

- Discrete alarms
- Analog alarms
- Logging tags

With the help of these tables you configure alarms and logging tags for the currently selected HMI tag.

Discrete alarms table

In the "Discrete alarms" table, you configure discrete alarms to the HMI tag selected in the HMI tag table. When you configure a discrete alarm, multiple selection in the HMI tag table is not possible. You configure the discrete alarms for each HMI tag separately.

Analog alarms table

In the "Analog alarms" table, you configure analog alarms to the HMI tag selected in the HMI tag table. When you configure an analog alarm, multiple selection in the HMI tag table is not possible. You configure the analog alarms for each HMI tag separately.

Logging tags table

In the "Logging tags" table, you configure logging tags to the HMI tag selected in the HMI tag table. When you configure a logging tag, multiple selection in the HMI tag table is not possible. You configure the logging tags for each HMI tag separately. The "Logging tags" table is only available if the HMI device used supports logging.
If WinCC Runtime Professional is used, you can also assign several log tags to a tag. With the other HMI devices, you can only assign one log tag to a tag.

External tags

Introduction

External tags allow communication (data exchange) between the components of an automation system, for example, between an HMI device and a PLC.

Principle

An external tag is the image of a defined memory location in the PLC. You have read and write access to this storage location from both the HMI device and from the PLC.

Since external tags are the image of a storage location in the PLC, the applicable data types depend on the PLC which is connected to the HMI device.

If you write a PLC control program in STEP 7, the PLC tags created in the control program will be added to the PLC tag table. If you want to connect an external tag to a PLC tag, access the PLC tags directly via the PLC tag table and connect them to the external tag.

Data types

All the data types which are available at the connected PLC are available at an external tag in WinCC. Information about data types which are available for connection to other PLCs can be found in the documentation about the respective communication drivers.

See "Communication between devices (Page 5994)" for additional information.

Note

As well as external tags, area pointers are also available for communication between the HMI device and PLC. You can set up and enable the area indicators in the "Connections" editor.

Central tag management in STEP 7

You can connect also connect DB instances of user-defined PLC data types (UDT) to the HMI tags.

The PLC data type and the corresponding DB instances are created and updated centrally in STEP 7. In WinCC, you can use the following sources as PLC tag (DB instances):

- Data block elements that use a UDT as data type
- Data block instances of a UDT

The data type is taken from STEP 7 and is not converted to an HMI data type. The access mode is always "Symbolic access". Depending on the release for WinCC in STEP 7, elements and structured elements of the PLC data type are applied to WinCC. Elements of a structured UDT are applied and displayed in the PLC tag table if the instance-specific properties "Visible in HMI" and "Accessible from HMI" have been set.

Note

Accessing PLC data types

Access to PLC data types is available only in conjunction with SIMATIC S7-1200 or S7-1500.

Synchronization with PLC tags

A variety of options for synchronizing external tags with the PLC tags are available in the Runtime settings under "Settings for tags".

When you perform synchronization, you have the option of automatically applying the tag names of the PLC to external tags and reconnecting the existing tags.

The generated tag name is derived from the position of the data value in the hierarchical structure of the data block.

Update of tag values

For external tags, the current tag values are transmitted in Runtime via the communication connection between WinCC and the connected automation systems and then saved in the Runtime memory. Next, the tag value will be updated to the set cycle time. For use in the Runtime project, WinCC accesses tag values in the Runtime memory that were read from the PLC at the previous scan cycle checkpoint. As a result, the value in the PLC can already change whilst the value from the Runtime memory is being processed.

Note

PLC array elements in conjunction with S7-1200 or S7-1500

The index of the PLC array elements can begin with any number. In WinCC, indexing always starts with 0.

A PLC tag "Array [1..3] of Int", for example, is mapped to "Array [0..2] of Int" in WinCC.

When you access an array in a script, pay attention to the correct indexing.

See also

Communication between devices (Page 5994)

Addressing external tags

Introduction

The options for addressing external tags depend on the type of connection between WinCC and the PLC in question. A distinction must be made between the following connection types:

- Integrated connection
  Connections of devices which are within a project and were created with the "Devices & Networks" editor are referred to as integrated connections.
- Non-integrated connection
  Connections of devices which were created with the "Connections" editor are referred to as non-integrated connections. It is not necessary that all of the devices be within a single project.

The connection type can also be recognized by its icon.

	Integrated connection
	Non-integrated connection

You can find additional information in the section "Basics of communication (Page 5994)".

Addressing with integrated connections

An integrated connection offers the advantage that you can address a tag both symbolically and absolutely.

For symbolic addressing, you select the PLC tag via its name and connect it to the HMI tag. The valid data type for the HMI tag is automatically selected by the system. You have to distinguish between the following cases when you address elements in data blocks:

Symbolic addressing of data blocks with optimized access and standard access:

During the symbolic addressing of a data block with optimized access and standard access, the address of an element in the data block is dynamically assigned and is automatically adopted in the HMI tag in the event of a change. You do not need to compile the connected data block or the WinCC project for this step.
For data blocks with optimized access, only symbolic addressing is available.

For symbolic addressing of elements in a data block, you only need to recompile and reload the WinCC project in case of the following changes:

- If the name or the data type of the linked data block element or global PLC tag has changed.
- If the name or the data type of the higher level structure node of a linked element in the data block element or global PLC tag has changed.
- If the name of the connected data block has changed.

Symbolic addressing is currently available with the following PLCs:

- SIMATIC S7 1200
- SIMATIC S7 1500

Symbolic addressing is also available if you have an integrated link.

You can also use absolute addressing with an integrated connection. You have to use absolute addressing for PLC tags from a SIMATIC S7 300/400 PLC. If you have connected an HMI tag with a PLC tag and the address of the PLC tag changes, you only have to recompile the control program to update the new address in WinCC. Then you recompile the WinCC project and load it onto the HMI device.

In WinCC, symbolic addressing is the default method. To change the default setting, select the menu command "Options > Settings". Select "Visualization > Tags" in the "Settings" dialog. If required, disable the "Symbolic access" option.

The availability of an integrated connection depends on the PLC used. The following table shows the availability:

PLC	Integrated connection	Comments
S7 300/400	Yes	The linking of tags is not checked in Runtime. If the tag address changes in the PLC and the HMI device is not compiled again and loaded, the change is not registered in runtime.
S7 1200	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the following behavior applies to the S7 300/400.
S7-1500	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the following behavior applies to the S7 300/400.

Create an integrated connection in the "Devices & Networks" editor. If the PLC is contained in the project and integrated connections are supported, you can then also have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.

Addressing with non-integrated connections

In the case of a project with a non-integrated connection, you always configure a tag connection with absolute addressing. Select the valid data type yourself. If the address of a PLC tag changes in a project with a non-integrated connection during the course of the project, you also have to make the change in WinCC. The tag connection cannot be checked for validity in Runtime, an error message is not issued.

A non-integrated connection is available for all supported PLCs.

Symbolic addressing is not available in a non-integrated connection.

With a non-integrated connection, the control program does not need to be part of the WinCC project. You can perform the configuration of the PLC and the WinCC project independently of each other. For configuration in WinCC, only the addresses used in the PLC and their function have to be known.

See also

Basics of communication (Page 5994)

Internal Tags

Introduction

Internal tags do not have any connection to the PLC. Internal tags transport values within the HMI device. The tag values are only available as long as runtime is running.

Principle

Internal tags are stored in the memory of the HMI device. Therefore, only this HMI device has read and write access to the internal tags. You can create internal tags to perform local calculations, for example.

You can use the HMI data types for internal tags. Availability depends on the HMI device being used.

The following HMI data types are available:

HMI data type	Data format
Array	One-dimensional array
Bool	Binary tag
DateTime	Date/time format
DInt	Signed 32-bit value
Int	Signed 16-bit value
LReal	Floating-point number 64-bit IEEE 754
Real	Floating-point number 32-bit IEEE 754
SInt	Signed 8-bit value
UDnt	Unsigned 32-bit value
UInt	Unsigned 16-bit value
USInt	Unsigned 8-bit value
WString	Text tag, 16-bit character set

12.11.4.2 Data exchange using area pointers

Basic information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

For example, area pointers are required for the following data:

- Recipes
- Job mailboxes
- Sign-of-life monitoring

Area pointer

The following area pointers are supported:

Area pointer

Area pointers can be configured for connections.

- Data record
- Date/time
- Coordination
- Job mailbox

Global area pointers of the HMI device

Global area pointers can be configured to only one connection for each project.

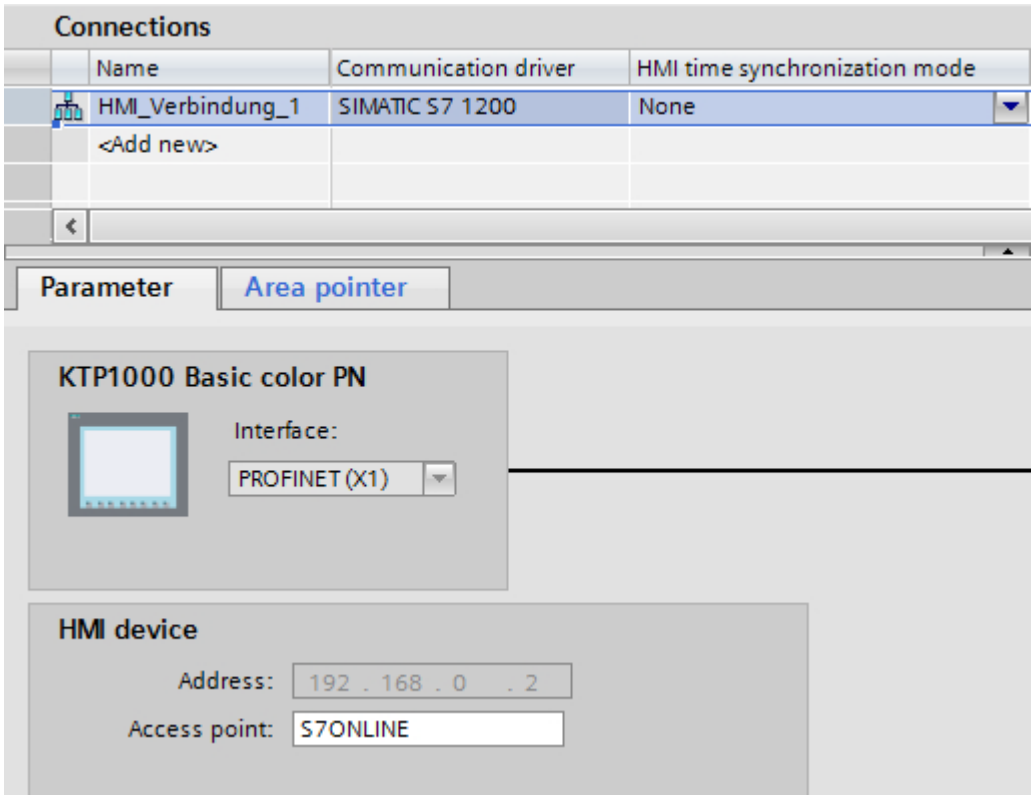
- Screen number
- Date/time PLC
- Project ID

Area pointers for connections

Introduction

Using the "Area pointer" tab of the "Connections" editor, you can configure the usage of the available area pointers.

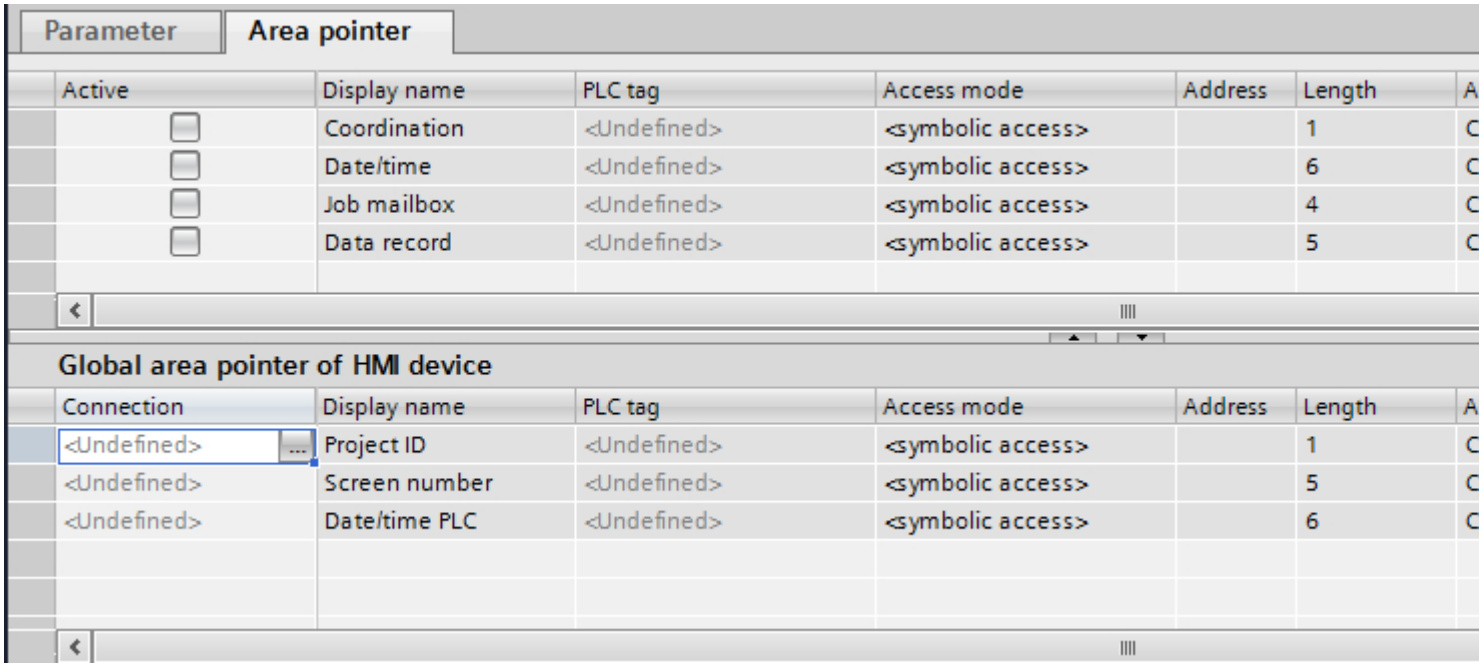
To configure the area pointers, open the "Connections" editor and open the "Area pointer" tab.



Structure

The "Area pointer" tab contains two tables of area pointers. The top part of the table contains the area pointers you can create and enable separately for each available connection.

The "Global area pointers of HMI device" table contains the area pointers which are created only once in the project and can be used for only one connection.

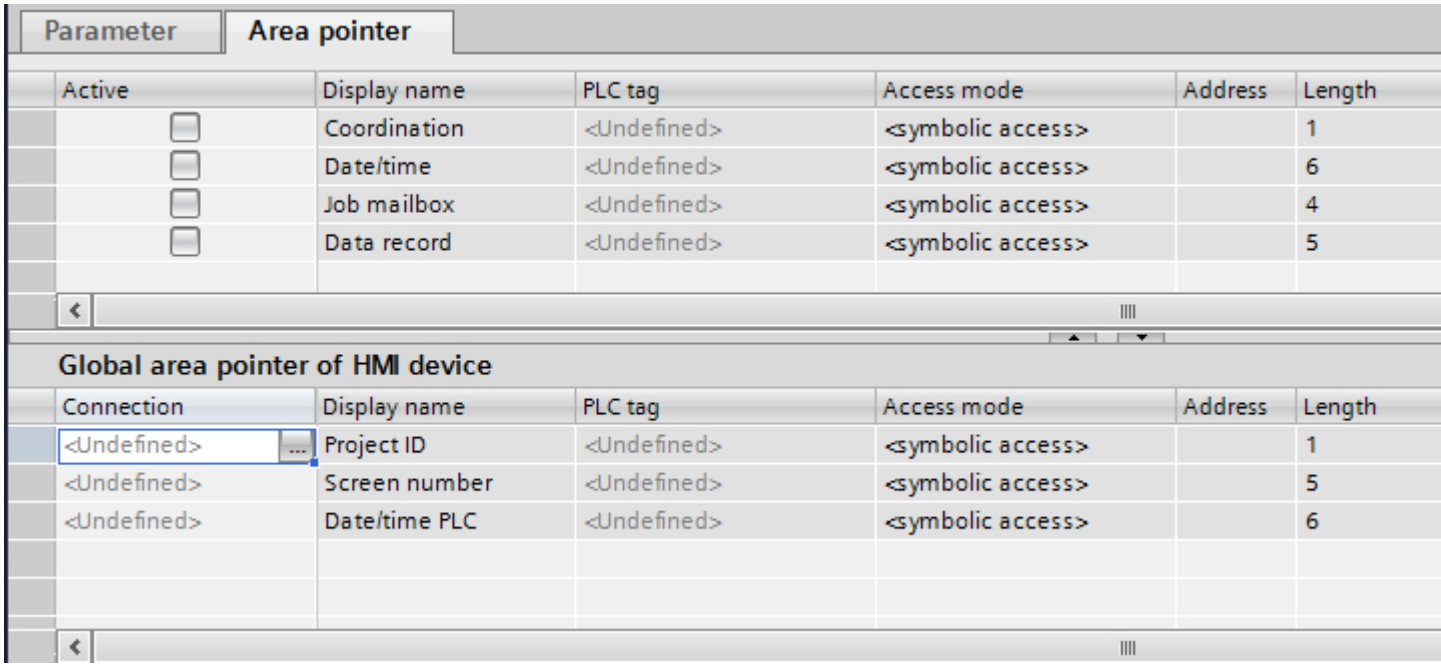


Use of area pointers

"Area pointer" tab

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You assign the following parameters in the "Area pointer" tab:



- Active
  Enables the area pointer.
- Pointer name
  Name of the area pointer specified by WinCC.
- PLC tag
  Here you select the PLC tag or the tag array that you have configured as the data area for the area pointer.
- Access mode
  Here you can select from the following access modes:
  - Symbolic access
  - Absolute access
- Address
  If you selected "Symbolic access", no address is entered in this field.
  If you selected "Absolute access", enter the address of a tag in the "Address" field.
- Length
  WinCC specifies the length of the area pointer.
- Acquisition cycle
  You specify the acquisition cycle in this field for area pointers that are read by the HMI device. Note that a very short acquisition time may have a negative impact on HMI device performance.
- Comment
  Enter a comment, for example, to describe the purpose of the area pointer.

Accessing data areas

Accessing data areas

Accessing data areas

The following table shows how HMI devices and PLCs access individual data areas for read (R) or write (W) operations.

Data area	Required for	HMI device	PLC
Screen number	Evaluation by the PLC in order to determine the active screen.	W	R
Data record	Transfer of data records with synchronization	R/W	R/W
Date/time	Transfer of the date and time from the HMI device to the PLC	W	R
Date/time PLC	Transfer of the date and time from the PLC to the HMI device	R	W
Coordination	Requesting the HMI device status in the PLC program	W	R
Project ID	Runtime checks for consistency between the WinCC project ID and the project in the PLC	R	W
Job mailbox	Triggering of HMI device functions by the PLC program	R/W	R/W

Configuring area pointers

Configuration of area pointers

Introduction

You use an area pointer to access a data area in the PLC. The data area is stored in the PLC.

Prior to configuring area pointers

Before you use the area pointer, you must enable and parameterize it under "Connections > Area Pointer".

Global data block

To access the data area in the PLC, you have to create a global data block in the PLC program. The following example shows the use of a data block.

Length of area pointers

For area pointers with a length >= 1, you set up the data area as a tag array in a global data block or instance data block.

You also have the option to use a PLC tag for area pointers with a length = 1.

The configuration of the tags in a data block is dependent on the length of the area pointer you want to use. The unit for the length of an area pointer is a 16-bit word.

If, for example, you want to use an area pointer with a length of "5", you must create an array with 5 array elements of the data type UINT in the data block.

Alternative procedure

Alternatively, you can also use the absolute access mode to access area pointers. Absolute access mode only works on standard PLC data blocks.

Parameterizing a global data block

Introduction

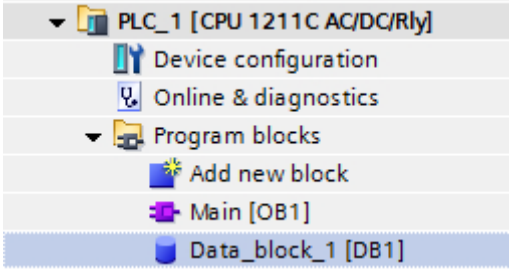
To access the data area in the PLC, a global data block for the area pointer must be parameterized in the PLC program.

Requirements

- A PLC is created in the project.
- A connection is configured between the PLC and the HMI device.
- The PLC program contains a global data block.

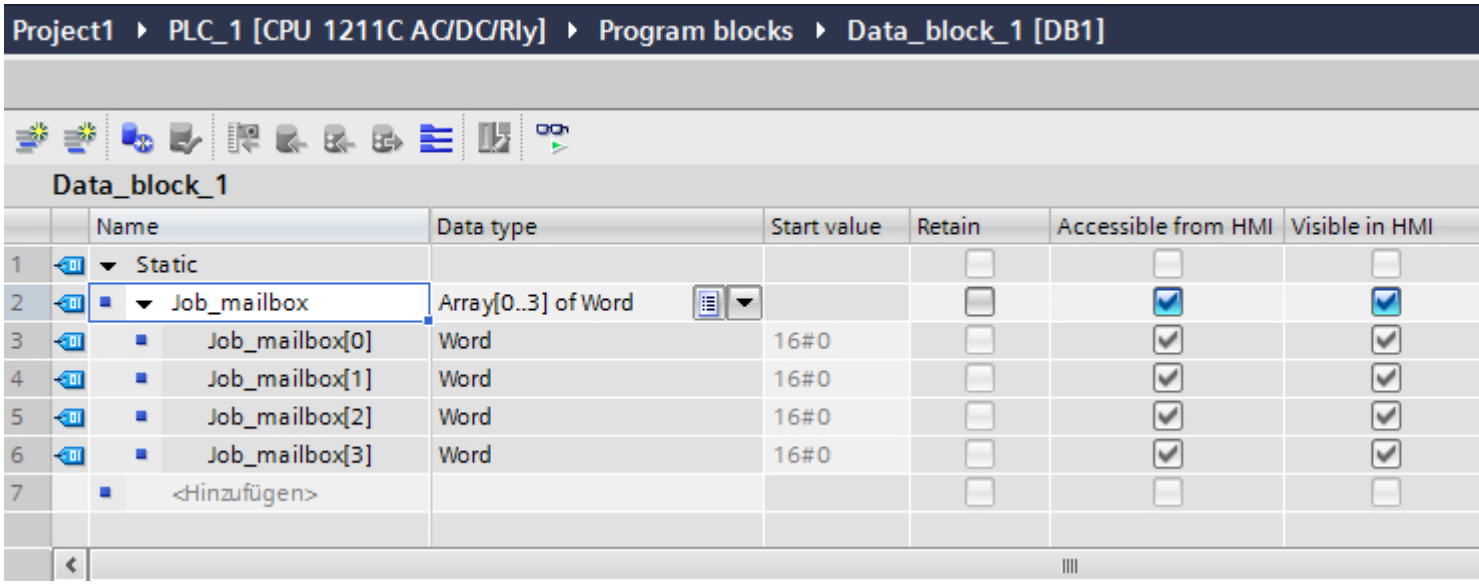
Procedure

1. Open "PLC > Program blocks" in the project tree.
2. Double-click the global data block you created previously.
   The data block opens.



3. Enter a tag name in the "Name" column.
4. Select "Array[lo .. hi] of type" as the data type in the "Data type" column.
5. Replace the "lo" entry by the low value for the dimension of the array.
6. Replace the "hi" entry by the high value for the dimension of the array.
   Example: If you configure an area pointer with the length "4", enter the value "0" for "lo" and the value "3" for "hi" inside the brackets.

7. Replace the "type" designation with the "word" data type.
   The full data type for an array of 4 tags appears as follows: "Array[0 .. 3] of word".
   The tag array is created after the entry is confirmed.
8. Click "Compile".
   The project is compiled.



Configuring an area pointer for a connection

Introduction

After you have parameterized the global data block, you now create the area pointer for the connection.

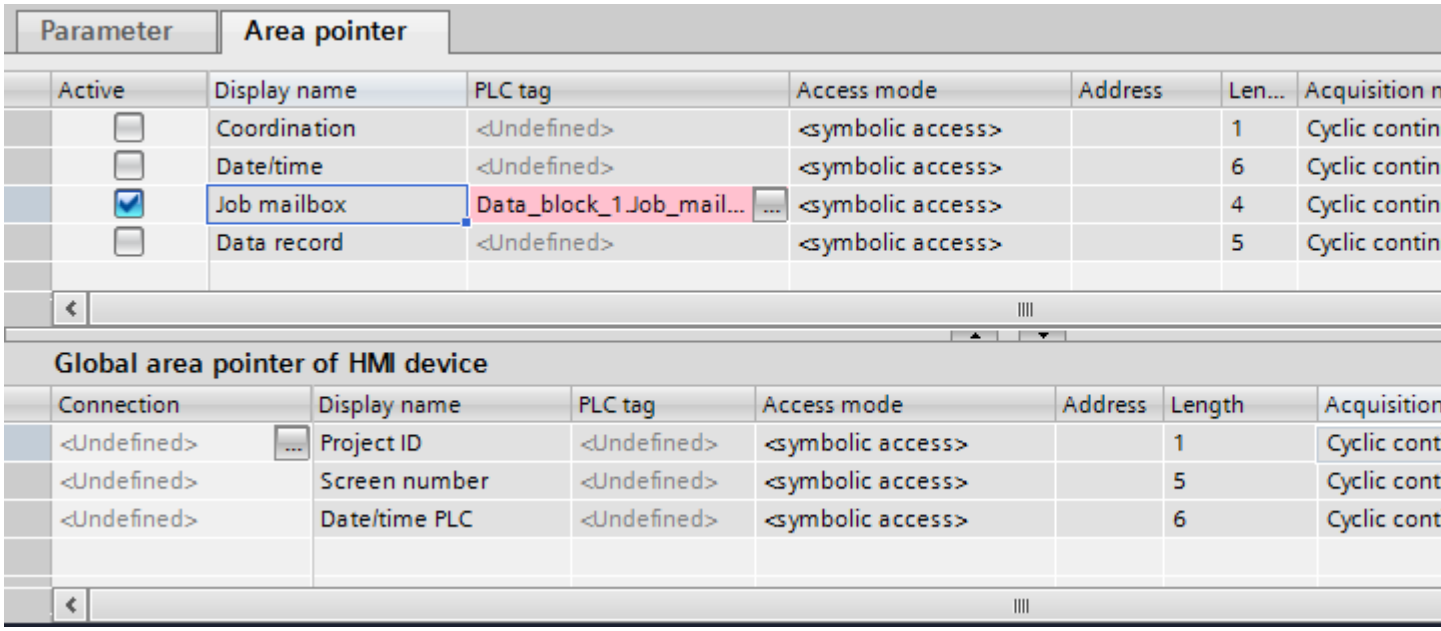
Requirements

- The global data block has been parameterized in the PLC program.

Procedure

1. Open "HMI >Connections" in the project tree.
2. Click the "Area pointer" tab.
3. Enable the required area pointer.
   You enable a global area pointer by selecting the connection in the "Connection" field.
4. Click the navigation button in the "PLC tag" field.
   The object list opens.

5. Navigate to the data block in the object list, and select the tag in the right window.
   You do not need an array tag to configure an area pointer with the length of "1".



6. Select the "Word" data type when creating the tag in the data block.

   If required, set additional parameters, such as the acquisition cycle, during configuration.

Result

The area pointer is enabled and connected to the PLC tag in the global data block.

12.11.5 Device dependency

12.11.5.1 Basic Panel

Communication drivers

SIMATIC communication drivers

Device dependency of the Basic Panels

Various communication drivers are configured for Basic Panels depending on the communication.

The following tables show the communication drivers released for integrated and non-integrated communication.

There are different versions of the HMI devices within the integrated communication.

Communication drivers for integrated communication (V11.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
KP300 Basic	Yes	Yes	Yes	Yes	Yes	No	Yes
KP400 Basic	Yes	Yes	Yes	Yes	Yes	No	Yes
KTP400 Basic PN	Yes	Yes	Yes	Yes	Yes	No	Yes
KTP600 Basic DP	Yes	Yes	Yes	Yes	Yes	No	Yes
KTP600 Basic PN	Yes	Yes	Yes	Yes	Yes	No	Yes
KTP1000 Basic DP	Yes	Yes	Yes	Yes	Yes	No	Yes
KTP1000 Basic PN	Yes	Yes	Yes	Yes	Yes	No	Yes
TP1500 Basic PN	Yes	Yes	Yes	Yes	Yes	No	Yes

Communication drivers for integrated communication (V12.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
KP300 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes
KP400 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP400 Basic PN	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP600 Basic DP	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP600 Basic PN	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP1000 Basic DP	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP1000 Basic PN	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1500 Basic PN	No	Yes	Yes	Yes	Yes	Yes	Yes

Communication drivers for integrated communication (V13.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
KTP400 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP700 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP900 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP1200 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes

Communication drivers for non-integrated communication

HMI devices	SIMATIC S7-1200	SIMATIC S7-1500	SIMATIC S7-300/400	SIMATIC S7-200	SIMATIC LOGO!	SIMATIC HTTP protocol
KP300 Basic	Yes	Yes	Yes	Yes	Yes	No
KP400 Basic	Yes	Yes	Yes	Yes	Yes	No
KTP400 Basic PN	Yes	Yes	Yes	Yes	Yes	No
KTP600 Basic DP	Yes	Yes	Yes	Yes	Yes	No
KTP600 Basic PN	Yes	Yes	Yes	Yes	Yes	No
KTP700 Basic PN	Yes	Yes	Yes	Yes	Yes	No
KTP900 Basic PN	Yes	Yes	Yes	Yes	Yes	No
KTP1000 Basic DP	Yes	Yes	Yes	Yes	Yes	No
KTP1000 Basic PN	Yes	Yes	Yes	Yes	Yes	No
TP1500 Basic PN	Yes	Yes	Yes	Yes	Yes	No

Other communication drivers

Device dependency of the Basic Panels

The following table shows which communication drivers you can configure with the various Basic Panels.

Communication drivers

HMI devices	OPC	SIMATIC HTTP protocol	Allen-Bradley EtherNet/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
KP300 Basic	No	No	Yes	No	Yes	No	Yes	No	No
KP400 Basic	No	No	Yes	No	Yes	No	Yes	No	No
KTP400 Basic PN	No	No	Yes	No	Yes	No	Yes	No	No
KTP600 Basic DP	No	No	No	Yes [2)]	No	Yes	No	Yes [1)]	Yes
KTP600 Basic PN	No	No	Yes	No	Yes	No	Yes	No	No
KTP700 Basic PN	No	No	Yes	No	Yes	No	Yes	No	No
KTP700 Basic DP	No	No	No	Yes [2)]	No	Yes	No	Yes [1)]	Yes
KTP900 Basic PN	No	No	Yes	No	Yes	No	Yes	No	No
KTP1000 Basic DP	No	No	No	Yes [2)]	No	Yes	No	Yes [1)]	Yes
KTP1000 Basic PN	No	No	Yes	No	Yes	No	Yes	No	No
KTP1200 Basic PN	No	No	Yes	No	Yes	No	Yes	No	No
KTP1200 Basic DP	No	No	No	Yes [2)]	No	Yes	No	Yes [1)]	Yes
TP1500 Basic PN	No	No	Yes	No	Yes	No	Yes	No	No

[1)] only with RS 422-RS232 converter

Order number of the converter: 6AV6 671-8XE00-0AX0

[2)] Direct communication with PLC 5 or KF2 module, otherwise only approved with RS422-RS232 converter (option).

Order number of the converter: 6AV6 671-8XE00-0AX0

Interfaces of the Basic Panels

Device dependency of the Basic Panels

The following table shows which HMI device interfaces are available for the communication driver protocols.

Table 12-149 Basic Panels

	KP300 Basic **KP400 Basic** **KTP400 Basic PN** **KTP600 Basic PN** **KTP700 Basic PN** **KTP900 Basic PN** **KTP1200 Basic PN** **KTP1000 Basic PN** **TP1500 Basic PN**	**KTP600 Basic DP** **KTP700 Basic DP** **KTP1000 Basic DP** **KTP1200 Basic DP**
SIMATIC S7 - PPI [1]	—	MPI/DP (X2)
SIMATIC S7 - MPI	—	MPI/DP (X2)
SIMATIC S7 - PROFIBUS	—	MPI/DP (X2)
SIMATIC S7 - PROFINET	PROFINET (X1)	—
SIMATIC HMI HTTP protocol	—	—
OPC	—	—
Allen-Bradley EtherNet/IP	PROFINET (X1)	—
Allen-Bradley DF1	—	MPI/DP (X2) [2]
Mitsubishi TCP/IP	PROFINET (X1)	—
Mitsubishi FX	—	MPI/DP (X2) (RS422)
Modicon Modbus TCP	PROFINET (X1)	—
Modicon Modbus RTU	—	MPI/DP (X2) [3]
Omron Host Link	—	MPI/DP (X2) (RS422)

[1] For SIMATIC S7-200 only

[2] Direct communication with PLC5 or KF2 module, otherwise only approved with RS422-RS232 converter (option).
Order number: 6AV6 671-8XE00-0AX0

[3] only approved with RS 422-RS232 converter
Order number: 6AV6 671-8XE00-0AX0

Area pointers for Basic Panels

Introduction

Area pointers are parameter fields from which the HMI device obtains information about the location and size of data areas in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

WinCC uses the following area pointers:

- Job mailbox
- Project ID
- Screen number
- Data record
- Date/time
- Date/time PLC
- Coordination

Availability of the area pointers

The following table shows the availability of the area pointers on the HMI devices. Note that the area pointers can be used only for available communication drivers.

Area pointer

	KP300 Basic	KTP400 Basic PN	KTP600 Basic PN	KTP600 Basic DP	KTP700 Basic PN / DP	KTP900 Basic PN	KTP1000 Basic PN / DP	KTP1200 Basic PN / DP	TP1500 Basic PN
Screen number	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Data record	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time PLC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Coordination	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Project ID	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Job mailbox	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

12.11.5.2 Panel

Communication drivers

SIMATIC communication drivers

Device dependency of the panels

Various communication drivers are configured for panels depending on the communication.

The following tables show the communication drivers released for integrated and non-integrated communication.

There are different versions of the HMI devices within the integrated communication.

Communication drivers for integrated communication (V11.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
OP 73	No	Yes	Yes	Yes	Yes	No	Yes
OP 77A	No	Yes	Yes	Yes	Yes	No	Yes
OP 77B	No	Yes	Yes	Yes	Yes	No	Yes
TP 177A	No	Yes	Yes	Yes	Yes	No	Yes
TP 177A Portrait	No	Yes	Yes	Yes	Yes	No	Yes
TP 177B 4"	No	Yes	Yes	Yes	No	No	Yes
TP 177B 6" mono	No	Yes	Yes	Yes	Yes	No	Yes
TP 177B 6"	No	Yes	Yes	Yes	Yes	No	Yes
OP 177B 6" mono	No	Yes	Yes	Yes	Yes	No	Yes
OP 177B 6"	No	Yes	Yes	Yes	Yes	No	Yes
TP 277 6"	No	Yes	Yes	Yes	Yes	No	Yes
OP 277 6"	No	Yes	Yes	Yes	Yes	No	Yes

Communication drivers for non-integrated communication

HMI devices	SIMATIC S7-1500	SIMATIC S7-1200	SIMATIC S7-300/400	SIMATIC S7-200	SIMATIC LOGO!	SIMATIC HTTP protocol
OP 73	No	No	Yes	Yes	No	No
OP 77A	No	Yes	Yes	Yes	No	No
OP 77B	No	Yes	Yes	Yes	No	No
TP 177A	No	Yes	Yes	Yes	No	No

HMI devices	SIMATIC S7-1500	SIMATIC S7-1200	SIMATIC S7-300/400	SIMATIC S7-200	SIMATIC LOGO!	SIMATIC HTTP protocol
TP 177A Portrait	No	Yes	Yes	Yes	No	No
TP 177B 4"	No	Yes	Yes	Yes	Yes	Yes
TP 177B 6" mono	No	Yes	Yes	Yes	Yes	No
TP 177B 6"	No	Yes	Yes	Yes	Yes	Yes
OP 177B 6" mono	No	Yes	Yes	Yes	Yes	No
OP 177B 6"	No	Yes	Yes	Yes	Yes	Yes
TP 277 6"	No	Yes	Yes	Yes	Yes	Yes
OP 277 6"	No	Yes	Yes	Yes	Yes	Yes

Other communication drivers

Device dependency of the panels

The following table shows the communication drivers you can configure with the various panels.

Communication drivers

HMI devices	OPC	Allen-Bradley EtherNet/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
OP 73	No	No	No	No	No	No	No	No
OP 77A	No	No	Yes[2) 3)]	No	Yes	No	Yes[1)]	Yes[3)]
OP 77B	No	No	Yes[2) 3)]	No	Yes	No	Yes	Yes[3)]
TP 177A	No	No	Yes[2) 3)]	No	Yes	No	Yes[1)]	Yes[3)]
TP 177A Portrait	No	No	Yes[2) 3)]	No	Yes	No	Yes[1)]	Yes[3)]
TP 177B 4"	No	Yes	Yes[2) 3)]	Yes	Yes	Yes	Yes[1) 3)]	Yes[3)]
TP 177B 6" mono	No	No	Yes[2) 3)]	No	Yes	No	Yes[1) 3)]	Yes[3)]
TP 177B 6"	No	Yes	Yes[2) 3)]	Yes	Yes	Yes	Yes[1) 3)]	Yes[3)]
OP 177B 6" mono	No	No	Yes[2) 3)]	No	Yes	No	Yes[1) 3)]	Yes[3)]
OP 177B 6"	No	Yes	Yes[2) 3)]	Yes	Yes	Yes	Yes[1) 3)]	Yes[3)]
TP 277 6"	No	Yes	Yes[2) 3)]	Yes	Yes	Yes	Yes[1) 3)]	Yes[3)]
OP 277 6"	No	Yes	Yes[2) 3)]	Yes	Yes	Yes	Yes[1) 3)]	Yes[3)]

1) only approved with RS 422-RS232 converter
Order number: 6AV6 671-8XE00-0AX0

2) Direct communication with PLC 5 or KF2 module, otherwise only approved with RS422-RS232 converter (option).
Order number: 6AV6 671-8XE00-0AX0

3) "PROFINET IO Enabled" must be disabled.

Interfaces of Panels

Device dependency of Panels

The following table shows which HMI device interfaces are available for the communication driver protocols.

Table 12-150 Panels

	OP 73	OP 77A	OP 77B [3]	TP 177A	TP 177B [3] OP 177B [3]	TP 277 [3] OP 277 [3]
SIMATIC S7 - PPI [1]	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B
SIMATIC S7 - MPI	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B
SIMATIC S7 - PROFIBUS	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B
SIMATIC S7 - PROFINET	—	—	—	—	ETHER-NET [2]	ETHER-NET [2]
SIMATIC HMI HTTP protocol	—	—	—	—	ETHER-NET [2]	ETHER-NET
OPC	—	—	—	—	—	—
Allen-Bradley Ethernet IP	—	—	—	—	ETHER-NET [2]	ETHER-NET
Allen-Bradley DF1	—	IF1B [5]	IF1A [3], IF1B [3][4]	IF1B [5]	IF1B [3] [5] [8]	IF1B [3] [5] [8]
Mitsubishi MC TCP/IP	—	—	—	—	ETHER-NET [2]	ETHER-NET
Mitsubishi FX	—	IF1B	IF1A [3], IF1B [3]	IF1B	IF1B [3] [8]	IF1B [3] [8]
Modicon Modbus TCP	—	—	—	—	ETHER-NET [2]	ETHER-NET
Modicon Modbus RTU	—	IF1B [7]	IF1A [3], IF1B [3][6]	IF1B [7]	IF1B [7] [3] [8]	IF1B [7] [3] [8]
Omron Host Link	—	IF1B	IF1A [3], IF1B [3]	IF1B	IF1B [3] [8]	IF1B [3] [8]

[1] For SIMATIC S7-200 only

[2] Not approved for TP 177B DP, OP 177B DP.

[3] For serial communication, deselect "Remote Control" of "Channel 1" in the "File > Transfer >Options" menu.

4) Only with PLC5 or KF2 module.

5) Direct communication with PLC5 or KF2 module, otherwise only approved with RS422-RS232 converter 6AV6 671-8XE00-0AX0 (option)

6) Can be selected and used, but not approved.

7) Only with RS 422-RS 232 converter 6AV6 671-8XE00-0AX0 (option)

8) "PROFINET IO Enabled" must be disabled.

Area pointers for Panels

Introduction

Area pointers are parameter fields from which the HMI device obtains information about the location and size of data areas in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

WinCC uses the following area pointers:

- Job mailbox
- Project ID
- Screen number
- Data record
- Date/time
- Date/time PLC
- Coordination

Availability of the area pointers

The following table shows the availability of the area pointers on the HMI devices. Note that the area pointers can be used only for available communication drivers.

Area pointers

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Screen number	Yes	Yes	Yes	Yes	Yes	Yes
Data record	No	Yes	Yes	Yes	Yes	Yes
Date/time	Yes	Yes	Yes	Yes	Yes	Yes
Date/time PLC	Yes	Yes	Yes	Yes	Yes	Yes
Coordination	Yes	Yes	Yes	Yes	Yes	Yes
Project ID	Yes	Yes	Yes	Yes	Yes	Yes
Job mailbox	Yes	Yes	Yes	Yes	Yes	Yes

12.11.5.3 Comfort Panel

Communication drivers

SIMATIC communication drivers

Device dependency of the Comfort Panels

Various communication drivers are configured for Comfort Panels depending on the communication.

The following tables show the communication drivers released for integrated and non-integrated communication.

There are different versions of the HMI devices within the integrated communication.

Communication drivers for integrated communication (V11.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
KP400 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
KTP400 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
KTP400 Comfort Portrait	No	Yes	Yes	Yes	Yes	No	Yes
KP700 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
TP700 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
TP700 Comfort Portrait	No	Yes	Yes	Yes	Yes	No	Yes
KP900 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
TP900 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
TP900 Comfort Portrait	No	Yes	Yes	Yes	Yes	No	Yes
KP1200 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
TP1200 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
TP1200 Comfort Portrait	No	Yes	Yes	Yes	Yes	No	Yes
KP1500 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
TP1500 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
TP1500 Comfort Portrait	No	Yes	Yes	Yes	Yes	No	Yes
TP1900 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
TP1900 Comfort Portrait	No	Yes	Yes	Yes	Yes	No	Yes

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
TP2200 Comfort	No	Yes	Yes	Yes	Yes	No	Yes
TP2200 Comfort Portrait	No	Yes	Yes	Yes	Yes	No	Yes

Communication drivers for integrated communication (V12.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
KP400 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP400 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP400 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
KP700 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP700 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP700 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
KP900 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP900 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP900 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
KP1200 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1200 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1200 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
KP1500 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1500 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1500 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1900 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1900 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
TP2200 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP2200 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes

Communication drivers for integrated communication (V13.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
KP400 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP400 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP400 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
KP700 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP700 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP700 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
KP900 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP900 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP900 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
KP1200 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1200 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1200 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
KP1500 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1500 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1500 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1900 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1900 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes
TP2200 Comfort	No	Yes	Yes	Yes	Yes	Yes	Yes
TP2200 Comfort Portrait	No	Yes	Yes	Yes	Yes	Yes	Yes

Communication drivers for non-integrated communication

HMI devices	SIMATIC S7-1500	SIMATIC S7-1200	SIMATIC S7-300/400	SIMATIC S7-200	SIMATIC LOGO!	SIMATIC HTTP protocol
KP400 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
KTP400 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
KTP400 Comfort Portrait	Yes	Yes	Yes	Yes	Yes	Yes
KP700 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
TP700 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
TP700 Comfort Portrait	Yes	Yes	Yes	Yes	Yes	Yes

HMI devices	SIMATIC S7-1500	SIMATIC S7-1200	SIMATIC S7-300/400	SIMATIC S7-200	SIMATIC LOGO!	SIMATIC HTTP protocol
KP900 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
TP900 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
TP900 Comfort Portrait	Yes	Yes	Yes	Yes	Yes	Yes
KP1200 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
TP1200 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
TP1200 Comfort Portrait	Yes	Yes	Yes	Yes	Yes	Yes
KP1500 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
TP1500 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
TP1500 Comfort Portrait	Yes	Yes	Yes	Yes	Yes	Yes
TP1900 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
TP1900 Comfort Portrait	Yes	Yes	Yes	Yes	Yes	Yes
TP2200 Comfort	Yes	Yes	Yes	Yes	Yes	Yes
TP2200 Comfort Portrait	Yes	Yes	Yes	Yes	Yes	Yes

Other communication drivers

Device dependency of the Comfort Panels

The following table shows which communication drivers you can configure with the various Comfort Panels.

Communication drivers

HMI devices	OPC [1]	SIMATIC HTTP protocol	Allen-Bradley EtherNet/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
KP400 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
KTP400 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
KTP400 Comfort Portrait	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
KP700 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP700 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP700 Comfort Portrait	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]

HMI devices	OPC [1]	SIMATIC HTTP protocol	Allen-Bradley EtherNet/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
KP900 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP900 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP900 Comfort Portrait	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
KP1200 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP1200 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP1200 Comfort Portrait	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
KP1500 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP1500 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP1500 Comfort Portrait	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP1900 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP1900 Comfort Portrait	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP2200 Comfort	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]
TP2200 Comfort Portrait	Yes	Yes	Yes	Yes [2] [3]	Yes	Yes [3]	Yes	Yes [3] [4]	Yes [3]

[1] OPC UA client (DA only)
OPC UA server (DA only)

[2] Direct communication with PLC 5 or KF2 module, otherwise only approved with RS422-RS232 converter (option).
Order number: 6AV6 671-8XE00-0AX0

[3] "PROFINET IO Enabled" must be disabled.

[4] Only approved with RS 422-RS232 converter
Order number: 6AV6 671-8XE00-0AX0

Interfaces of Comfort Panels

Device dependency of Comfort Panels

The following table shows which HMI device interfaces are available for the communication driver protocols.

Table 12-151 Comfort Panels

	KP400 Comfort	KTP400 Comfort	KP700 Comfort	TP700 Comfort	KP900 Comfort	TP900 Comfort	KP1200 Comfort	TP1200 Comfort	KP1500 Comfort	TP1500 Comfort	TP1900 Comfort	TP2200 Comfort
SIMATIC S7 - PPI [1]	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B
SIMATIC S7 - MPI	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B
SIMATIC S7 - PROFIBUS	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B	IF1B
SIMATIC S7 - PROFINET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET
SIMATIC HMI HTTP protocol	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET
OPC	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET
Allen-Bradley EtherNet/IP	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET
Allen-Bradley DF1	IF1B [1)3)4)]	IF1B [1)3)4)]	IF1B [1)3)4)]	IF1B [1)3)4)]	IF1B [1)3)4)]	IF1B [1)3)4)]	IF1B [1)3)4)]	IF1B [1)3)4)]	IF1B [1)3)4)]	IF1B [1)3)4)]	IF1B [1)3)4)]	IF1B [1)3)4)]
Mitsubishi MC TCP/IP	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET
Mitsubishi FX	IF1B [3)4)]	IF1B [3)4)]	IF1B [3)4)]	IF1B [3)4)]	IF1B [3)4)]	IF1B [3)4)]	IF1B [3)4)]	IF1B [3)4)]	IF1B [3)4)]	IF1B [3)4)]	IF1B [3)4)]	IF1B [3)4)]
Modicon Modbus TCP/	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET

	KP400 Comfort	KTP400 Comfort	KP700 Comfort	TP700 Comfort	KP900 Comfort	TP900 Comfort	KP1200 Comfort	TP1200 Comfort	KP1500 Comfort	TP1500 Comfort	TP1900 Comfort	TP2200 Comfort
Modicon Modbus RTU	IF1B 2)3)4)	IF1B 2)3)4)	IF1B 2)3)4)	IF1B 2)3)4)	IF1B 2)3)4)	IF1B 2)3)4)	IF1B 2)3)4)	IF1B 2)3)4)	IF1B 2)3)4)	IF1B 2)3)4)	IF1B 2)3)4)	IF1B 2)3)4)
Omron Host Link	IF1B 3)4)	IF1B 3)4)	IF1B 3)4)	IF1B 3)4)	IF1B 3)4)	IF1B 3)4)	IF1B 3)4)	IF1B 3)4)	IF1B 3)4)	IF1B 3)4)	IF1B 3)4)	IF1B 3)4)

1) Direct communication with PLC5 or KF2 module, otherwise only with RS422-RS232 converter.
   Order number: 6AV6 671-8XE00-0AX0
2) only with RS422-RS232 converter (option)
   Order number: 6AV6 671-8XE00-0AX0
3) For serial communication, deselect "Remote Control" of "Channel 1" in the "File > Transfer >Options" menu.
4) "PROFINET IO Enabled" must be disabled.

Area pointers for Comfort Panel

Introduction

Area pointers are parameter fields from which the HMI device obtains information about the location and size of data areas in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

WinCC uses the following area pointers:

- Job mailbox
- Project ID
- Screen number
- Data record
- Date/time
- Date/time PLC
- Coordination

Availability of the area pointers

The following table shows the availability of the area pointers on the HMI devices. Note that the area pointers can be used only for available communication drivers.

Area pointer

	KP400 Comfort	KTP400 Comfort	KP700 Comfort	TP700 Comfort	KP900 Comfort	TP900 Comfort	KP1200 Comfort	TP1200 Comfort	KP1500 Comfort	TP1500 Comfort	TP1900 Comfort	TP2200 Comfort
Screen number	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Data record	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time PLC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Coordination	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Project ID	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Job mailbox	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

12.11.5.4 Multi Panel

Communication drivers

SIMATIC communication drivers

Device dependency of the Multi Panels

Various communication drivers are configured for Comfort Panels depending on the communication.

The following tables show the communication drivers released for integrated and non-integrated communication.

There are different versions of the HMI devices within the integrated communication.

Communication drivers for integrated communication (V11.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
MP 177 6" Touch	No	Yes	Yes	Yes	No	No	No
MP 277 8" Key	No	Yes	Yes	Yes	No	No	Yes
MP 277 10" Key	No	Yes	Yes	Yes	No	No	Yes
MP 277 10" Touch	No	Yes	Yes	Yes	No	No	Yes
MP 377 12" Key	No	Yes	Yes	Yes	No	No	Yes
MP 377 12" Touch	No	Yes	Yes	Yes	No	No	Yes
MP 377 15" Touch	No	Yes	Yes	Yes	No	No	Yes
MP 377 19" Touch	No	Yes	Yes	Yes	No	No	Yes

Communication drivers for integrated communication (V12.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
MP 177 6" Touch	No	Yes	Yes	Yes	Yes	Yes	No
MP 277 8" Key	No	Yes	Yes	Yes	Yes	Yes	Yes
MP 277 10" Key	No	Yes	Yes	Yes	Yes	Yes	Yes
MP 277 10" Touch	No	Yes	Yes	Yes	Yes	Yes	Yes
MP 377 12" Key	No	Yes	Yes	Yes	Yes	Yes	Yes
MP 377 12" Touch	No	Yes	Yes	Yes	Yes	Yes	Yes
MP 377 15" Touch	No	Yes	Yes	Yes	Yes	Yes	Yes
MP 377 19" Touch	No	Yes	Yes	Yes	Yes	Yes	Yes

Communication drivers for non-integrated communication

HMI devices	SIMATIC S7-1500	SIMATIC S7-1200	SIMATIC S7-300/400	SIMATIC S7-200	SIMATIC LOGO!	SIMATIC HTTP protocol
MP 177 6" Touch	Yes	Yes	Yes	Yes	Yes	Yes
MP 277 8" Key	Yes	Yes	Yes	Yes	Yes	Yes
MP 277 10" Key	Yes	Yes	Yes	Yes	Yes	Yes
MP 277 10" Touch	Yes	Yes	Yes	Yes	Yes	Yes
MP 377 12" Key	Yes	Yes	Yes	Yes	Yes	Yes
MP 377 12" Touch	Yes	Yes	Yes	Yes	Yes	Yes
MP 377 15" Touch	Yes	Yes	Yes	Yes	Yes	Yes
MP 377 19" Touch	Yes	Yes	Yes	Yes	Yes	Yes

Other communication drivers

Device dependency of the Multi Panels

The following table shows which communication drivers you can configure with the various Multi Panels.

Communication drivers

HMI devices	OPC [1]	Allen-Bradley EtherNet/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
MP 177 6" Touch	No	Yes	Yes[3)4)]	Yes	Yes[4)]	Yes	Yes[2)4)]	Yes[4)]
MP 277 8" Key	Yes	Yes	Yes[3)4)]	Yes	Yes[4)]	Yes	Yes[2)4)]	Yes[4)]
MP 277 10" Key	Yes	Yes	Yes[3)4)]	Yes	Yes[4)]	Yes	Yes[2)4)]	Yes[4)]

HMI devices	OPC [1]	Allen-Bradley Ether-Net/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
MP 277 10" Touch	Yes	Yes	Yes[3)4)]	Yes	Yes[4)]	Yes	Yes[2)4)]	Yes[4)]
MP 377 12" Key	Yes	Yes	Yes[3)4)]	Yes	Yes[4)]	Yes	Yes[2)4)]	Yes[4)]
MP 377 12" Touch	Yes	Yes	Yes[3)4)]	Yes	Yes[4)]	Yes	Yes[2)4)]	Yes[4)]
MP 377 15" Touch	Yes	Yes	Yes[3)4)]	Yes	Yes[4)]	Yes	Yes[2)4)]	Yes[4)]
MP 377 19" Touch	Yes	Yes	Yes[3)4)]	Yes	Yes[4)]	Yes	Yes[2)4)]	Yes[4)]

1) OPC-XML DA Server
OPC-DA Client (Inproc)

2) only approved with RS 422-RS232 converter
Order number: 6AV6 671-8XE00-0AX0

3) Direct communication with PLC 5 or KF2 module, otherwise only with RS422-RS232 converter (option).
Order number: 6AV6 671-8XE00-0AX0

4) "PROFINET IO Enabled" must be disabled.

Interfaces of Multi Panels

Device dependency of Multi Panels

The following table shows which HMI device interfaces are available for the communication driver protocols.

Table 12-152 Multi Panels

	MP 177 [3)]	MP 277 [3)]	MP 377 [3)]
SIMATIC S7 - PPI [1)]	IF1B	IF1B	IF1B
SIMATIC S7 - MPI	IF1B	IF1B	IF1B
SIMATIC S7 - PROFIBUS	IF1B	IF1B	IF1B
SIMATIC S7 - PROFINET	ETHERNET	ETHERNET	ETHERNET
SIMATIC HMI HTTP protocol	ETHERNET	ETHERNET	ETHERNET
OPC	-	ETHERNET	ETHERNET
Allen-Bradley EtherNet/IP	ETHERNET	ETHERNET	ETHERNET
Allen-Bradley DF1	IF1B [3) 4) 5)]	IF1B [3) 4) 5)]	IF1B [3) 4) 5)]
Mitsubishi MC TCP/IP	ETHERNET	ETHERNET	ETHERNET
Mitsubishi FX	IF1B [3)5)]	IF1B [3)5)]	IF1B [3)5)]
Modicon Modbus TCP	ETHERNET	ETHERNET	ETHERNET

	MP 177 3)	MP 277 3)	MP 377 3)
Modicon Modbus RTU	IF1B 2) 3) 5)	IF1B 2) 3) 5)	IF1B 2) 3) 5)
Omron Host Link	IF1B 3)5)	IF1B 3)5)	IF1B 3)5)

1) For SIMATIC S7-200 only

2) only with RS 422-RS232 converter (option)
Order number: 6AV6 671-8XE00-0AX0

3) For serial communication, deselect "Remote Control" of "Channel 1" in the "File > Transfer >Options" menu.

4) Direct communication with PLC5 or KF2 module, otherwise only approved with RS422-RS232 converter (option).
Order number: 6AV6 671-8XE00-0AX0

5) "PROFINET IO Enabled" must be disabled.

Area pointers for Multi Panel

Introduction

Area pointers are parameter fields from which the HMI device obtains information about the location and size of data areas in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

WinCC uses the following area pointers:

- Job mailbox
- Project ID
- Screen number
- Data record
- Date/time
- Date/time PLC
- Coordination

Availability of the area pointers

The following table shows the availability of the area pointers on the HMI devices. Note that the area pointers can be used only for available communication drivers.

Area pointers

	MP 177	MP 277	MP 377
Screen number	Yes	Yes	Yes
Data record	Yes	Yes	Yes

	MP 177	MP 277	MP 377
Date/time	Yes	Yes	Yes
Date/time PLC	Yes	Yes	Yes
Coordination	Yes	Yes	Yes
Project ID	Yes	Yes	Yes
Job mailbox	Yes	Yes	Yes

12.11.5.5 Mobile Panel

Communication drivers

SIMATIC communication drivers

Device dependency of the Mobile Panels

Various communication drivers are configured for Comfort Panels depending on the communication.

The following tables show the communication drivers released for integrated and non-integrated communication.

There are different versions of the HMI devices within the integrated communication.

Communication drivers for integrated communication (V11.0)

HMI devices	SIMAT-IC S7-1200 (V1)	SIMAT-IC S7-1200 (V2)	SIMAT-IC S7-1200 (V2.2)	SIMAT-IC S7-1200 (V3)	SIMAT-IC S7-1200 (V4)	SIMAT-IC S7-1500	SIMAT-IC S7-300/400
Mobile Panel 177 6" DP	No	Yes	Yes	Yes	Yes	No	Yes
Mobile Panel 177 6" PN	No	Yes	Yes	Yes	Yes	No	Yes
Mobile Panel 277 8"	No	Yes	Yes	Yes	Yes	No	Yes
Mobile Panel 277 8" IWLAN V2	No	Yes	Yes	Yes	Yes	No	Yes
Mobile Panel 277F 8" IWLAN V2	No	No	No	No	No	No	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	No	No	No	No	No	No	Yes
Mobile Panel 277 10"	No	Yes	Yes	Yes	Yes	No	Yes

Communication drivers for integrated communication (V12.0)

HMI devices	SIMAT-IC S7-1200 (V1)	SIMAT-IC S7-1200 (V2)	SIMAT-IC S7-1200 (V2.2)	SIMAT-IC S7-1200 (V3)	SIMAT-IC S7-1200 (V4)	SIMAT-IC S7-1500	SIMAT-IC S7-300/400
Mobile Panel 177 6" DP	No	Yes	Yes	Yes	Yes	Yes	Yes
Mobile Panel 177 6" PN	No	Yes	Yes	Yes	Yes	Yes	Yes
Mobile Panel 277 8"	No	Yes	Yes	Yes	Yes	Yes	Yes
Mobile Panel 277 8" IWLAN V2	No	Yes	Yes	Yes	Yes	Yes	Yes
Mobile Panel 277F 8" IWLAN V2	No	Yes[1)]	Yes[1)]	Yes[1)]	Yes[1)]	Yes	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	No	Yes[1)]	Yes[1)]	Yes[1)]	Yes[1)]	Yes	Yes
Mobile Panel 277 10"	No	Yes	Yes	Yes	Yes	Yes	Yes

[1)] Not as a master system

Communication drivers for integrated communication (V13.0.1)

HMI devices	SIMAT-IC S7-1200 (V1)	SIMAT-IC S7-1200 (V2)	SIMAT-IC S7-1200 (V2.2)	SIMAT-IC S7-1200 (V3)	SIMAT-IC S7-1200 (V4)	SIMAT-IC S7-1500	SIMAT-IC S7-300/400
KTP700 Mobile	Yes	Yes	Yes	Yes	Yes	Yes	Yes
KTP700F Mobile	Yes	Yes	Yes	Yes	Yes	Yes	Yes
KTP900 Mobile	Yes	Yes	Yes	Yes	Yes	Yes	Yes
KTP900F Mobile	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Communication drivers for non-integrated communication

HMI devices	SIMAT-IC S7-1500	SIMAT-IC S7-1200	SIMAT-IC S7-300/400	SIMAT-IC S7-200	SIMATIC LOGO!	SIMATIC HTTP protocol
Mobile Panel 177 6" DP	Yes[2)]	Yes	Yes	Yes	No	No
Mobile Panel 177 6" PN	Yes[2)]	Yes	Yes	Yes	Yes	Yes
Mobile Panel 277 8"	Yes	Yes	Yes	Yes	Yes	Yes
Mobile Panel 277 8" IWLAN V2	Yes	Yes	Yes	Yes	Yes	Yes

HMI devices	SIMAT-IC S7-1500	SIMAT-IC S7-1200	SIMAT-IC S7-300/400	SIMAT-IC S7-200	SIMATIC LOGO!	SIMATIC HTTP protocol
Mobile Panel 277F 8" IWLAN V2	Yes	Yes	Yes	Yes	Yes	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	Yes	Yes	Yes	Yes	Yes	Yes
Mobile Panel 277 10"	Yes	Yes	Yes	Yes	Yes	Yes
KTP700 Mobile	Yes	Yes	Yes	Yes	Yes	Yes
KTP700F Mobile	Yes	Yes	Yes	Yes	Yes	Yes
KTP900 Mobile	Yes	Yes	Yes	Yes	Yes	Yes
KTP900F Mobile	Yes	Yes	Yes	Yes	Yes	Yes

1) "PROFINET IO Enabled" must be disabled.

2) You must activate "Access via PUT/GET communication" for communication with the SIMATIC S7-1500 PLC.

Activate PUT/GET communication in the PLC properties at:

"General > Security > Enable PUT/GET communication with remote partner (PLC, HMI, OPC, etc.)"

Other communication drivers

Device dependency of theMobile Panels

The following table shows which communication drivers you can configure with the various Mobile Panels.

Communication drivers

HMI devices	OPC	SIMATIC HTTP protocol	Allen-Bradley Ether-Net/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
Mobile Panel 177 6" DP	No	No	No	Yes 3)	No	Yes 3)	No	Yes 3)	Yes 3)
Mobile Panel 177 6" PN	No	Yes	Yes	No	Yes	No	Yes	No	No
Mobile Panel 277 8"	No 1)	Yes	Yes	Yes 3)	Yes	Yes 3)	Yes	Yes 3)	Yes 3)

HMI devices	OPC	SIMATIC HTTP protocol	Allen-Bradley Ether-Net/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
Mobile Panel 277 8" IWLAN V2	No [1]	Yes	Yes	No	Yes	No	Yes	No	No
Mobile Panel 277F 8" IWLAN V2	No [1]	Yes	Yes	No	No	No	No	No	No
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	No [1]	Yes	Yes	No	No	No	No	No	No
Mobile Panel 277 10"	No [1]	Yes	Yes	Yes [3]	Yes	Yes [3]	Yes	Yes [3]	Yes [3]
KTP700 Mobile	Yes [2]	Yes	Yes	No	Yes	No	Yes	No	No
KTP700F Mobile	Yes [2]	Yes	Yes	No	Yes	No	Yes	No	No
KTP900 Mobile	Yes [2]	Yes	Yes	No	Yes	No	Yes	No	No
KTP900F Mobile	Yes [2]	Yes	Yes	No	Yes	No	Yes	No	No

[1] OPC-XML DA server (DA only)

[2] OPC UA client (DA only)
OPC UA server (DA only)

[3] "PROFINET IO Enabled" must be disabled.

Interfaces of Mobile Panels

Device dependency of Mobile Panels

The following table shows which HMI device interfaces are available for the communication driver protocols.

Table 12-153 Mobile Panels

	Mobile Panel 177 DP [3]	Mobile Panel 177 PN	Mobile Panel 277 [2][3]	Mobile Panel 277 IWLAN V2	Mobile Panel 277F V2 IW-LAN	Mobile Panel 277F IWLAN V2 (RFID tag)
SIMATIC S7 - PPI [1]	IF1B	—	IF1B	—	—	—
SIMATIC S7 - MPI	IF1B	—	IF1B	—	—	—

	Mobile Panel 177 DP [3]	Mobile Panel 177 PN	Mobile Panel 277 [2][3]	Mobile Panel 277 IWLAN V2	Mobile Panel 277F V2 IWLAN	Mobile Panel 277F IWLAN V2 (RFID tag)
SIMATIC S7 - PROFIBUS	IF1B	—	IF1B	—	—	—
SIMATIC S7 - PROFINET	—	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET
SIMATIC HMI HTTP protocol	—	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET
OPC	—	—	—	ETHERNET	ETHERNET	ETHERNET
Allen-Bradley DF1	IF1A [3], IF1B [3] [4] [6] (terminal box)	—	IF1A [3], IF1B [3] [4] [6] (terminal box)	—	—	—
Allen-Bradley EtherNet/IP	---	ETHERNET	ETHERNET	ETHERNET	ETHERNET	ETHERNET
Mitsubishi MC TCP/IP	--	ETHERNET	ETHERNET	ETHERNET	—	—
Mitsubishi FX	IF1A [3], IF1B [3] [6] (terminal box)	—	IF1A [3], IF1B [3] [6] (terminal box)	—	—	—
Modicon Modbus RTU	IF1A [3], IF1B [3] [5] [6] (terminal box)	—	IF1A [3], IF1B [3] [5] [6] (terminal box)	—	—	—
Modicon Modbus TCP/IP	—	ETHERNET	ETHERNET	ETHERNET	—	—
Omron Host Link	IF1A [3], IF1B [3] [6] (terminal box)	—	IF1A [3], IF1B [3][6] (terminal box)	—	—	—

1) For SIMATIC S7-200 only
2) Depending on the terminal box used
3) For serial communication, deselect "Remote Control" of "Channel 1" in the "File > Transfer >Options" menu.
4) With PLC5 and KF2 module only
5) Can be selected and used, but not approved.
6) "PROFINET IO Enabled" must be disabled.

Area pointers for Mobile Panel

Introduction

Area pointers are parameter fields from which the HMI device obtains information about the location and size of data areas in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

WinCC uses the following area pointers:

- Job mailbox
- Project ID
- Screen number
- Data record
- Date/time
- Date/time PLC
- Coordination

Availability of the area pointers

The following table shows the availability of the area pointers on the HMI devices. Note that the area pointers can be used only for available communication drivers.

Area pointer

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IW-LAN V2	Mobile Panel 277F IW-LAN V2 (RFID tag)	KTP700 Mobile KTP700F Mobile	KTP900 Mobile KTP900F Mobile
Screen number	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Data record	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time PLC	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Coordination	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Project ID	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Job mailbox	Yes	Yes	Yes	Yes	Yes	Yes	Yes

12.11.5.6 PC systems

Communication drivers

SIMATIC communication drivers

Device dependency

The following table shows which communication drivers you can configure with the various Runtimes on a PC.

Communication drivers for integrated communication (V11.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
WinCC RT Advanced	No	Yes	Yes	Yes	No	No	Yes
WinCC RT Professional	No	Yes	Yes	Yes	No	No	Yes

Communication drivers for integrated communication (V12.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
WinCC RT Advanced	No	Yes	Yes	Yes	Yes	Yes	Yes
WinCC RT Professional	No	Yes	Yes	Yes	Yes	Yes	Yes

Communication drivers for integrated communication (V13.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
WinCC RT Advanced	No	Yes	Yes	Yes	Yes	Yes	Yes
WinCC RT Professional	No	Yes	Yes	Yes	Yes	Yes	Yes

Communication drivers for non-integrated communication

HMI devices	SIMATIC S7-1500	SIMATIC S7-1200	SIMATIC S7-300/400	SIMATIC S7-200	SIMATIC LOGO!	SIMATIC HTTP protocol
WinCC RT Advanced	Yes	Yes	Yes	Yes	Yes	Yes
WinCC RT Professional	Yes	Yes	Yes	No	No	No

1) OPC DA server
OPC DA client
OPC XML DA client
OPC UA server (DA only)
OPC UA client (DA only)

2) OPC DA server
OPC DA client
OPC XML DA server
OPC XML DA client
OPC UA server (DA only)

Other communication drivers

Device dependency

The following table shows which communication drivers you can configure with the various Runtimes on a PC.

Communication drivers

HMI devices	OPC	Allen-Bradley EtherNet/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
WinCC RT Advanced	Yes[1)]	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WinCC RT Professional	Yes[2)]	Yes	No	Yes	No	Yes	No	No

1) OPC DA server
OPC DA client
OPC XML DA client
OPC UA server (DA only)
OPC UA client (DA only)

2) OPC DA server
OPC DA client
OPC XML DA server
OPC XML DA client
OPC UA server (DA only)

Interfaces for PC systems

Device dependency

The following table shows which HMI device interfaces are available for the communication driver protocols.

	WinCC RT Advanced	WinCC RT Professional
SIMATIC S7 - PPI	MPI/DP [2]	--
SIMATIC S7 - MPI	MPI/DP [2]	MPI/DP [2]
SIMATIC S7 - PROFIBUS	MPI/DP [2]	MPI/DP [2]
SIMATIC S7 - PROFINET	ETHERNET	ETHERNET
SIMATIC HMI HTTP protocol	ETHERNET	ETHERNET
OPC	ETHERNET	ETHERNET
OPC UA	ETHERNET	--
Allen-Bradley EtherNet/IP	ETHERNET	ETHERNET
Allen-Bradley DF1	COM1 to COM4 [1]	--
Mitsubishi MC TCPI/IP	ETHERNET	ETHERNET
Mitsubishi FX	COM1 to COM4 [1]	--
Modicon Modbus TCP/IP	ETHERNET	ETHERNET
Modicon Modbus RTU	COM1 to COM4 [1]	--
Omron Host Link	COM1 to COM4 [1]	--

[1] COM2 is blocked for Panel PC 477

[2] for SIMATIC PC, via integrated interface; for standard PC, e.g., via SIMATIC NET CP 5611 A2

Area pointers for PC systems

Introduction

Area pointers are parameter fields from which WinCC RT Advanced obtains information about the location and size of data areas in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

WinCC uses the following area pointers:

- Job mailbox
- Project ID
- Screen number
- Data record
- Date/time

- Date/time PLC
- Coordination

Availability of the area pointers

The following table shows the availability of the area pointers on the HMI devices. Note that the area pointers can be used only for available communication drivers.

Table 12-154 WinCC RT Advanced

	WinCC RT Advanced
Screen number	Yes
Data record	Yes
Date/time	Yes
Date/time PLC	Yes
Coordination	Yes
Project ID	Yes
Job mailbox	Yes

12.11.5.7 Parallel communication

Parallel communication of communication drivers

The following table shows an overview of which communication drivers you can use simultaneously on one HMI device.

Note

Parallel communication is not approved for Basic Panels.

Parallel communication over Ethernet interfaces

The approved combinations can be operated via the same Ethernet interface. Several Ethernet interfaces are not required.

Parallel communication only concerns the Ethernet-based communication drivers.

	Allen-Bradley Ether-Net/IP	Mitsubishi MC TCP/IP	Modicon Modbus TCPIP	OPC (DA/ XML DA)	OPC UA (DA)	SIMATIC LOGO!	SIMATIC S7 200	SIMATIC S7 300/400	SIMATIC S7 1200	SIMATIC S7 1500	SIMATIC HTTP protocol	Sinumerik NC
Allen-Bradley Ether-Net/IP	--	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Mitsubishi MC TCP/IP	No	--	No	Yes	Yes	No	No	No	No	No	Yes	No
Modicon Modbus TCPIP	No	No	--	Yes	Yes	No	No	No	No	No	Yes	No
OPC (DA/ XML DA)	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OPC UA (DA)	Yes	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SIMATIC LOGO!	Yes	No	No	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes	Yes
SIMATIC S7 200	Yes	No	No	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes
SIMATIC S7 300/400	Yes	No	No	Yes	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes
SIMATIC S7 1200	Yes	No	No	Yes	Yes	Yes	Yes	Yes	--	Yes	Yes	Yes
SIMATIC S7 1500	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	--	Yes	Yes
SIMATIC HTTP protocol	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	--	Yes
Sinumerik NC	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	--

Parallel communication over serial interfaces

The following applies for parallel communication over serial interfaces:

- One communication driver per interface.
- One interface per communication driver.

12.11.6 Communicating with SIMATIC S7 1500

12.11.6.1 Communication with SIMATIC S7 1500

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 1500 PLC.

You can configure the following communication channels for the SIMATIC S7 1500 PLC:

- PROFINET
- PROFIBUS

HMI connection for communication

Configure connections between the HMI device and a SIMATIC S7 1500 in the "Devices & Networks" Editor.

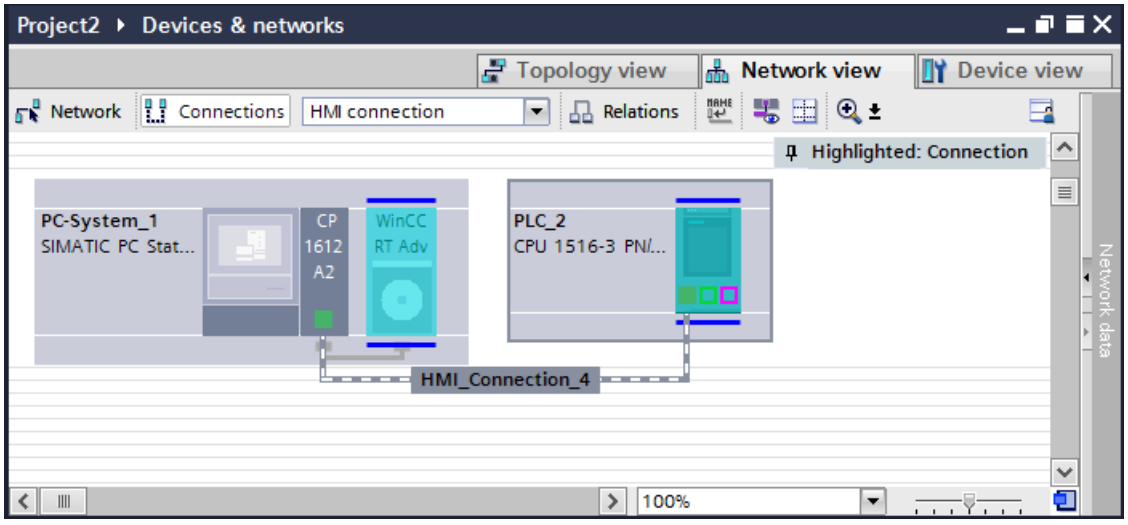
12.11.6.2 Communication via PROFINET

Configuring an HMI connection

Communication via PROFINET

HMI connections via PROFINET

If you have inserted an HMI device and a SIMATIC S7 1500 into the project, interconnect the two PROFINET interfaces in the "Devices & Networks" editor.



You can also connect multiple HMI devices to a single SIMATIC S7 1500 and multiple SIMATIC S7 1500 to a single HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 1500 via PROFINET or Ethernet in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

Requirements

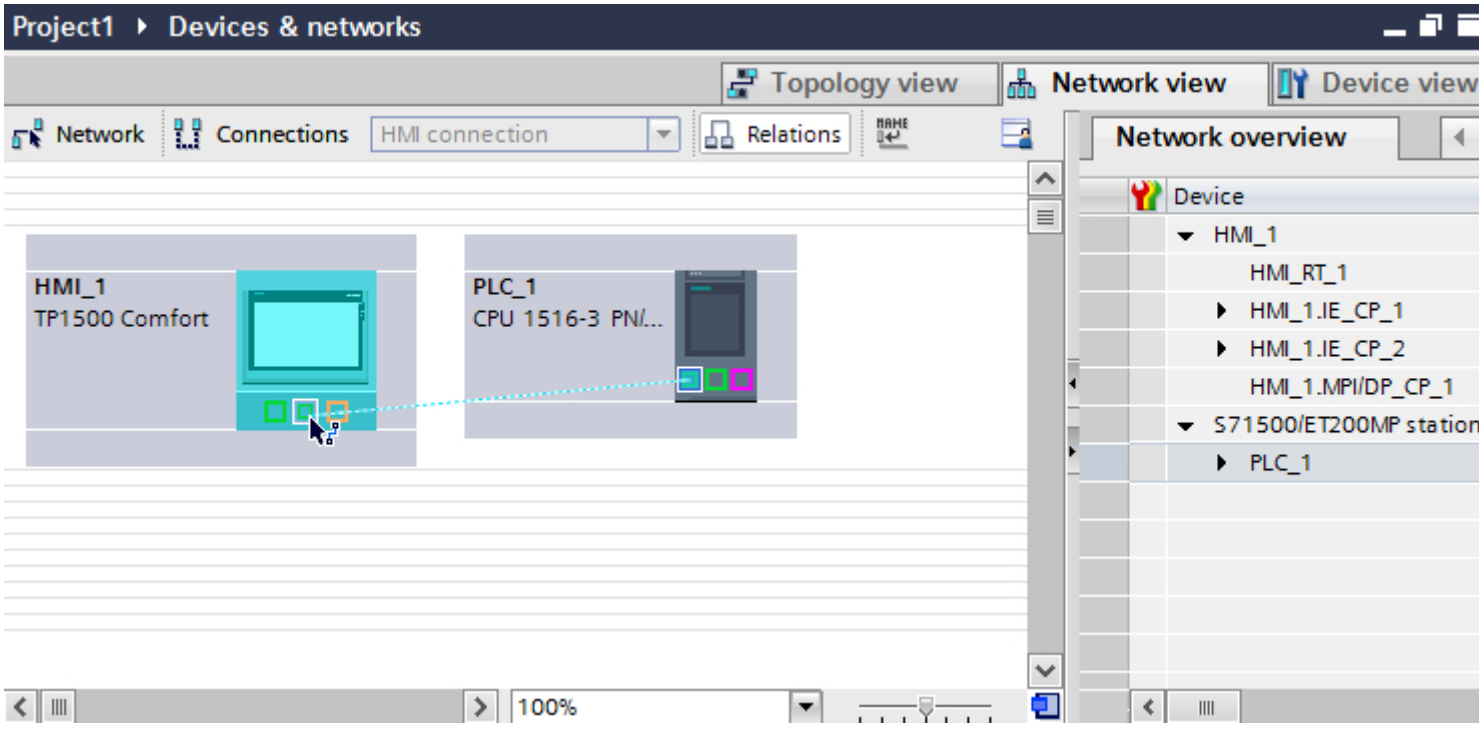
The following communication partners are created in the "Devices & Networks" editor:

- HMI device with PROFINET or Ethernet interface
- SIMATIC S7 1500 with PROFINET interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

3. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the HMI device.



4. Click the connecting line.
5. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
6. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFINET parameters (Page 6093)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 1500. The IP address and subnet mask connection parameters are configured.

Configuring an HMI connection

Communication via PROFINET

Communication via PROFINET

This chapter describes communication over PROFINET between a WinCC Runtime and the SIMATIC S7 1500 PLC.

You can use the following WinCC Runtimes as an HMI device:

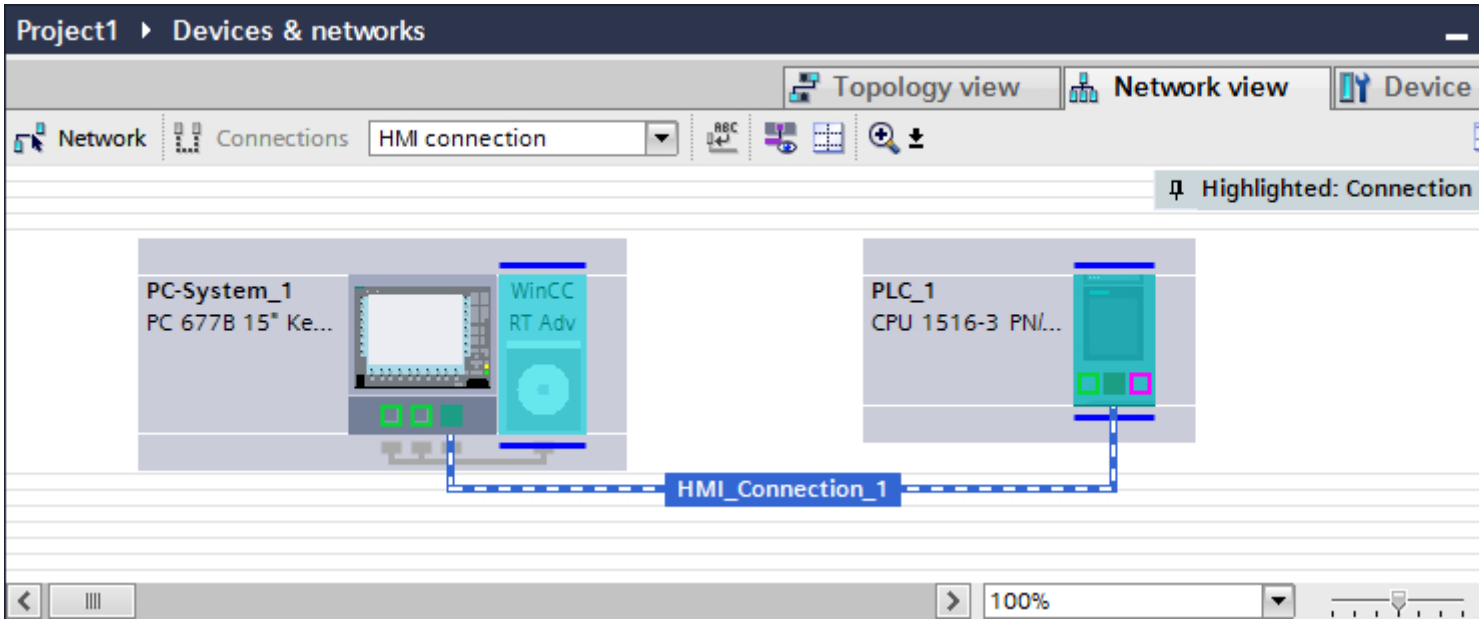
- WinCC RT Advanced

WinCC Runtime as HMI device

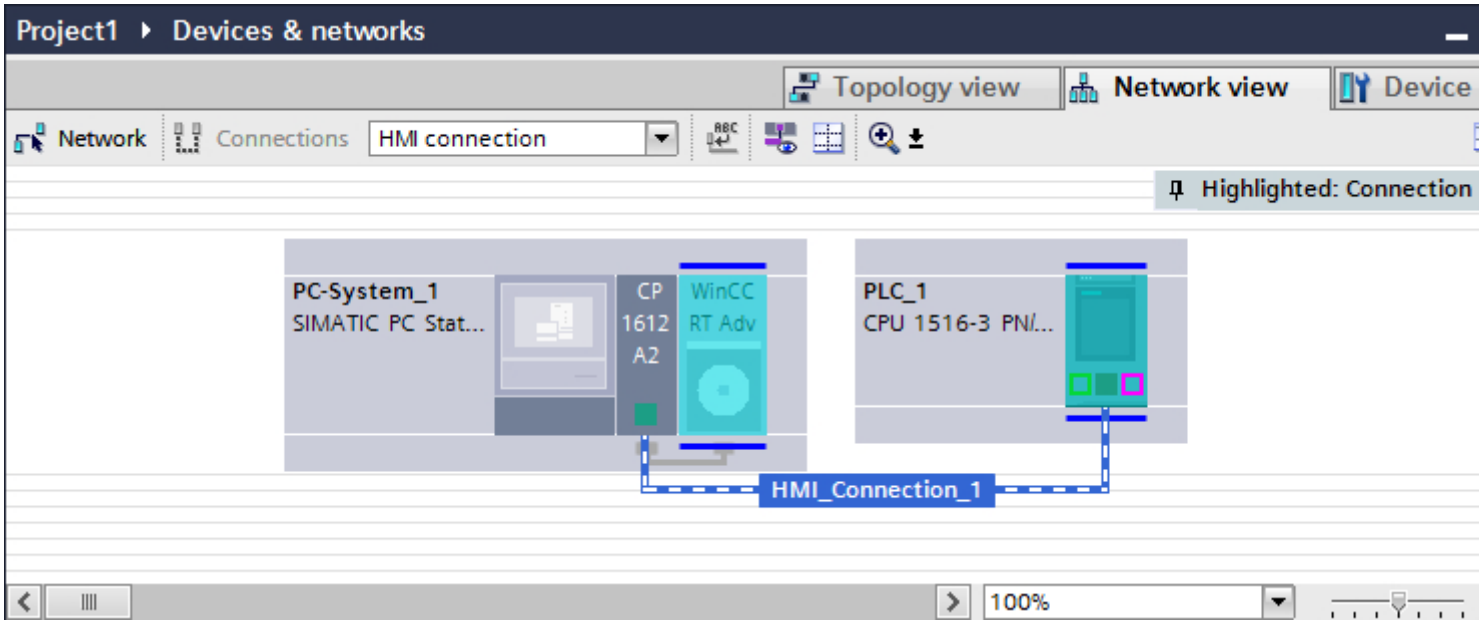
Configure the HMI connections between a WinCC Runtime and SIMATIC S7 1500 in the "Devices & Networks" editor.

If you want to use a WinCC Runtime as an HMI device, you have the following options:

1. You create a SIMATIC PC and configure a WinCC Runtime on this SIMATIC PC.
   In this way you use the SIMATIC PC with a WinCC Runtime as an HMI device.



2. You create a WinCC Runtime and configure a communication processor to the Runtime.
   In this way you use your configuration PC with a WinCC Runtime as an HMI device.



You can also connect multiple HMI devices to a single SIMATIC S7 1500 and multiple SIMATIC S7 1500 to a single HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET with a PC

Introduction

You configure an HMI connection over PROFINET or Ethernet between HMI devices and a SIMATIC S7 1500 in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

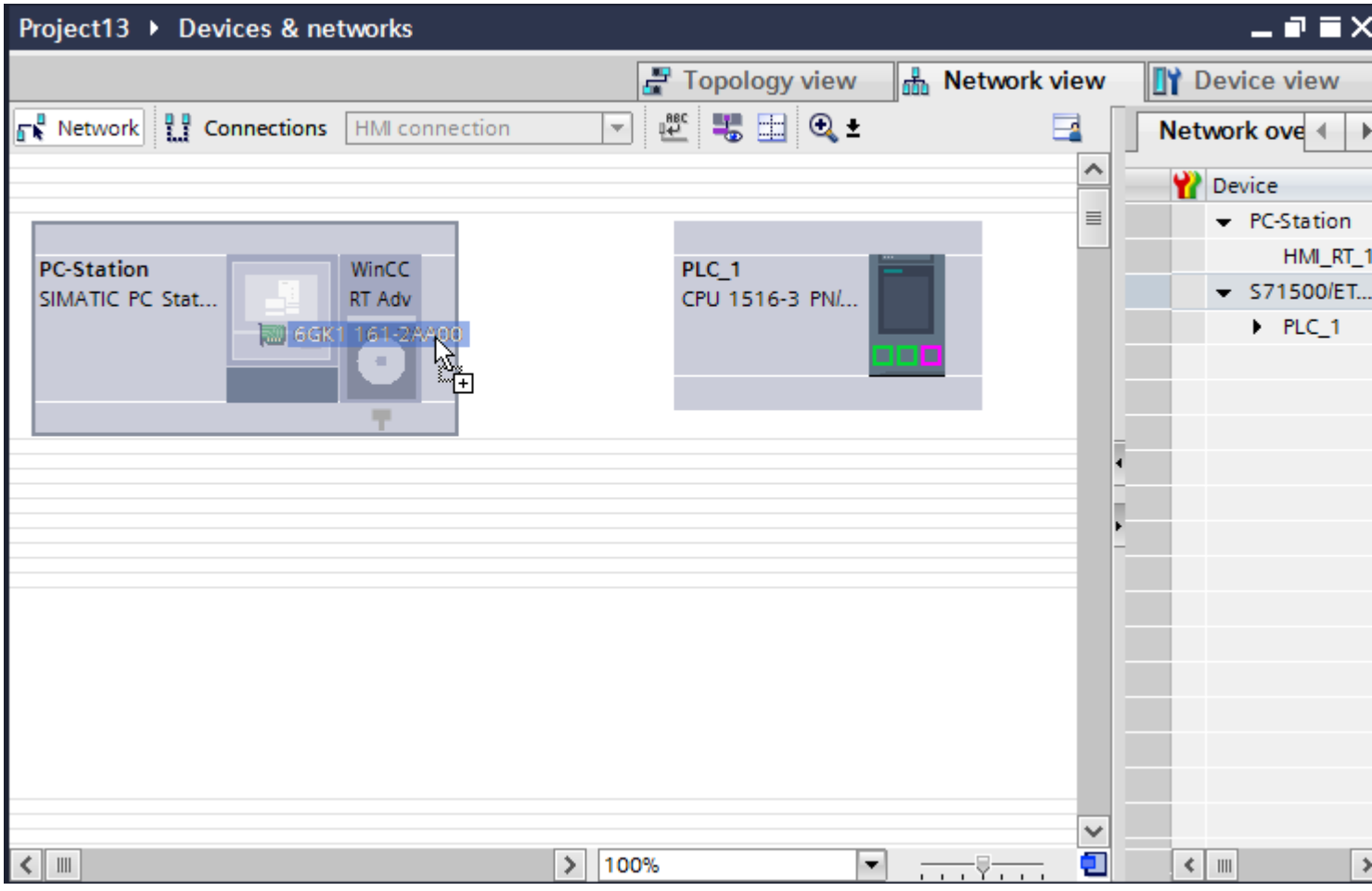
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 1500 with PROFINET interface
- PC station with a "WinCC Runtime" application

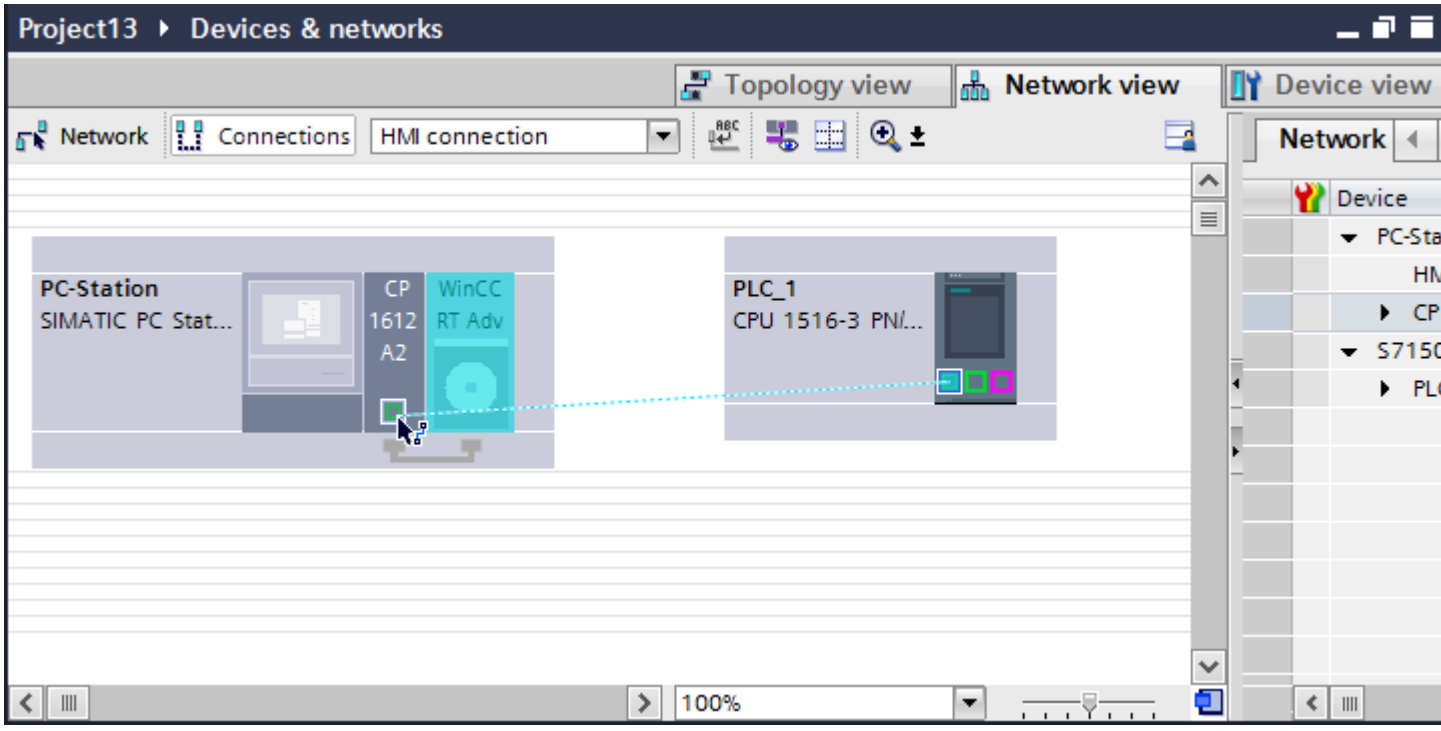
Procedure

1. Double-click the "Devices & Networks" item in the project navigation.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a PROFINET-capable communication processor from the hardware catalog to the WinCC Runtime.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET interface of the communication processor.



5. Click the connected interface on the PC station and select the entry "PC station" under "Interface assignment" in the Inspector window.
6. Click the connecting line.
   The connection is displayed graphically in the Inspector window.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFINET parameters (Page 6093)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 1500. The IP address and subnet mask connection parameters are configured.

Configuring an HMI connection via PROFINET with a SIMATIC PC

Introduction

You configure an HMI connection over PROFINET or Ethernet between HMI devices and SIMATIC S7 1500 in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

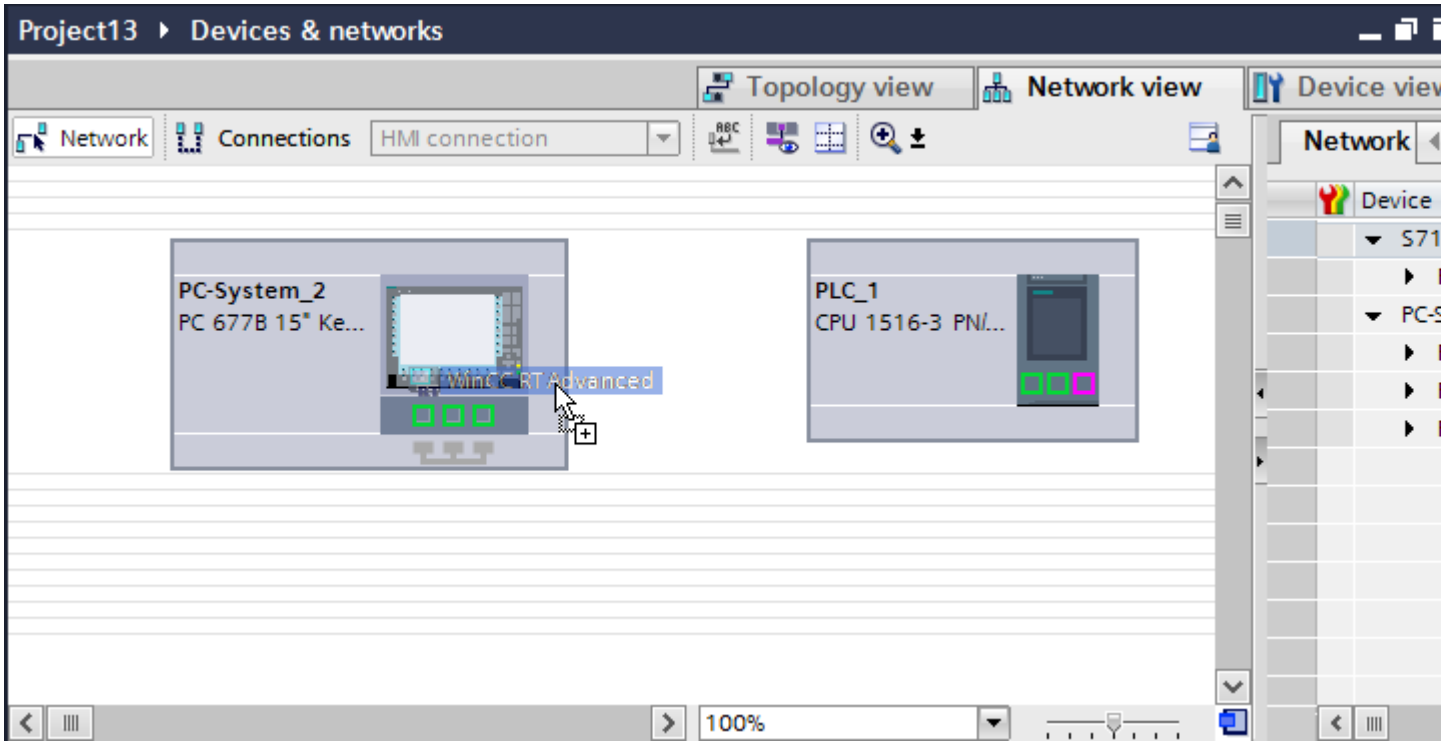
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 1500 with PROFINET interface
- SIMATIC PC with PROFINET interface

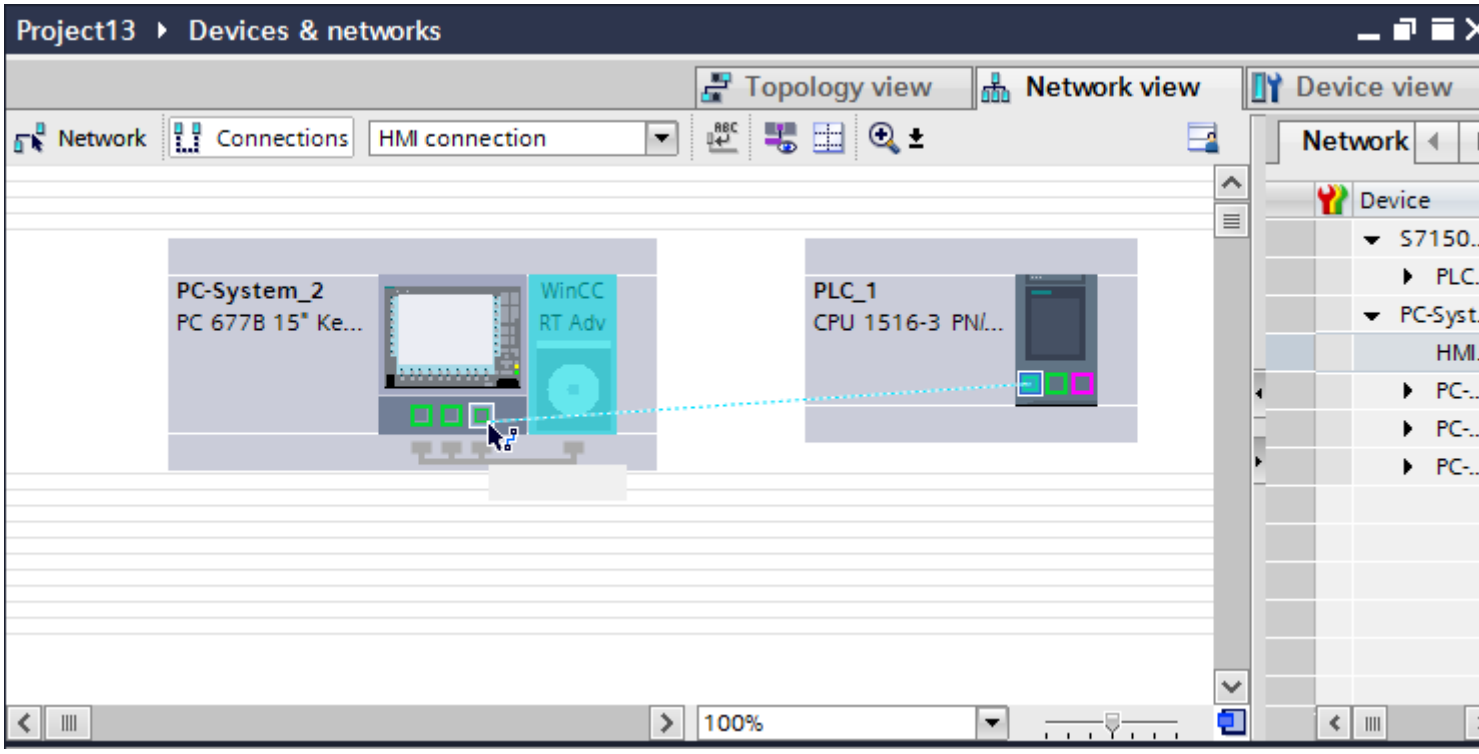
Procedure

1. Double-click the "Devices & Networks" item in the project navigation.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a WinCC Runtime from the hardware catalog to the PC.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the PC.



5. Click the connected interface on the PC station and select the entry "PC station" under "Interface assignment" in the Inspector window.
6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFINET parameters (Page 6093)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 1500. The IP address and subnet mask connection parameters are configured.

PROFINET parameters

PROFINET parameters for the HMI connection

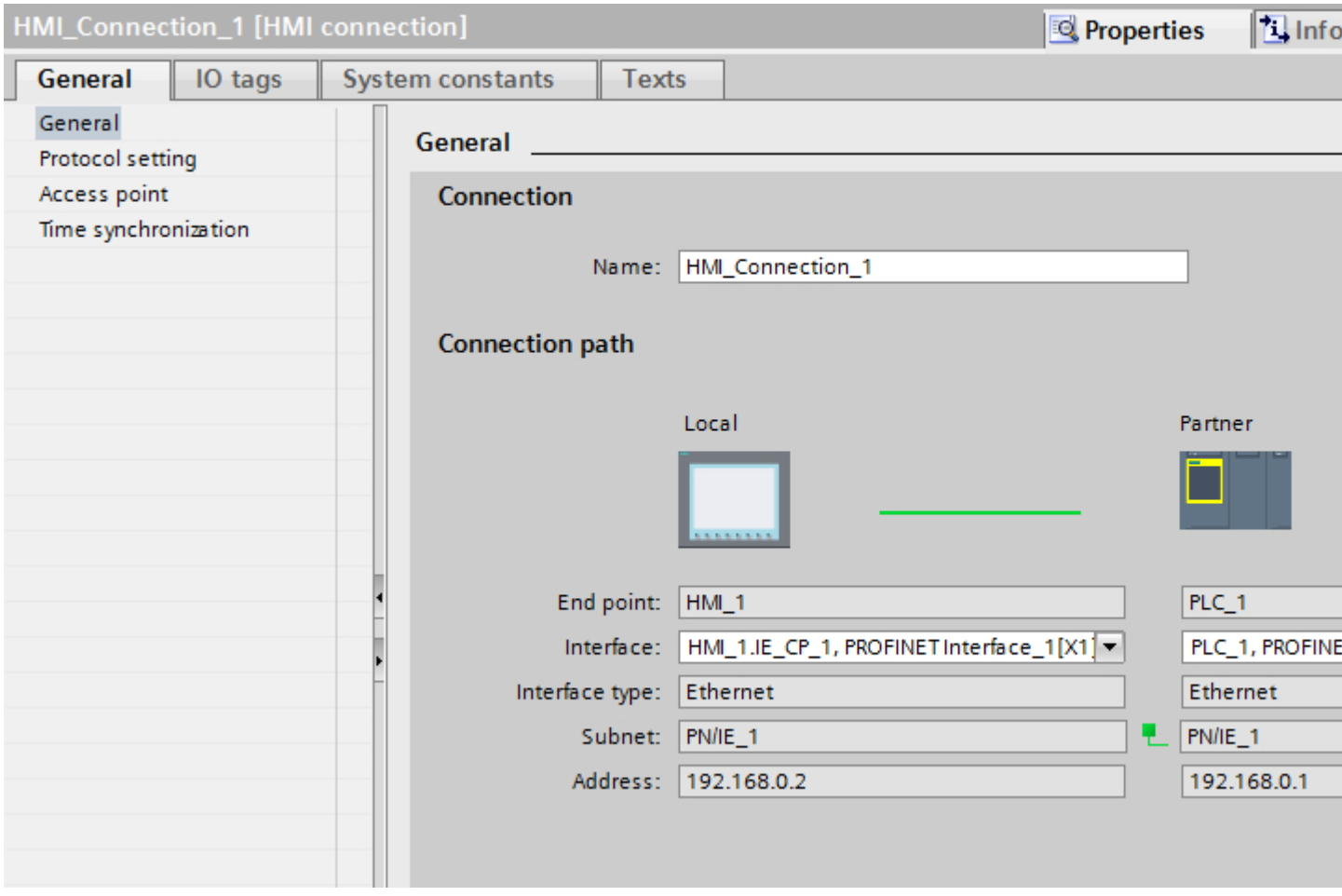
PROFINET parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Shows the name of the HMI connection.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFINET parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area cannot be edited.
- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the selected IP address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

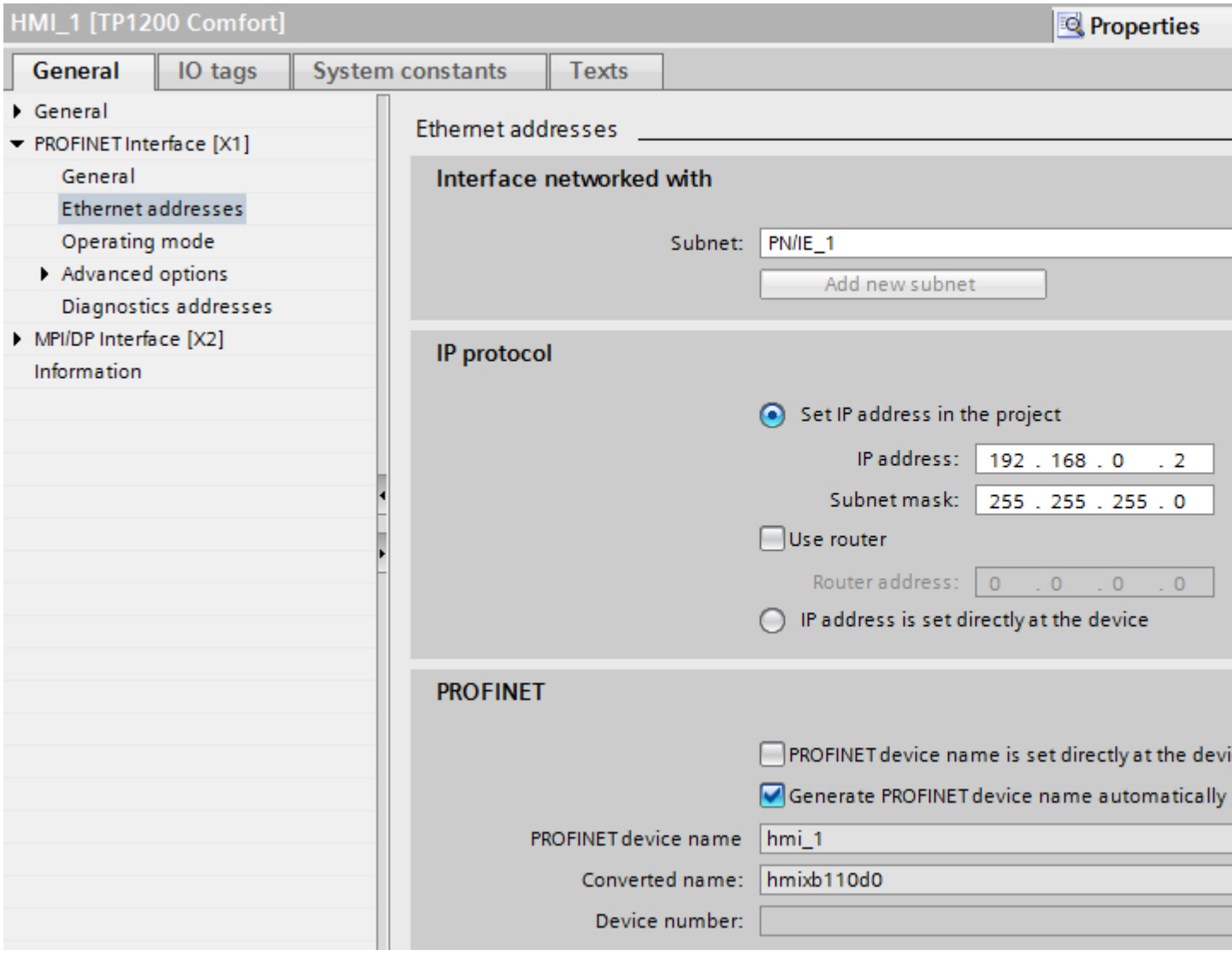
PROFINET parameters for the HMI device

PROFINET parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Set IP address in the project"
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

  Note

  The device is automatically restarted in the case of HMI devices with the Windows CE 3.0 operating system.

  HMI devices with Windows CE 3.0:
  - Mobile Panel 177 PN
  - Mobile Panel 177 DP

- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
- "Use IP router"
  If you are using an IP router, select "Use IP router" and enter the router address in the "Router address" field.
- "Set IP address directly on the device"
  If the "Set IP address on the device" function is enabled, the IP address is not taken from the project. You have to enter the IP address directly in the Control Panel of the HMI device.

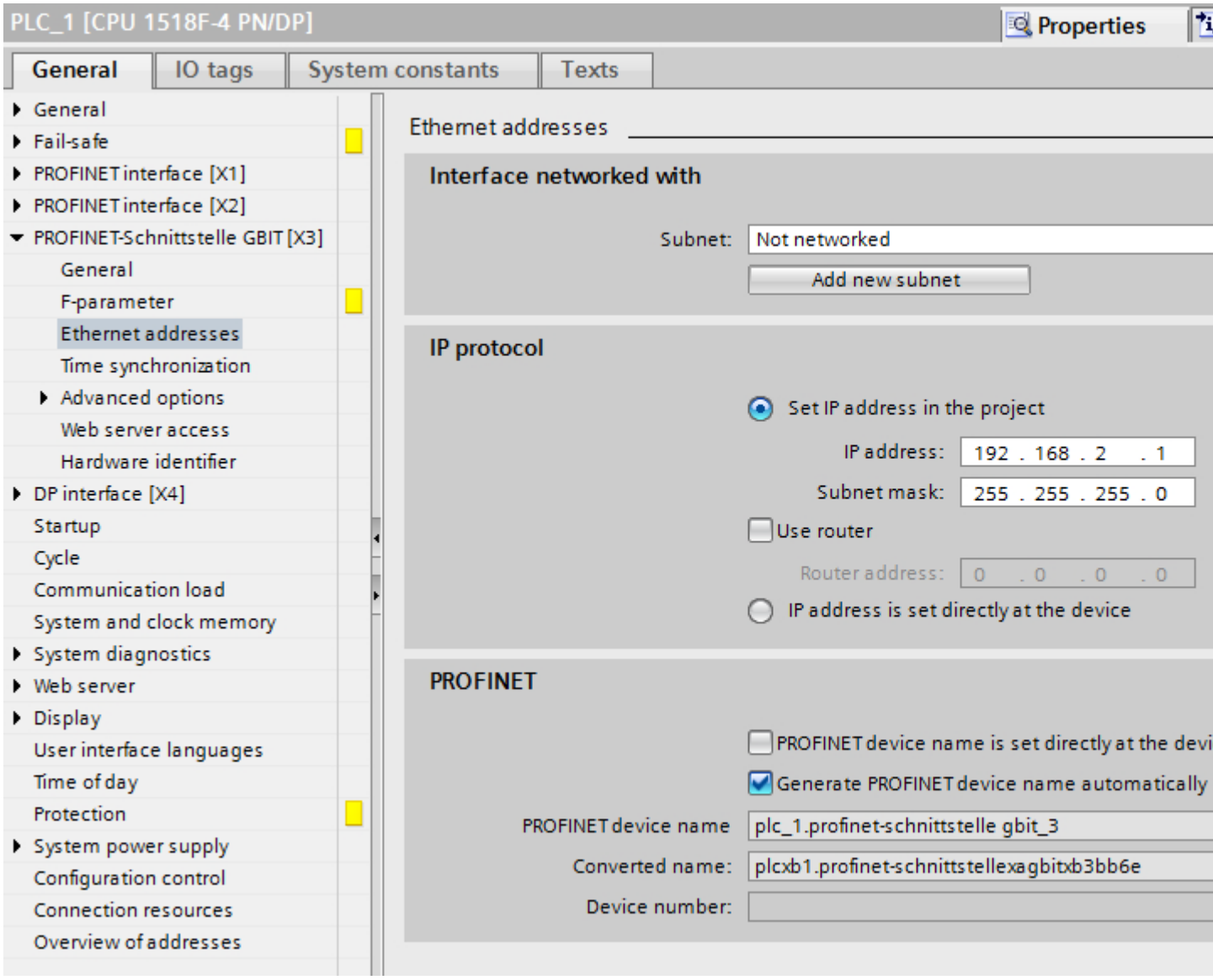
PROFINET parameters for the PLC

PROFINET parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFINET parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "IP address"
  You assign the IP address of the HMI device in the "IP address" area.
- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
  If you are using an IP router, select "Use IP router" and enter the router address in the field.

Protection of communication

Security levels

If you want to protect the controller and HMI device communication, you can assign protection levels for the communication.

For a SIMATIC S7-1500 CPU, you can enter multiple passwords and thereby set up different access rights for various user groups.

The passwords are entered in a table, so that exactly one protection level is assigned to each password.

The effect of the password is given in the "Protection" column.

For the SIMATIC S7-1500 controller, several aspects need to be considered when setting protection levels.

For additional information on this, see:Setting options for the protection (Page 6099)

Example

When configuring the controller, you select the "Complete protection" protection level for a standard CPU (i.e., not an F-CPU).

Afterwards, you enter a separate password for every protection level above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the assigned passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (no protection) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (write protection) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (read and write protection) allows access as if the CPU were read-protected and write-protected, so that only HMI access is possible for users who know this password.

Setting options for the protection

Access levels

The following section describes how to use the various access levels of the S7-1500 CPUs.

S7-1500 CPUs provide various access levels to limit the access to specific functions.

The individual access levels as well as their associated passwords are specified in the object properties of the CPU. You assign parameters for the access level in a table.

Access level	Access			Access permission
	HMI	Read	Write	Password
Full access (no protection)	✓	✓	✓	
Read access	✓	✓		
HMI access	✓			
No access (complete protection)				

The green checkmarks in the columns to the right of the respective access level specify which operations are possible without knowing the password of this access level.

If you want to use the functions of fields that are not selected in the "Access" column, a password has to be entered:

Example:

You set the access level "Read access". You can see from the table that write access is not possible during operation without entering a password.

The table also shows that full access is necessary for the "Write" function.

To use a function requiring write access during operation, the password for full access must therefore be entered.

Notice

Configuring an access level does not replace know-how protection

Configuring access levels offers a high degree of protection against unauthorized changes to the CPU by restricting download privileges to the CPU. However, blocks on the memory card are not write- or read-protected. Use know-how protection to protect the code of blocks on the memory card.

Default characteristics

The default access level is "Full access (no protection)". Every user can read and change the hardware configuration and the blocks. A password is not set and is also not required for online access.

The access levels in detail

Below you can find an explanation of the existing access levels and which functions are possible at the respective access level.

- Full access (no protection):
  The hardware configuration and the blocks can be read and changed by all users.
- Read access for F-blocks (F-CPUs only):
  F-blocks of the safety program cannot be changed without authorization by the password of this access level or a higher level.
  Further information is available in the programming and operating manual *SIMATIC Safety - Configuring and Programming.*
- Read access:
  With this access level, read-only access to the hardware configuration and the blocks is possible without entering a password, which means you can download hardware configuration and blocks to the programming device. In addition, HMI access and access to diagnostics data is possible, as is changing the operating state (RUN/STOP).
  You cannot load blocks or a hardware configuration into the CPU. Moreover, no writing test functions and firmware updates are possible.
- HMI access:
  Only HMI access and access to diagnostics data is possible. Tags can be read and written via a HMI device.
  At this access level, you can neither load blocks and the hardware configuration into the CPU nor load blocks and the hardware configuration from the CPU into the programming device.
  In addition, the following is **not** possible: Writing test functions, changing the operating state (RUN/STOP) and firmware updates.
- No access (complete protection):
  Only identification data can be read, via "Accessible devices", for example.
  Neither read nor write access to the hardware configuration and the blocks is possible. HMI access is also not possible. The server function for PUT/GET communication is disabled in this access level (cannot be changed).
  Legitimation with a configured password provides you with access in accordance with the associated protection level.

Behavior of functions at different access levels

The table below describes which online functions are possible in the various protection levels.

Function	Full access	Read access	HMI access	No access
Identification of the device, via "Accessible devices", for example	Yes	Yes	Yes	Yes
HMI diagnostics view	Yes	Yes	Yes	No
Monitoring tags (M, I, Q, DB content) via HMI device	Yes	Yes	Yes	No
Modifying tags (M, I, Q, DB content) via HMI device	Yes	Yes	Yes	No

Diagnostics display (for example, device information, connection display, alarm display, diagnostic buffer)	Yes	Yes	Yes	No
Reading cycle time statistics (Online & Diagnostics)	Yes	Yes	Yes	No
Reading information from the hardware configuration (Online & Diagnostics)	Yes	Yes	Yes	No
Reading time-of-day	Yes	Yes	Yes	No
Executing online functions within the hardware configuration (Online & Diagnostics)	Yes	Yes	Yes	No
Acknowledging alarms	Yes	Yes	Yes	No
Receiving alarms	Yes	Yes	Yes	No
Enabling/disabling alarms	Yes	Yes	No	No
Reading tags via test function (STEP 7, tag table or watch table)	Yes	Yes	No	No
Requesting operating state change online (RUN/STOP/ warm restart)	Yes	Yes	No	No
Downloading data blocks, code blocks, hardware configuration to PG/PC	Yes	Yes	No	No
Set time-of-day	Yes	Yes	No	No
Deleting data blocks, code blocks, hardware configuration in the CPU	Yes	No	No	No
Downloading individual data blocks, code blocks, hardware configurations to the CPU	Yes	No	No	No
Loading PLC program to the device and resetting	Yes	No	No	No
Firmware update of CPUs or I/O modules	Yes	No	No	No
Modifying tags via test function (STEP 7, watch table)	Yes	No	No	No
Reading tags in the program status	Yes	No	No	No
Online editing of blocks	Yes	No	No	No
Modifying outputs in STOP mode	Yes	No	No	No

Behavior of a password-protected module during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Before an online function is executed, the necessary permission is checked and, if necessary, the user is prompted to enter a password.

Example: The module was configured with read access and you want to execute the "Modify tags" function. This requires write access to a test function; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on.

Access authorization to the protected data is in effect for the duration of the online connection or until the access authorization is manually rescinded with "Online > Delete access rights".

Each access level allows unrestricted access to certain functions without entering a password, for example, identification using the "Accessible devices" function.

Access to a password-protected S7-1500 CPU can be restricted locally on the display. The restriction is only in effect when the operating mode switch is set to RUN.

Access password for the HMI connection

Introduction

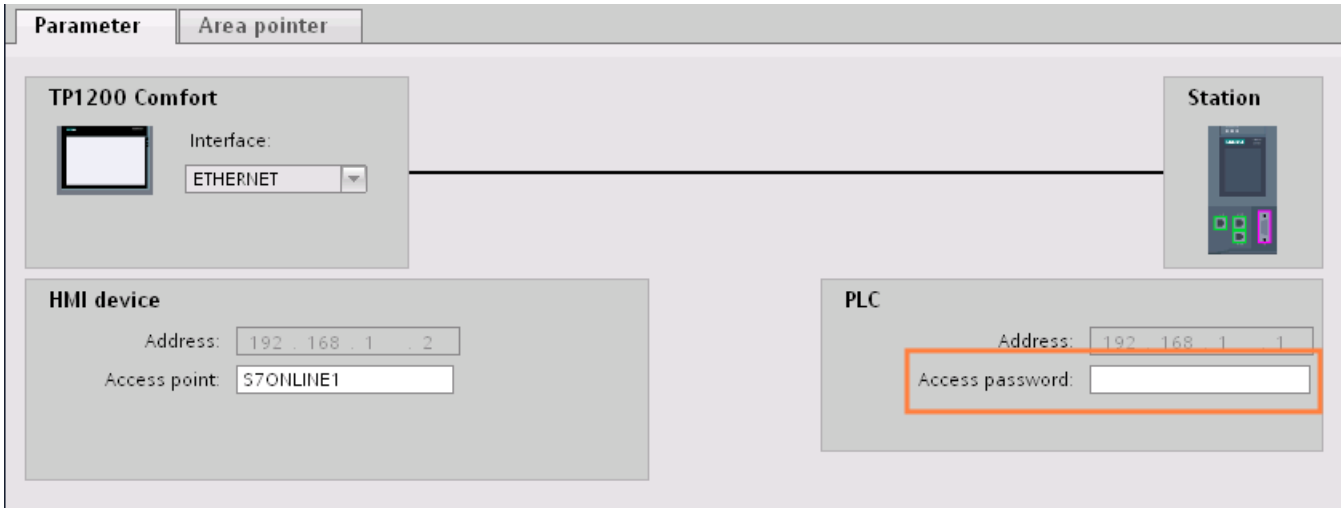
Communication with a PLC with the"Complete protection" protection level is protected by a password. The password is stored in the properties of the PLC.

Enter the password from the PLC in the "Access password" area.

Communication to the PLC is denied if an incorrect password or no password is entered.

Entering the access password

Enter the access password for the HMI connection in the "Connections" editor.



Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission rate / duplex

Depending on the selected device, you can make the following settings for "Transmission rate / duplex":

- Automatic setting
  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the peer port. The "Enable autonegotiation" option is also enabled as a default, in other words, you can use cross cables or patch cables for the connection.
- TP/ITP at x Mbps full duplex (half duplex)
  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:
  - Autonegotiation enabled
    You can use both cross cable and patch cable.
  - Autonegotiation disabled
    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.
- Deactivated
  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option enables or disables port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because with this option the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can accordingly not be optimally set.

Note

When a local port is connected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not accept the setting, an error message is generated.

Wiring rules for disabled autonegotiation

Requirements

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission rate
- Autonegotiation incl. autocrossing disabled

The time for negotiating the transmission rate during startup has been saved.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

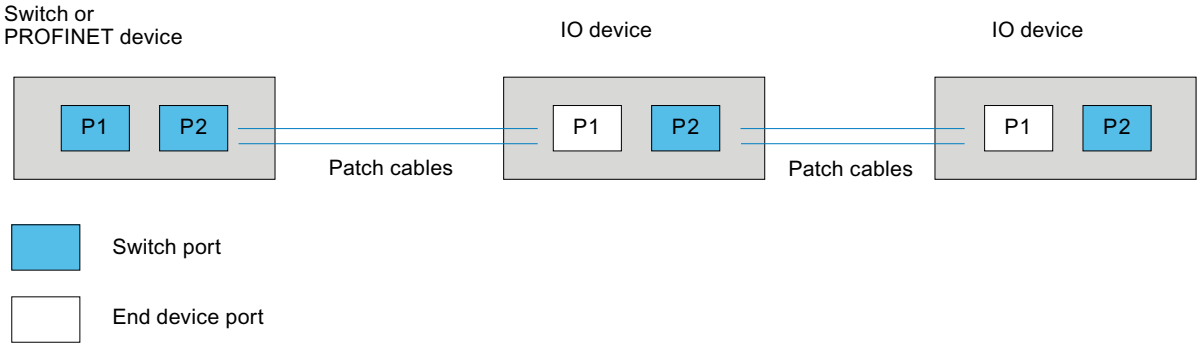
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in line using a single cable type (patch cable). To do this, you connect port 2 of the IO device (distributed I/O) with port 1 of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirements

To use boundaries, the respective device must have more than one port. If the PROFINET does not support boundary settings, they are not shown.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of discovery of accessible devices"
  No forwarding of DCP frames to identify accessible devices. Devices downstream from this port cannot be reached by the project tree under "Accessible devices". Devices downstream from this port cannot be reached by the CPU.
- "End of topology discovery"
  LLDP frames (Link Layer Discovery Protocol) are not forwarded for topology detection.
- "End of sync domain"
  No forwarding of sync frames transmitted to synchronize nodes within a sync domain.
  If you operate, for example, a PROFINET device with more than two ports in a ring, you should prevent the sync frame from being fed into the ring by setting a sync boundary (at the ports not inside the ring).
  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device from the other sync domain.

Restrictions

The following restrictions must be observed:

- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been determined for the port, the following check boxes cannot be used:
  - "End of discovery of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

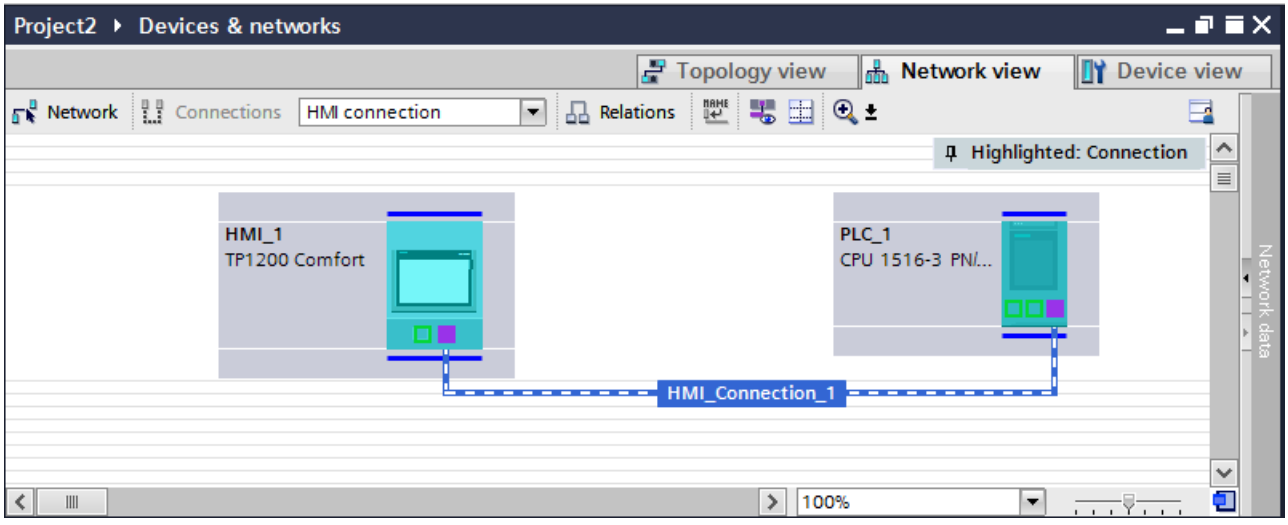
12.11.6.3 Communication via PROFIBUS

Configuring an HMI connection

Communication via PROFIBUS

HMI connections via PROFIBUS

If you have inserted an HMI device and a SIMATIC S7 1500 into the project, interconnect the two PROFIBUS interfaces in the "Devices & Networks" editor.



HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS

Introduction

You configure an HMI connection over PROFIBUS between HMI devices and a SIMATIC S7 1500 in the "Devices & Networks" editor.

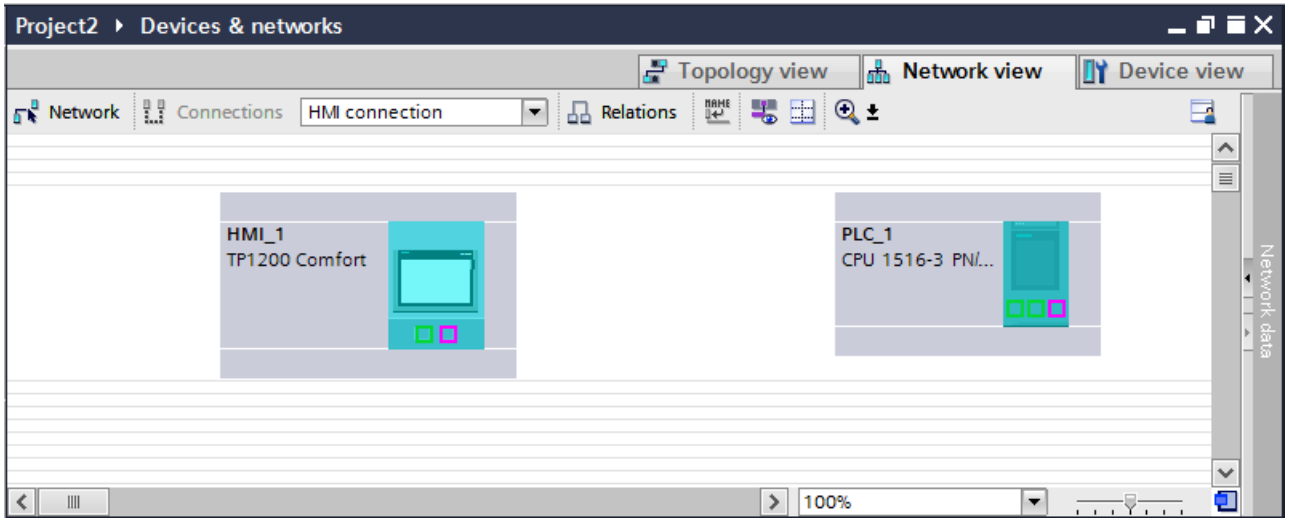
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- HMI device with MPI/DP interface
- SIMATIC S7 1500 with PROFIBUS interface

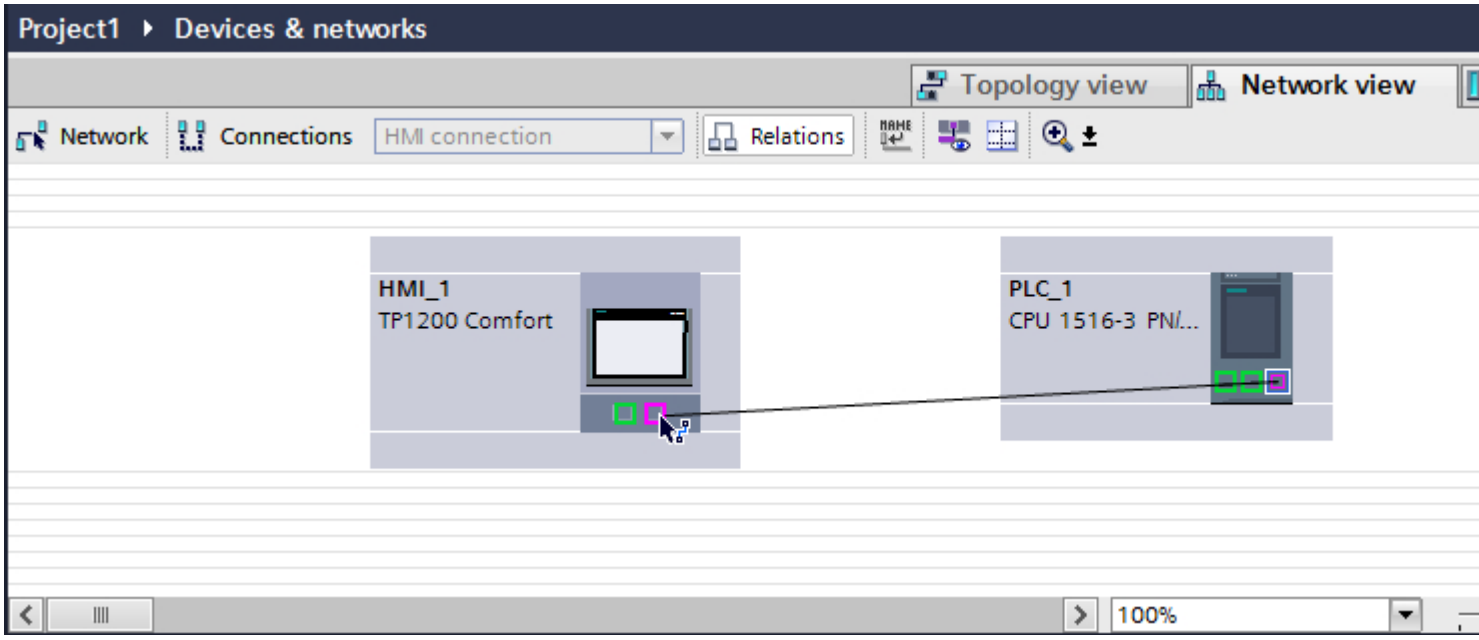
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.



2. Click the "Connections" button.
   The devices available for connection are highlighted in color.
3. Click the HMI device interface.
4. Select the "PROFIBUS" interface type in the Inspector window under "Properties > General > HMI MPIDP > Parameters".

5. Click the interface of the PLC and use a drag-and-drop operation to draw a connection to the HMI device.



6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFIBUS parameters (Page 6116)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection over PROFIBUS between an HMI device and a SIMATIC S7 1500.

Configuring an HMI connection

Communication via PROFIBUS

Communication via PROFIBUS

This chapter describes communication over PROFIBUS between a WinCC Runtime and the SIMATIC S7 1500 PLC.

You can use the following WinCC Runtimes as an HMI device:

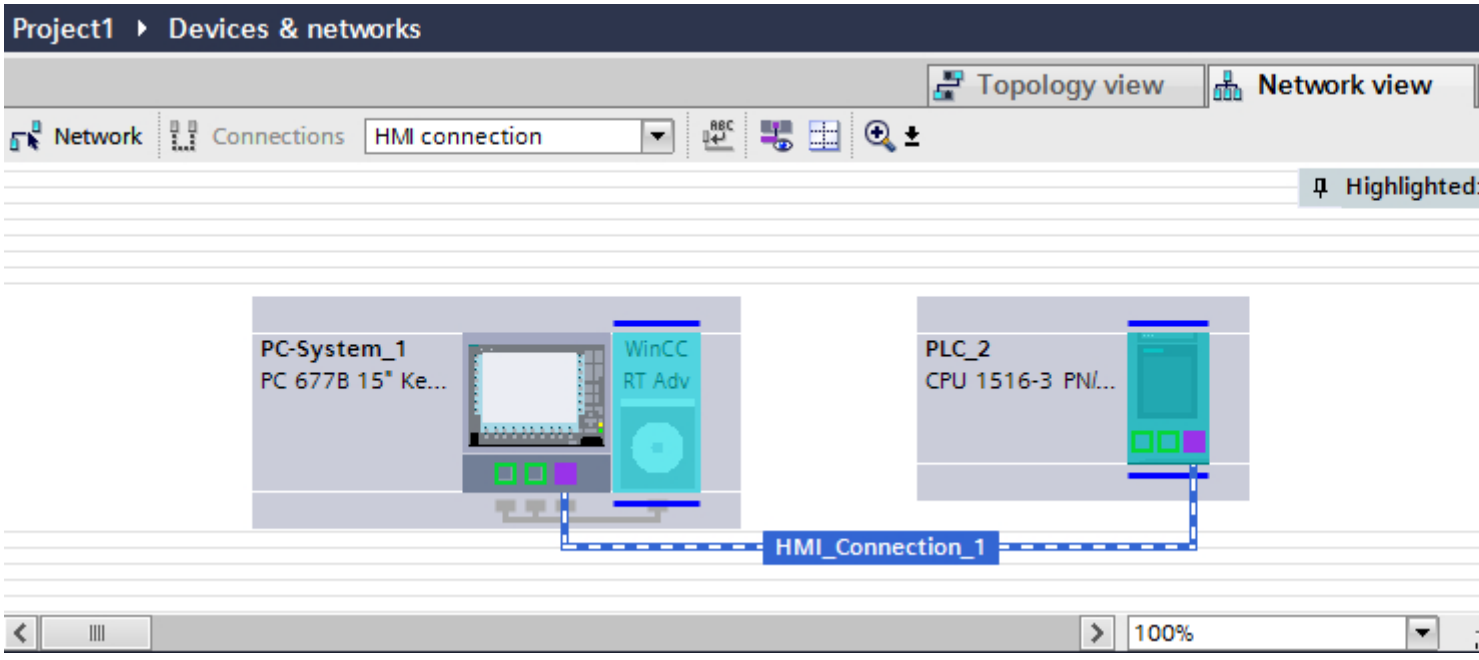
- WinCC RT Advanced
- WinCC RT Professional

WinCC Runtime as HMI device

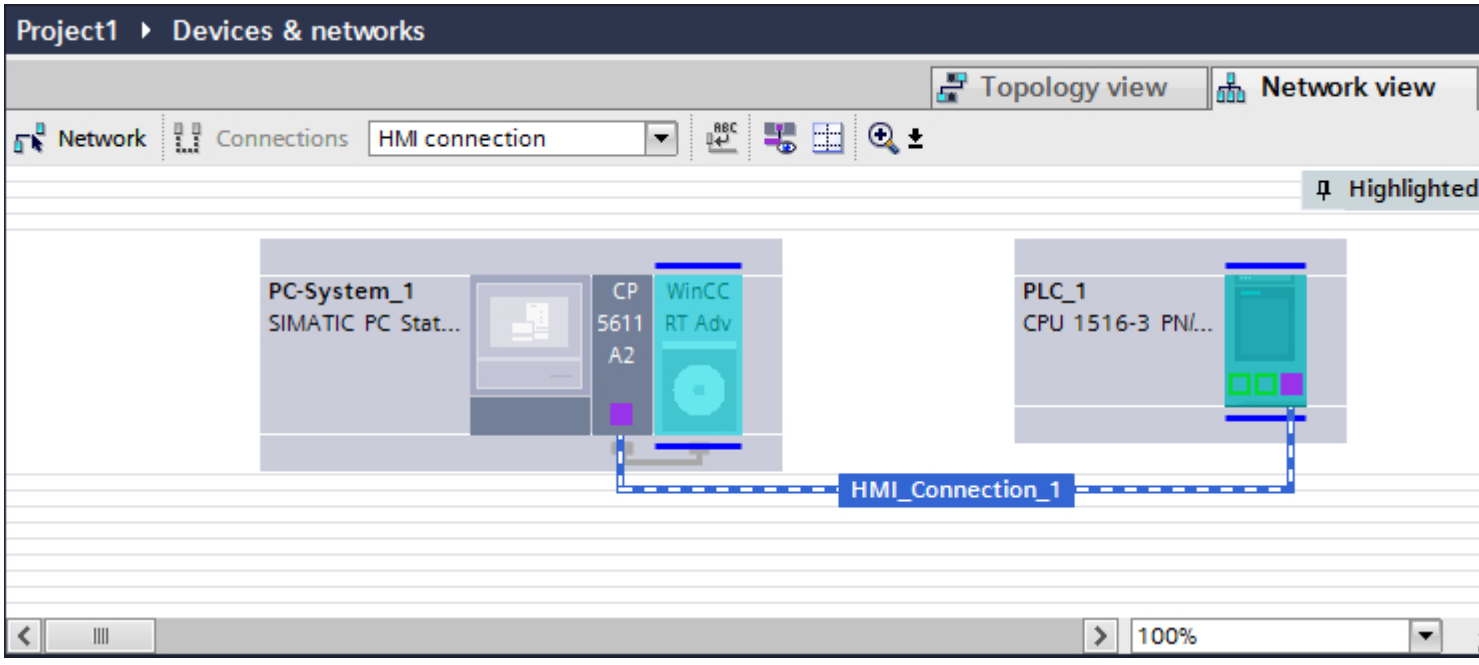
Configure the HMI connections between WinCC Runtime and SIMATIC S7 1500 in the "Devices & Networks" editor.

If you want to use a WinCC Runtime as an HMI device, you have the following options:

1. You create a SIMATIC PC and configure a WinCC Runtime on this SIMATIC PC.
   In this way you use the SIMATIC PC with a WinCC Runtime as an HMI device.



2. You create a WinCC Runtime and configure a communication processor to the Runtime.
   In this way you use your configuration PC with a WinCC Runtime as an HMI device.



You can also connect multiple HMI devices to a single SIMATIC S7 1500 and multiple SIMATIC S7 1500 to a single HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFIBUS in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS with a PC

Introduction

You configure an HMI connection over PROFIBUS between the HMI device and SIMATIC S7 1500 in the "Devices & Networks" editor.

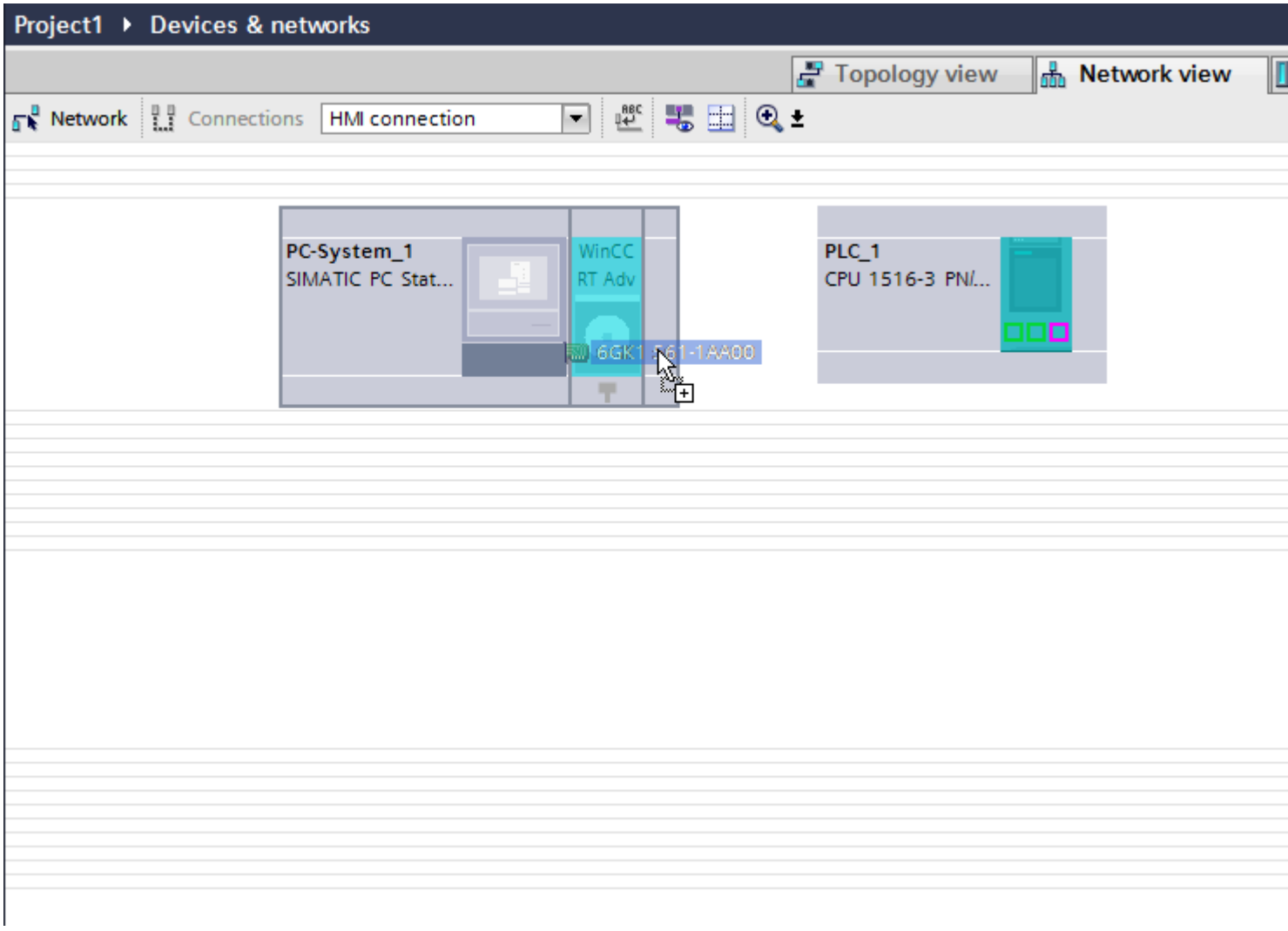
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 1500 with PROFIBUS interface
- PC station with WinCC RT Advanced or WinCC RT Professional

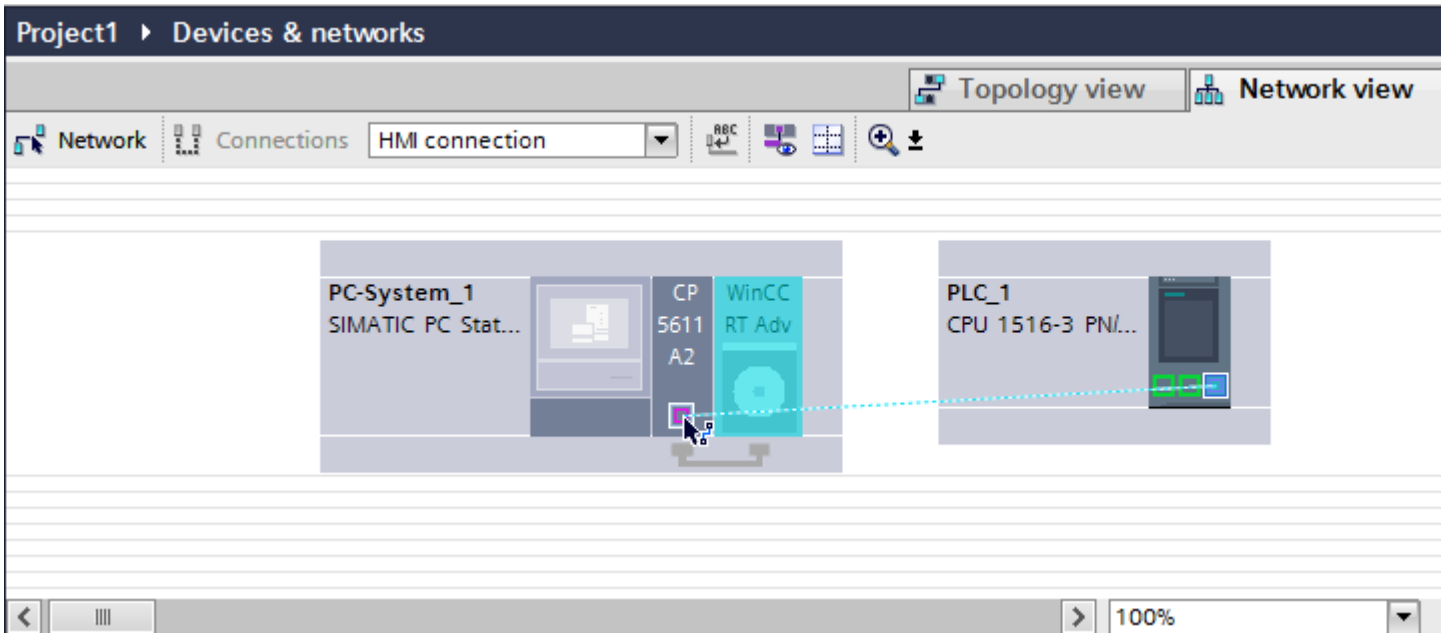
Procedure

1. Double-click the "Devices & Networks" item in the project navigation.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a PROFIBUS-capable communication processor from the hardware catalog to the WinCC Runtime.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click the PROFIBUS interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFIBUS interface of the communication processor.



5. Click the connecting line.
6. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
7. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFIBUS parameters (Page 6116)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection over PROFIBUS between an HMI device and a SIMATIC S7 1500.

Configuring an HMI connection via PROFIBUS with a SIMATIC PC

Introduction

You configure an HMI connection over PROFIBUS between HMI devices and a SIMATIC S7 1500 in the "Devices & Networks" editor.

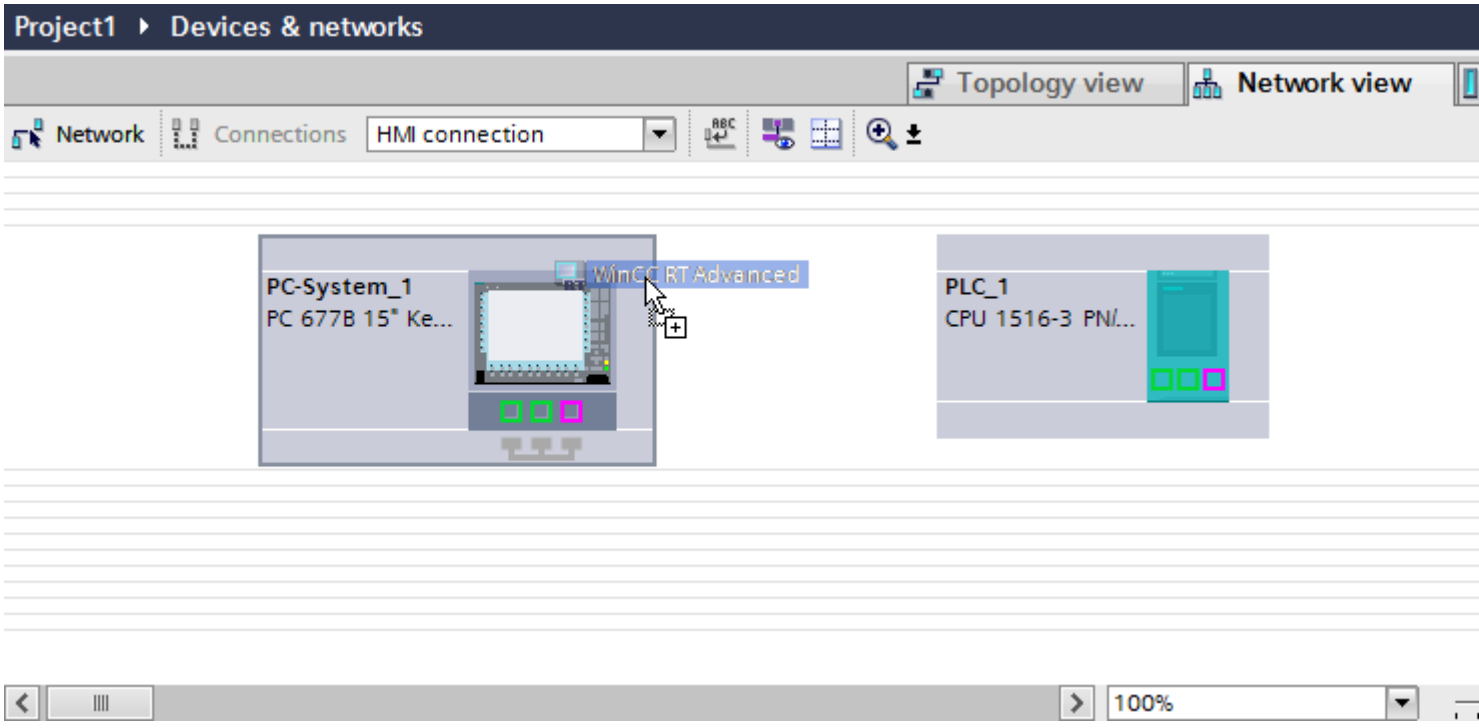
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 1500 with PROFIBUS interface
- SIMATIC PC with PROFIBUS interface

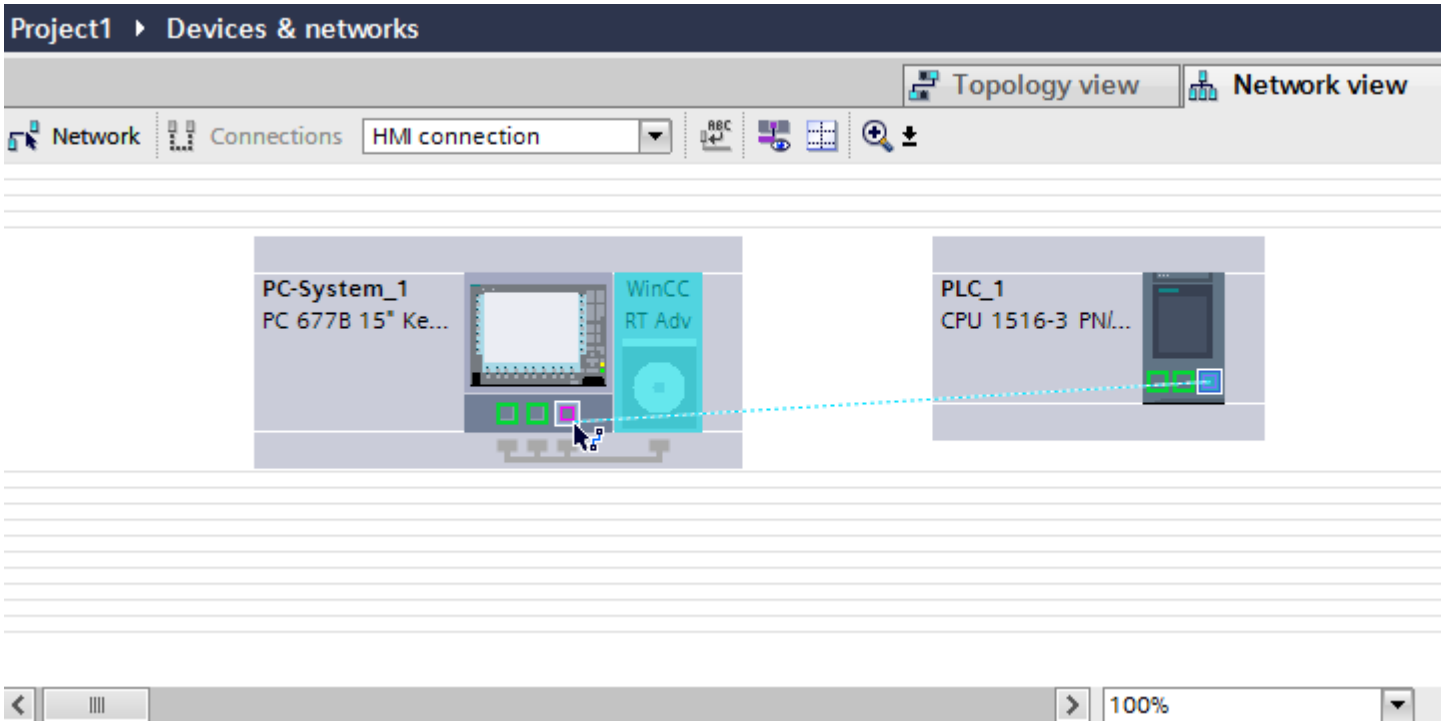
Procedure

1. Double-click the "Devices & Networks" item in the project navigation.
   The available communication partners in the project are displayed in the network view.
2. Click the MPI interface of the PC and select "PROFIBUS" for the interface type in the Inspector window.
3. Use a drag-and-drop operation to move a WinCC Runtime from the hardware catalog to the PC.



4. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

5. Click the PROFIBUS interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFIBUS interface of the PC.



6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFIBUS parameters (Page 6116)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection over PROFIBUS between an HMI device and a SIMATIC S7 1500.

PROFIBUS parameters

PROFIBUS parameters for the HMI connection

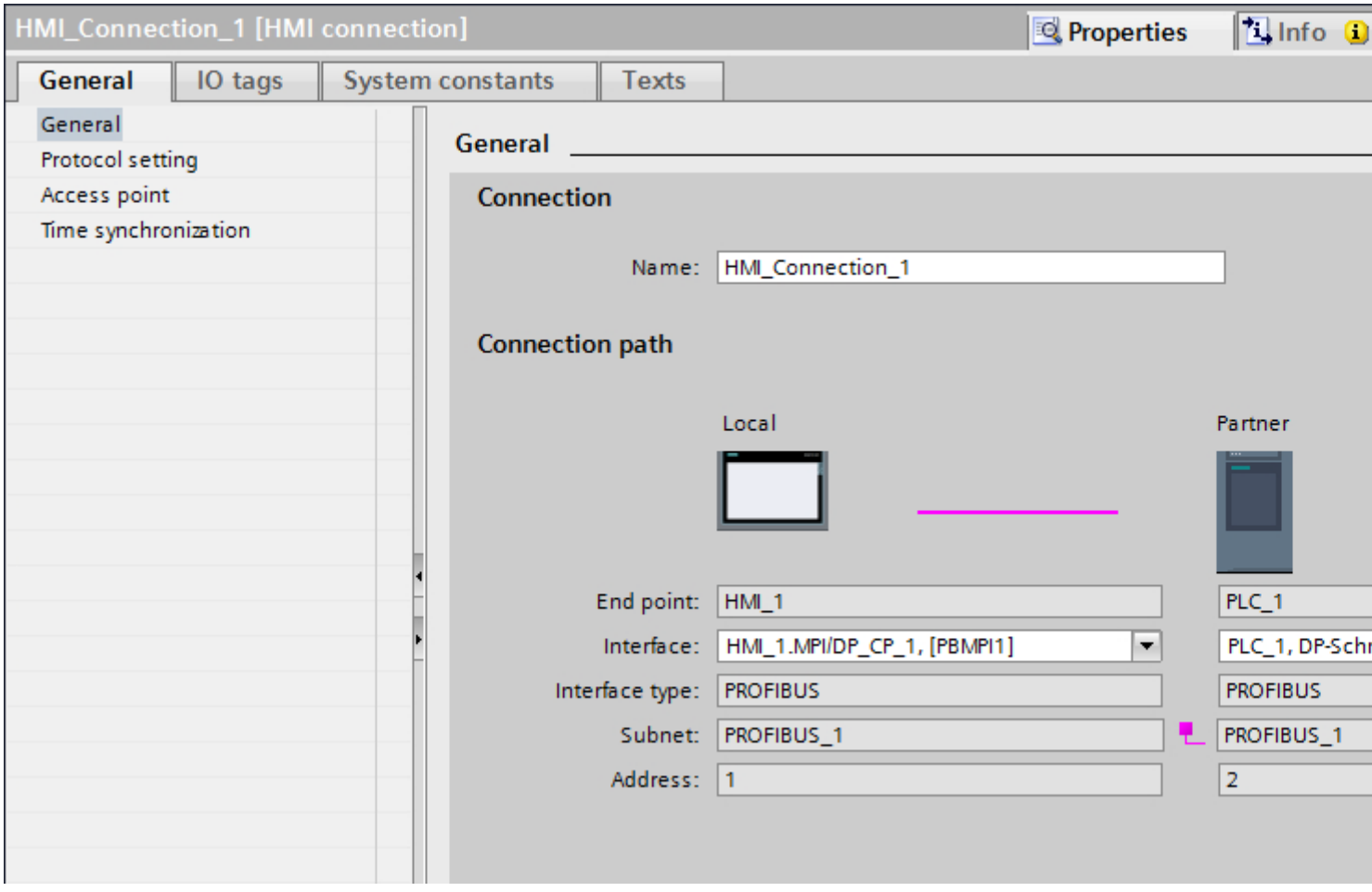
PROFIBUS parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Displays whether the devices are networked together.

- displayed if the devices are networked together.

- displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFIBUS parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area cannot be edited.
- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the PROFIBUS address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

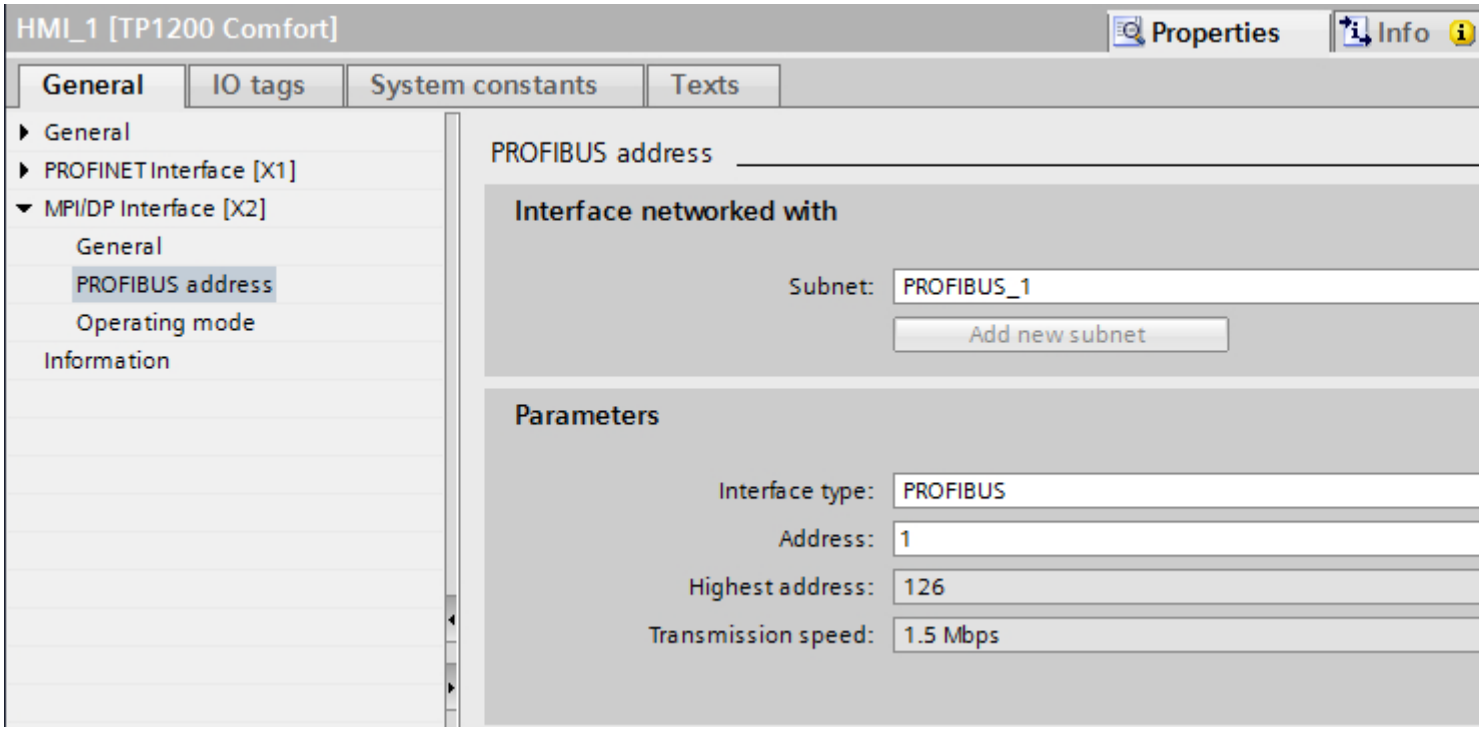
PROFIBUS parameters for the HMI device

PROFIBUS parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  You assign the interface type in the "Interface type" area. Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

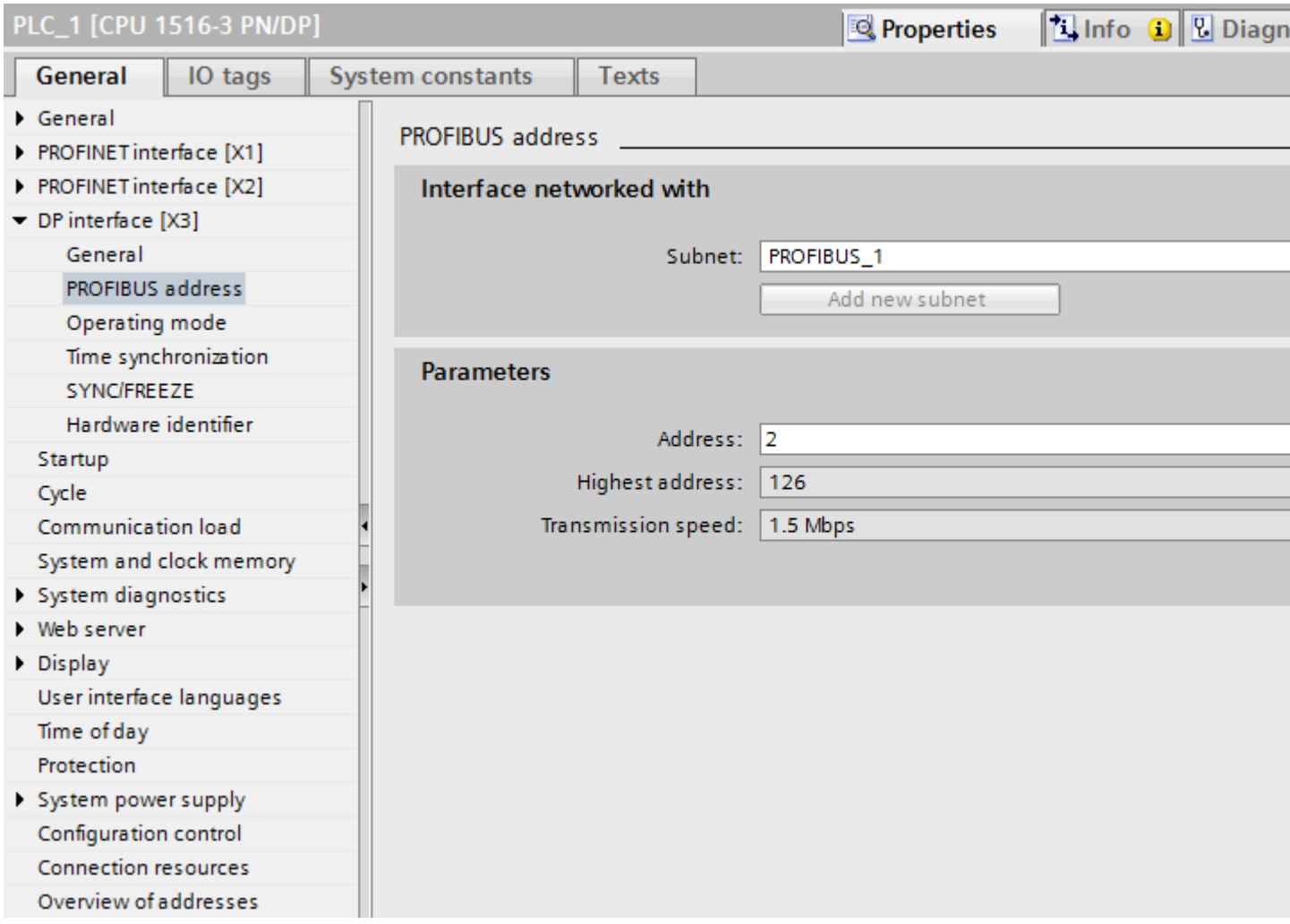
PROFIBUS parameters for the PLC

PROFIBUS parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Protection of communication

Security levels

If you want to protect the PLC and HMI device communication, you can assign protection levels for the communication.

For a SIMATIC S7-1500 CPU, you can enter multiple passwords and thereby set up different access rights for various user groups.

The passwords are entered in a table, so that exactly one protection level is assigned to each password.

The effect of the password is given in the "Protection" column.

For the SIMATIC S7-1200 controller, several aspects need to be considered when setting protection levels. For additional information on this, see: Auto-Hotspot

Example

You select the "Complete protection" protection level for a standard CPU (i.e., not an F-CPU) when configuring it.

Afterwards, you enter a separate password for every protection level above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the assigned passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (no protection) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (write protection) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (read and write protection) allows access as if the CPU were read-protected and write-protected, so that only HMI access is possible for users who know this password.

Access password for the HMI connection

Introduction

Secure access to a PLC by assigning a password.

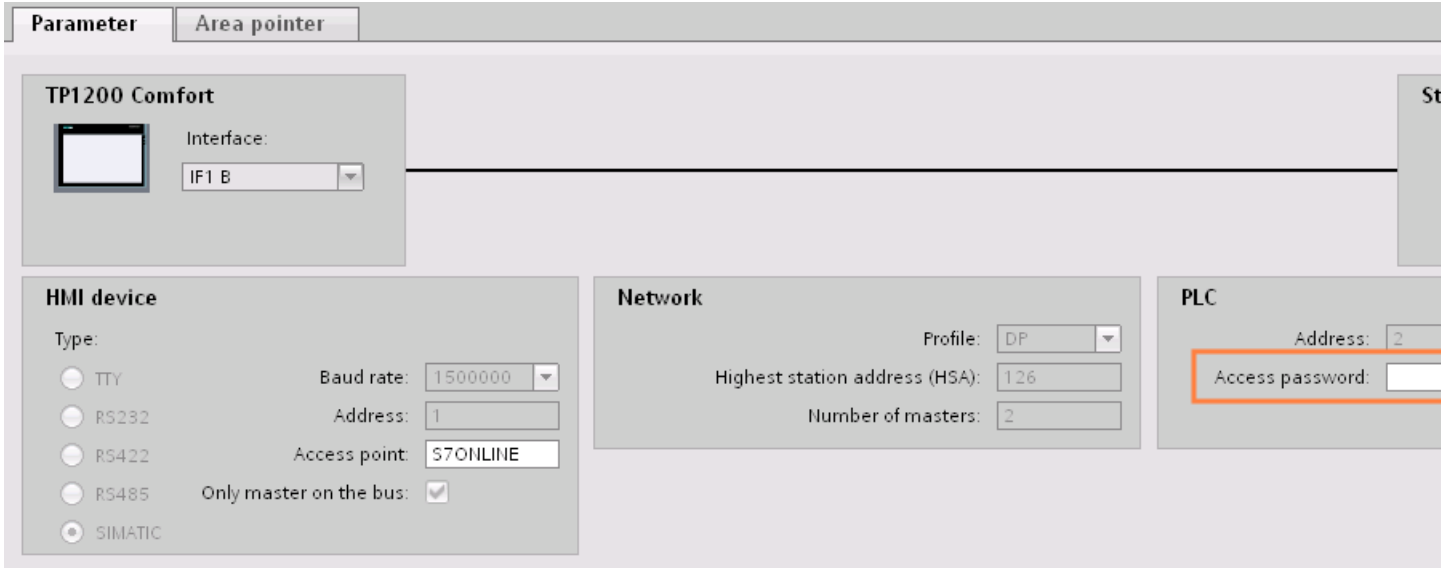
Assign the password when configuring the connection.

Input of the password of the PLC is mandatory as of "Complete protection" security level.

Communication to the PLC is denied if an incorrect password or no password is entered.

Assigning password

Enter the "Access password" for the PLC in the "Connections" editor.



12.11.6.4 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Data exchange using area pointers (Page 5999)

Restrictions

You can only configure the following data types for communication with SIMATIC S7 1500 for data exchange using area pointers:

- UInt and array of UInt
- Word and array of Word
- Int and array of Int
- "Array[0..15] of Bool" for area pointer "Coordination"
- Date_And_Time
- DTL and LDT

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Use

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1st word	Current screen type															
2nd word	Current screen number															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3rd word	Reserved															
4th word	Current field number															
5th word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" or "40" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Using data types

The data types "Date_And_Time, DTL" and "LDT" can only be used with the "Date/time" and "Date/time PLC" area pointers.

The data format of the "Date/time" area pointer depends on job mailbox 40/41.

If there are no control tags linked to the area pointer, or a control tag is linked with the data type "Array[0..5] of UInt/Word/Int", the following applies:

The configuration of the "Date/time" area pointer is only used for job mailbox 41.

If job mailbox 40 is used, the data format "DATE_AND_TIME (BCD-encoded)" is used (shown in the next section).

If the "Date/time" and "Date/time PLC" area pointers are linked to a control tag with the data type "DATE_AND_TIME", "DTL" or "LDT", the associated data format is used in the corresponding area pointer.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the Date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Using data types

The data types "Date_And_Time, DTL" and "LDT" can only be used with the "Date/time" and "Date/time PLC" area pointers.

The data format of the "Date/time" area pointer depends on job mailbox 40/41.

If there are no control tags linked to the area pointer, or a control tag is linked with the data type "Array[0..5] of UInt/Word/Int", the following applies:

The configuration of the "Date/time" area pointer is only used for job mailbox 41.

If job mailbox 40 is used, the data format "DATE_AND_TIME (BCD-encoded)" is used (shown in the next section).

If the "Date/time" and "Date/time PLC" area pointers are linked to a control tag with the data type "DATE_AND_TIME", "DTL" or "LDT", the associated data format is used in the corresponding area pointer.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

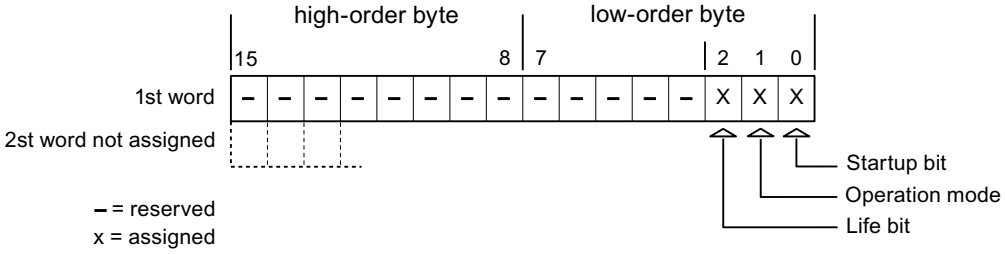
By default, the "Coordination" area pointer has the length of one word and cannot be changed.

Usage

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of the bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

"Project ID" area pointer

Function

When Runtime starts, a check can be carried out as to whether the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program. If there is no concordance, a system event is given on the HMI device and Runtime is stopped.

Use

Note

HMI connections cannot be switched "online".

The HMI connection in which the "Project ID" area pointer is used must be switched "online".

To use this area pointer, set up the following during the configuration:

- Define the version of the configuration. Values between 1 and 255 are possible.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:
  You enter the data address in the editor "Communication > Connections" under "Address".

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

"Job mailbox" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	**Setting the date (BCD coded)** [3) 4)]	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	**User logon**	

No.	Function	
14	**Set time (BCD-coded)**	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	**User logoff**	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	**Transfer date/time to PLC**	
	(in the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	**Transfer date/time to PLC**	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	**Update tag**	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in Parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	**Screen selection** [2]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	**Reading data record from PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	**Writing data record to PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

1)	Only for devices supporting recipes.
2)	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
3)	The weekday is ignored on HMI device KTP 600 BASIC PN.
4)	The weekday is ignored when you configure the "Date/Time PLC" area pointer.

"Data record" area pointer

"Data record" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization via the data mailbox

Data records are always transferred directly, which means that the tag values are read straight from an address or written straight to an address configured for this tag without being redirected via an interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system alarm.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then, for example, process, edit, or save these values in the HMI device.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data mailbox free
2	0000 0010	Transferring
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxs No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	

	Left byte (LB)	Right byte (RB)
Word 3	Data record number (1 to 65535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the function from the data medium and writes the values to the PLC.	

Step	Action	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible
- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system alarms
- Triggered by function
  Output of system alarms
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data record.

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Displaying tags with Runtime Advanced and Panels (Page 4270)

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

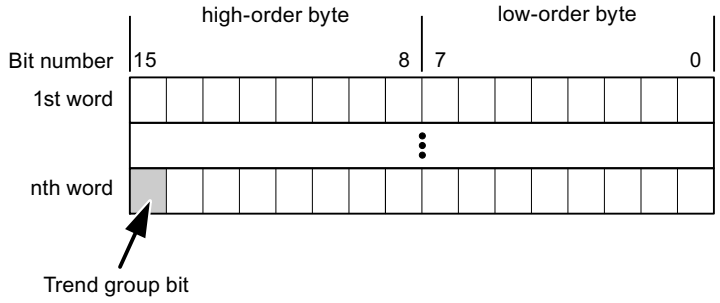
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Permitted data types for trends

For SIMATIC S7

You assign one bit to each trend during configuration. Tags and array tags of the "Word" or "Int" data type are permitted.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Configuring Alarms (Page 4295)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

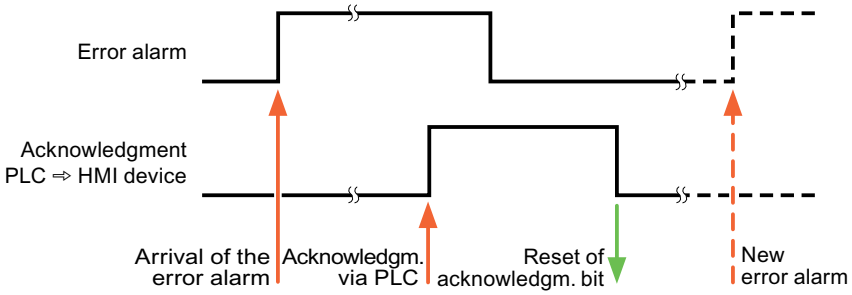
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

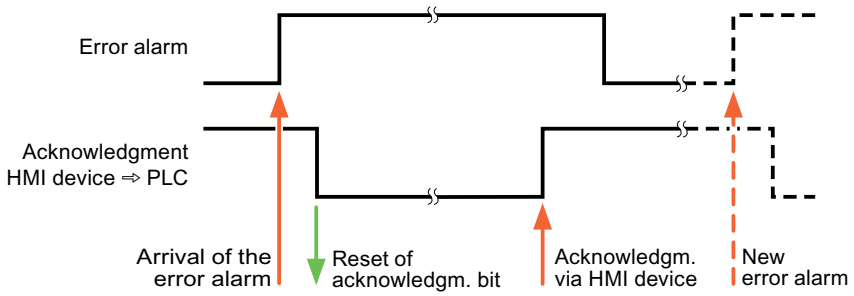
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



LED mapping

Function

Keyboard devices have LEDs in the function keys. These LEDs can be activated from the PLC. Thus, it is possible, for example, to signal to the user which key he should press in a given situation by lighting up an LED.

Note

The LED function cannot be configured for Basic Panels.

Requirements

In order to activate an LED, an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration.

LED assignment

The assignment of the individual LEDs to the bits in the LED tags is specified when the function keys are configured. In this process, the "LED tag" and the assigned "Bit" are specified for each function key in the "General" group of the properties window.

The "Bit" bit number designates the first of two consecutive bits that control the following LED statuses.

Bit n+ 1	Bit n	LED function	
		All Mobile Panels, all Operator Panels, all Multi Panels, all Comfort Panels	Panel PCs
0	0	Off	Off
0	1	Rapid flashing	Flashing
1	0	Slow flashing	Flashing
1	1	On permanently	On permanently

12.11.6.5 Performance features of communication

S7-1500 device dependency

Device dependency

If you use devices from an earlier version of the TIA Portal with TIA Portal V13, it may not be possible to configure connections to certain HMI devices.

Basic Panels V11.0

HMI devices	SIMATIC S7-1500
KP300 Basic	No
KP400 Basic	No
KTP400 Basic PN	No
KTP600 Basic DP	No
KTP600 Basic PN	No
KTP1000 Basic DP	No
KTP1000 Basic PN	No
TP1500 Basic PN	No

Basic Panels V12.0

HMI devices	SIMATIC S7-1500
KP300 Basic	Yes
KP400 Basic	Yes
KTP400 Basic PN	Yes
KTP600 Basic DP	Yes
KTP600 Basic PN	Yes
KTP1000 Basic DP	Yes
KTP1000 Basic PN	Yes
TP1500 Basic PN	Yes

Basic Panels V13.0

HMI devices	SIMATIC S7-1500
KTP400 Basic	Yes
KTP700 Basic	Yes
KTP900 Basic	Yes
KTP1200 Basic	Yes

Basic Panels V13.0.1

HMI devices	SIMATIC S7-1500
KTP400 Basic	Yes
KTP700 Basic	Yes
KTP900 Basic	Yes
KTP1200 Basic	Yes

Panels V11.0

HMI devices	SIMATIC S7-1500
OP 73	No
OP 77A	No
OP 77B	No
TP 177A	No
TP 177A Portrait	No
TP 177B 4"	No
TP 177B 6" mono	No
TP 177B 6"	No
OP 177B 6" mono	No
OP 177B 6"	No
TP 277 6"	No
OP 277 6"	No

Panels V12.0

HMI devices	SIMATIC S7-1500
OP 73	No
OP 77A	No
OP 77B	No
TP 177A	No
TP 177A Portrait	No
TP 177B 4"	No
TP 177B 6" mono	No
TP 177B 6"	No
OP 177B 6" mono	No
OP 177B 6"	No
TP 277 6"	No
OP 277 6"	No

Multi Panels V11.0

HMI devices	SIMATIC S7-1500
MP 177 6" Touch	No
MP 277 8" Key	No
MP 277 10" Key	No
MP 277 10" Touch	No
MP 377 12" Key	No
MP 377 12" Touch	No
MP 377 15" Touch	No
MP 377 19" Touch	No

Multi Panels V12.0

HMI devices	SIMATIC S7-1500
MP 177 6" Touch	Yes
MP 277 8" Key	Yes
MP 277 10" Key	Yes
MP 277 10" Touch	Yes
MP 377 12" Key	Yes
MP 377 12" Touch	Yes
MP 377 15" Touch	Yes
MP 377 19" Touch	Yes

Mobile Panels V11.0

HMI devices	SIMATIC S7-1500
Mobile Panel 177 6" DP	No
Mobile Panel 177 6" PN	No
Mobile Panel 277 8"	No
Mobile Panel 277 8" IWLAN V2	No
Mobile Panel 277F 8" IWLAN V2	No
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	No
Mobile Panel 277 10"	No

Mobile Panels V12.0

HMI devices	SIMATIC S7-1500
Mobile Panel 177 6" DP	Yes
Mobile Panel 177 6" PN	Yes
Mobile Panel 277 8"	Yes
Mobile Panel 277 8" IWLAN V2	Yes

HMI devices	SIMATIC S7-1500
Mobile Panel 277F 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	Yes
Mobile Panel 277 10"	Yes

Mobile Panels V13.0.1

HMI devices	SIMATIC ET 200 CPU
KTP 700 Mobile	Yes
KTP 900 Mobile	Yes

Comfort Panels V11.0

HMI devices	SIMATIC S7-1500
KP400 Comfort	No
KTP400 Comfort	No
KTP400 Comfort Portrait	No
KP700 Comfort	No
TP700 Comfort	No
TP700 Comfort Portrait	No
KP900 Comfort	No
TP900 Comfort	No
TP900 Comfort Portrait	No
KP1200 Comfort	No
TP1200 Comfort	No
TP1200 Comfort Portrait	No
KP1500 Comfort	No
TP1500 Comfort	No
TP1500 Comfort Portrait	No
TP1900 Comfort	No
TP1900 Comfort Portrait	No
TP2200 Comfort	No
TP2200 Comfort Portrait	No

Comfort Panels V12.0

HMI devices	SIMATIC S7-1500
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes

HMI devices	SIMATIC S7-1500
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V13.0

HMI devices	SIMATIC S7-1500
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V13.0.1

HMI devices	SIMATIC S7-1500
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Runtime V11.0

HMI devices	SIMATIC S7-1500
WinCC RT Advanced	No

Runtime V12.0

HMI devices	SIMATIC S7-1500
WinCC RT Advanced	Yes

Runtime V13.0

HMI devices	SIMATIC S7-1500
WinCC RT Advanced	Yes

Runtime V13.0.1

HMI devices	SIMATIC S7-1500
WinCC RT Advanced	Yes

Valid data types for SIMATIC S7 1500

Valid data types for connections with SIMATIC S7 1500

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length	
BOOL	1 bit	
BYTE	1 byte	
WORD	2 bytes	
DWORD	4 bytes	
CHAR	1 byte	
WCHAR	2 bytes	RT Professional
ARRAY of WCHAR	--	
INT	2 bytes	
DINT	4 bytes	
REAL	4 bytes	
TIME	4 bytes	
DATE	2 bytes	
TIME_OF_DAY	4 bytes	
S5TIME	2 bytes	
COUNTER	2 bytes	
TIMER	2 bytes	
DATE_AND_TIME	8 bytes	
STRING	(2+n) bytes, n = 0 to 254	
WSTRING	(4+2*n) bytes, n = 0 to 254	Basic Panels
	(4+2*n) bytes, n = 0 to 4094	Panels, RT Advanced
	(4+2*n) bytes, n = 0 to 65534	RT Professional
DTL	12 bytes	
LDT	8 bytes	
LINT	8 bytes	
LREAL	8 bytes	
LTIME	8 bytes	
LTIME_OF_DAY	8 bytes	
SINT	1 byte	
UDINT	4 bytes	
UINT	2 bytes	
ULINT	8 bytes	
USINT	1 byte	

12.11.6.6 Configuring connections in the "Connections" editor

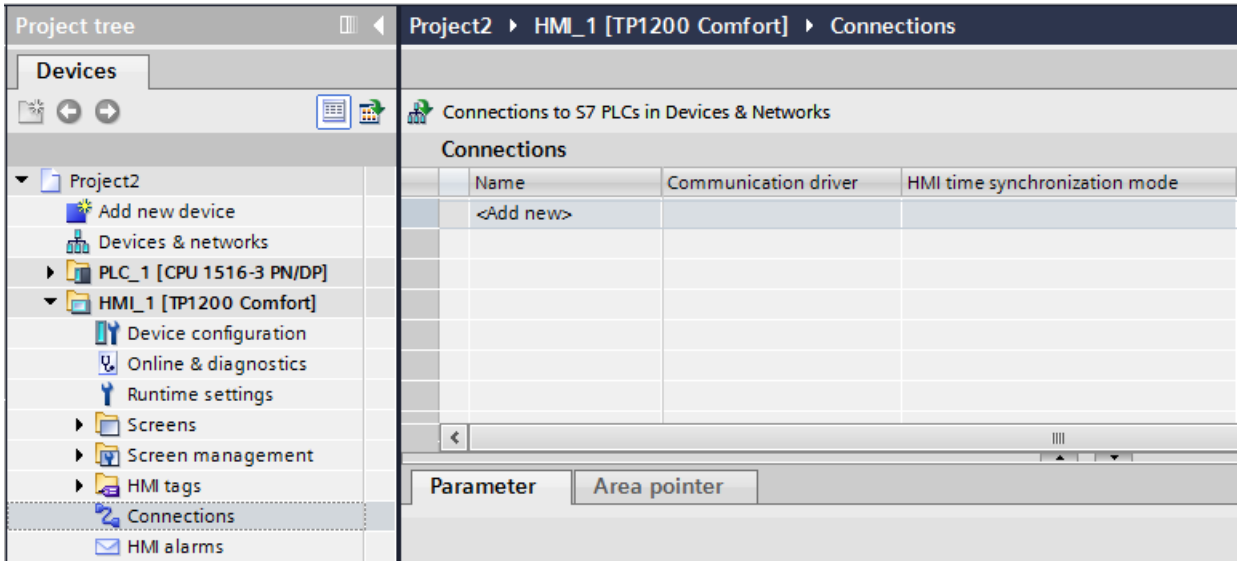
Creating a PROFINET connection

Requirements

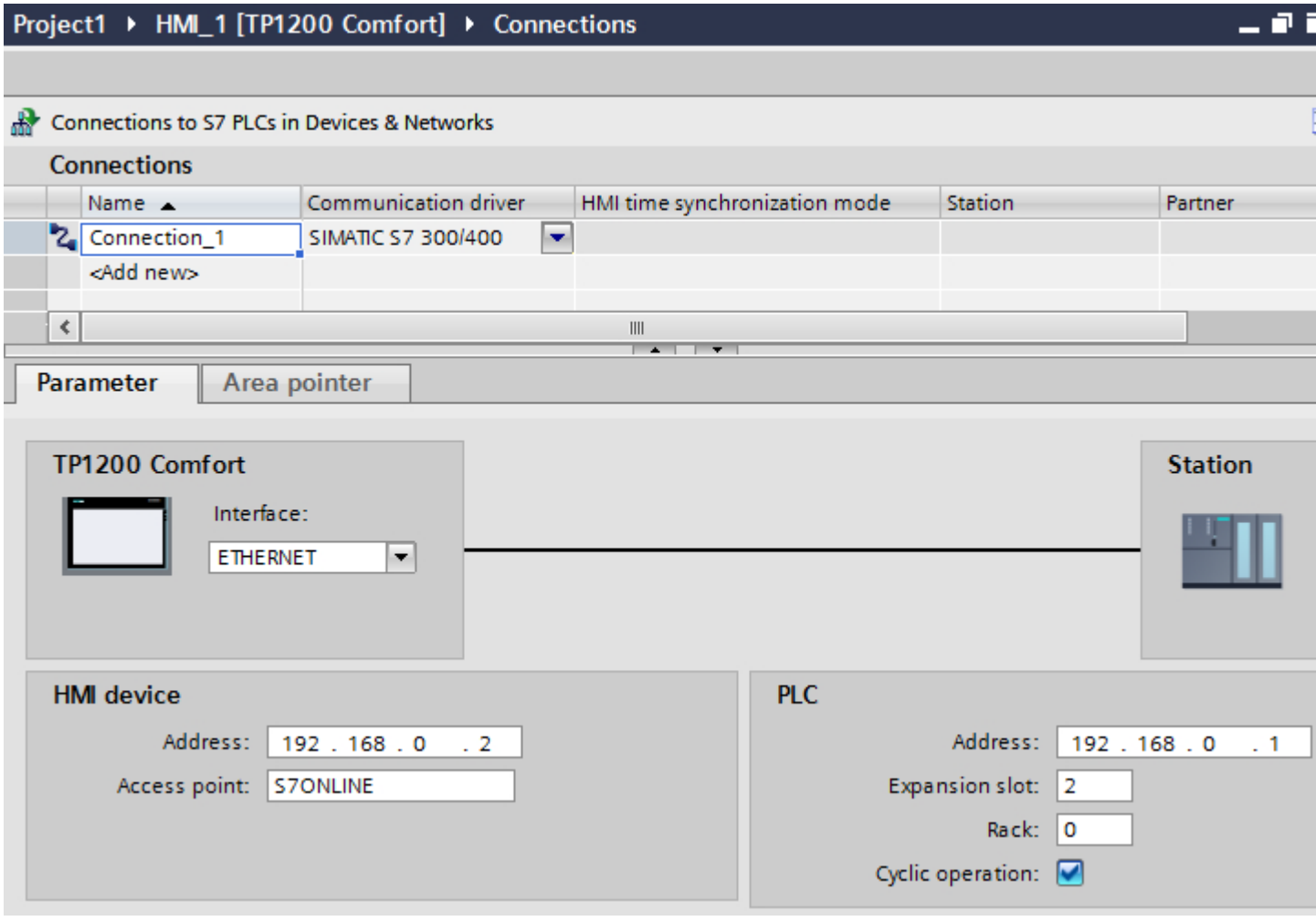
- A project is open.
- An HMI device with a PROFINET interface has been created.

Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.



4. Click the name of the connection.
5. Select a PROFINET interface of the HMI device in the Inspector window under "Parameters > Interface".
6. Set the IP addresses of the communication partners in the Inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

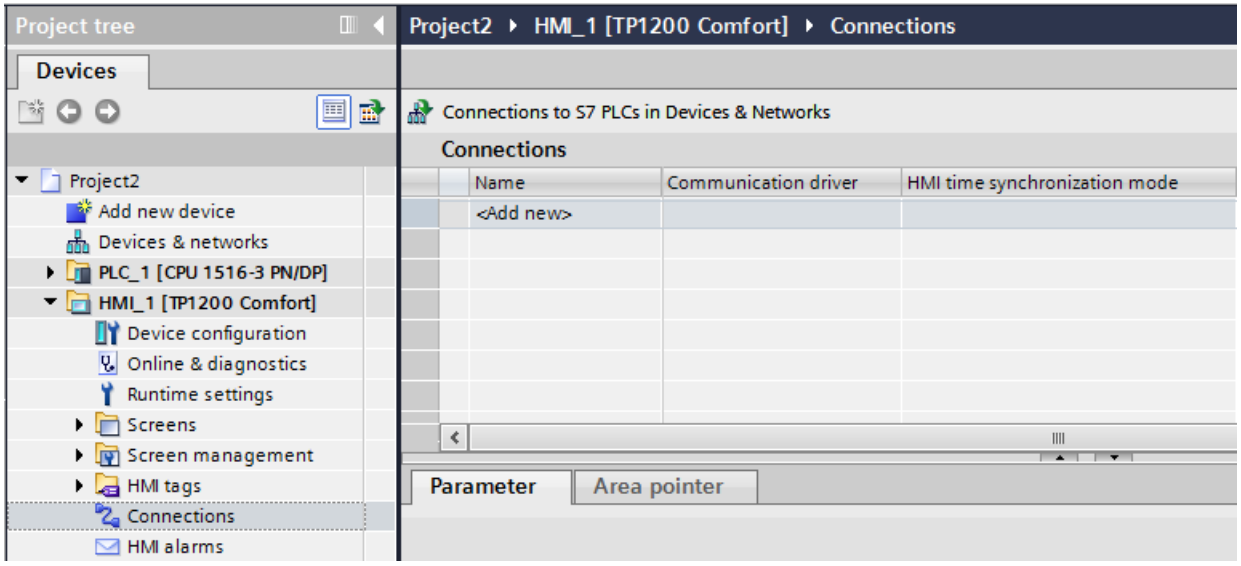
Creating a PROFIBUS connection

Requirements

- A project is open.
- An HMI device with a PROFIBUS interface has been created.

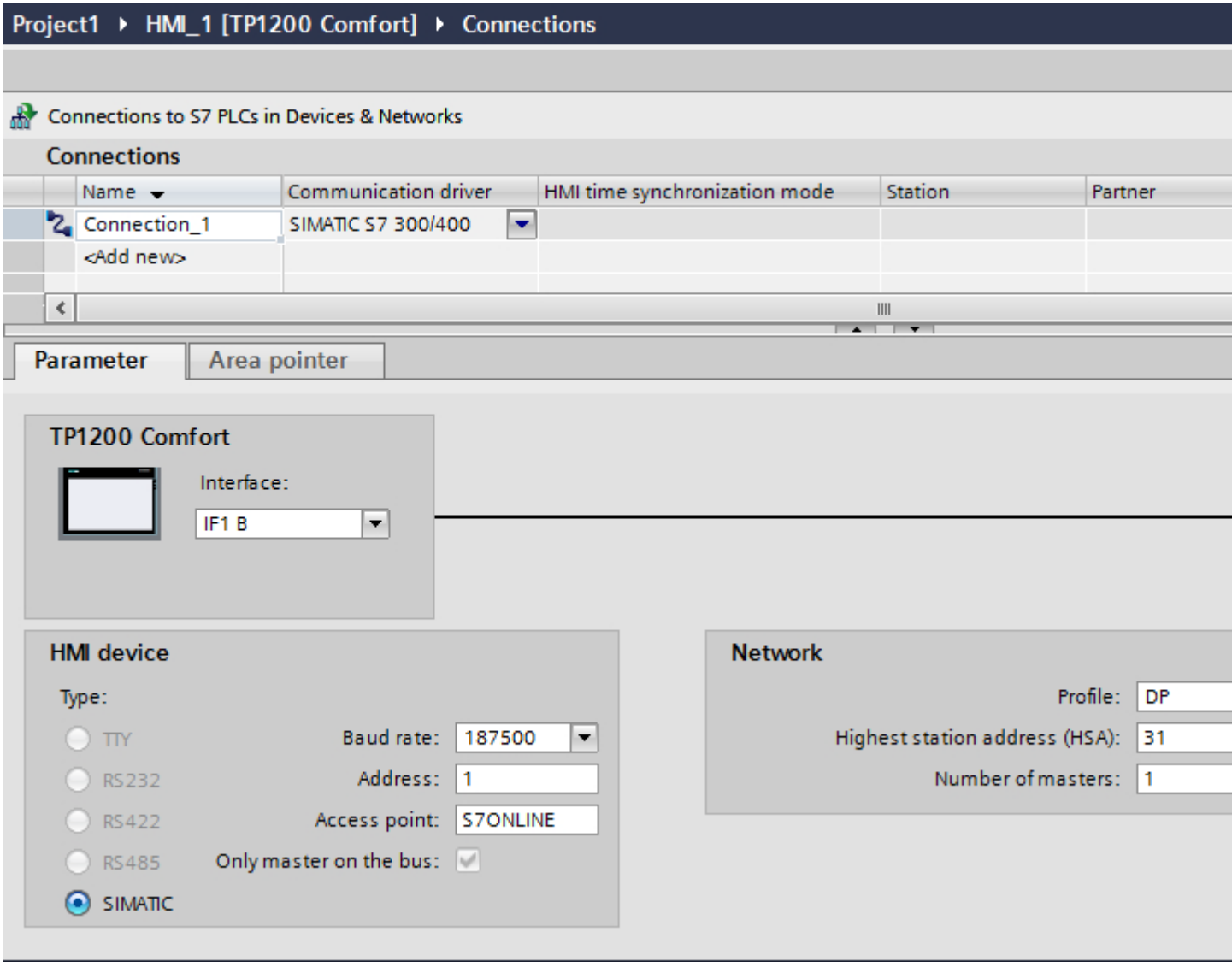
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select the "IF 1 B" interface from "Parameters > interface" in the Inspector window.

6. Select the "DP" profile in the Inspector window under "Parameters > Network".



7. Set the addresses of the communication partners in the inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

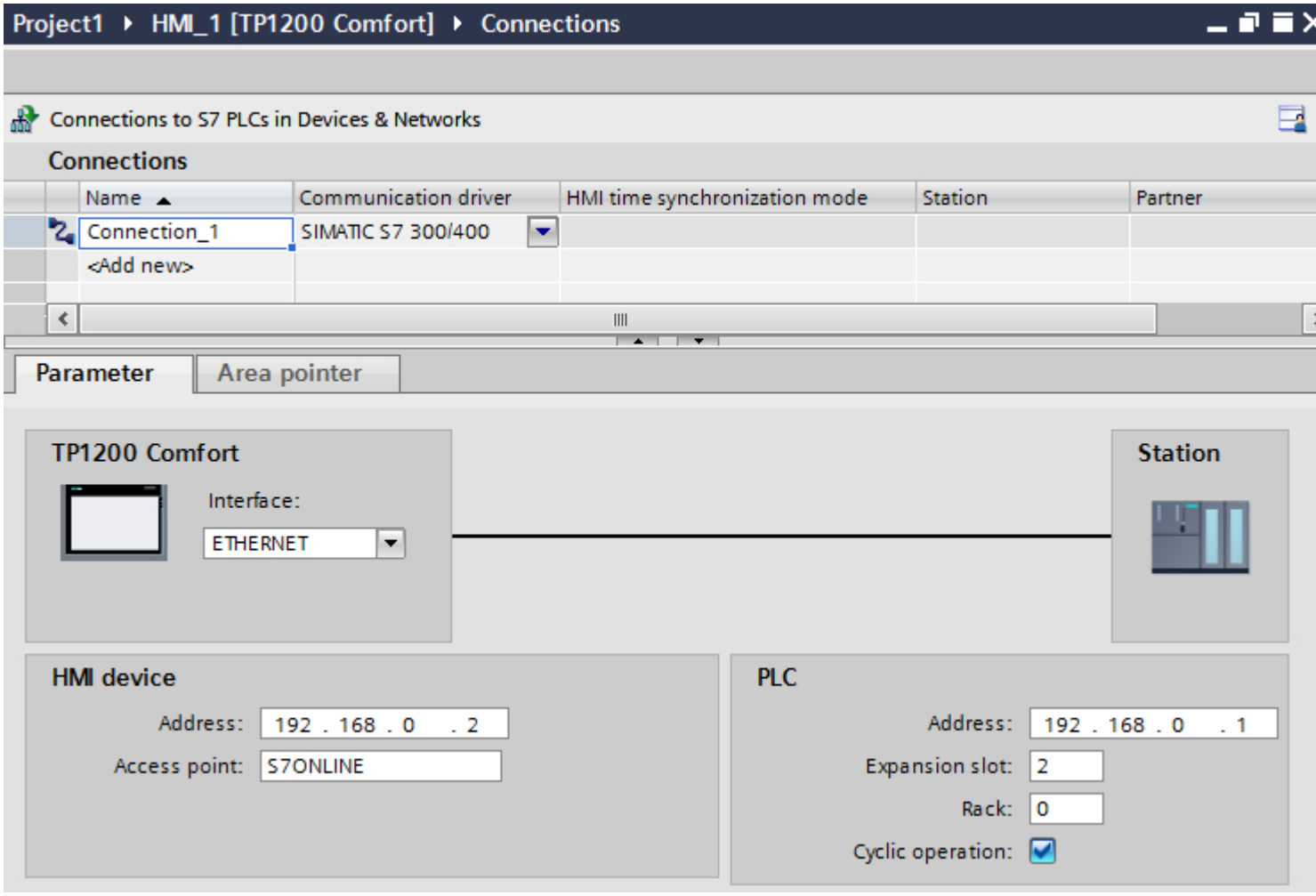
Connection parameters

Connection parameters

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.

- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.
- "Only master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).
  In S7-200, you must set an HMI device as the master.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"
  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.

- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the PLC.

  Note

  You only need a password if you have set "Complete protection" at the PLC.

  No connection is set up to the PLC if the "Complete protection" security level is stored on the PLC and you do not enter a password.

Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"

- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

- "Access point"
  Specifies the access point for the PG/PC interface that can be used to reach the communication partner.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the PLC.

Note

You only need a password if you have set "Complete protection" at the PLC.

No connection is set up to the PLC if the "Complete protection" security level is stored on the PLC and you do not enter a password.

12.11.6.7 Configuring connections in the "Connections" editor

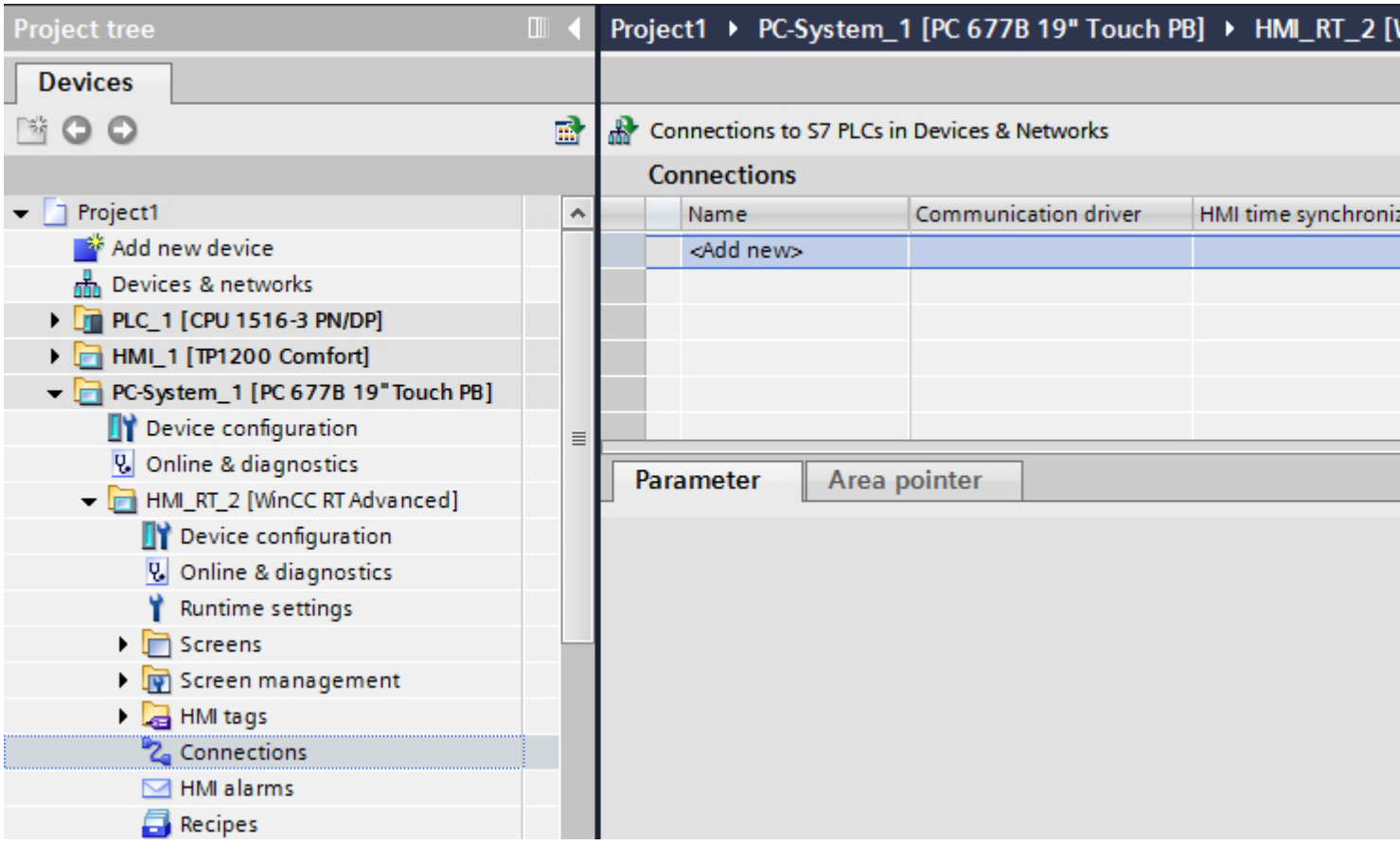
Creating a connection

Requirements

- A project is open.
- WinCC RT Advanced is created.

Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select an interface of the HMI device in the Inspector window under "Parameters > Interface".
6. Set the parameters for the connection in the Inspector window.

Interfaces

Select one of the following interfaces in the Inspector window under "Parameters > WinCC RT Advanced > Interfaces".:

- Industrial Ethernet
- MPI
- PROFIBUS

For additional information on the parameters of the interfaces see:

Connection parameters (Page 6157)

Connection parameters

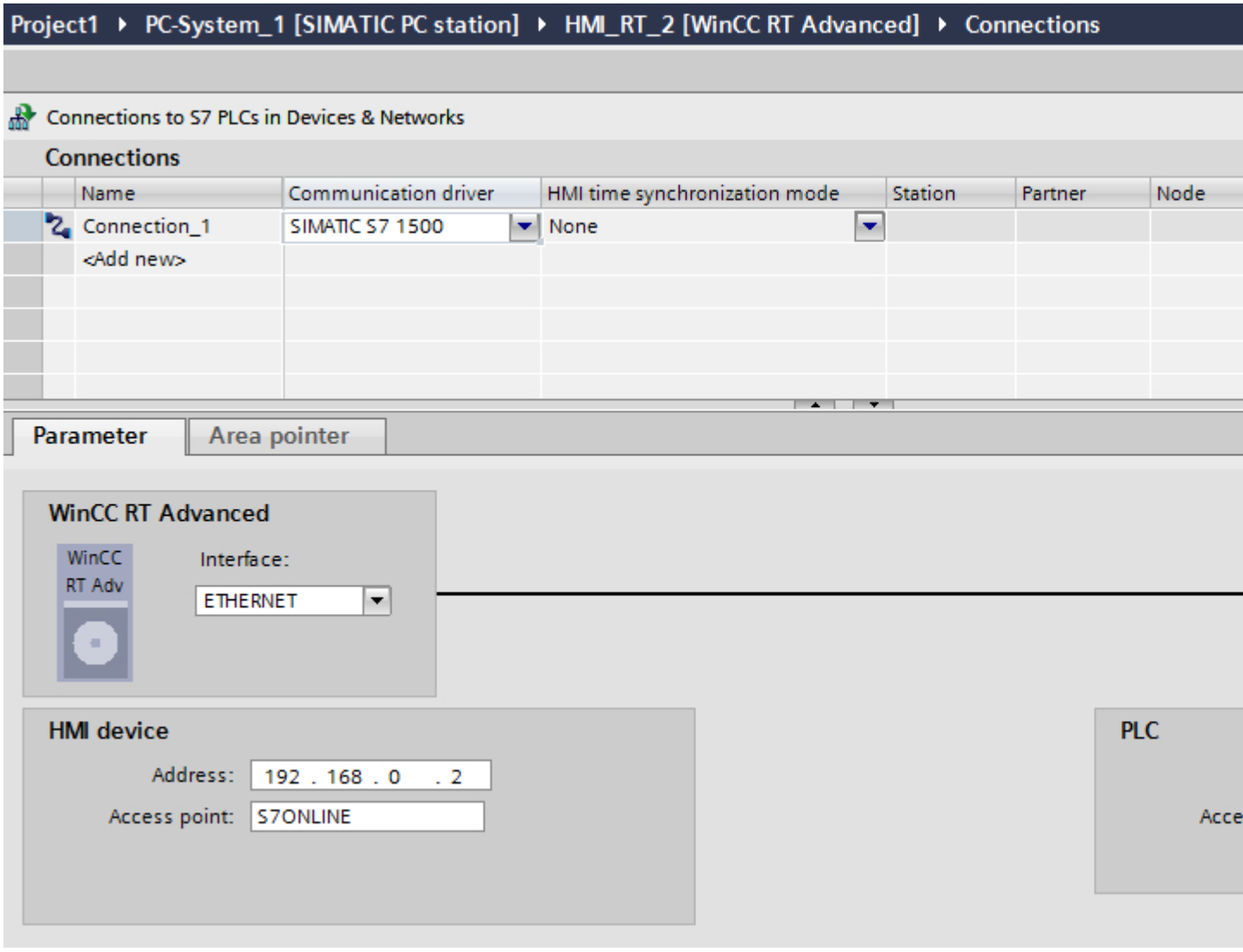
ETHERNET

Introduction

You assign the parameters for the HMI device and the controller in the network under "Parameters".

The parameters described in the following apply to the following interfaces:

- TCP/IP



HMI device

- "Address"
  Enter the IP address of the HMI device under "Address".
- "Access point"
  Specifies the access point for the PG/PC interface that can be used to reach the communication partner.

Controller

- "Address"
  Enter the IP address of the controller under "Address".
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the controller.

Note

You only need a password if you have set the "Complete protection" protection level at the controller.

No connection to the controller is established if the "Complete protection" protection level is stored on the controller and you do not enter a password.

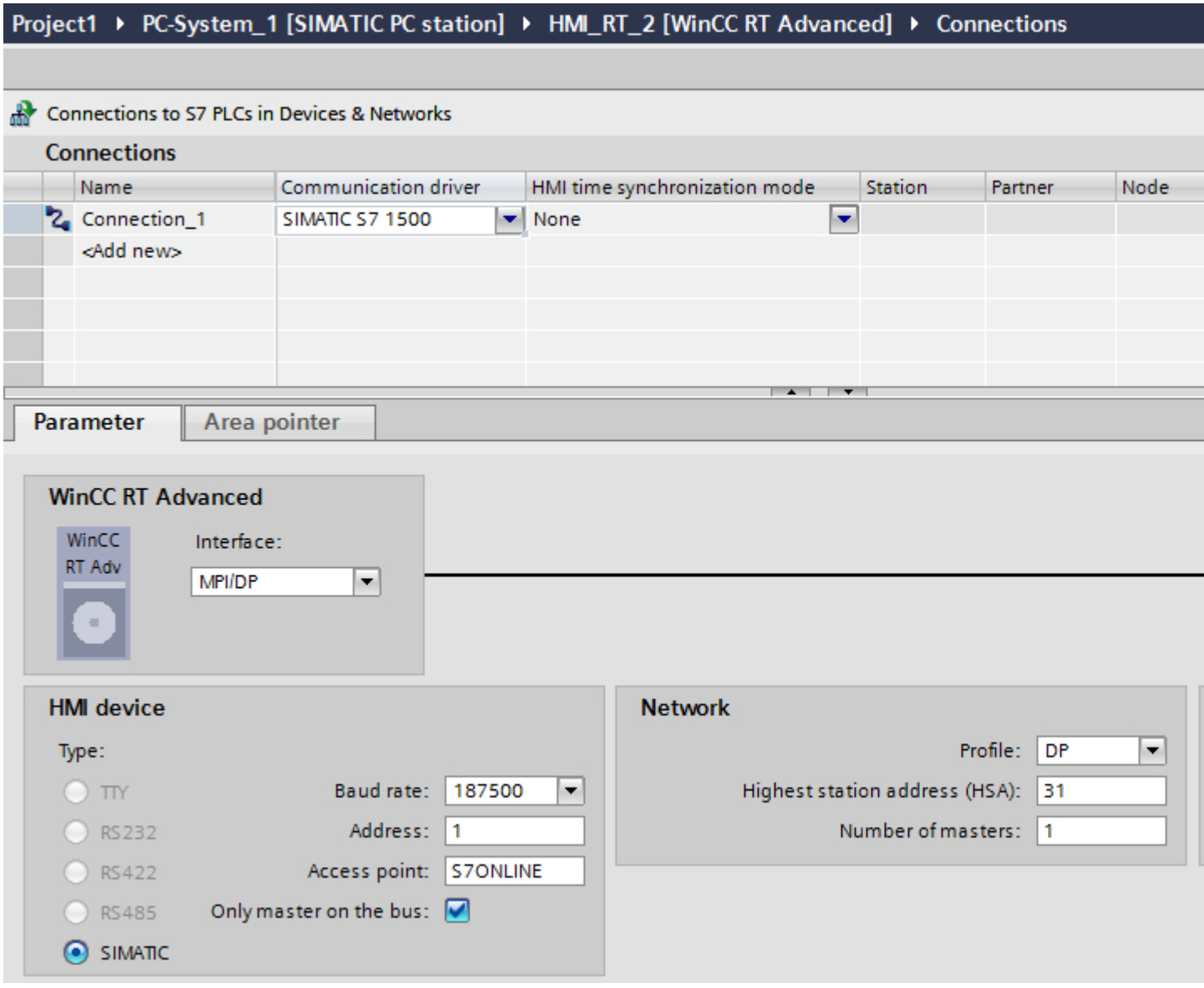
MPI/DP

Introduction

You assign the parameters for the HMI device and the controller in the network under "Parameters".

The parameters described in the following apply to the following interfaces:

- PROFIBUS



HMI device

- "Baud rate"
  Select the baud rate of the HMI device under "Baud rate".
- "Address"
  Enter the address of the HMI device under "Address".
- "Access point"
  Enter the logical device name under "Access point".

Controller

- "Address"
  Enter the address of the controller under "Address".
- "Access password"
  If you have configured the degree of protection "Complete protection" for the controller, enter the password for the controller under "Access password".

12.11.6.8 Configuring time synchronization

Time synchronization

Introduction

To have the same time of day throughout the plant, you can synchronize the time on various plant components using time synchronization. WinCC time synchronization is operated as a master-slave system.

One system component must be a clock for all components of a plant to work with identical time. The component functioning as the clock is referred to as the time master. The components that receive the time are time slaves.

Properties of time synchronization

- The HMI device can define the time as master or can accept the time of the PLC as slave.
- In "master mode", the time is synchronized at each connection setup.
- In "slave mode", the time is synchronized at each connection setup and than at cyclic intervals of 10 minutes.
- The first time synchronization is performed on the HMI device immediately after the start of runtime.
- Time synchronization is only performed on the HMI device during operation of runtime.

Time synchronization restrictions

Approved HMI devices

You can configure the time synchronization between a SIMATIC S7-1200 or SIMATIC S7-1500 and an HMI device with the following HMI devices:

Device	Operating system
Basic Panels	-
TP177 4"	Windows CE 5.0
Multi Panel 177	Windows CE 5.0

Device	Operating system
Multi Panel 277	Windows CE 5.0
Multi Panel 377	Windows CE 5.0
Mobile Panel 277	Windows CE 5.0
Mobile 277 IWLAN V2	Windows CE 5.0
Comfort Panels	Windows CE 6.0
PC systems with WinCC RT Advanced	Microsoft Windows XP Microsoft Windows 7

Configuration restrictions

- If an HMI device has several connections to SIMATIC S7-1200 or SIMATIC S7-1500, you can only configure one connection as "slave".
- If you have enabled time synchronization for the HMI device as "slave", you can no longer use the global area pointer "Date/time PLC".
- An HMI device can only request the time from a PLC with "Complete protection" security type configuration if the correct "Access password" is configured.
  Configure the "Access password" for communication with a PLC with "Complete protection" security type in the "Connections" editor of the HMI device.
  This "Access password" must match the password configured on the PLC. The PLC password is assigned in the PLC properties at: "General > Security"
- Basic Panels can only be configured as "Slave".
- If you use Basic Panels for the configuration, it is not possible to use time synchronization via NTP and the "Date/time PLC" area pointer simultaneously.
- Time synchronization with SIMATIC S7-1200 (V1.0) controllers is not possible.
- Time synchronization between the HMI device TP177 4" and SIMATIC S7-1200 (V4.0) controllers is not possible.
- Time synchronization between the HMI device TP177 4" and SIMATIC S7-1500 is not possible.

Configuring time synchronization for integrated connections

Introduction

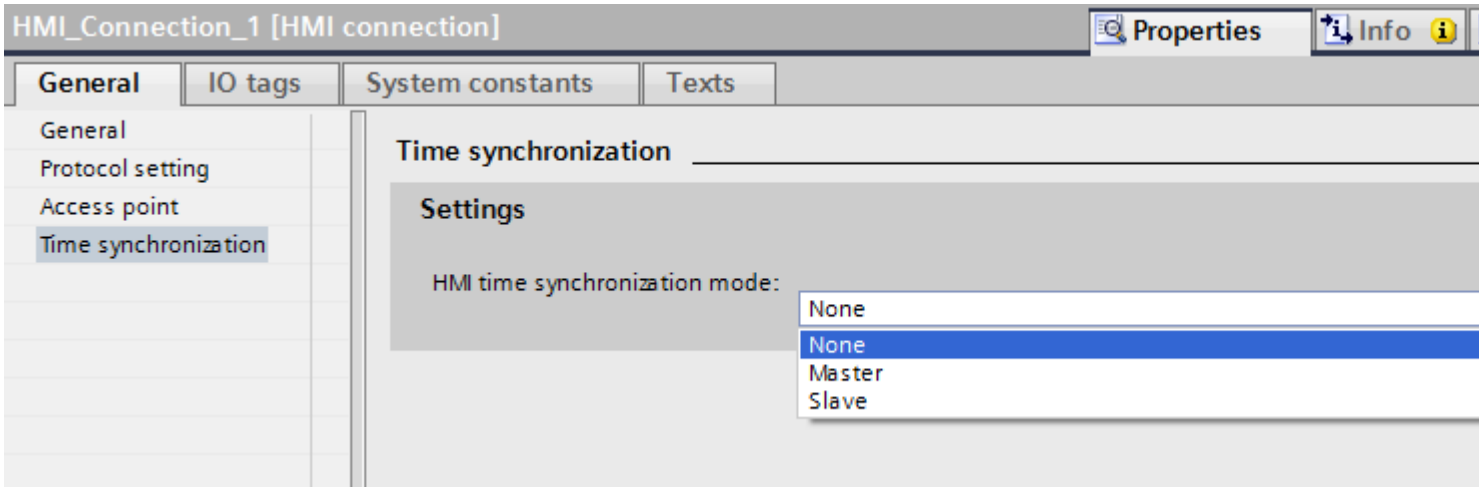
You configure time synchronization for an integrated connection in the "Devices & Networks" editor.

Requirements

- An HMI connection between an HMI device and a SIMATIC S7 1200 or SIMATIC S7 1500 has been configured.
- The HMI device must support the "time synchronization" function.
- The "Devices & Networks" editor is open.

Procedure

1. Click the line of the HMI connection in the "Devices & networks" editor.
2. Select the following in the inspector window under "General > Time synchronization > Settings":
   - None: No time synchronization is used.
   - Master: The HMI device sets the time.
   - Slave: The PLC sets the time.



Configuring time synchronization for non-integrated connections

Introduction

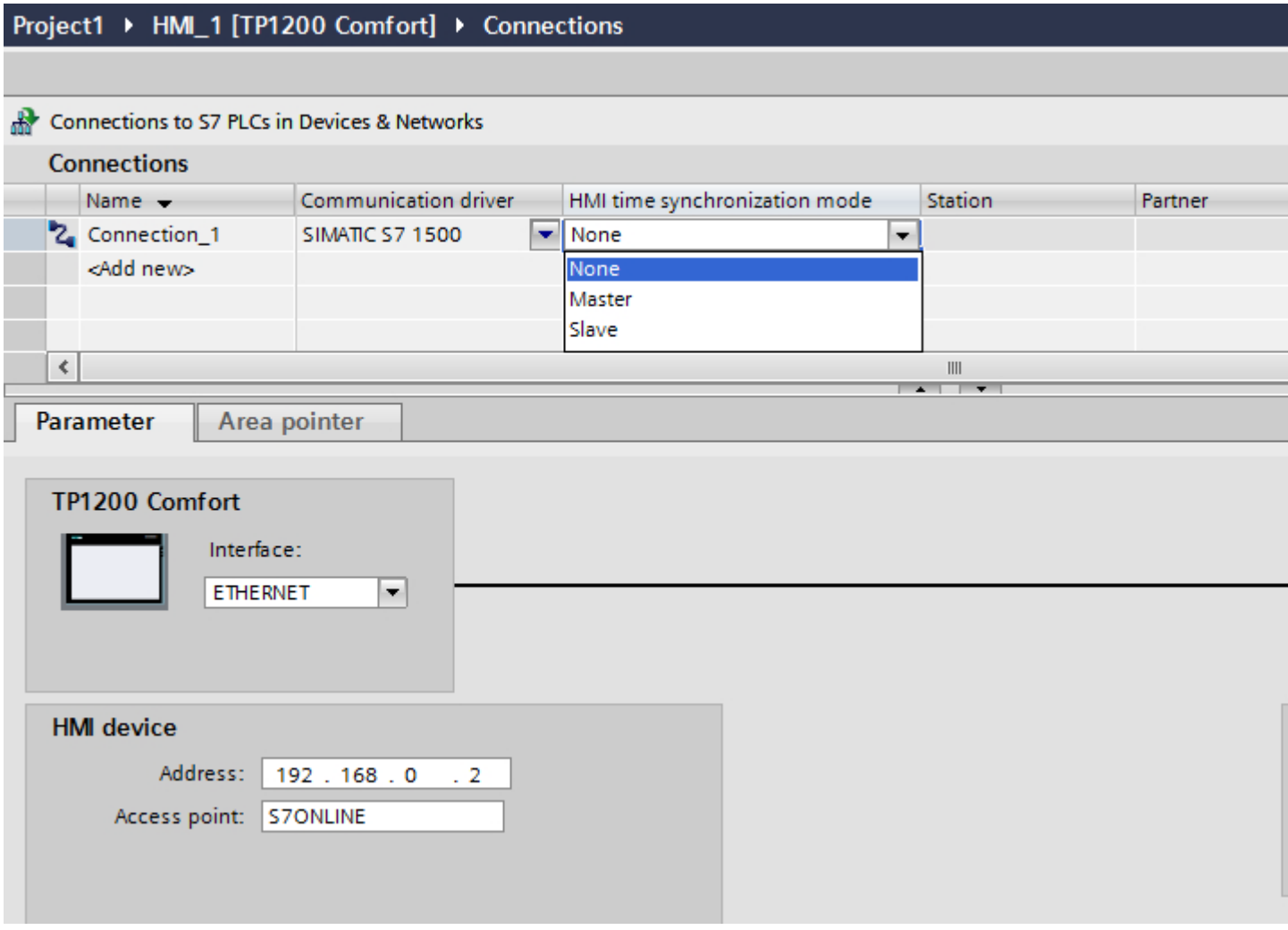
You configure time synchronization for a non-integrated connection in the "Connections" editor.

Requirements

- An HMI device which supports the "time synchronization" function has been created.
- "Connections" editor is open.

Procedure

1. Double-click "<Add>".
2. In the "Communication drivers" column, select the "SIMATIC S7 1500" PLC.
3. Select the following in the "HMI time synchronization mode" column:
   - None: No time synchronization is used.
   - Master: The HMI device sets the time.
   - Slave: The PLC sets the time.



12.11.7 Communicating with SIMATIC S7 1200

12.11.7.1 Communication with SIMATIC S7 1200

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 1200 PLC.

You can configure the following communication channels for the SIMATIC S7 1200 PLC:

- PROFINET
- PROFIBUS

HMI connection for communication

You configure connections between the HMI device and a SIMATIC S7 1200 in the "Devices & Networks" editor. If you have configured a HMI device with a serial port, you must configure a PROFIBUS-capable communication module to the SIMATIC S7 1200.

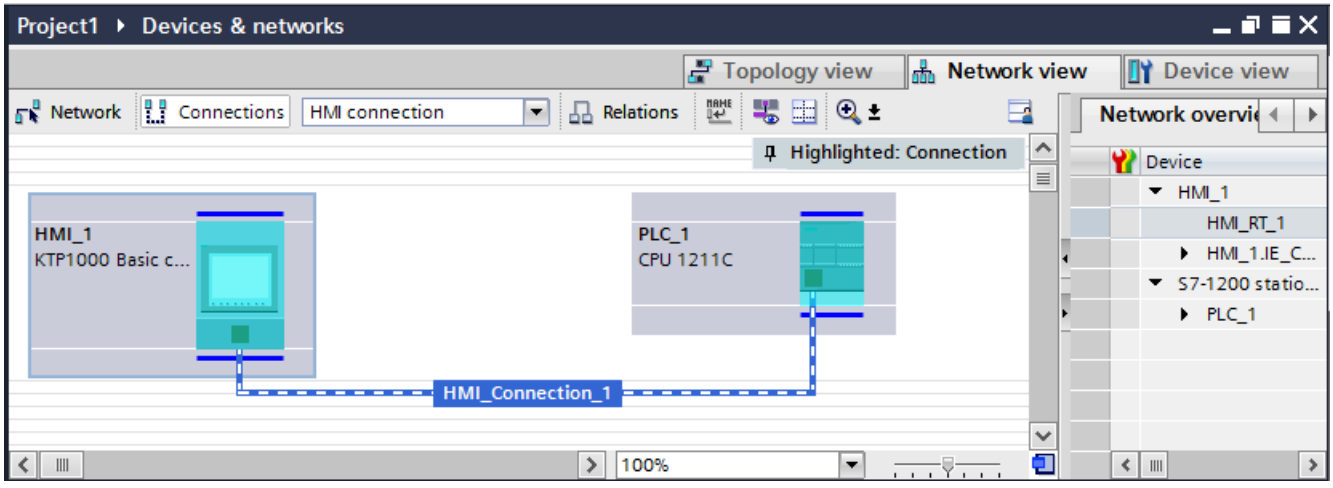
12.11.7.2 Communication via PROFINET

Configuring an HMI connection

Communication via PROFINET

HMI connections via PROFINET

If you have inserted an HMI device and a SIMATIC S7 1200 into the project, you interconnect the two PROFINET interfaces in the "Devices & Networks" editor.



You can also connect multiple HMI devices to one SIMATIC S7 1200 and multiple SIMATIC S7 1200s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 1200 via PROFINET or Ethernet in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

Requirements

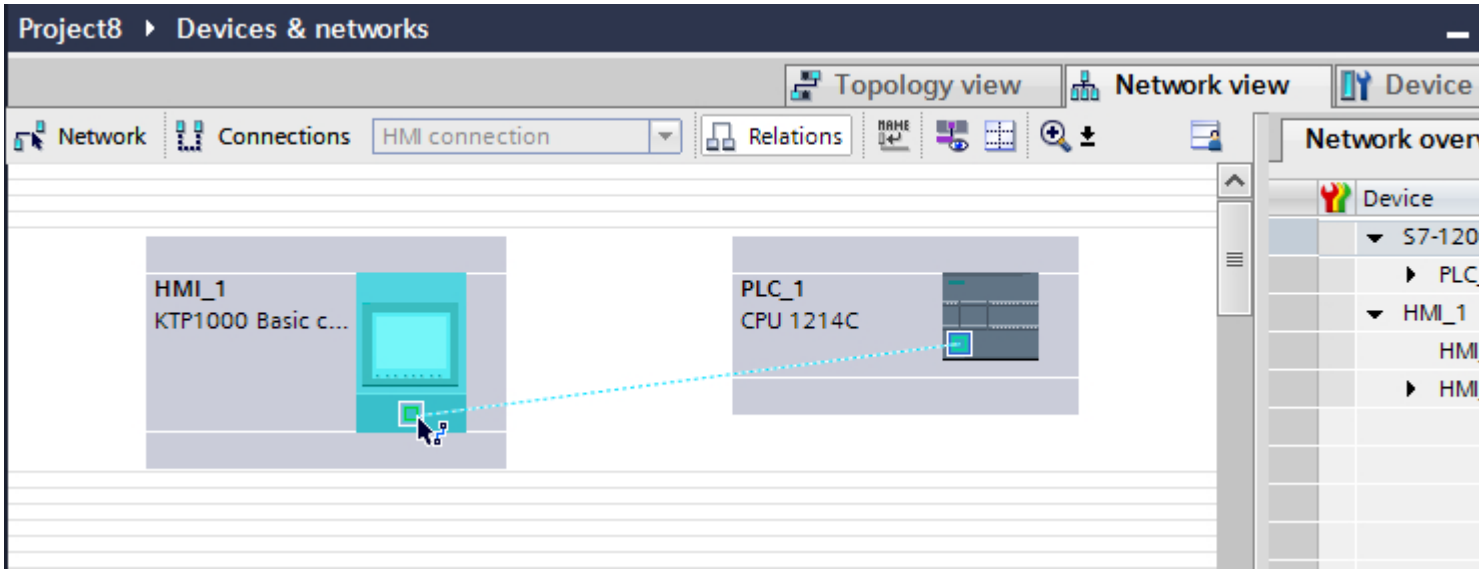
The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 1200
- HMI device with PROFINET or Ethernet interface

Procedure

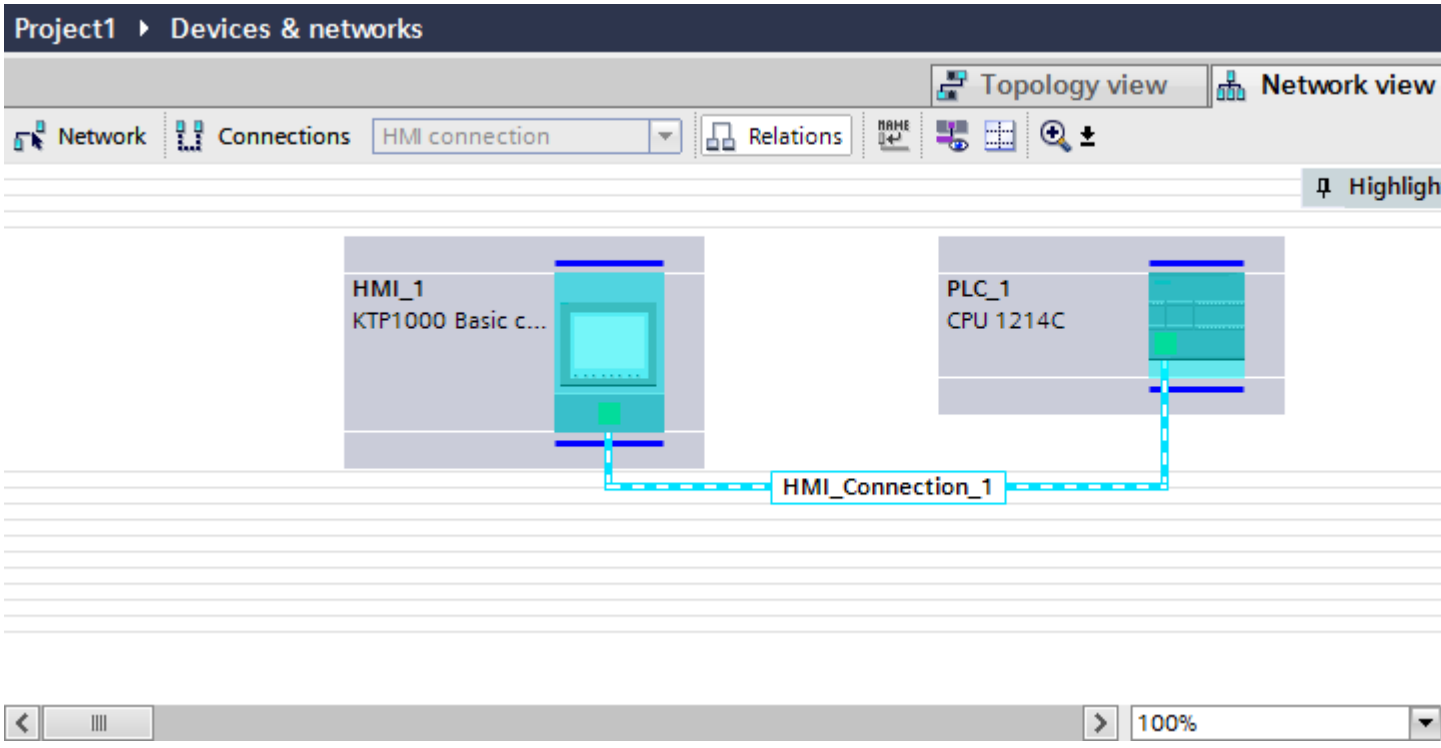
1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

3. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the HMI device.



4. Click the connecting line.

5. Click "Highlight HMI connection" and select the HMI connection.



The connection is displayed graphically in the Inspector window.

6. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project. See the chapter "PROFINET parameters (Page 6175)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 1200. The IP address and subnet mask connection parameters are configured.

Configuring an HMI connection

Communication via PROFINET

Communication via PROFINET

This section describes the communication between a WinCC Runtime and the SIMATIC S7 1200 PLC via PROFINET.

You can use the following WinCC Runtimes as an HMI device:

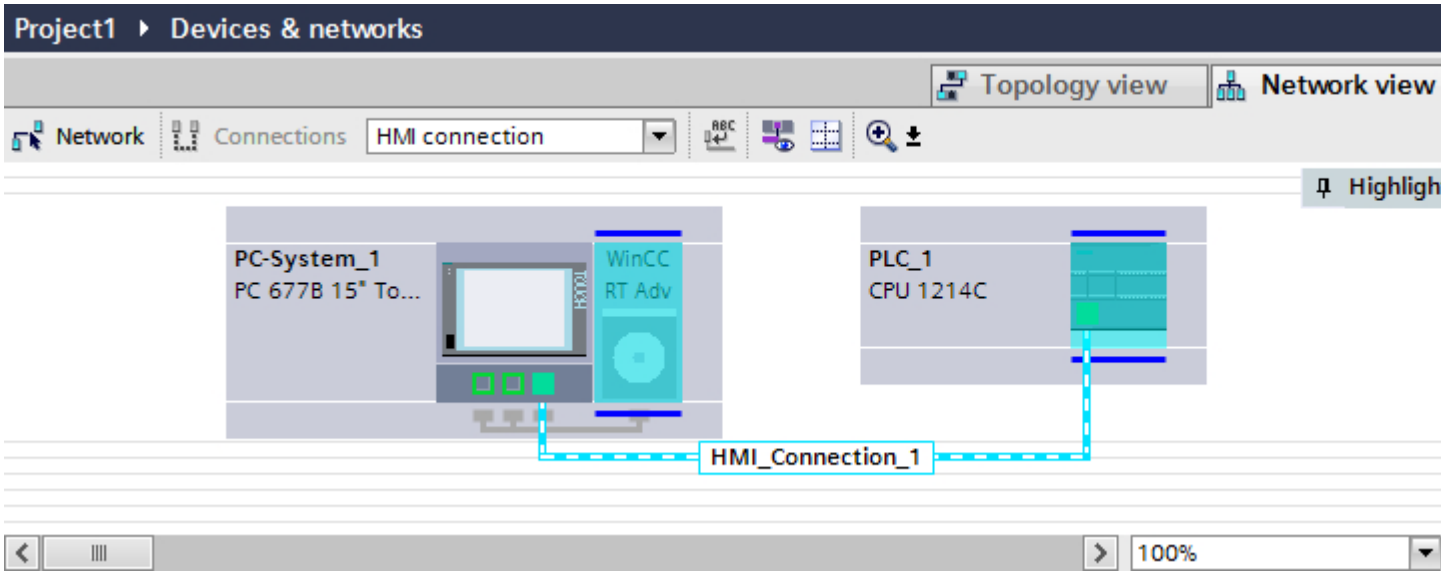
- WinCC RT Advanced

WinCC Runtime as HMI device

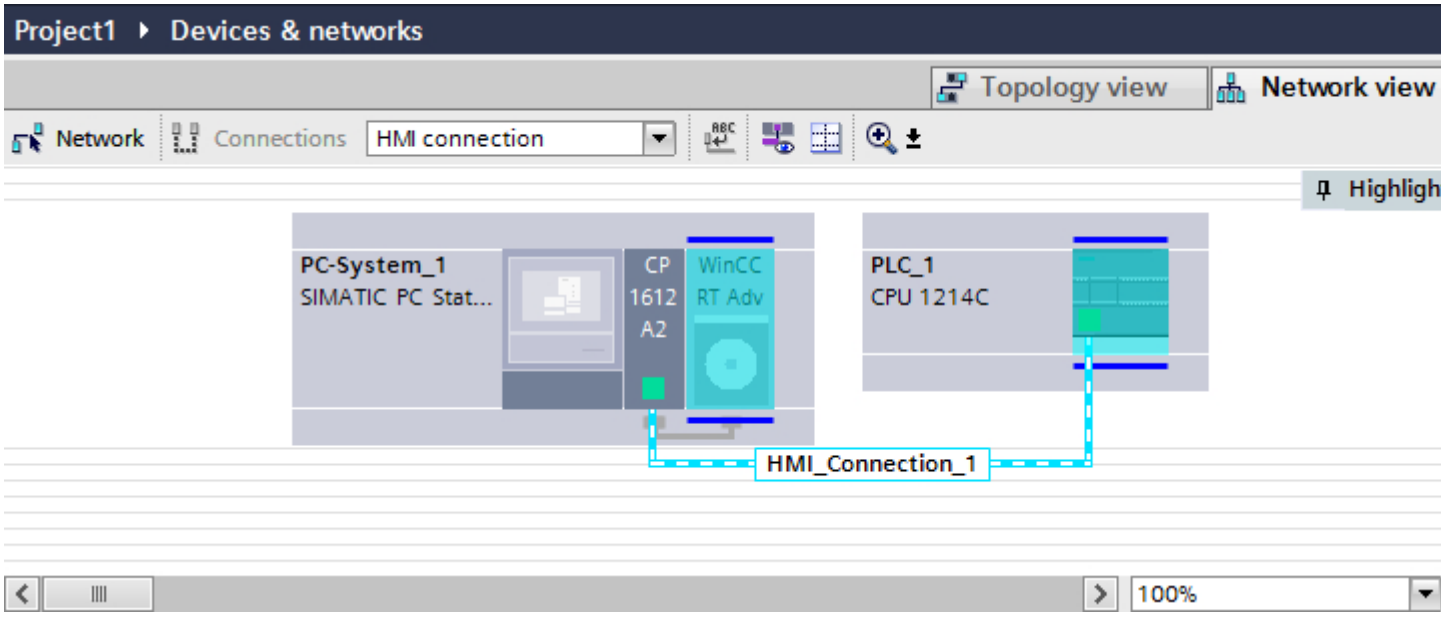
You configure the HMI connections between WinCC Runtime and SIMATIC S7 1200 in the "Devices & Networks" editor.

If you want to use a WinCC Runtime as an HMI device, you have the following options:

1. You create a SIMATIC PC and configure a WinCC Runtime on this SIMATIC PC.
   In this way you use the SIMATIC PC with a WinCC Runtime as an HMI device.



2. You create a Runtime and configure a communication processor to the Runtime.
   In this way you use your configuration PC as an HMI device.



You can also connect multiple HMI devices to one SIMATIC S7 1200 and multiple SIMATIC S7 1200s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET with a SIMATIC PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 1200 via PROFINET or Ethernet in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

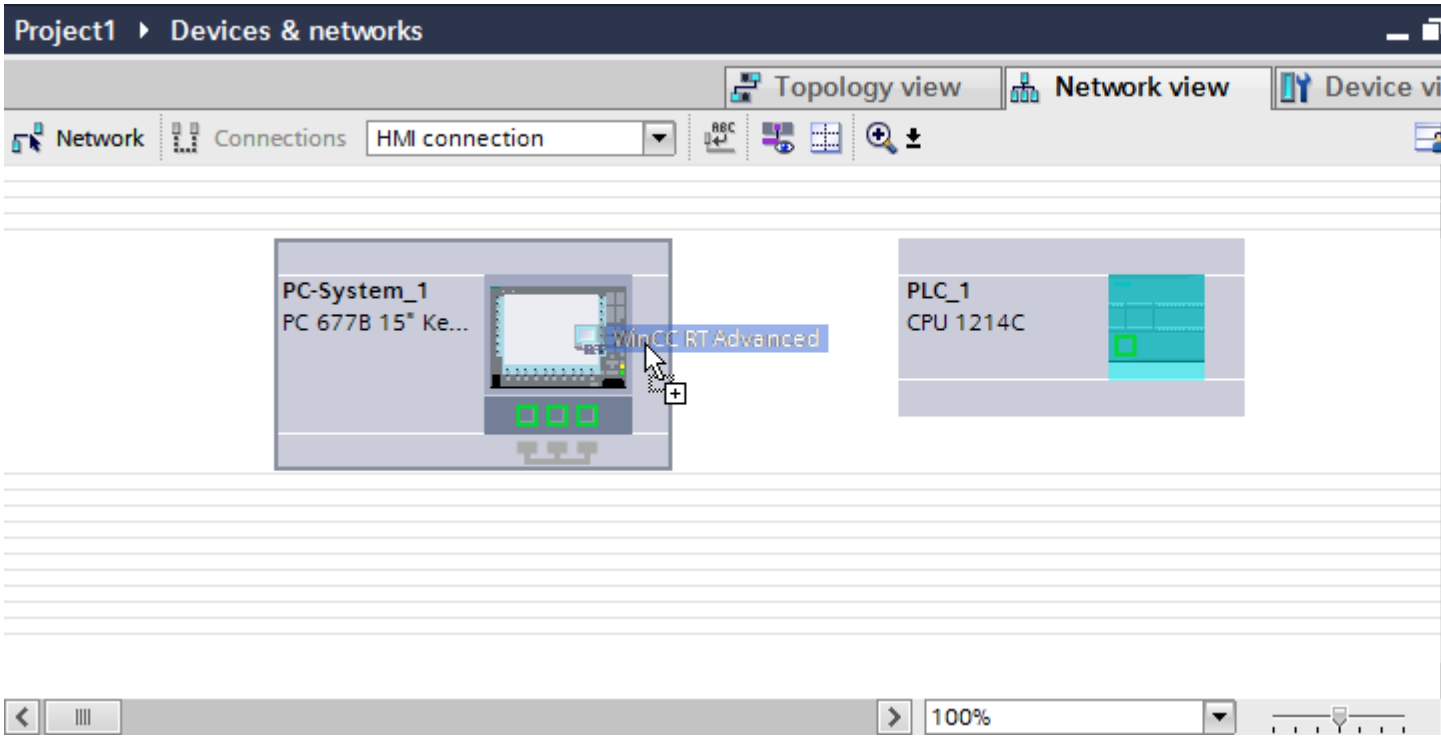
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 1200
- SIMATIC PC with PROFINET interface

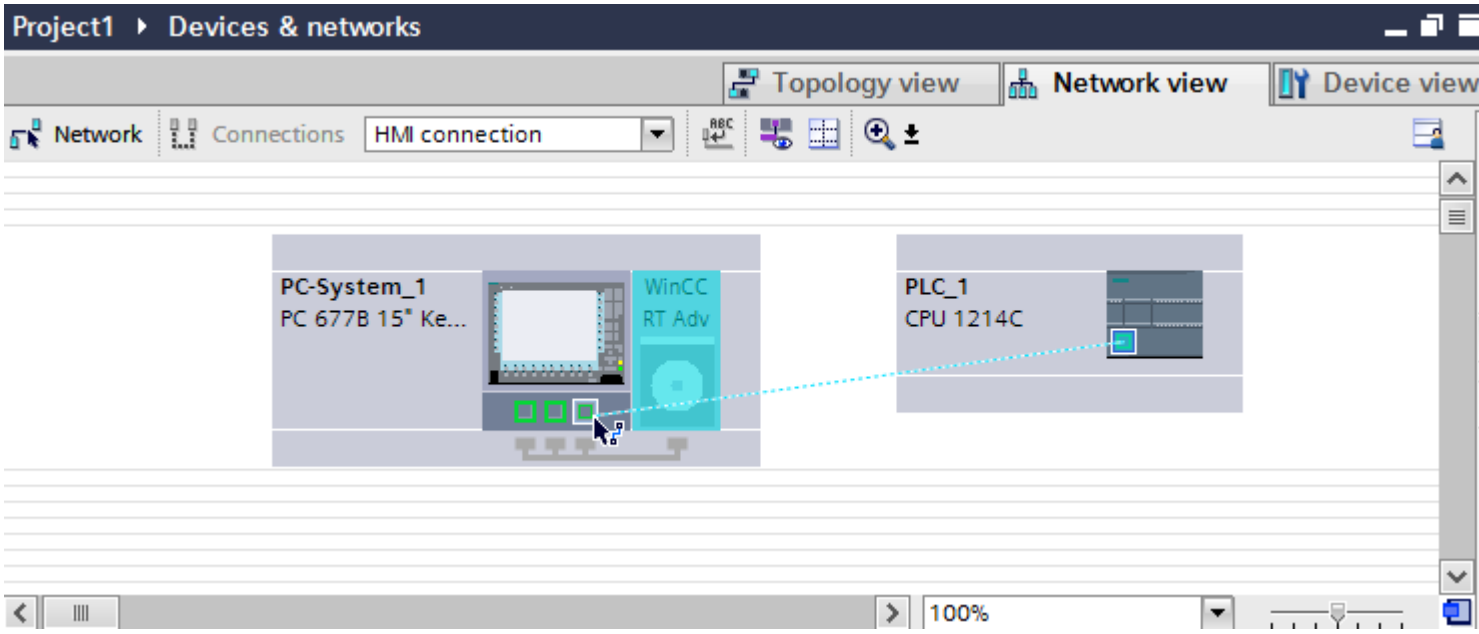
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a WinCC Runtime from the hardware catalog to the PC.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET interface of the PC.



5. Click the connecting line.
6. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
7. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFINET parameters (Page 6175)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 1200. The IP address and subnet mask connection parameters are configured.

Configuring an HMI connection via PROFINET with a PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 1200 via PROFINET or Ethernet in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

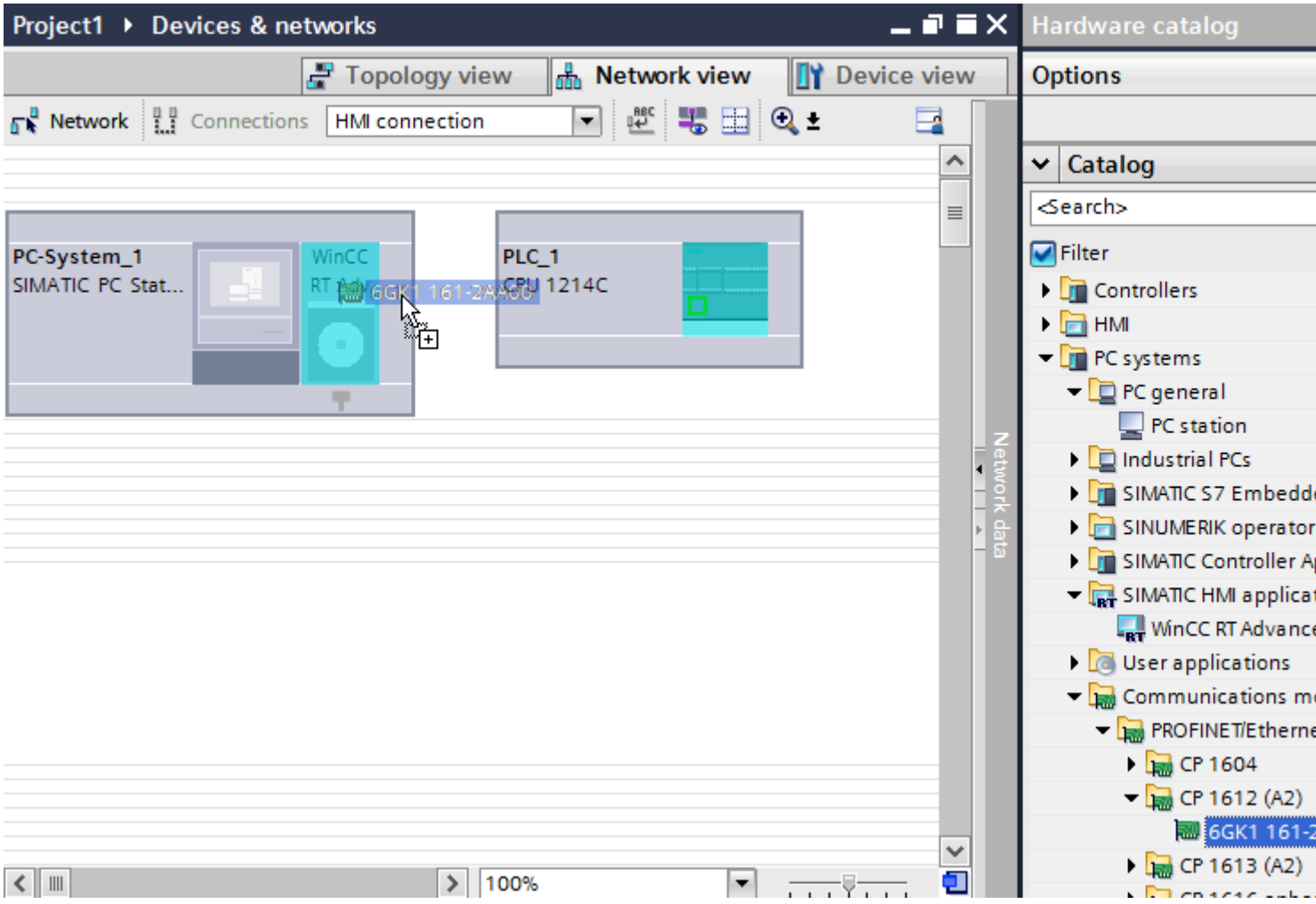
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 1200
- PC station with WinCC RT Advanced

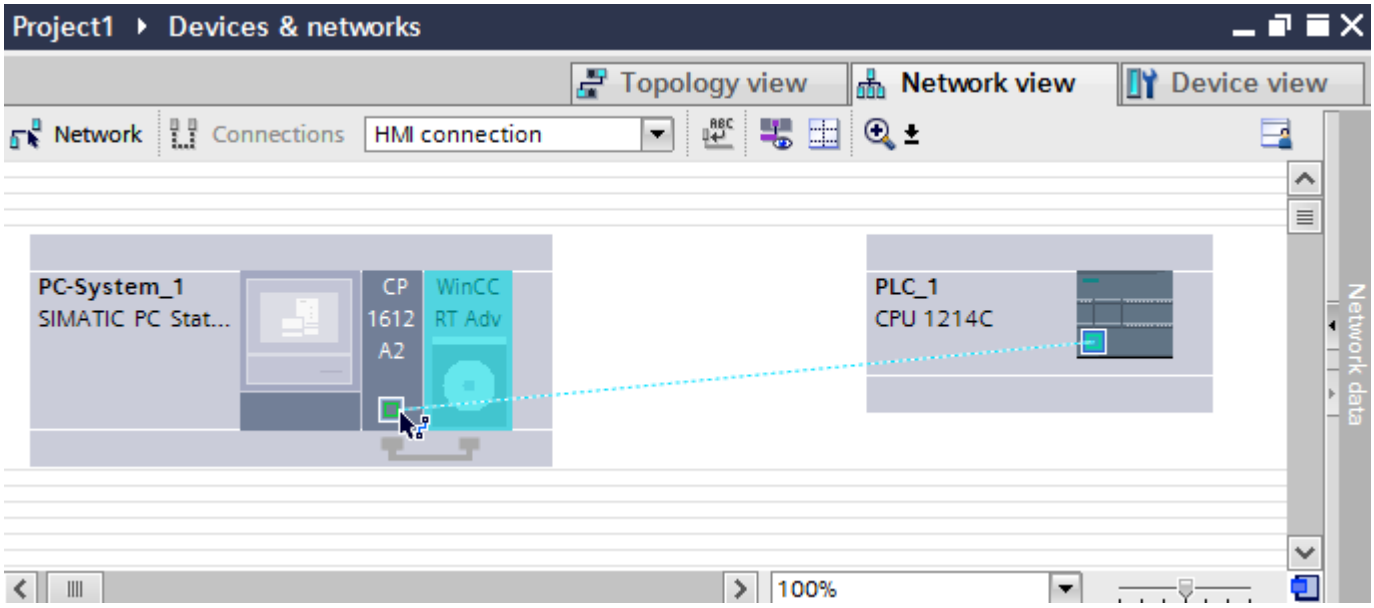
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a communication processor from the hardware catalog to the WinCC Runtime.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET interface of the communication processor.



5. Click the connecting line.
6. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
7. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFINET parameters (Page 6175)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 1200. The IP address and subnet mask connection parameters are configured.

PROFINET parameters

PROFINET parameters for the HMI connection

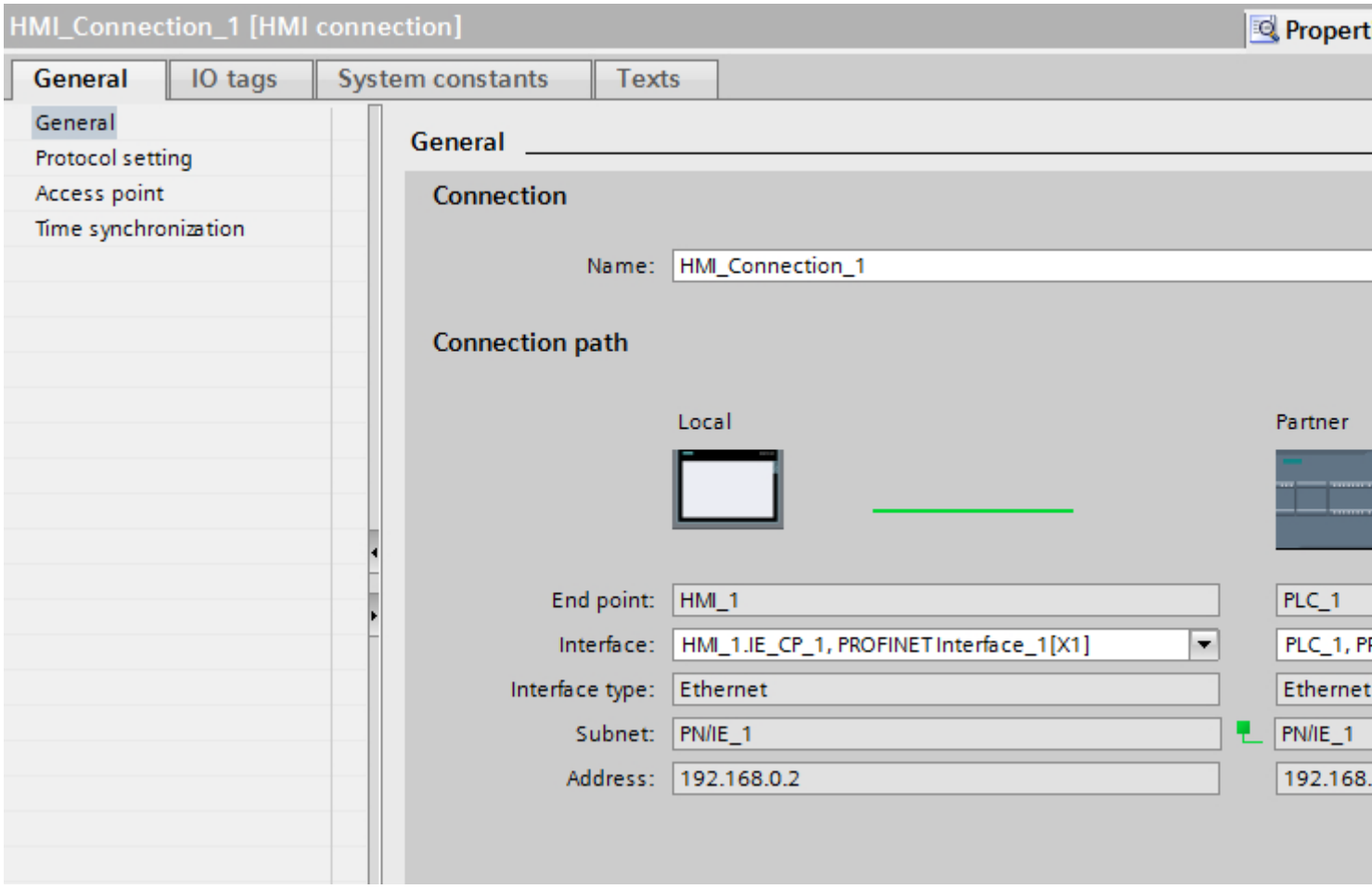
PROFINET parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and editing HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



Connection

The "Connection" area displays the HMI connection created for communication between the devices.

You can edit the name of the HMI connection in this area.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFINET parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area cannot be edited.
- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the selected IP address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

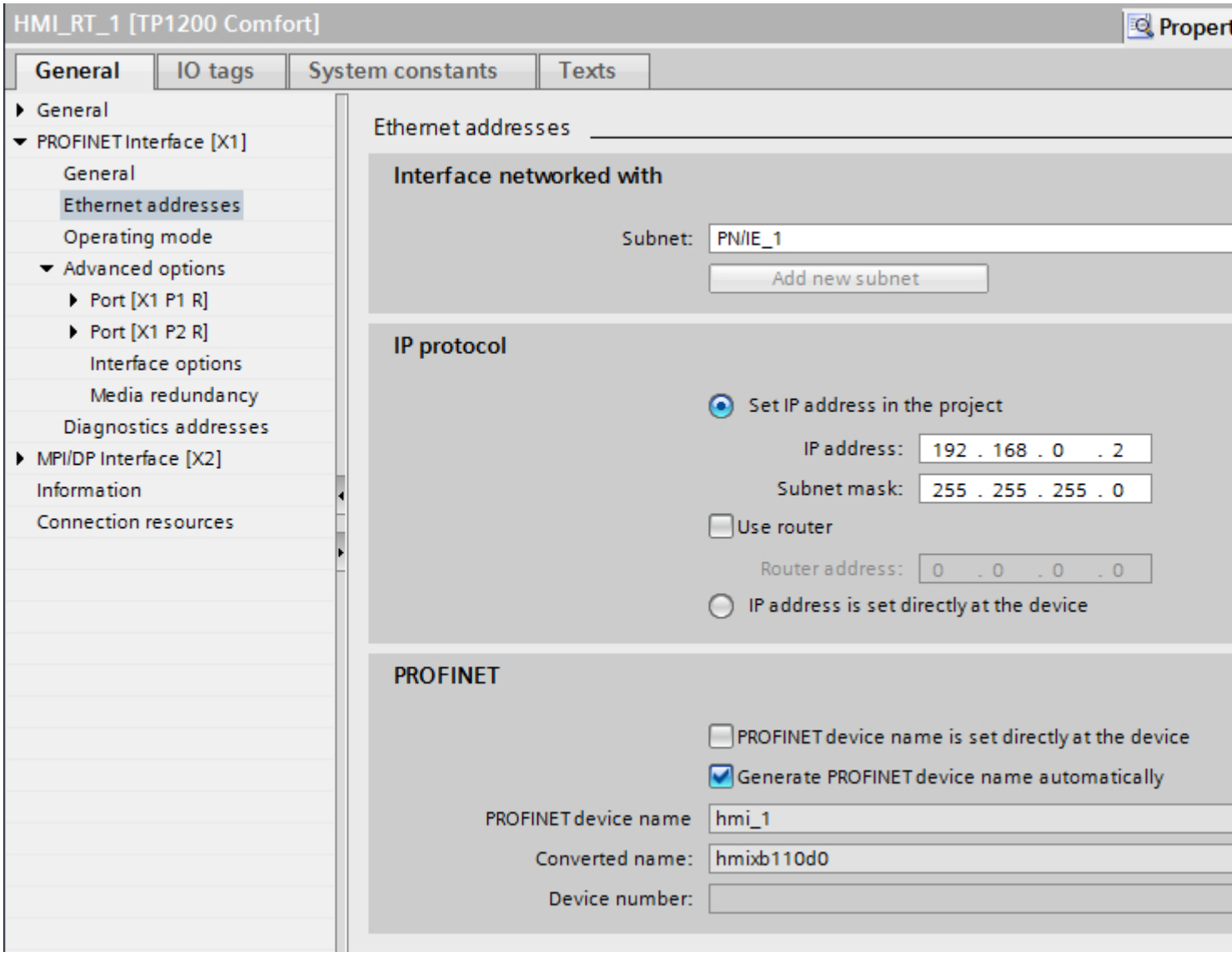
PROFINET parameters for the HMI device

PROFINET parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Set IP address in the project"
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

  Note

  The device is automatically restarted in the case of HMI devices with the Windows CE 3.0 operating system.

  HMI devices with Windows CE 3.0:
  - OP 77B
  - TP 177B color PN/DP
  - TP 177B mono DP
  - OP 177B color PN/DP
  - OP 177B mono DP
  - Mobile Panel 177 PN
  - Mobile Panel 177 DP
  - TP 277 6"
  - OP 277 6"

- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
- "Use IP router"
  If you are using an IP router, select "Use IP router" and enter the router address in the "Router address" field.
- "Set IP address using a different method"
  If the function "Set IP address using a different method" is activated, the IP address is not taken from the project. You have to enter the IP address directly in the Control Panel of the HMI device.

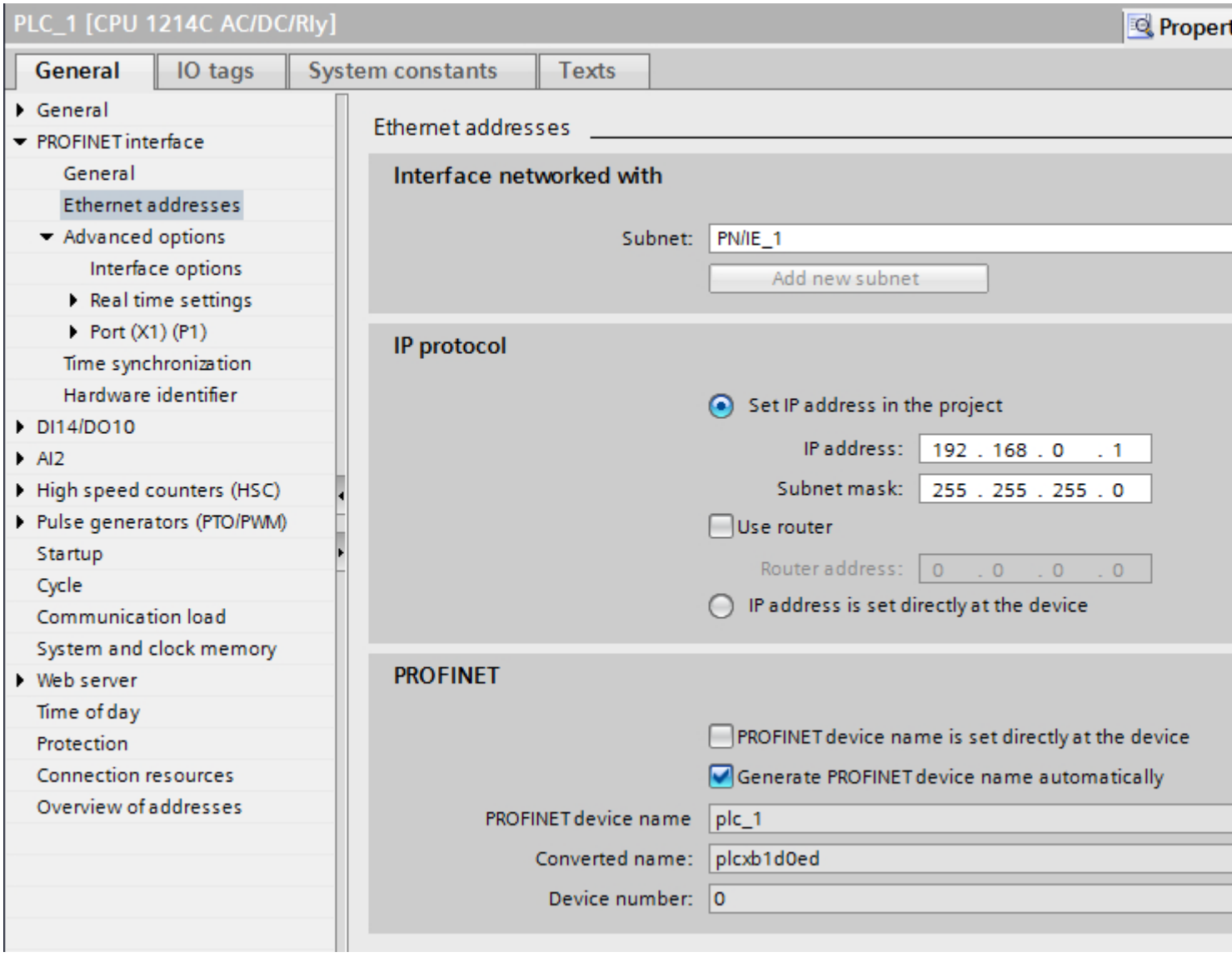
PROFINET parameters for the PLC

PROFINET parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFINET parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "IP address"
  You assign the IP address of the HMI device in the "IP address" area.
- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
  If you are using an IP router, select "Use IP router" and enter the router address in the field.

Configuring Industrial Ethernet

Rules for the network configuration

The Ethernet interfaces of the devices have a default IP address that you can change.

IP address

The IP parameters are visible if the communication-capable devices support the TCP/IP protocol.

The IP address consists of 4 decimal figures in the range of 0 to 255. The decimal figures are separated from one another by a dot.

Example: 140.80.0.2

The IP address consists of the following:

- The address of the (sub) net
- The address of the node (generally also called host or network node)

Subnet mask

The subnet mask splits these two addresses. It determines which part of the IP address addresses the network and which part of the IP address addresses the node.

The set bits of the subnet mask determine the network part of the IP address.

Example:

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

In the example given for the above IP address, the subnet mask shown here has the following meaning:

The first 2 bytes of the IP address identify the subnet - i.e. 140.80. The last two bytes address the node, thus 0.2.

It is generally true that:

- The network address results from AND linking the IP address and subnet mask.
- The node address results from AND NOT linking the IP address and subnet mask.

Relation between IP address and default subnet mask

An agreement exists relating to the assignment of IP address ranges and so-called "Default subnet masks". The first decimal number (from the left) in the IP address determines the structure of the default subnet mask. It determines the number of "1" values (binary) as follows:

IP address (decimal)	IP address (binary)	Address class	Default subnet mask
0 to 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 to 191	10xxxxxx.xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

Note

Range of values for the first decimal point

A value of between 224 and 255 is also possible for the first decimal number of the IP address (address class D etc). This is, however, not recommended because there is no address check for these values.

Masking other subnets

You can use the subnet mask to add further structures and form "private" subnets for a subnet that is assigned one of the address classes A, B or C. This is done by setting other lower points of the subnet mask to "1". For each bit set to "1", the number of "private" networks doubles and the number of nodes they contain is halved. Externally, the network functions like an individual network as it did previously.

Example:

You have a subnet of address class B (e.g. IP address 129.80.xxx.xxx) and change the default subnet mask as follows:

Masks	Decimal	Binary
Default subnet mask	255.255.0.0	11111111.11111111.00000000.00000000
Subnet mask	255.255.128.0	11111111.11111111.10000000.00000000

Result:

All nodes with addresses between 129.80.001.xxx and 129.80.127.xxx are on one subnet, all nodes with addresses between 129.80.128.xxx and 129.80.255.xxx are on another subnet.

Router

The job of the routers is to connect the subnets. If an IP datagram is to be sent to another network, it first has to be conveyed to a router. To make this possible, in this case you have to enter the address of the router for each node in the subnet.

The IP address of a node in the subnet and the address of the router may only differ at the points at which there is a "0" in the subnet mask.

Configuring time synchronization for integrated connections

Introduction

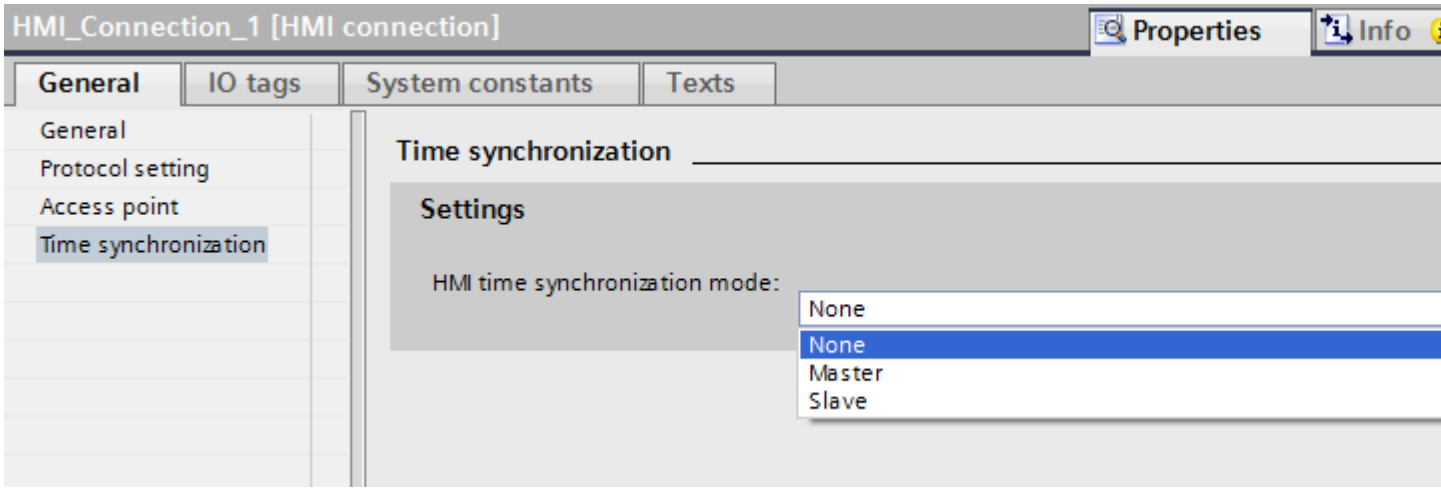
You configure time synchronization for an integrated connection in the "Devices & Networks" editor.

Requirements

- An HMI connection between an HMI device and a SIMATIC S7 1200 has been configured.
- The HMI device must support the "time synchronization" function.
- The "Devices & Networks" editor is open.

Procedure

1. Click the line of the HMI connection in the "Devices & networks" editor.
2. Select the following in the inspector window under "General > Time synchronization > Settings":
   - None: No time synchronization is used.
   - Master: The HMI device sets the time.
   - Slave: The PLC sets the time.



Protection of communication

Security levels

If you want to protect the controller and HMI device communication, you can assign protection levels for the communication.

For a SIMATIC S7-1200 CPU, you can enter multiple passwords and thereby set up different access rights for various user groups.

The passwords are entered in a table, so that exactly one protection level is assigned to each password.

The effect of the password is given in the "Protection" column.

For the SIMATIC S7-1200 controller, several aspects need to be considered when setting protection levels.

For additional information on this, see:

Setting options for the protection (FW as of V4) (Page 6184)

Setting options for the protection level (FW V1 to V3) (Page 6186)

Example

When configuring the controller, you select the "Complete protection" protection level for a standard CPU (i.e., not an F-CPU).

Afterwards, you enter a separate password for every protection level above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the assigned passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (no protection) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (write protection) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (read and write protection) allows access as if the CPU were read-protected and write-protected, so that only HMI access is possible for users who know this password.

Setting options for the protection (FW as of V4)

Protection level

The following section describes how to use the various access levels of the S7-1200 CPUs as of V4.

S7-1200 CPUs provide various access levels to limit the access to specific functions.

The parameters for the access levels are assigned in a table. The green checkmarks in the columns to the right of the respective access level specify which operations are possible

without knowing the password of this access level. If you want to use the functions of check boxes that are not selected, a password has to be entered.

Notice

Configuring an access level does not replace know-how protection

Configuring access levels prevents unauthorized changes to the CPU by restricting download privileges. However, blocks on the memory card are not write- or read-protected. Use know-how protection to protect the code of blocks on the memory card.

Default characteristics

The default access level is "Full access (no protection)". Every user can read and change the hardware configuration and the blocks. A password is not set and is also not required for online access.

The access levels in detail

With an S7-1200 CPU, you can configure the following access levels:

- Full access (no protection): The hardware configuration and the blocks can be read and changed by all users.
- Read access: With this access level, only read access to the hardware configuration and the blocks is possible without entering a password - meaning that you can load the hardware configuration and blocks into the programming device. In addition, HMI access and access to diagnostics data is possible.
  You cannot load blocks or a hardware configuration into the CPU without entering the password. Moreover, writing test functions and firmware updates are **not** possible without a password.
- HMI access: With this access level, only HMI access and access to diagnostics data is possible without entering the password.
  Without entering the password, you can neither load blocks and hardware configuration into the CPU, nor load blocks and hardware configuration from the CPU into the programming device. In addition, the following is **not** possible without a password: Writing test functions, changing the operating state (RUN/STOP) and firmware updates.
- No access (complete protection): When the CPU is completely protected, no read or write access to the hardware configuration and the blocks is possible. HMI access is also not possible. The server function for PUT/GET communication is disabled in this access level (cannot be changed).
  Authorization with the password again provides you full access to the CPU.

Behavior of a password-protected module during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Validity is checked before the online function is executed. If password protection is in place, you are prompted to enter a password.

Example: The module was configured with read access and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on.

Access authorization to the protected data is in effect for the duration of the online connection or until the access authorization is manually rescinded with "Online > Delete access rights".

Each access level allows unrestricted access to certain functions without entering a password, for example, identification using the "Accessible devices" function.

Setting options for the protection level (FW V1 to V3)

Protection level

The following section describes how to use the various protection levels of the S7-1200 CPUs V1 to V3.

Effects of the protection level setting

You can choose between the following protection levels:

- No protection: This corresponds to the default behavior. You cannot enter a password. Read and write access is always permitted.
- Write protection: Only read-only access is possible. You cannot change any data on the CPU and cannot load any blocks or a configuration. HMI access and communication between CPUs are excluded from the write protection. Assignment of a password is required to select this protection level.
- Write/read protection: No write or read access is possible in the "Accessible devices" area or in the project for devices that are switched online. Only the CPU type and the identification data can be displayed in the project tree under "Accessible devices". Display of online information or blocks under "Accessible devices", or in the project for devices interconnected online, is possible.
  HMI access and communication between CPUs are excluded from the write protection.
  Assignment of a password is required to select this protection level.

Behavior of a password-protected CPU during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Validity is checked before the online function is executed. If password protection is in place, you are prompted to enter a password.

Example: The module was assigned write protection and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on with a password.

Access authorization to the protected data is in effect for the duration of the online connection or until the access authorization is manually rescinded with "Online > Delete access rights". Access authorization will also expire when the project is closed.

Note

You can not restrict functions for process control, monitoring, and communications.

Some functions are still protected due to their use as online data. RUN/STOP in the "Online Tools" task card or "Set the time" in the diagnostics and online editor is therefore write-protected.

Access password for the HMI connection

Introduction

Secure access to a PLC by assigning a password.

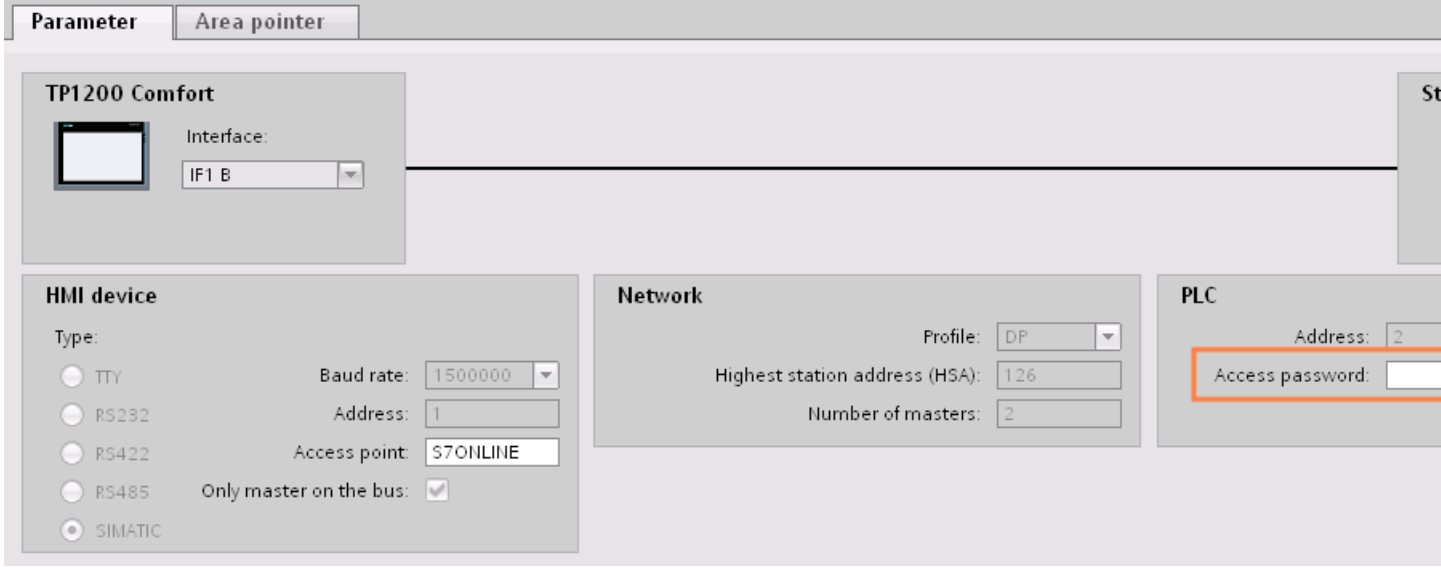
Assign the password when configuring the connection.

Input of the password of the PLC is mandatory as of "Complete protection" security level.

Communication to the PLC is denied if an incorrect password or no password is entered.

Assigning password

Enter the "Access password" for the PLC in the "Connections" editor.



Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission rate / duplex

Depending on the selected device, you can make the following settings for "Transmission rate / duplex":

- Automatic setting
  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the peer port. The "Enable autonegotiation" option is also enabled as a default, in other words, you can use cross cables or patch cables for the connection.
- TP/ITP at x Mbps full duplex (half duplex)
  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:
  - Autonegotiation enabled
    You can use both cross cable and patch cable.
  - Autonegotiation disabled
    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.
- Deactivated
  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option enables or disables port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because with this option the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can accordingly not be optimally set.

Note

When a local port is connected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not accept the setting, an error message is generated.

Wiring rules for disabled autonegotiation

Requirements

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission rate
- Autonegotiation incl. autocrossing disabled

The time for negotiating the transmission rate during startup has been saved.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

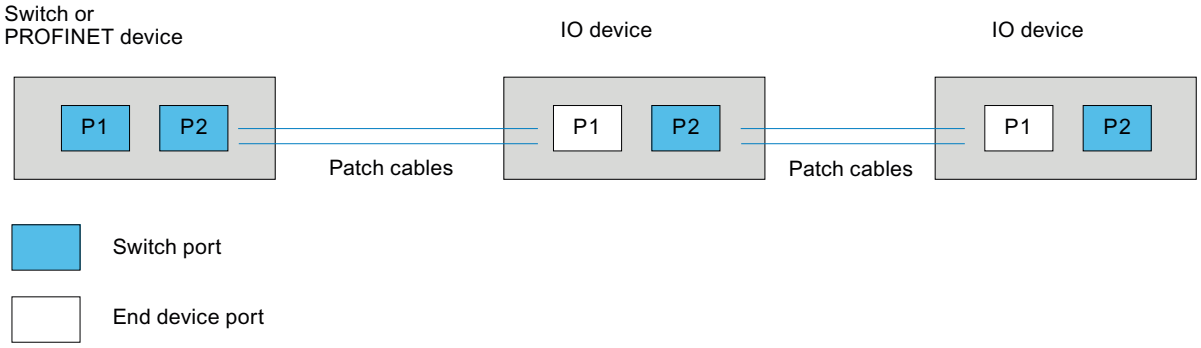
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in line using a single cable type (patch cable). To do this, you connect port 2 of the IO device (distributed I/O) with port 1 of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirements

To use boundaries, the respective device must have more than one port. If the PROFINET does not support boundary settings, they are not shown.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of discovery of accessible devices"
  No forwarding of DCP frames to identify accessible devices. Devices downstream from this port cannot be reached by the project tree under "Accessible devices". Devices downstream from this port cannot be reached by the CPU.
- "End of topology discovery"
  LLDP frames (Link Layer Discovery Protocol) are not forwarded for topology detection.
- "End of sync domain"
  No forwarding of sync frames transmitted to synchronize nodes within a sync domain.
  If you operate, for example, a PROFINET device with more than two ports in a ring, you should prevent the sync frame from being fed into the ring by setting a sync boundary (at the ports not inside the ring).
  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device from the other sync domain.

Restrictions

The following restrictions must be observed:

- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been determined for the port, the following check boxes cannot be used:
  - "End of discovery of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

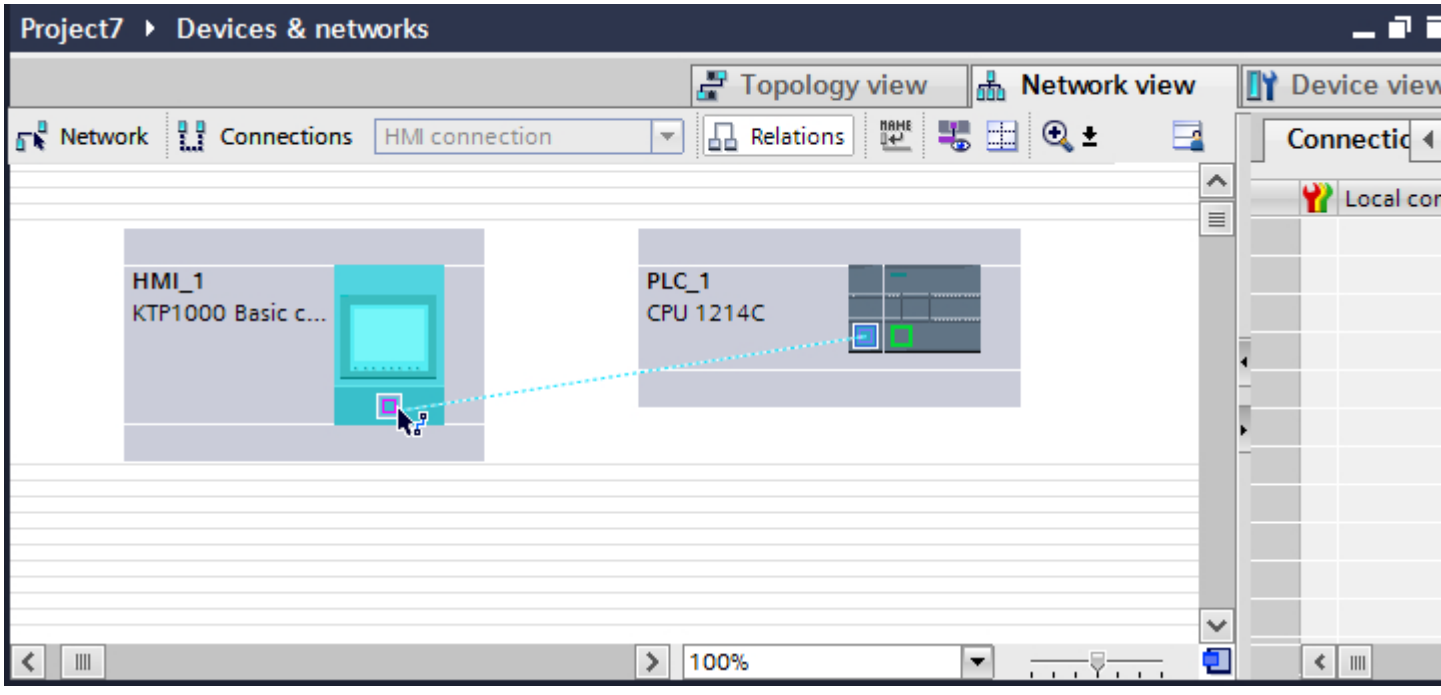
12.11.7.3 Communication via PROFIBUS

Configuring an HMI connection

Communication via PROFIBUS

HMI connections via PROFIBUS

If you want to connect a SIMATIC S7 1200 to a HMI device via PROFIBUS, you must configure a PROFIBUS-capable communication module to a slot of the controller first.



HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7-1200 via PROFIBUS in the "Devices & Networks" editor.

Requirements

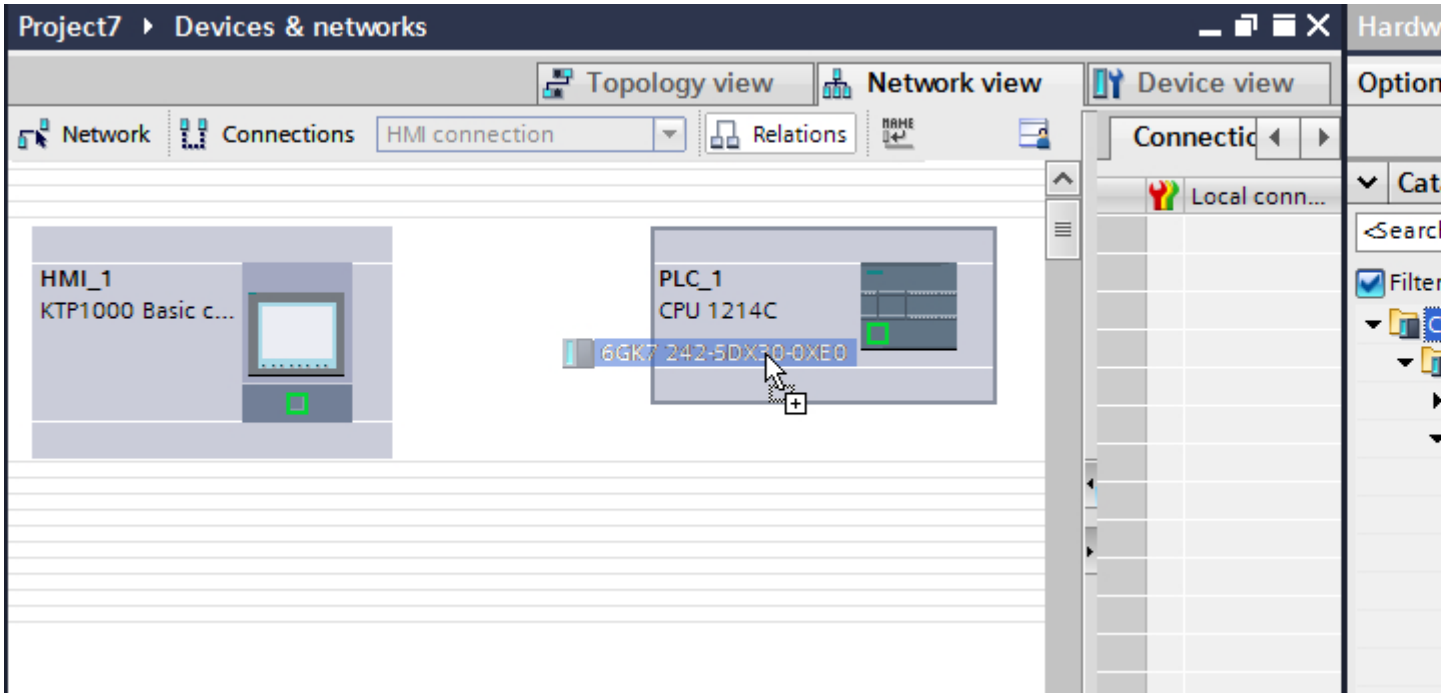
The following communication partners are created in the "Devices & Networks" editor:

- HMI device with MPI/DP interface
- SIMATIC S7-1200

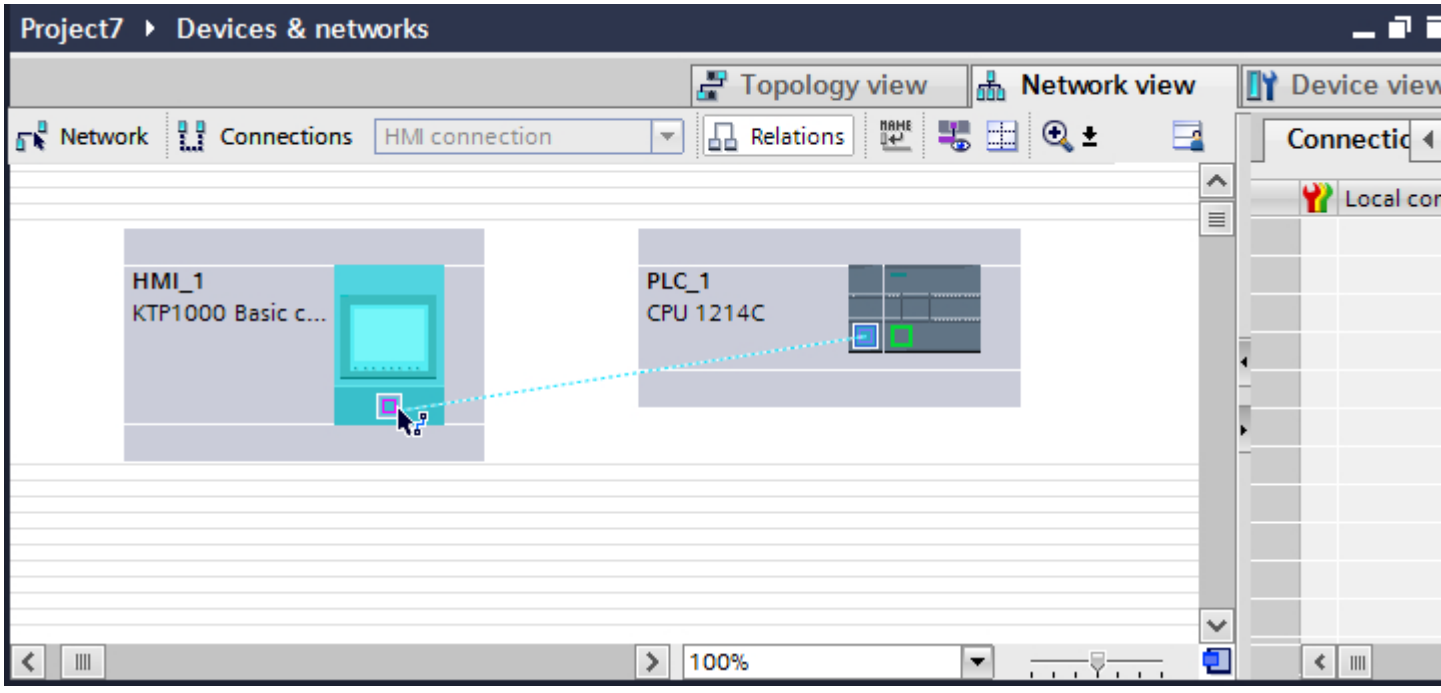
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button.
   The devices available for connection are highlighted in color.

3. Use a drag-and-drop operation to move a PROFIBUS-capable communication module from the hardware catalog to the PLC.



4. Click the HMI device interface.
5. Select the "PROFIBUS" interface type in the Inspector window under "Properties > General > PROFIBUS address/ MPI address > Parameters".
6. Click the interface of the communication module and use a drag-and-drop operation to draw a connection to the HMI device.



7. Click the name of the connection.
   The connection is displayed graphically in the Inspector window.
8. Click "Highlight HMI connection" and select the HMI connection.
9. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the section "PROFIBUS parameters (Page 6202)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7-1200 via PROFIBUS.

Configuring an HMI connection

Communication via PROFIBUS

Communication via PROFIBUS

This section describes the communication between a WinCC Runtime and the SIMATIC S7 1200 PLC via PROFIBUS.

You can use the following WinCC Runtimes as an HMI device:

- WinCC RT Advanced

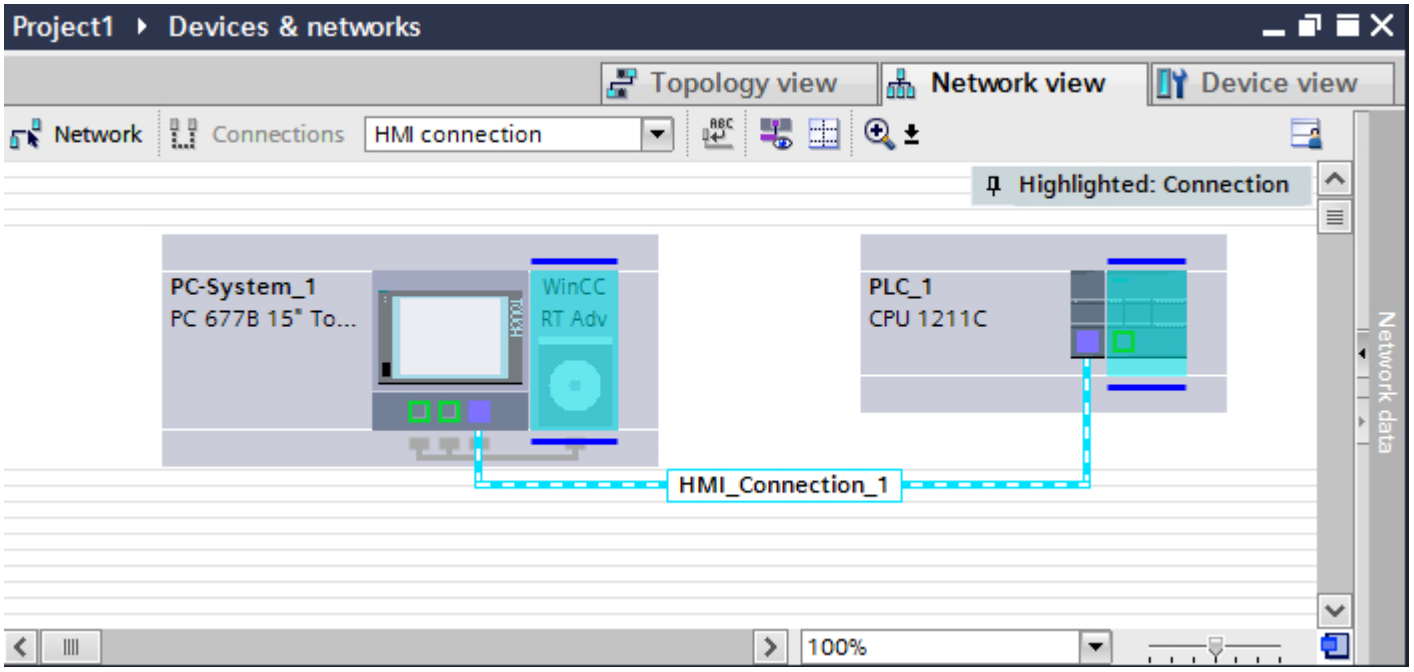
If you want to connect a SIMATIC S7 1200 to an HMI device via PROFIBUS, you must configure a PROFIBUS-capable communication module in a slot of the PLC.

WinCC Runtime as HMI device

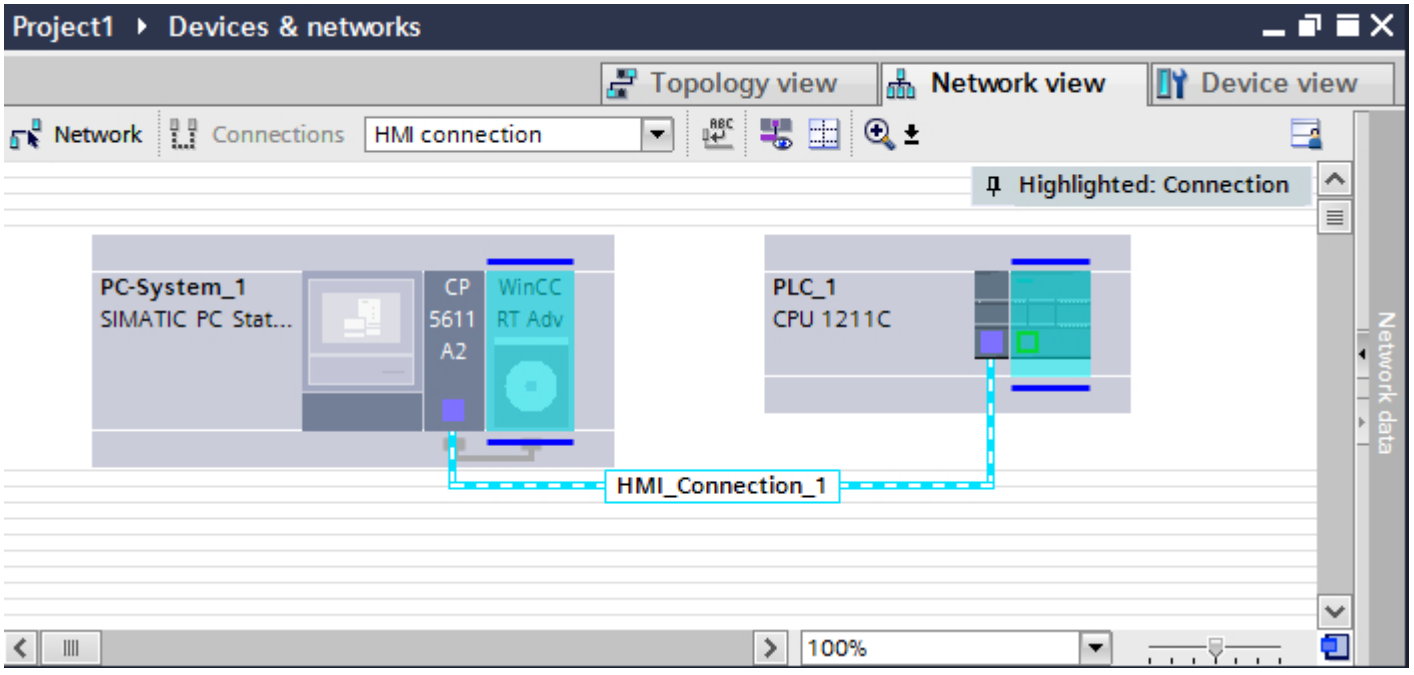
You configure the HMI connections between WinCC Runtime and SIMATIC S7 1200 in the "Devices & Networks" editor.

If you want to use a WinCC Runtime as an HMI device, you have the following options:

1. You create a SIMATIC PC and configure a WinCC Runtime on this SIMATIC PC.
   In this way you use the SIMATIC PC with a WinCC Runtime as an HMI device.



2. You create a WinCC Runtime and configure a communication processor to the Runtime.
   In this way you use your configuration PC with a WinCC Runtime as an HMI device.



You can also connect multiple HMI devices to one SIMATIC S7 1200 and multiple SIMATIC S7 1200s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFIBUS in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS with a SIMATIC PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 1200 via PROFIBUS in the "Devices & Networks" editor.

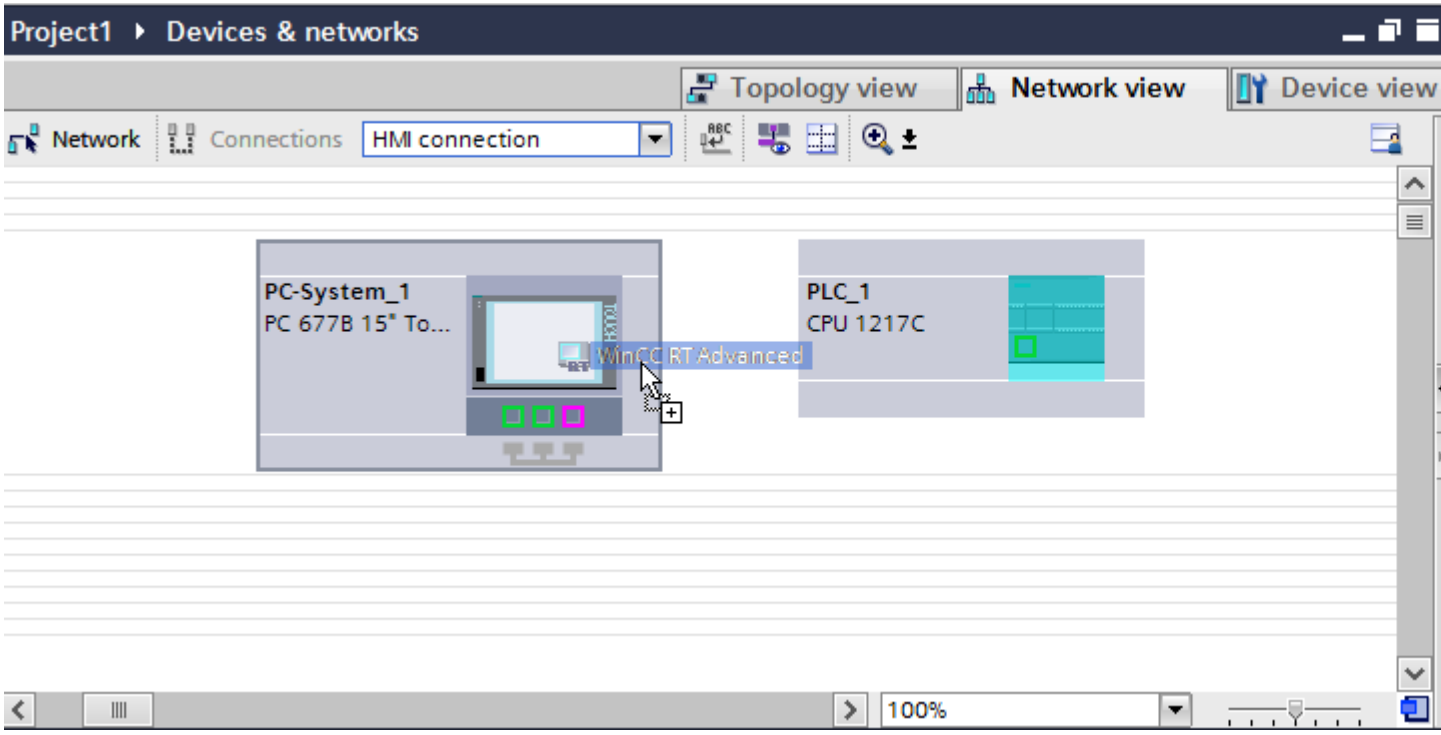
Requirements

The following communication partners are created in the "Devices & Networks" editor:

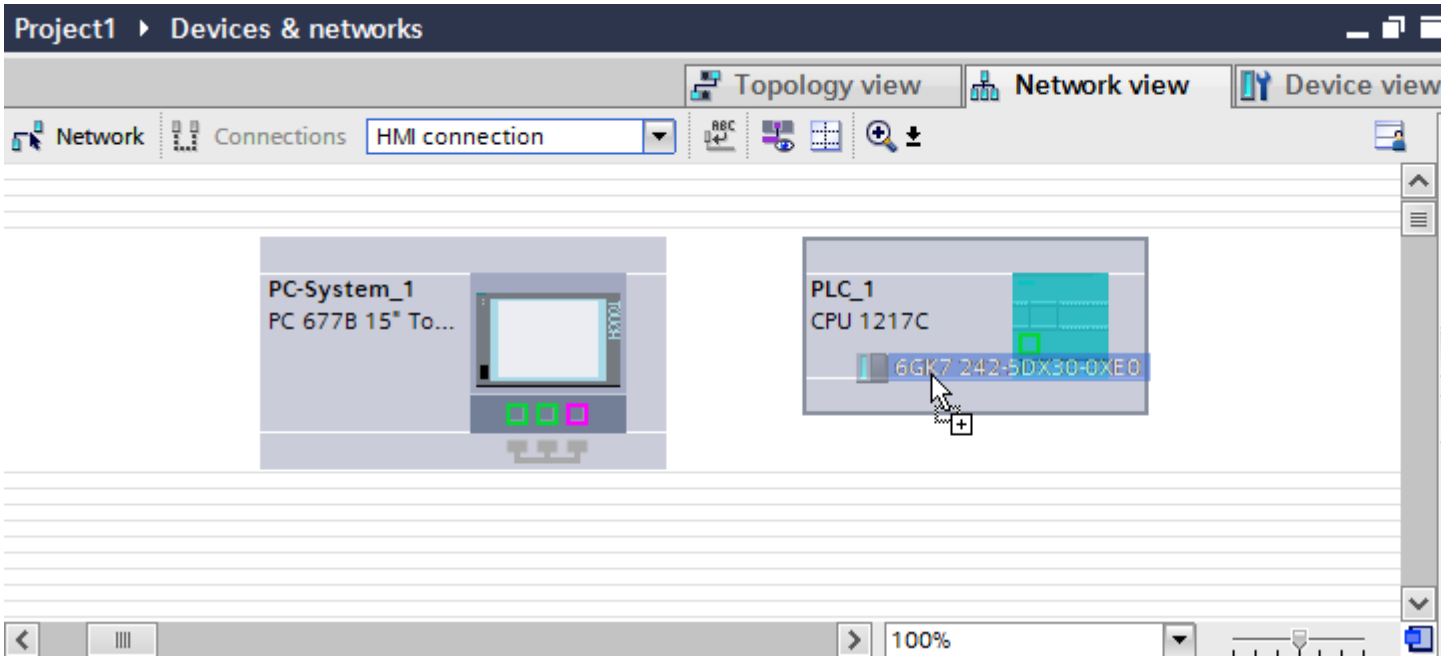
- SIMATIC S7 1200
- SIMATIC PC with PROFIBUS interface

Procedure

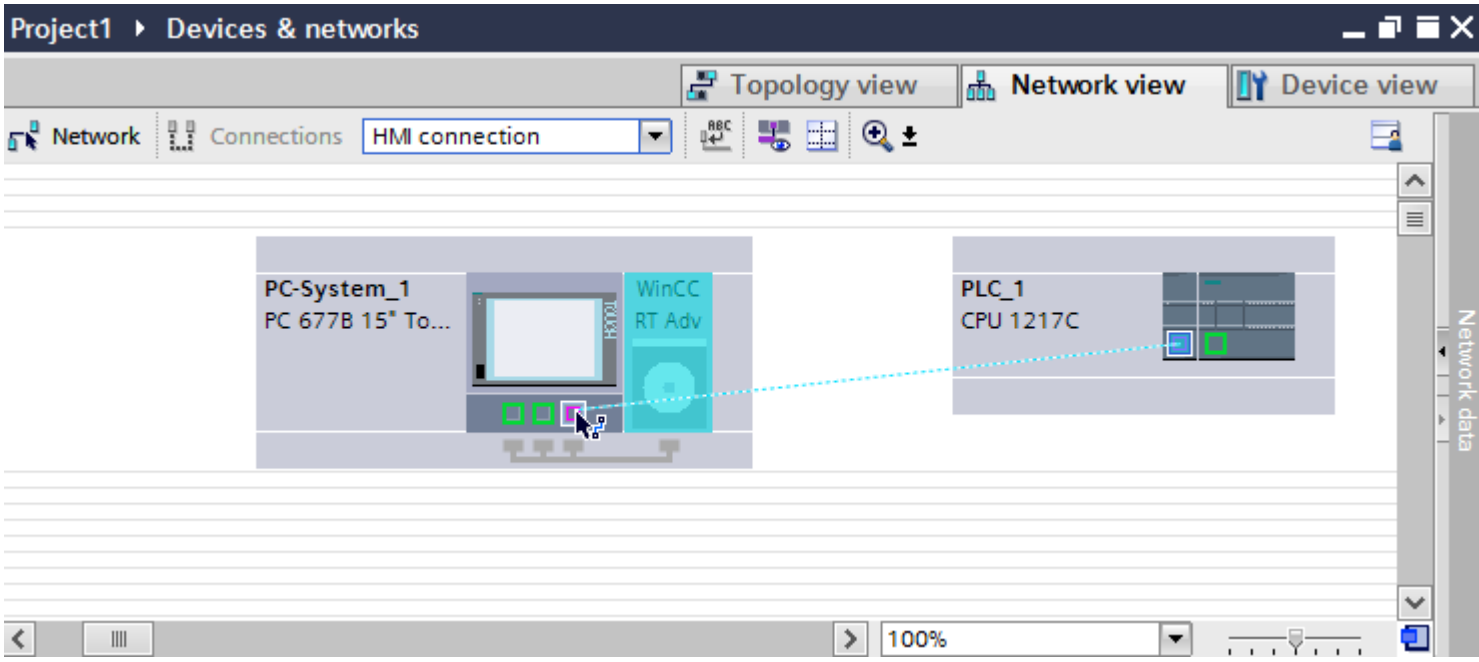
1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a WinCC Runtime from the hardware catalog to the PC.



3. Use a drag-and-drop operation to move a PROFIBUS-capable communication module from the hardware catalog to the PLC.



4. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.
5. Click the PROFIBUS interface of the communication module and use a drag-and-drop operation to draw a connection to the PROFIBUS interface of the PC.



6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFIBUS parameters (Page 6202)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 1200 via PROFIBUS.

Configuring an HMI connection via PROFIBUS with a PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 1200 via PROFIBUS in the "Devices & Networks" editor.

Requirements

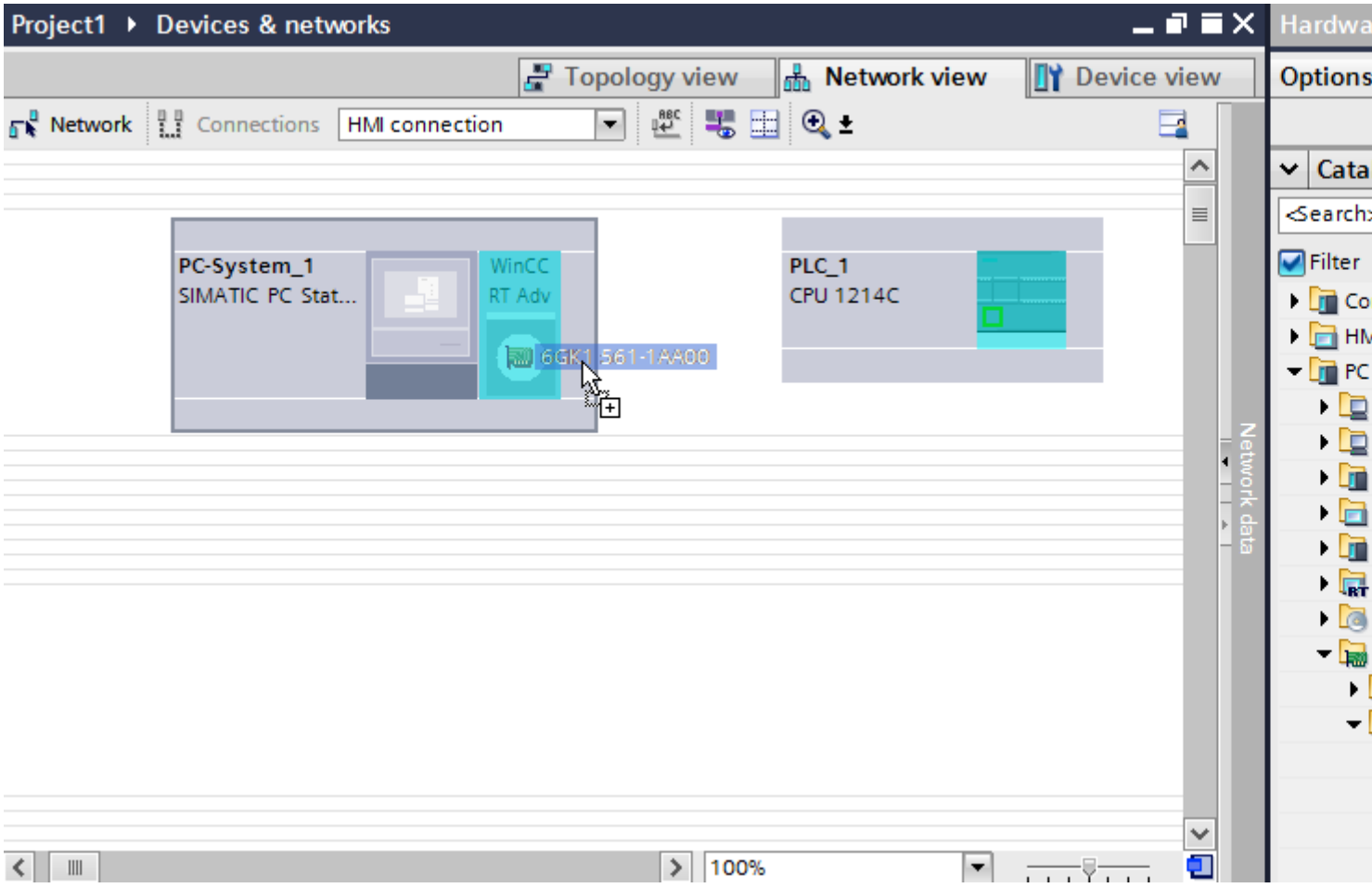
The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 1200
- PC station with WinCC RT Advanced

Procedure

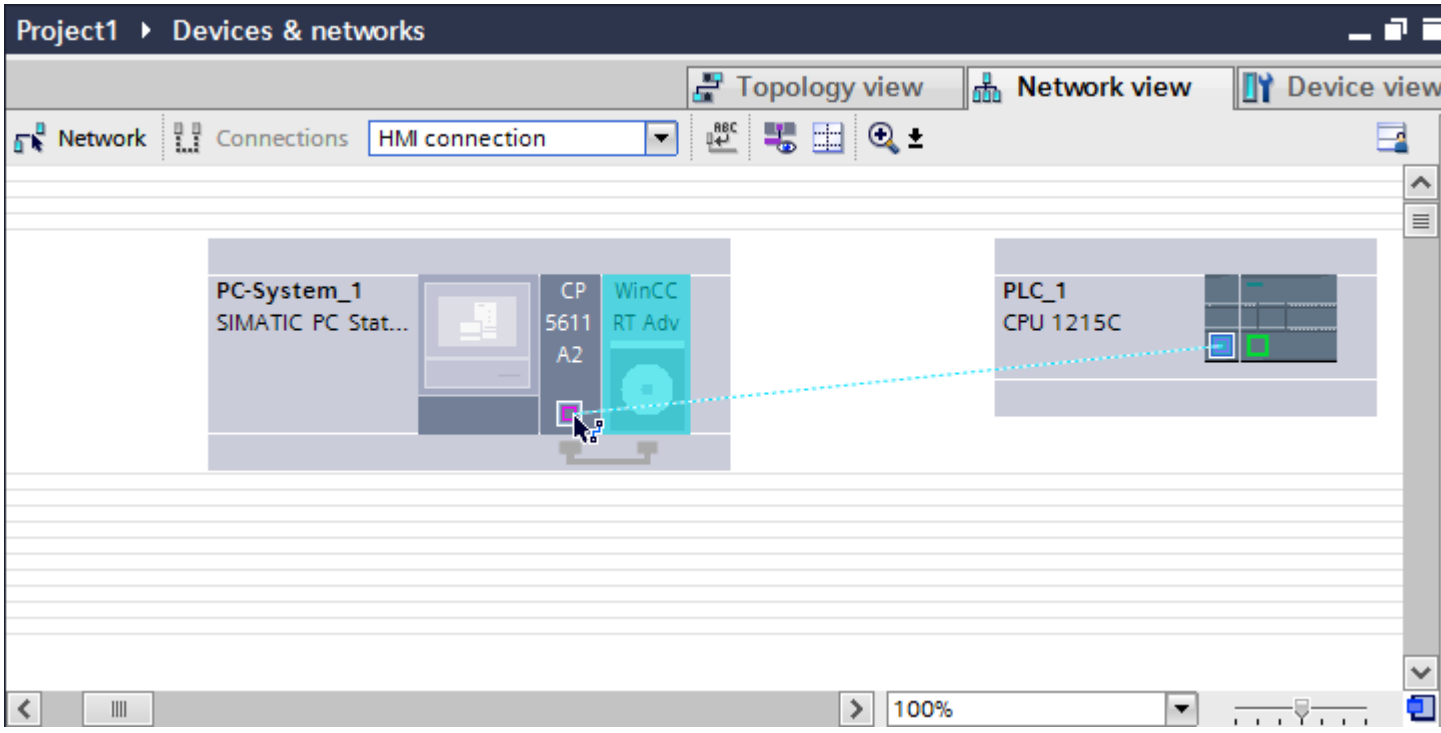
1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a PROFIBUS-capable communication processor from the hardware catalog to the WinCC Runtime.

3. Use a drag-and-drop operation to move a PROFIBUS-capable communication module from the hardware catalog to the PLC.



4. Click the "Connections" button and select "HMI connection" for the connection type. The devices available for connection are highlighted in color.

5. Click the PROFIBUS interface of the communication module and use a drag-and-drop operation to draw a connection to the PROFIBUS interface of the communication processor.



6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFIBUS parameters (Page 6202)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 1200 via PROFIBUS.

PROFIBUS parameters

PROFIBUS parameters for the HMI connection

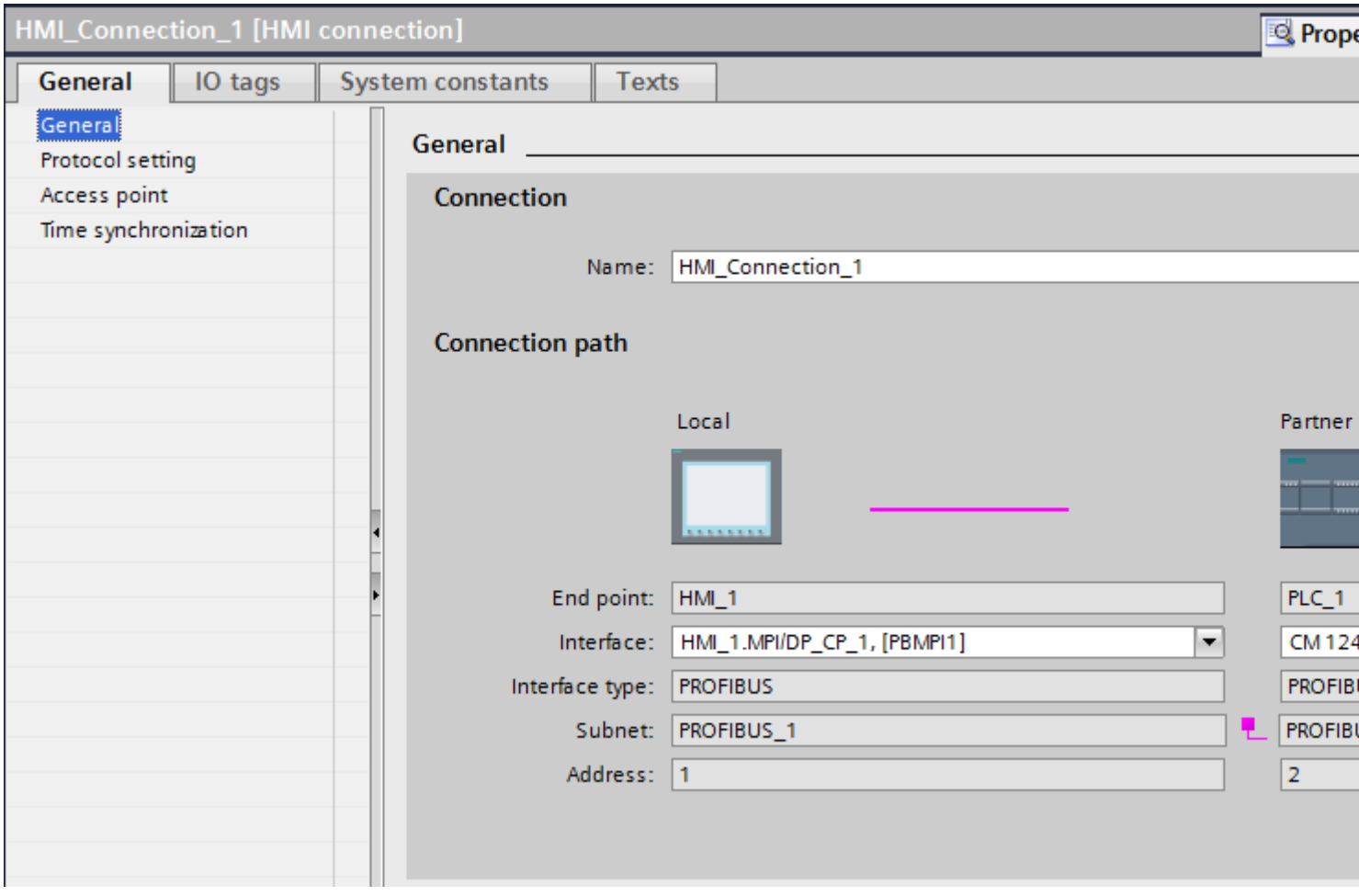
PROFIBUS parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and editing HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

The "Connection" area displays the HMI connection created for communication between the devices.

You can edit the name of the HMI connection in this area.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFIBUS parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"

Displays the selected interface type. This area cannot be edited.

- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the PROFIBUS address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

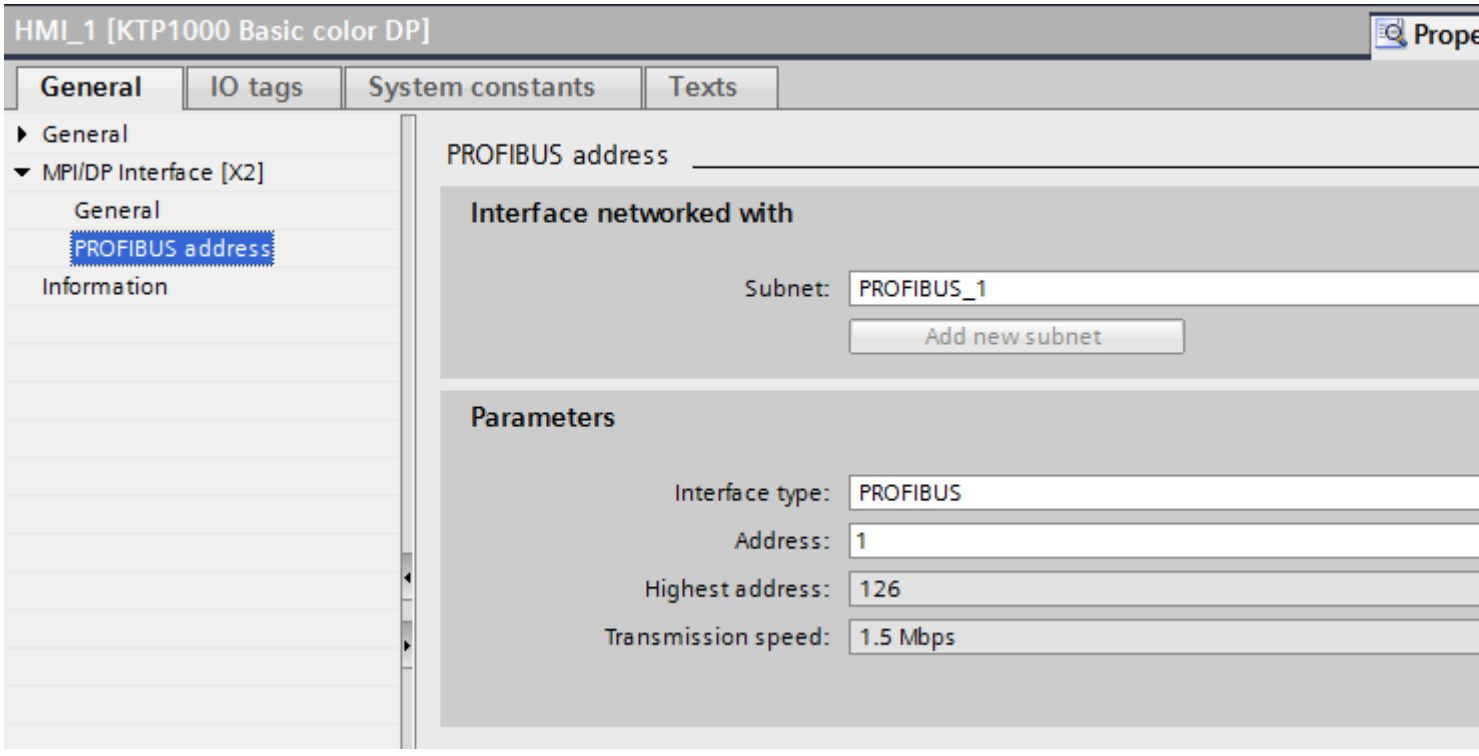
PROFIBUS parameters for the HMI device

PROFIBUS parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFIBUS parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

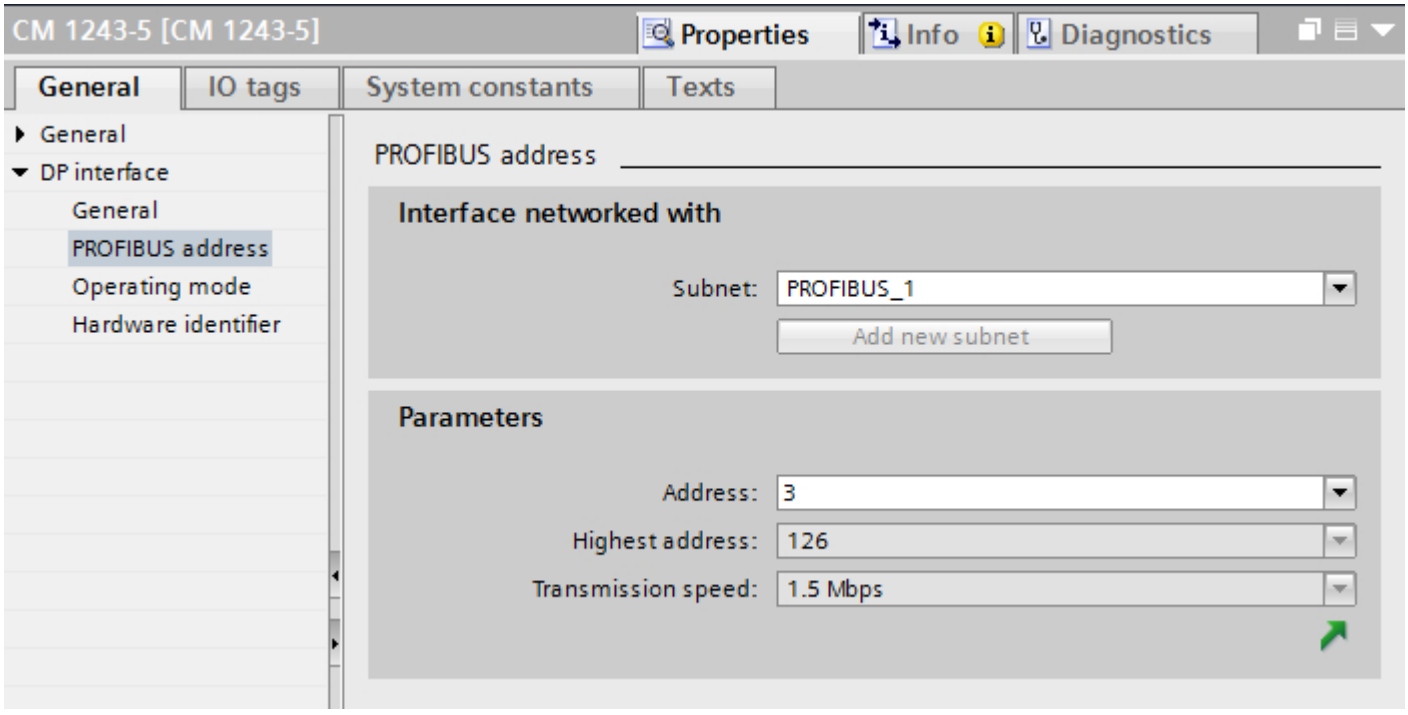
PROFIBUS parameters for the PLC

PROFIBUS parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.

- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Bus profiles with PROFIBUS

Introduction

Depending on the device types connected and protocols used on the PROFIBUS, different profiles are available. The profiles differ in terms of the setting options and calculation of bus parameters. The profiles are explained below.

Devices with different profiles on the same PROFIBUS subnet

The PROFIBUS subnet only functions without problem if the bus parameters of all devices have the same values.

Profiles and transmission rates

Profiles	Supported transmission speeds in Kbits/s
DP	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Standard	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Universal	9,6 19,2 93,75 187,5 500 1500

Meaning of profiles

Profile	Meaning
DP	Select the "DP" bus profile when the only devices connected to the PROFIBUS subnet are those which satisfy the requirements of standard EN 50170 Volume 2/3, Part 8-2 PROFIBUS. The bus parameter setting is optimized on these devices. This includes devices with DP master and DP slave interfaces of the SIMATIC S7 and distributed I/Os of other manufacturers.
Standard	Compared to the "DP" profile, the "Standard" profile also offers scope for devices of another project or devices which have not been configured here to be taken into account when calculating the bus parameters. The bus parameters are then calculated following a simple, non-optimized algorithm.
Universal	Select the "Universal" bus profile when individual devices on the PROFIBUS subnet use the PROFIBUS-FMS service. This includes the following devices for example: • CP 343-5 • PROFIBUS-FMS devices of other manufacturers As with the "Standard" profile, this profile allows you to take other devices into account when calculating the bus parameters.

Protection of communication

Security levels

If you want to protect the PLC and HMI device communication, you can assign protection levels for the communication.

For a SIMATIC S7-1500 CPU, you can enter multiple passwords and thereby set up different access rights for various user groups.

The passwords are entered in a table, so that exactly one protection level is assigned to each password.

The effect of the password is given in the "Protection" column.

For the SIMATIC S7-1200 controller, several aspects need to be considered when setting protection levels. For additional information on this, see: Auto-Hotspot

Example

You select the "Complete protection" protection level for a standard CPU (i.e., not an F-CPU) when configuring it.

Afterwards, you enter a separate password for every protection level above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the assigned passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (no protection) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (write protection) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (read and write protection) allows access as if the CPU were read-protected and write-protected, so that only HMI access is possible for users who know this password.

Access password for the HMI connection

Introduction

Secure access to a PLC by assigning a password.

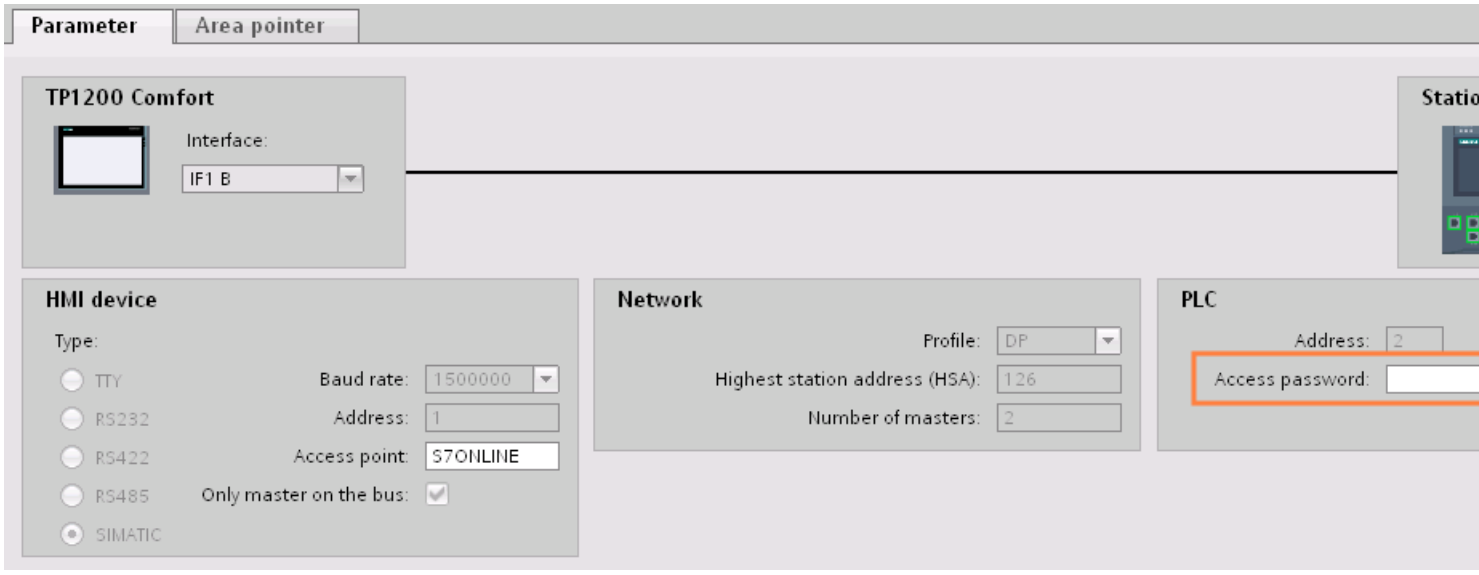
Assign the password when configuring the connection.

Input of the password of the PLC is mandatory as of "Complete protection" security level.

Communication to the PLC is denied if an incorrect password or no password is entered.

Assigning password

Enter the "Access password" for the PLC in the "Connections" editor.



12.11.7.4 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Configuring area pointers (Page 6044)

Area pointer "Date/time"

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Permitted data types

You can use the following data types when you configure the "Date/Time" area pointer:

- Int
- UInt
- Word
- DTL

Use of the "DTL" data type

Use of data type "DTL" with communication driver S7-1200.

A tag of the "DTL" data type has a length of 12 bytes and saves information on date and time in a predefined structure.

The "DTL" data type has the following structure:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2554
1			
2	Month	USINT	0 to 12
3	Day	USINT	1 to 31
4	Day of week	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanoseconds	UDINT	0 to 999 999 999
9			
10			
11			

The "DTL" data type supports time information down to the nanosecond range. Because panels only support time information down to the millisecond, you may encounter the following restriction when using the area pointers:

For the transmission of time information from a panel to the controller, the smallest unit of time is 1 millisecond. The value range from microseconds to nanoseconds of the "DTL" data type is filled with zeros.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the date/time area pointer PLC to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Permitted data types

You can use the following data types when you configure the "Date/Time PLC" area pointer:

- DTL

Use of the "DTL" data type

Use of data type "DTL" with communication driver S7-1200. A tag of the "DTL" data type has a length of 12 bytes and saves information on date and time in a predefined structure.

The "DTL" data type has the following structure:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2554
1			

Byte	Component	Data type	Value range
2	Month	USINT	0 to 12
3	Day	USINT	1 to 31
4	Day of week	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanoseconds	UDINT	0 to 999 999 999
9			
10			
11			

The HMI devices do not support the use of nanoseconds. Values in the nanosecond range will be ignored during processing in Runtime.

The "DTL" data type supports time information down to the nanosecond range. Because panels only support time information down to the millisecond, you may encounter the following restriction when using the area pointers:

For the transmission of time information from a controller to a panel, the range from microseconds to nanoseconds is ignored. The time information is processed on the panel down to milliseconds.

Area pointer "Coordination"

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

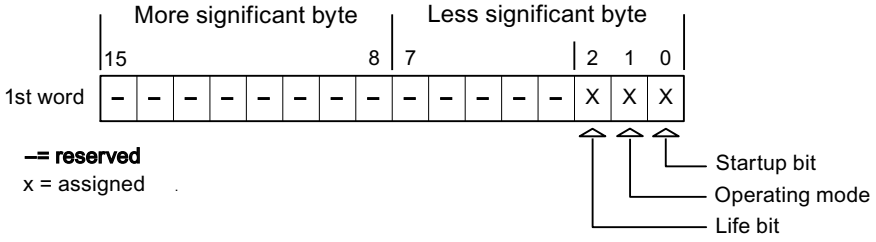
By default, the "Coordination" area pointer has the length of one word and cannot be changed.

Usage

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of the bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

Processing in the PLC

For a simpler evaluation in the PLC program, use a Bool array for this area pointer when using the SIMATIC S7 1200 communication driver. You will have to map the complete 16-bit word of the area pointer. Configure a tag of the data type "Array [0 .. 15] of bool" for this purpose.

Permitted data types

You can use the following data types when you configure the "Coordination" area pointer.

- Word
- UInt
- Bool

Area pointer "Screen number"

Function

The HMI devices store information about the screen called up on the HMI device in the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. Certain reactions can be triggered in the PLC, such as the call of a different screen.

Use

Before the "Screen number" area pointer can be used, it must be set up and activated by selecting "Communication ▸ Area pointer". You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1. Word	Current screen type															
2. Word	Current screen number															
3. Word	Reserved															
4th word	Current field number															
5. Word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

Note

Device dependency

Permanent windows are not available on Basic Panels.

Permitted data types

You can use the following data types when you configure the "Screen number" area pointer.

- Word
- UInt

Area pointer "Project ID"

Function

When Runtime starts, a check can be carried out as to whether the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program.

A missing compatibility results in a corresponding alarm and Runtime will not be started.

Use

Note

HMI connections cannot be switched "online".

The HMI connection in which the "Project ID" area pointer is used must be switched "online".

To use this area pointer, set up the following during the configuration:

- Define the version of the configuration. Values between 1 and 255 are possible.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- This is where you select the PLC tag or the tag array that you have configured as the data area for the area pointer.

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

Permitted data types

You can use the following data types when you configure the "Project ID" area pointer.

- Word
- UInt

Area pointer "Job mailbox"

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	**Set date (BCD-coded)**	

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	**User logon**	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	**User logoff**	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	**Transfer date/time to PLC**	
	An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	**Transfer date/time to PLC**	
	An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	**Update tag**	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in Parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	**Clear event buffer**	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	**Clear error alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	**Display selection**	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	**Reading data record from PLC** [1)]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	**Writing data record to PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1]	Only devices supporting recipes
[2]	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
[3]	The weekday is ignored on HMI device KTP 600 BASIC PN.

Permitted data types

You can use the following data types when you configure the "Screen number" area pointer:

- Word
- UInt

"Data record" area pointer

"Data mailbox" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization over the data mailbox

Data records are always transferred directly. That is, the tag values are read from an address or written to an address configured for this tag directly, without redirecting the values by means of interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a job mailbox, the data in the recipe view will be updated as well. Avoid operating the recipe view while job mailboxes for transfer of data records are being triggered. If you have already started editing a data record and a job mailbox is triggered for transfer of data records, then this job mailbox will be rejected.

Permitted data types

You can use the following data types when you configure the "Data record" area pointer.

- Word
- UInt

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example, when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then process, edit, or save these values, for example.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. Use this mechanism to prevent uncontrolled overwriting of data in either direction of your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Connections ▸ Area pointer" editor.
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data record free
2	0000 0010	Transfer is busy
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer started by the operator in the recipe display

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data mailbox and sets the data record number to 0.	Abort with system alarm.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data mailbox.	Abort with system alarm.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxes No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data mailboxes from the PLC to the HMI device. The job mailbox is structured as follows:

	Most significant byte	Least significant byte
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox no. 70 transfers data mailboxes from the HMI device to the PLC. The job mailbox is structured as follows:

	Most significant byte	Least significant byte
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device reads the values and stores the values in the data record specified in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed". • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the job from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The PLC program can now evaluate the transferred data. To allow further transfers, the PLC program must set the status word to 0 again.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system alarm.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	

Step	Action	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system alarm.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible

- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system alarms
- Triggered by function
  Output of system alarms
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data mailbox.

Note

Availability for specific devices

Notes in the status bar of the recipe view are not available in Basic Panels.

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Displaying tags with Runtime Advanced and Panels (Page 4270)

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

See also

Trends (Page 4270)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

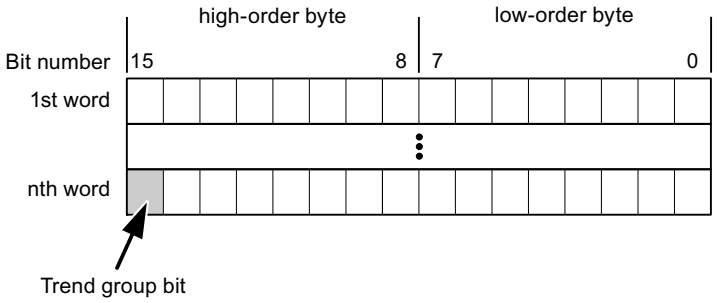
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Permitted data types for trends

For SIMATIC S7

You assign one bit to each trend during configuration. Tags and array tags of the "Word" or "Int" data type are permitted.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with Alarms (Page 4293)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

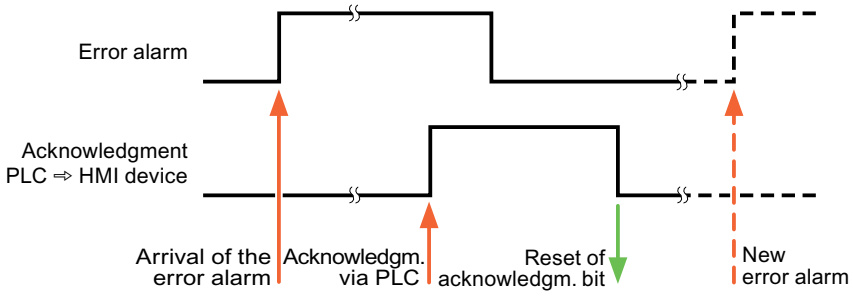
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment

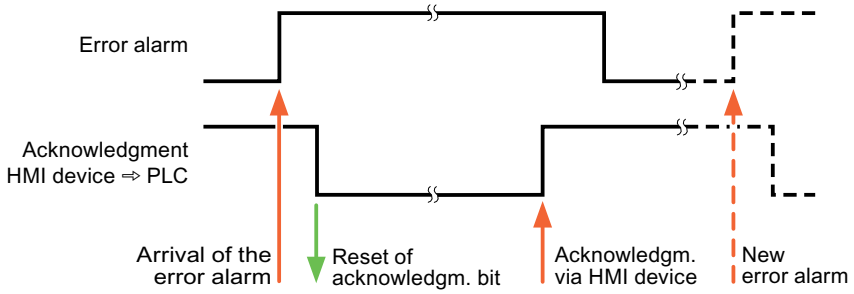
tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



LED mapping

Function

Keyboard devices have LEDs in the function keys. These LEDs can be activated from the PLC. Thus, it is possible, for example, to signal to the user which key he should press in a given situation by lighting up an LED.

Note

The LED function cannot be configured for Basic Panels.

Requirements

In order to activate an LED, an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration.

LED assignment

The assignment of the individual LEDs to the bits in the LED tags is specified when the function keys are configured. In this process, the "LED tag" and the assigned "Bit" are specified for each function key in the "General" group of the properties window.

The "Bit" bit number designates the first of two consecutive bits that control the following LED statuses.

Bit n+ 1	Bit n	LED function	
		All Mobile Panels, all Operator Panels, all Multi Panels, all Comfort Panels	Panel PCs
0	0	Off	Off
0	1	Rapid flashing	Flashing
1	0	Slow flashing	Flashing
1	1	On permanently	On permanently

12.11.7.5 Performance features of communication

S7-1200 device dependency

Communication with the SIMATIC S7-1200 controller

If you use devices from an earlier version of the TIA Portal with TIA Portal V13, it may not be possible to configure integrated connections to certain HMI devices.

Basic Panels V11.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KP300 Basic	Yes	Yes	Yes	Yes	Yes
KP400 Basic	Yes	Yes	Yes	Yes	Yes
KTP400 Basic PN	Yes	Yes	Yes	Yes	Yes
KTP600 Basic DP	Yes	Yes	Yes	Yes	Yes
KTP600 Basic PN	Yes	Yes	Yes	Yes	Yes
KTP1000 Basic DP	Yes	Yes	Yes	Yes	Yes
KTP1000 Basic PN	Yes	Yes	Yes	Yes	Yes
TP1500 Basic PN	Yes	Yes	Yes	Yes	Yes

Basic Panels V12.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KP300 Basic	No	Yes	Yes	Yes	Yes
KP400 Basic	No	Yes	Yes	Yes	Yes
KTP400 Basic PN	No	Yes	Yes	Yes	Yes
KTP600 Basic DP	No	Yes	Yes	Yes	Yes
KTP600 Basic PN	No	Yes	Yes	Yes	Yes

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KTP1000 Basic DP	No	Yes	Yes	Yes	Yes
KTP1000 Basic PN	No	Yes	Yes	Yes	Yes
TP1500 Basic PN	No	Yes	Yes	Yes	Yes

Basic Panels V13.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KTP400 Basic PN	No	Yes	Yes	Yes	Yes
KTP700 Basic PN/ DP	No	Yes	Yes	Yes	Yes
KTP900 Basic PN	No	Yes	Yes	Yes	Yes
KTP1200 Basic PN/DP	No	Yes	Yes	Yes	Yes

Basic Panels V13.0.1

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KTP400 Basic PN	No	Yes	Yes	Yes	Yes
KTP700 Basic PN/ DP	No	Yes	Yes	Yes	Yes
KTP900 Basic PN	No	Yes	Yes	Yes	Yes
KTP1200 Basic PN/DP	No	Yes	Yes	Yes	Yes

Panels V11.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
OP 73	No	Yes	Yes	Yes	Yes
OP 77A	No	Yes	Yes	Yes	Yes
OP 77B	No	Yes	Yes	Yes	Yes
TP 177A	No	Yes	Yes	Yes	Yes
TP 177A Portrait	No	Yes	Yes	Yes	Yes
TP 177B 4"	No	Yes	Yes	Yes	No
TP 177B 6" mono	No	Yes	Yes	Yes	Yes

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
TP 177B 6"	No	Yes	Yes	Yes	Yes
OP 177B 6" mono	No	Yes	Yes	Yes	Yes
OP 177B 6"	No	Yes	Yes	Yes	Yes
TP 277 6"	No	Yes	Yes	Yes	Yes
OP 277 6"	No	Yes	Yes	Yes	Yes

Multi Panels V11.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
MP 177 6" Touch	No	Yes	Yes	Yes	No
MP 277 8" Key	No	Yes	Yes	Yes	No
MP 277 10" Key	No	Yes	Yes	Yes	No
MP 277 10" Touch	No	Yes	Yes	Yes	No
MP 377 12" Key	No	Yes	Yes	Yes	No
MP 377 12" Touch	No	Yes	Yes	Yes	No
MP 377 15" Touch	No	Yes	Yes	Yes	No
MP 377 19" Touch	No	Yes	Yes	Yes	No

Multi Panels V12.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
MP 177 6" Touch	No	Yes	Yes	Yes	Yes
MP 277 8" Key	No	Yes	Yes	Yes	Yes
MP 277 10" Key	No	Yes	Yes	Yes	Yes
MP 277 10" Touch	No	Yes	Yes	Yes	Yes
MP 377 12" Key	No	Yes	Yes	Yes	Yes
MP 377 12" Touch	No	Yes	Yes	Yes	Yes
MP 377 15" Touch	No	Yes	Yes	Yes	Yes
MP 377 19" Touch	No	Yes	Yes	Yes	Yes

Mobile Panels V11.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
Mobile Panel 177 6" DP	No	Yes	Yes	Yes	Yes
Mobile Panel 177 6" PN	No	Yes	Yes	Yes	Yes
Mobile Panel 277 8"	No	Yes	Yes	Yes	Yes

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
Mobile Panel 277 8" IWLAN V2	No	Yes	Yes	Yes	Yes
Mobile Panel 277F 8" IWLAN V2	No	No	No	No	No
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	No	No	No	No	No
Mobile Panel 277 10"	No	Yes	Yes	Yes	Yes

Mobile Panels V12.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
Mobile Panel 177 6" DP	No	Yes	Yes	Yes	Yes
Mobile Panel 177 6" PN	No	Yes	Yes	Yes	Yes
Mobile Panel 277 8"	No	Yes	Yes	Yes	Yes
Mobile Panel 277 8" IWLAN V2	No	Yes	Yes	Yes	Yes
Mobile Panel 277F 8" IWLAN V2	No	Yes [1]	Yes [1]	Yes [1]	Yes [1]
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	No	Yes [1]	Yes [1]	Yes [1]	Yes [1]
Mobile Panel 277 10"	No	Yes	Yes	Yes	Yes

[1] Not as master system

Mobile Panels V13.0.1

HMI devices	SIMATIC ET 200 CPU
KTP 700 Mobile	Yes
KTP 900 Mobile	Yes

Comfort Panels V11.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KP400 Comfort	No	Yes	Yes	Yes	Yes
KTP400 Comfort	No	Yes	Yes	Yes	Yes
KTP400 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP700 Comfort	No	Yes	Yes	Yes	Yes
TP700 Comfort	No	Yes	Yes	Yes	Yes
TP700 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP900 Comfort	No	Yes	Yes	Yes	Yes

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
TP900 Comfort	No	Yes	Yes	Yes	Yes
TP900 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP1200 Comfort	No	Yes	Yes	Yes	Yes
TP1200 Comfort	No	Yes	Yes	Yes	Yes
TP1200 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP1500 Comfort	No	Yes	Yes	Yes	Yes
TP1500 Comfort	No	Yes	Yes	Yes	Yes
TP1500 Comfort Portrait	No	Yes	Yes	Yes	Yes
TP1900 Comfort	No	Yes	Yes	Yes	Yes
TP1900 Comfort Portrait	No	Yes	Yes	Yes	Yes
TP2200 Comfort	No	Yes	Yes	Yes	Yes
TP2200 Comfort Portrait	No	Yes	Yes	Yes	Yes

Comfort Panels V12.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KP400 Comfort	No	Yes	Yes	Yes	Yes
KTP400 Comfort	No	Yes	Yes	Yes	Yes
KTP400 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP700 Comfort	No	Yes	Yes	Yes	Yes
TP700 Comfort	No	Yes	Yes	Yes	Yes
TP700 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP900 Comfort	No	Yes	Yes	Yes	Yes
TP900 Comfort	No	Yes	Yes	Yes	Yes
TP900 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP1200 Comfort	No	Yes	Yes	Yes	Yes
TP1200 Comfort	No	Yes	Yes	Yes	Yes
TP1200 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP1500 Comfort	No	Yes	Yes	Yes	Yes
TP1500 Comfort	No	Yes	Yes	Yes	Yes
TP1500 Comfort Portrait	No	Yes	Yes	Yes	Yes
TP1900 Comfort	No	Yes	Yes	Yes	Yes
TP1900 Comfort Portrait	No	Yes	Yes	Yes	Yes
TP2200 Comfort	No	Yes	Yes	Yes	Yes
TP2200 Comfort Portrait	No	Yes	Yes	Yes	Yes

Comfort Panels V13.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KP400 Comfort	No	Yes	Yes	Yes	Yes
KTP400 Comfort	No	Yes	Yes	Yes	Yes
KTP400 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP700 Comfort	No	Yes	Yes	Yes	Yes
TP700 Comfort	No	Yes	Yes	Yes	Yes
TP700 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP900 Comfort	No	Yes	Yes	Yes	Yes
TP900 Comfort	No	Yes	Yes	Yes	Yes
TP900 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP1200 Comfort	No	Yes	Yes	Yes	Yes
TP1200 Comfort	No	Yes	Yes	Yes	Yes
TP1200 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP1500 Comfort	No	Yes	Yes	Yes	Yes
TP1500 Comfort	No	Yes	Yes	Yes	Yes
TP1500 Comfort Portrait	No	Yes	Yes	Yes	Yes
TP1900 Comfort	No	Yes	Yes	Yes	Yes
TP1900 Comfort Portrait	No	Yes	Yes	Yes	Yes
TP2200 Comfort	No	Yes	Yes	Yes	Yes
TP2200 Comfort Portrait	No	Yes	Yes	Yes	Yes

Comfort Panels V13.0.1

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KP400 Comfort	No	Yes	Yes	Yes	Yes
KTP400 Comfort	No	Yes	Yes	Yes	Yes
KTP400 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP700 Comfort	No	Yes	Yes	Yes	Yes
TP700 Comfort	No	Yes	Yes	Yes	Yes
TP700 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP900 Comfort	No	Yes	Yes	Yes	Yes
TP900 Comfort	No	Yes	Yes	Yes	Yes
TP900 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP1200 Comfort	No	Yes	Yes	Yes	Yes
TP1200 Comfort	No	Yes	Yes	Yes	Yes
TP1200 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP1500 Comfort	No	Yes	Yes	Yes	Yes

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
TP1500 Comfort	No	Yes	Yes	Yes	Yes
TP1500 Comfort Portrait	No	Yes	Yes	Yes	Yes
TP1900 Comfort	No	Yes	Yes	Yes	Yes
TP1900 Comfort Portrait	No	Yes	Yes	Yes	Yes
TP2200 Comfort	No	Yes	Yes	Yes	Yes
TP2200 Comfort Portrait	No	Yes	Yes	Yes	Yes

Runtime V11.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
WinCC RT Advanced	No	Yes	Yes	Yes	No

Runtime V12.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
WinCC RT Advanced	No	Yes	Yes	Yes	Yes

Runtime V13.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
WinCC RT Advanced	No	Yes	Yes	Yes	Yes

Runtime V13.0.1

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
WinCC RT Advanced	No	Yes	Yes	Yes	Yes

Valid data types for SIMATIC S7-1200

Valid data types for connections with SIMATIC S7-1200

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
BOOL	1 bit
SINT	1 byte

Data type	Length	
INT	2 bytes	
DINT	4 bytes	
USINT	1 byte	
UINT	2 bytes	
UDINT	4 bytes	
REAL	4 bytes	
LREAL	8 bytes	
TIME	4 bytes	
DATE	2 bytes	
DTL	12 bytes	Basic Panels, Panels, RT Advanced
	8 bytes	RT Professional
TIME_OF_DAY, TOD	4 bytes	
STRING	(2+n) bytes, n = 0 to 254	
WSTRING	(4+2*n) bytes, n = 0 to 254	Basic Panels
	(4+2*n) bytes, n = 0 to 4094	Panels, RT Advanced
	(4+2*n) bytes, n = 0 to 65534	RT Professional
CHAR	1 byte	
Array of CHAR	--	
BYTE	1 byte	
WORD	2 bytes	
DWORD	4 bytes	
LDT	8 bytes	RT Professional
DATE_AND_TIME	8 bytes	RT Professional

12.11.7.6 Creating connections in the "Connections" editor

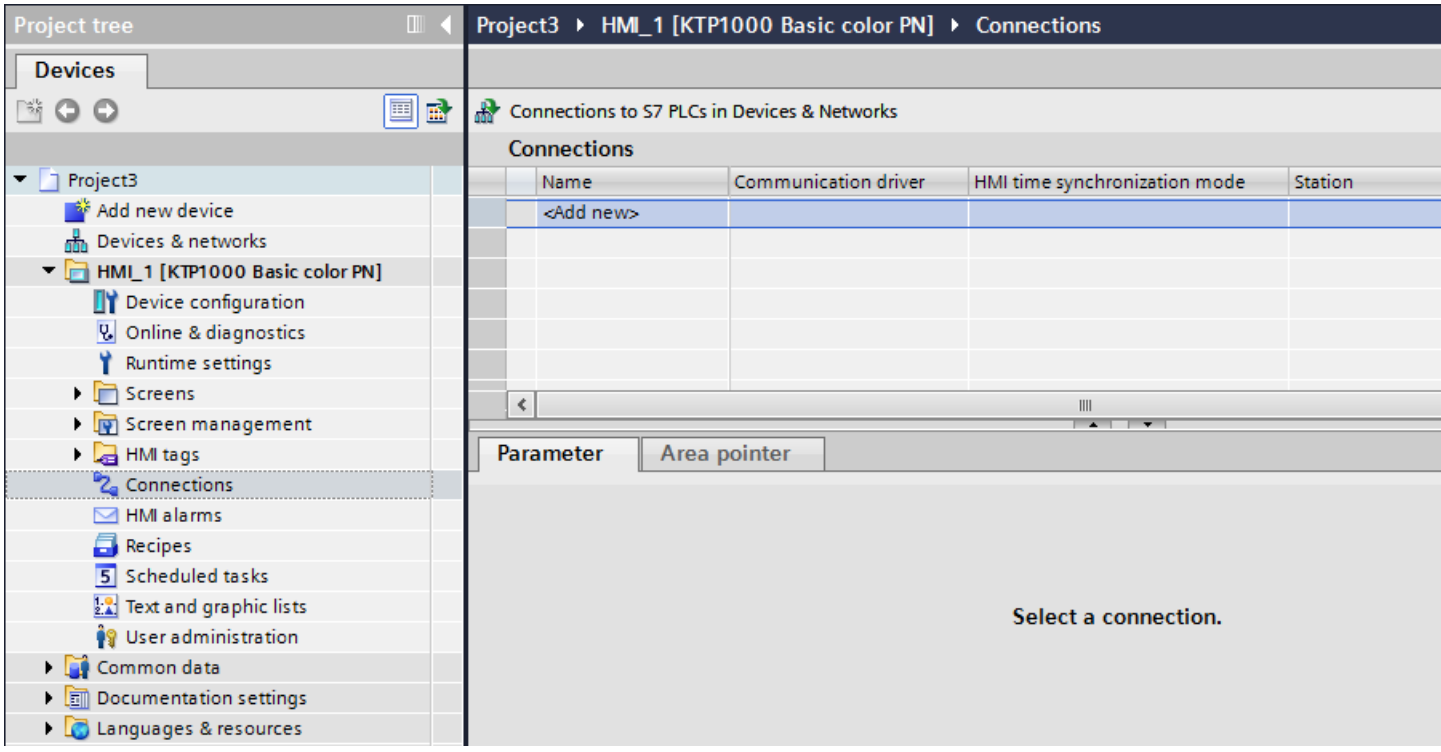
Creating a PROFINET connection

Requirements

- A project is open.
- An HMI device with a PROFINET interface has been created.

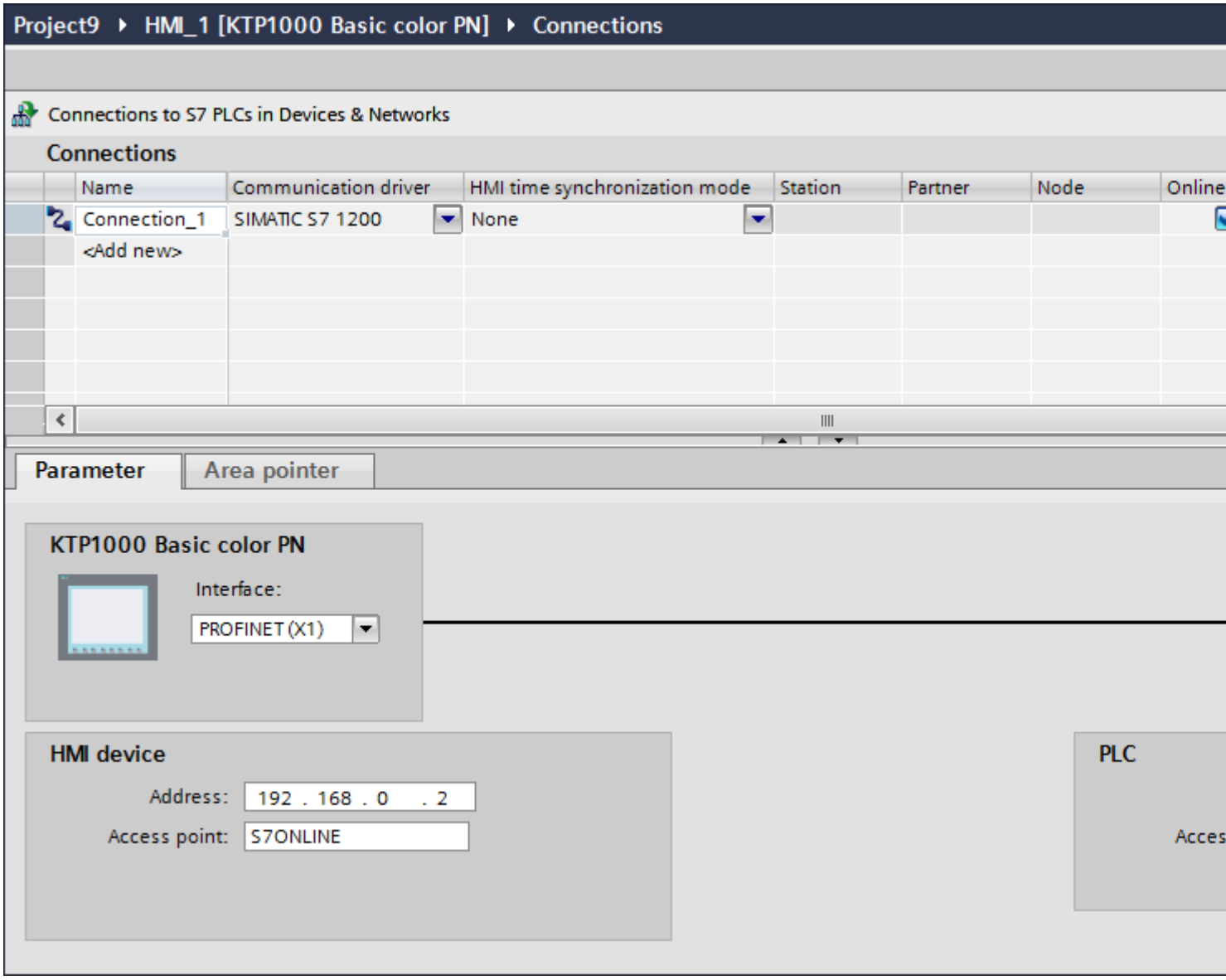
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. In the "Communication drivers" column, select the "SIMATIC S7 1200" driver.
4. Click the name of the connection.

5. Select a PROFINET interface of the HMI device in the Inspector window under "Parameters > Interface".



6. Set the IP addresses of the communication partners in the Inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

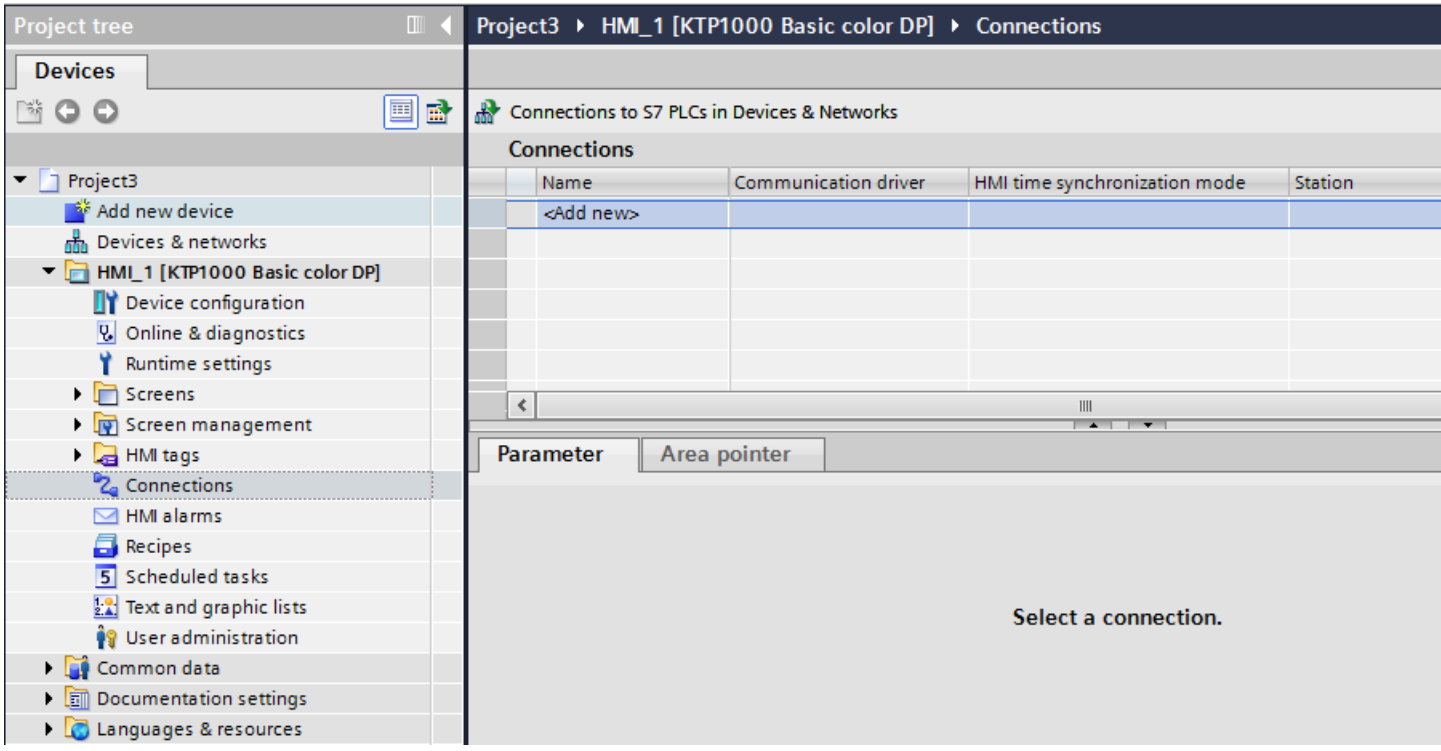
Creating a PROFIBUS DP connection

Requirements

- A project is open.
- An HMI device with a PROFIBUS interface has been created.

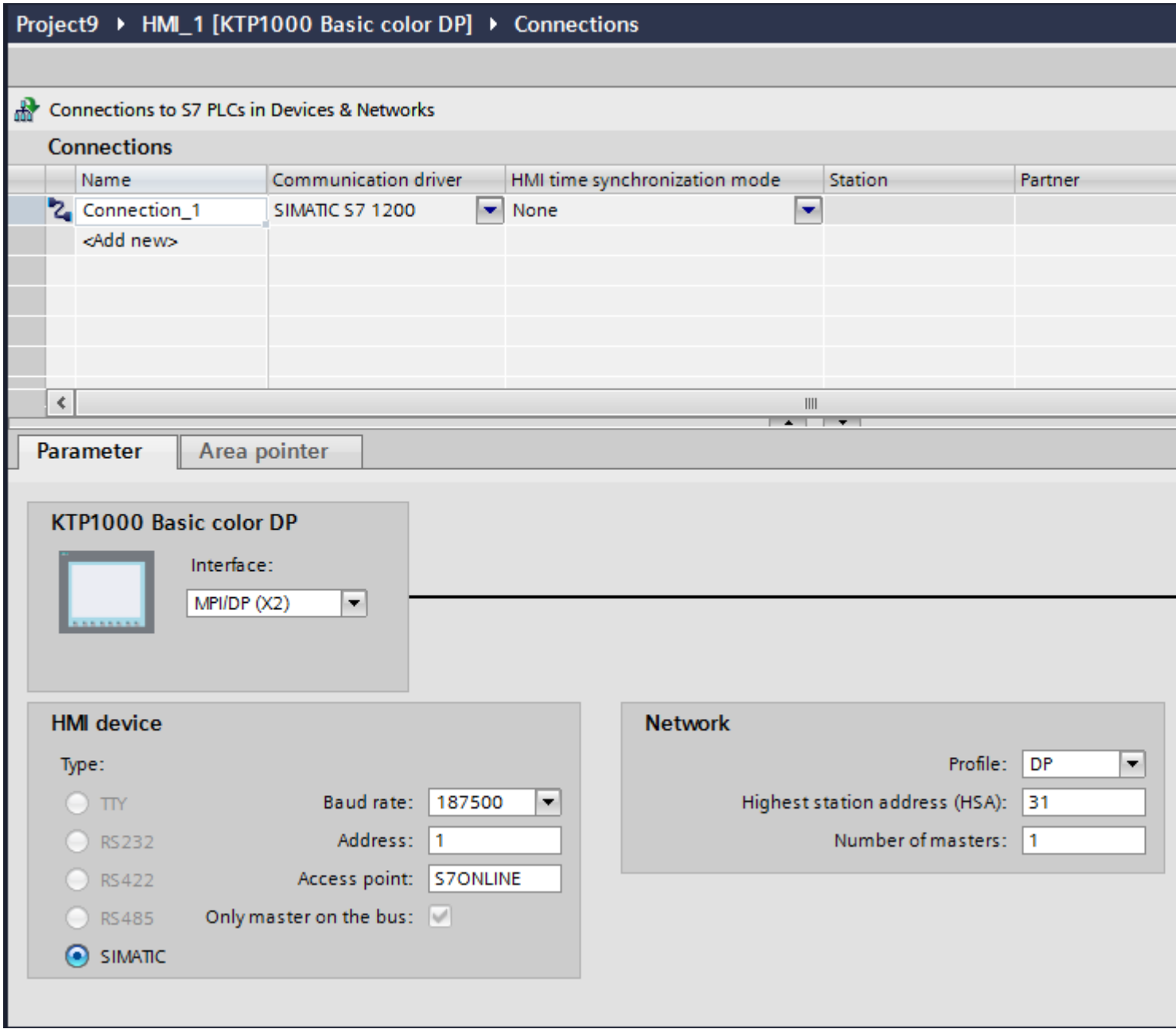
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. In the "Communication drivers" column, select the "SIMATIC S7 1200" driver.
4. Click the name of the connection.
5. Select the "MPI/DP" interface in the Inspector window under "Parameters".

6. Select the "DP" profile in the Inspector window under "Parameters > Network".



7. Set the addresses of the communication partners in the Inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

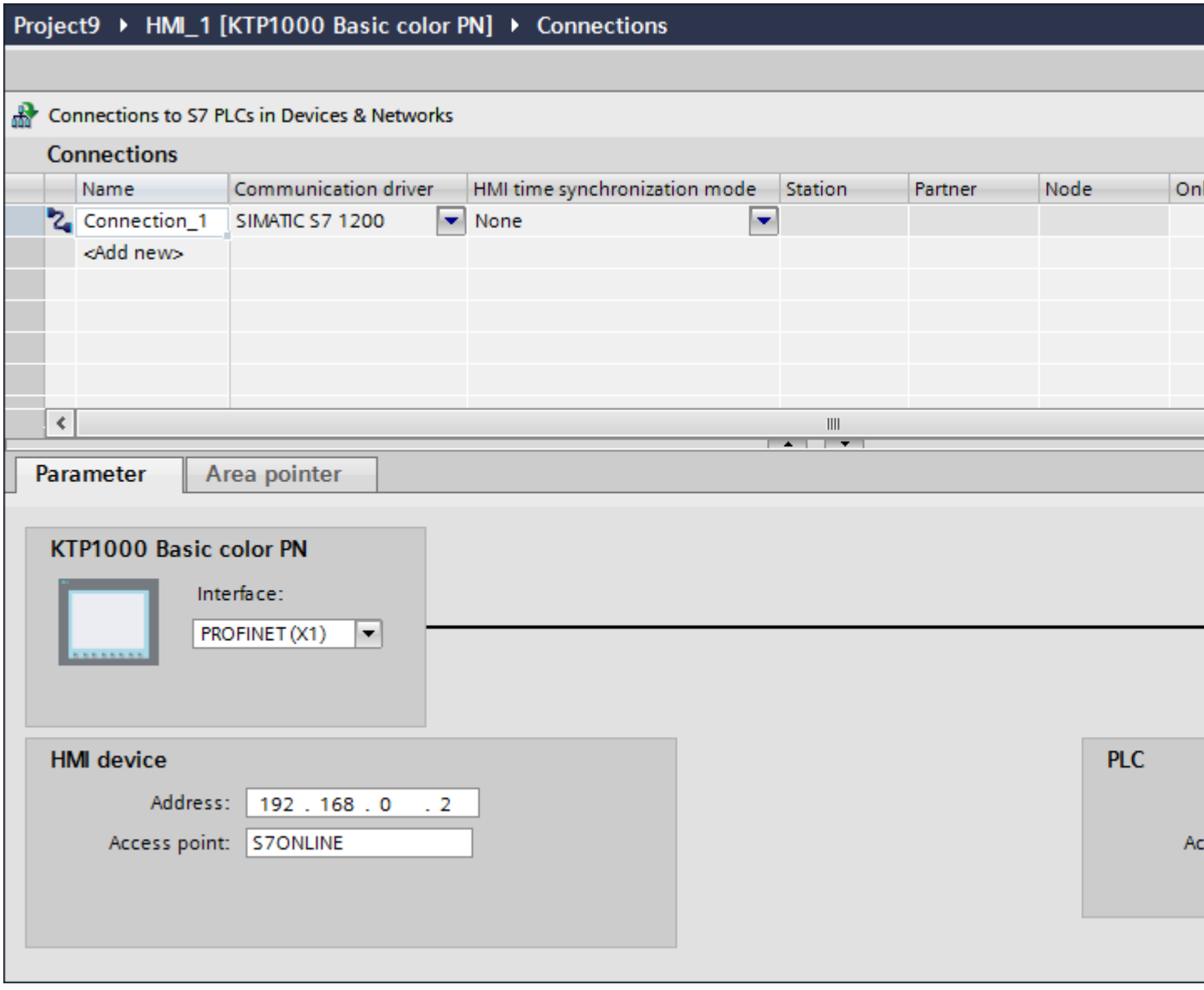
Parameters for the connection

Parameters for the connection (SIMATIC S7 1200)

Parameters to be set

To assign the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"
- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.
- "Access point"
  Specifies the access point for the PG/PC interface that can be used to reach the communication partner.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the PLC.

Note

You only need a password if you have set "Complete protection" at the PLC.

No connection is set up to the PLC if the "Complete protection" security level is stored on the PLC and you do not enter a password.

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

Note

If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.
- "Only master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).
  In S7-200, you must set an HMI device as the master.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"
  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the PLC.

  Note

  You only need a password if you have set "Complete protection" at the PLC.

  No connection is set up to the PLC if the "Complete protection" security level is stored on the PLC and you do not enter a password.

12.11.7.7 Time synchronization

Time synchronization

Introduction

To have the same time of day throughout the plant, you can synchronize the time on various plant components using time synchronization. WinCC time synchronization is operated as a master-slave system.

One system component must be a clock for all components of a plant to work with identical time. The component functioning as the clock is referred to as the time master. The components that receive the time are time slaves.

Properties of time synchronization

- The HMI device can define the time as master or can accept the time of the PLC as slave.
- In "master mode", the time is synchronized at each connection setup.
- In "slave mode", the time is synchronized at each connection setup and than at cyclic intervals of 10 minutes.
- The first time synchronization is performed on the HMI device immediately after the start of runtime.
- Time synchronization is only performed on the HMI device during operation of runtime.

Time synchronization restrictions

Approved HMI devices

You can configure the time synchronization between a SIMATIC S7-1200 or SIMATIC S7-1500 and an HMI device with the following HMI devices:

Device	Operating system
Basic Panels	-
TP177 4"	Windows CE 5.0
Multi Panel 177	Windows CE 5.0
Multi Panel 277	Windows CE 5.0
Multi Panel 377	Windows CE 5.0
Mobile Panel 277	Windows CE 5.0
Mobile 277 IWLAN V2	Windows CE 5.0
Comfort Panels	Windows CE 6.0
PC systems with WinCC RT Advanced	Microsoft Windows XP Microsoft Windows 7

Configuration restrictions

- If an HMI device has several connections to SIMATIC S7-1200 or SIMATIC S7-1500, you can only configure one connection as "slave".
- If you have enabled time synchronization for the HMI device as "slave", you can no longer use the global area pointer "Date/time PLC".

- An HMI device can only request the time from a PLC with "Complete protection" security type configuration if the correct "Access password" is configured.
  Configure the "Access password" for communication with a PLC with "Complete protection" security type in the "Connections" editor of the HMI device.
  This "Access password" must match the password configured on the PLC. The PLC password is assigned in the PLC properties at: "General > Security"
- Basic Panels can only be configured as "Slave".
- If you use Basic Panels for the configuration, it is not possible to use time synchronization via NTP and the "Date/time PLC" area pointer simultaneously.
- Time synchronization with SIMATIC S7-1200 (V1.0) controllers is not possible.
- Time synchronization between the HMI device TP177 4" and SIMATIC S7-1200 (V4.0) controllers is not possible.
- Time synchronization between the HMI device TP177 4" and SIMATIC S7-1500 is not possible.

Configuring time synchronization for integrated connections

Introduction

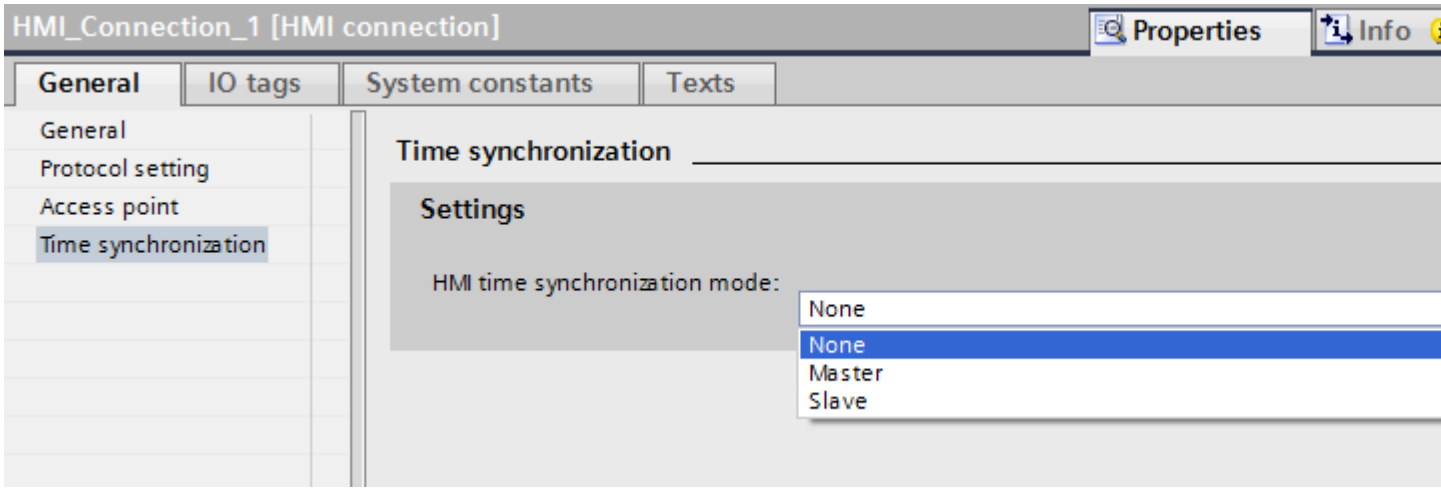
You configure time synchronization for an integrated connection in the "Devices & Networks" editor.

Requirements

- An HMI connection between an HMI device and a SIMATIC S7 1200 has been configured.
- The HMI device must support the "time synchronization" function.
- The "Devices & Networks" editor is open.

Procedure

1. Click the line of the HMI connection in the "Devices & networks" editor.
2. Select the following in the inspector window under "General > Time synchronization > Settings":
   - None: No time synchronization is used.
   - Master: The HMI device sets the time.
   - Slave: The PLC sets the time.



Configuring time synchronization for non-integrated connections

Introduction

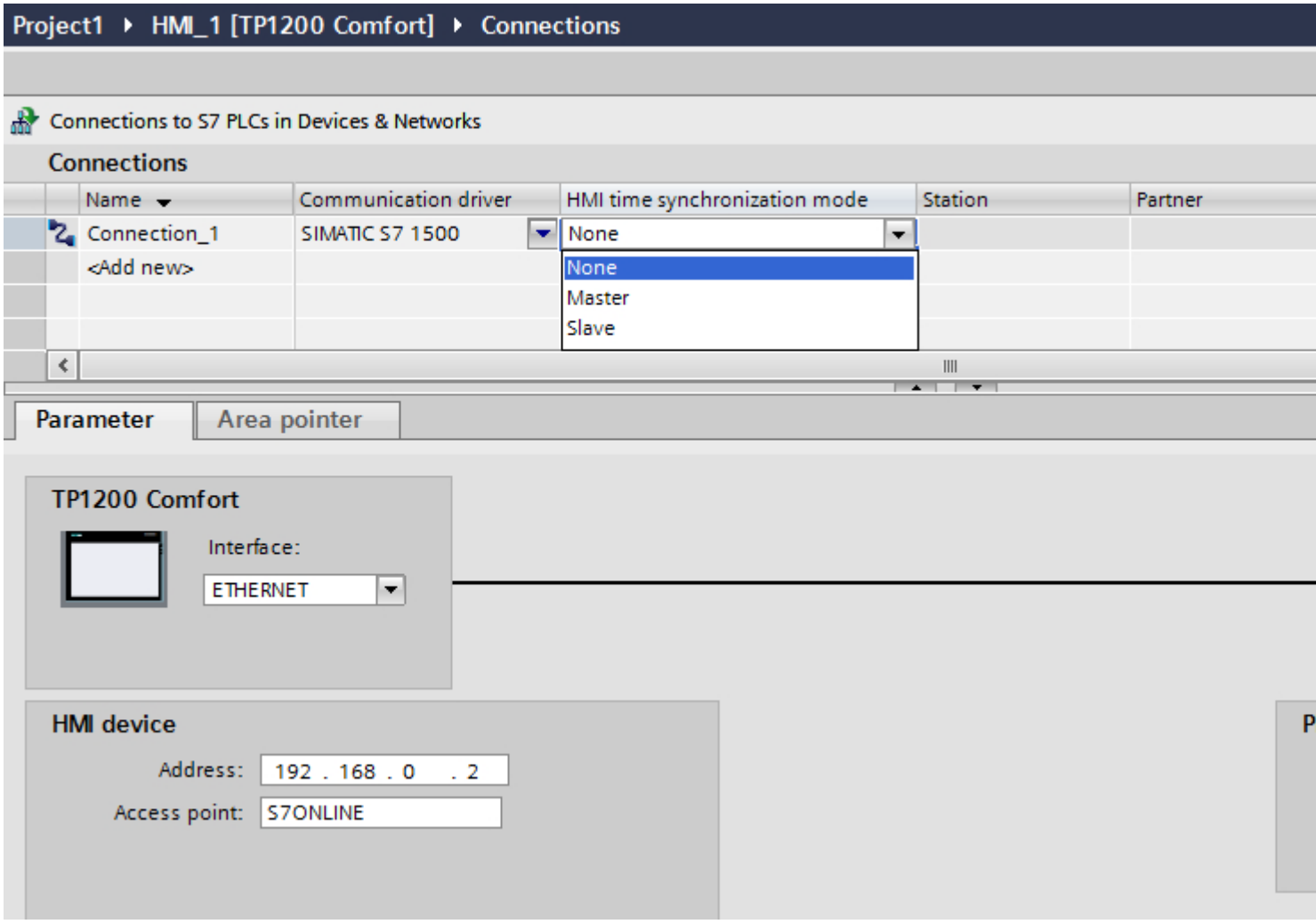
You configure time synchronization for a non-integrated connection in the "Connections" editor.

Requirements

- An HMI device which supports the "time synchronization" function has been created.
- "Connections" editor is open.

Procedure

1. Double-click "<Add>".
2. In the "Communication drivers" column, select the "SIMATIC S7 1200" PLC.
3. Select the following in the "HMI time synchronization mode" column:
   - None: No time synchronization is used.
   - Master: The HMI device sets the time.
   - Slave: The PLC sets the time.



12.11.8 Communicating with SIMATIC S7 300/400

12.11.8.1 Communication with SIMATIC S7 300/400

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 300 and S7 400 PLCs. These two PLCs will be referred to jointly as SIMATIC S7 300/400.

You can configure the following communication channels for the SIMATIC S7 300/400 PLC:

- PROFINET
- PROFIBUS
- MPI

HMI connection for communication

You configure connections between the HMI device and a SIMATIC S7 300/400 in the "Devices & Networks" editor.

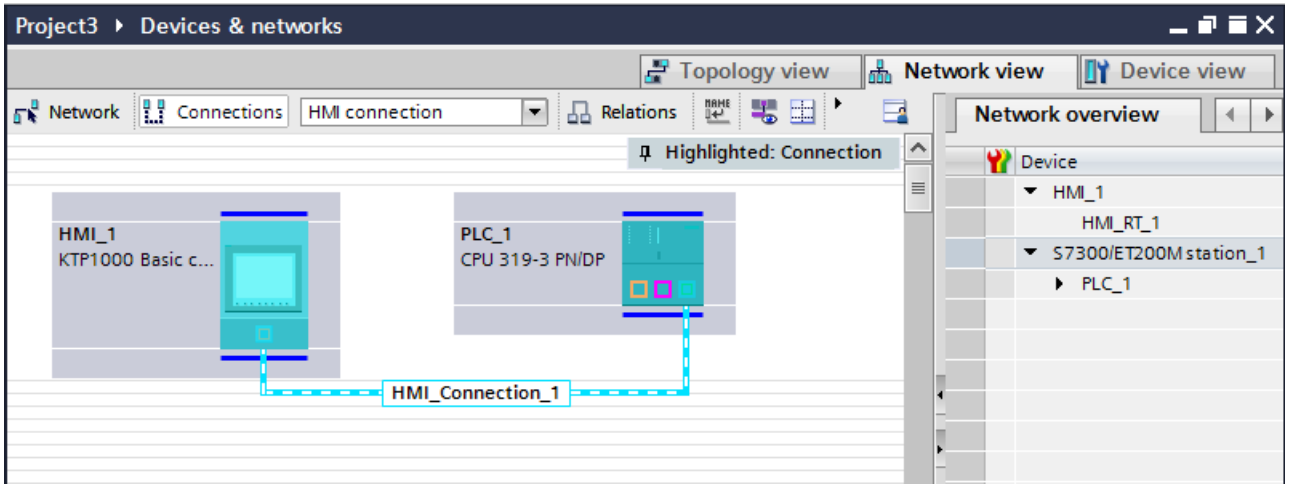
12.11.8.2 Communication via PROFINET

Configuring an HMI connection

Communication via PROFINET

HMI connections via PROFINET

If you have inserted an HMI device and a SIMATIC S7 300/400 into the project, you interconnect the two PROFINET interfaces in the "Devices & Networks" editor.



You can also connect multiple HMI devices to one SIMATIC S7 300/400 and multiple SIMATIC S7 300/400s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFINET or Ethernet in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

Requirements

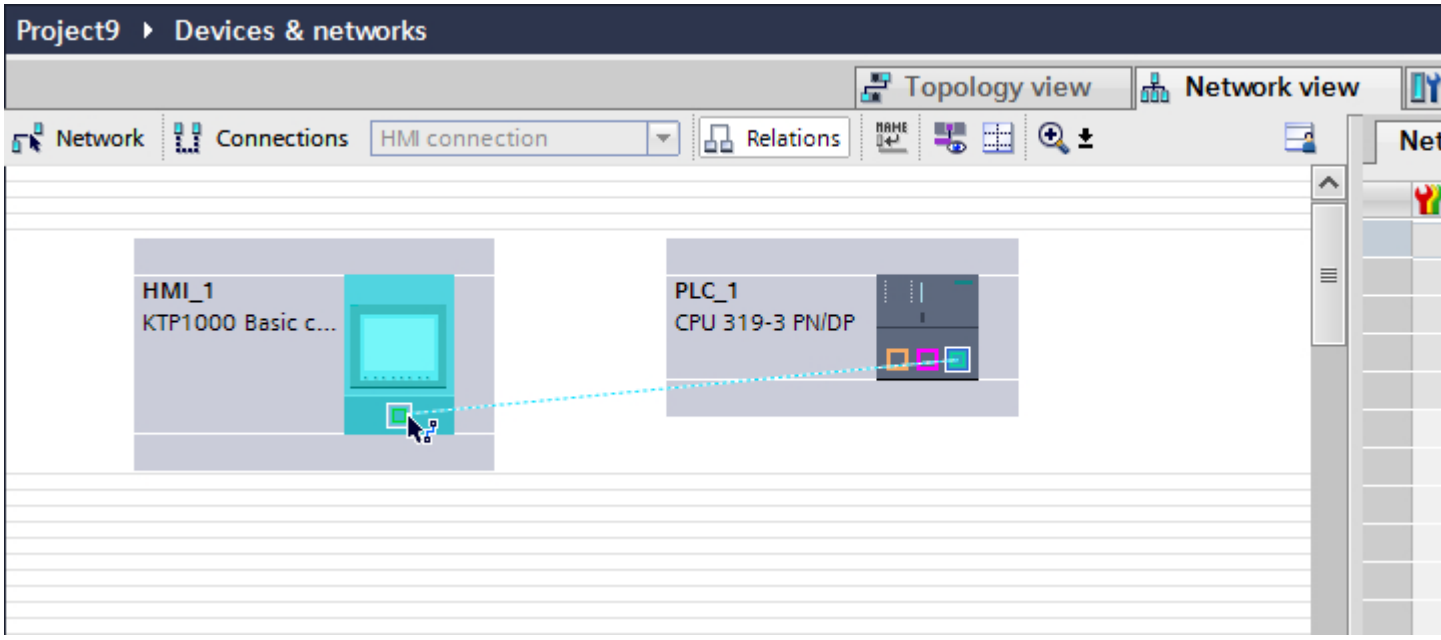
The following communication partners are created in the "Devices & Networks" editor:

- HMI device with PROFINET or Ethernet interface
- SIMATIC S7 300/400 with PROFINET interface.

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

3. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the HMI device.



4. Click the connecting line.
5. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
6. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFINET parameters (Page 6262)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 300/400. The IP address and subnet mask connection parameters are configured.

Configuring an HMI connection

Communication via PROFINET

Communication via PROFINET

This section describes the communication between a WinCC Runtime and the SIMATIC S7 300/400 PLC via PROFINET.

You can use the following WinCC Runtimes as an HMI device:

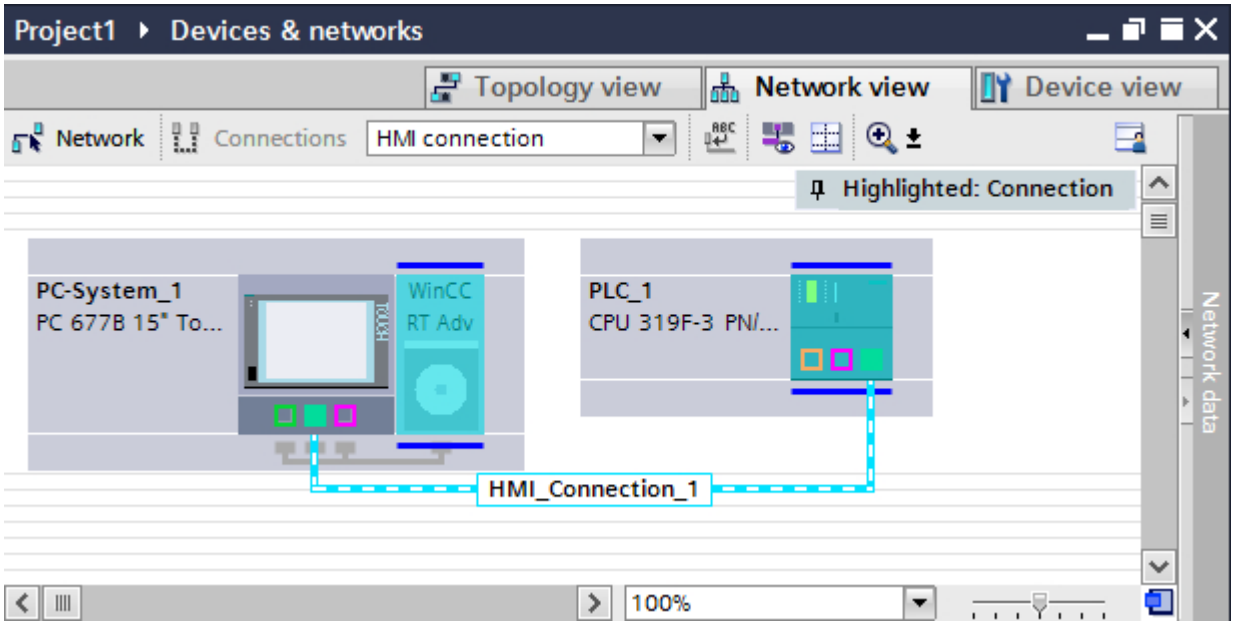
- WinCC RT Advanced

WinCC Runtime as HMI device

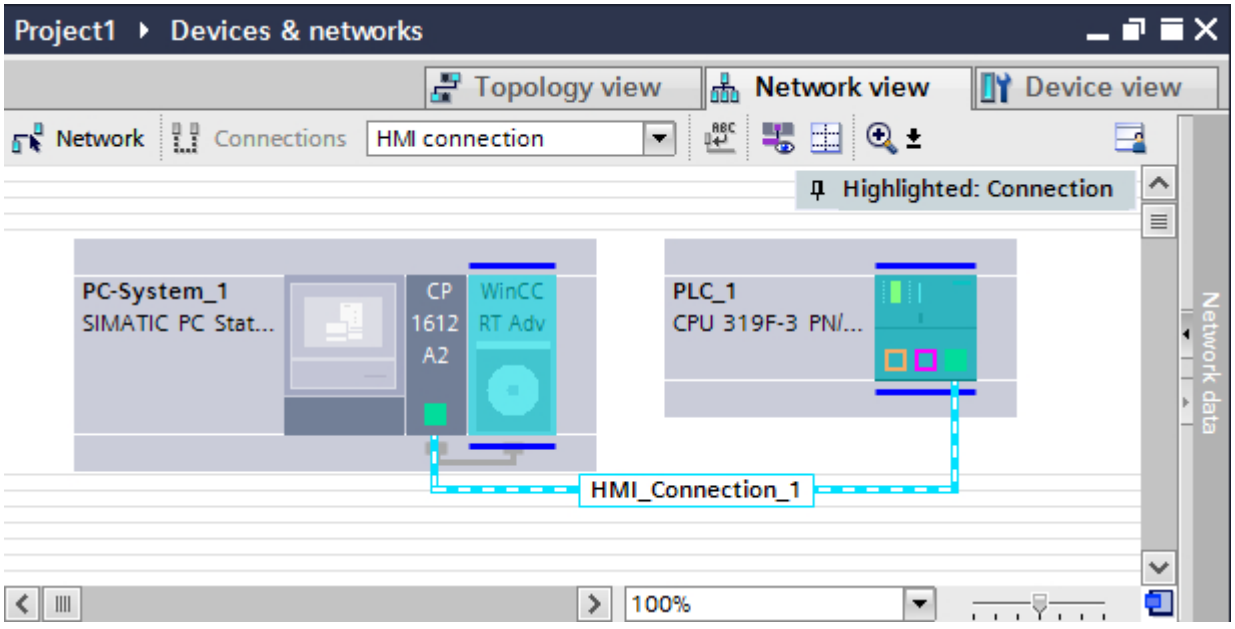
You configure the HMI connections between WinCC Runtime and SIMATIC S7 300/400 in the "Devices & Networks" editor.

If you want to use a WinCC Runtime as an HMI device, you have the following options:

1. You create a SIMATIC PC and configure a WinCC Runtime on this SIMATIC PC.
   In this way you use the SIMATIC PC with a WinCC Runtime as an HMI device.



2. You create a WinCC Runtime and configure a communication processor to the Runtime.
   In this way you use your configuration PC with a WinCC Runtime as an HMI device.



You can also connect multiple HMI devices to one SIMATIC S7 300/400 and multiple SIMATIC S7 300/400s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET with a SIMATIC PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFINET or Ethernet in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

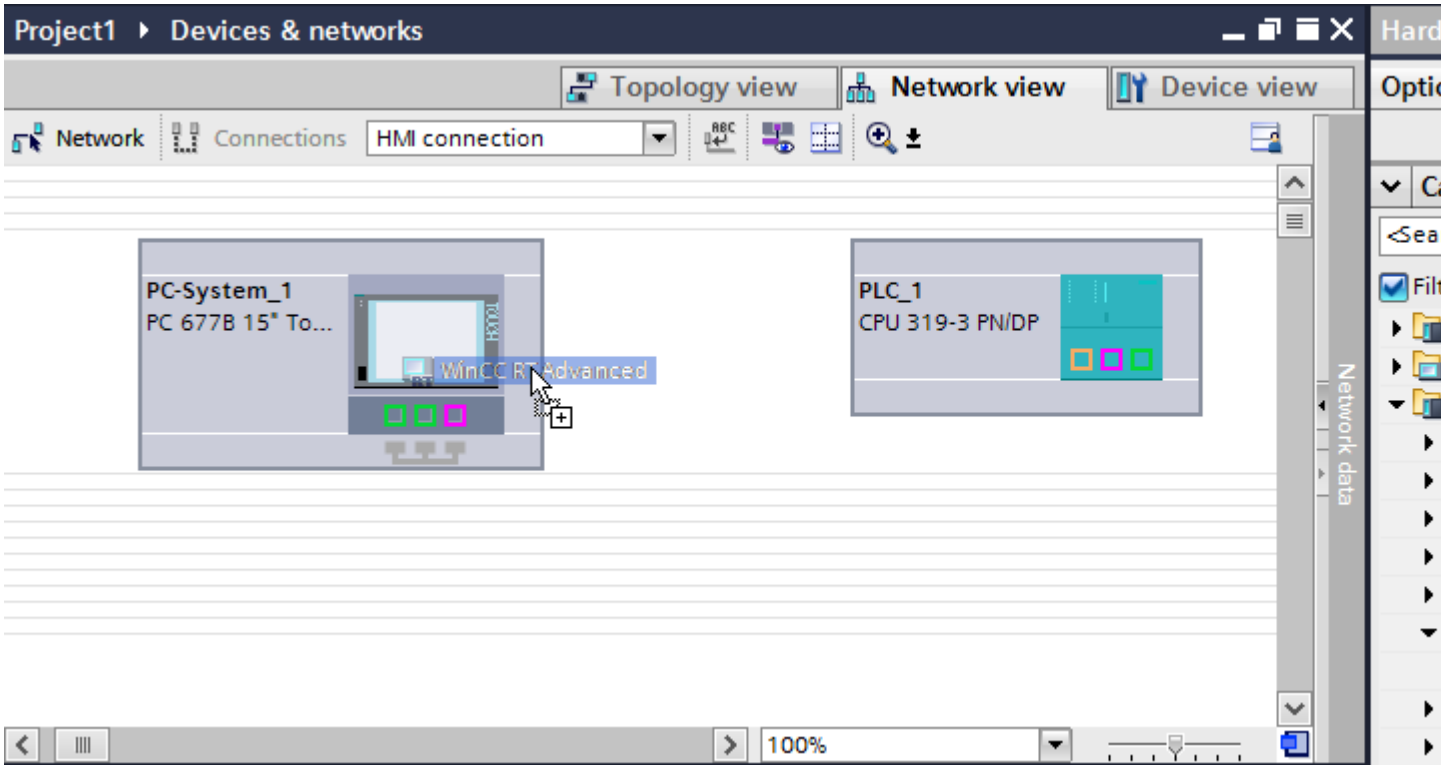
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 300/400 with PROFINET interface
- SIMATIC PC with PROFINET interface

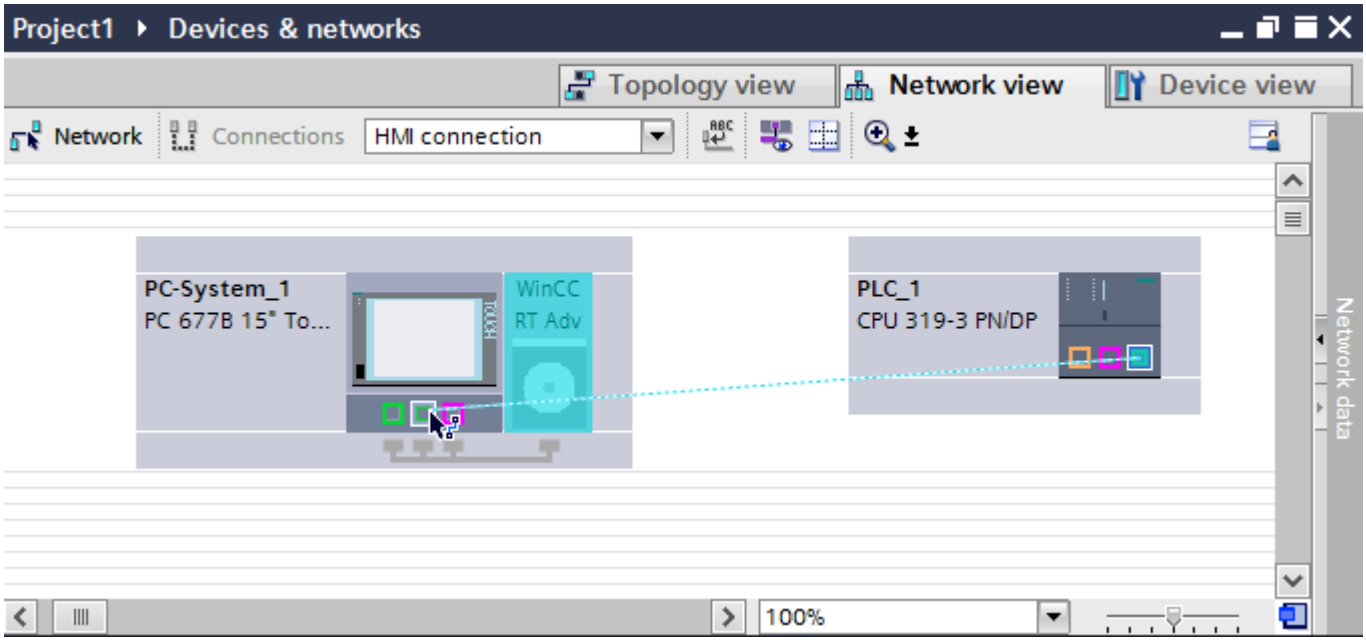
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a WinCC Runtime from the hardware catalog to the PC.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the PC.



5. Click the connecting line.
6. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
7. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFINET parameters (Page 6262)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 300/400. The IP address and subnet mask connection parameters are configured.

Configuring an HMI connection via PROFINET with a PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFINET or Ethernet in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

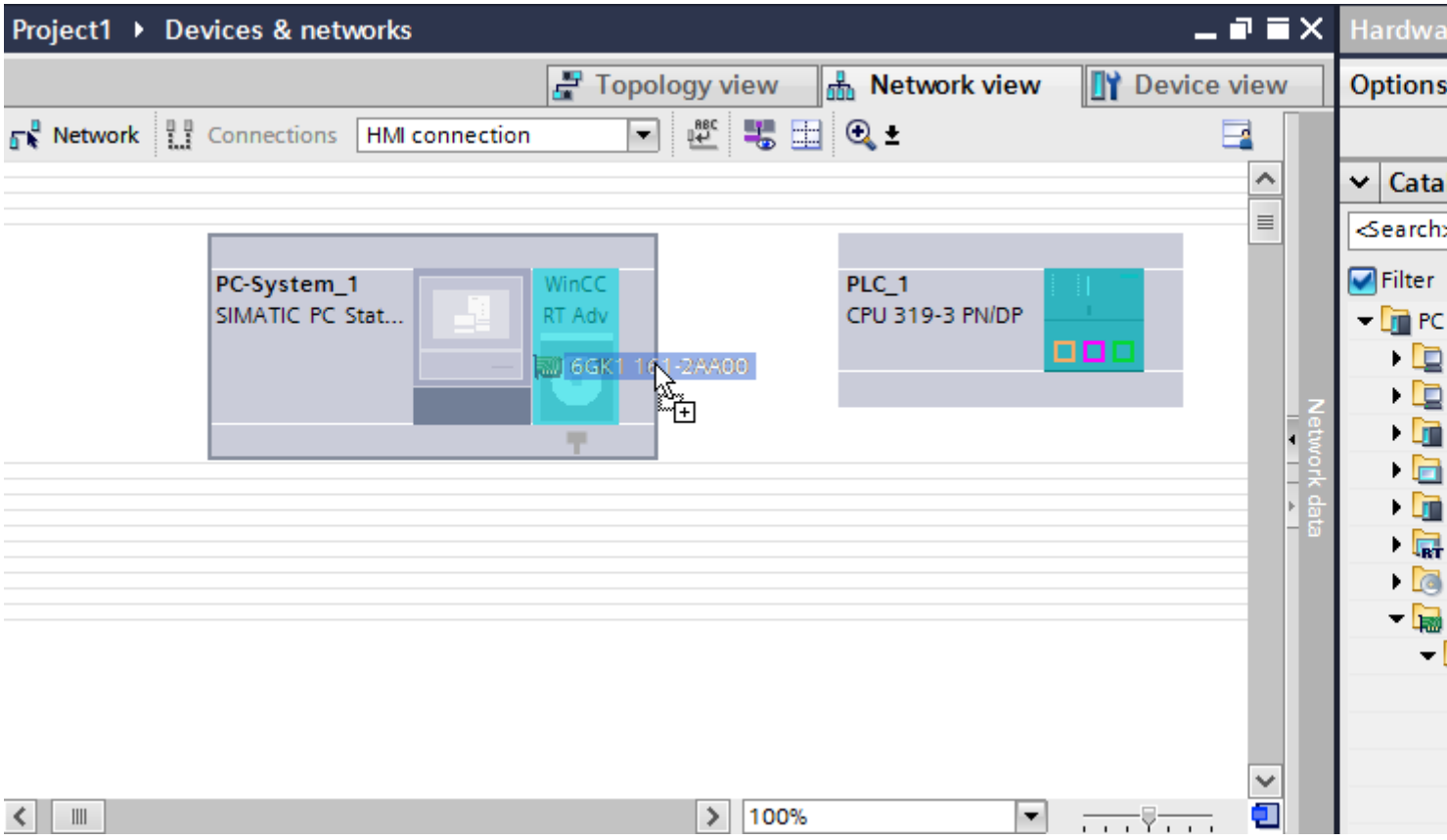
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 300/400 with PROFINET interface
- PC station with WinCC RT Advanced

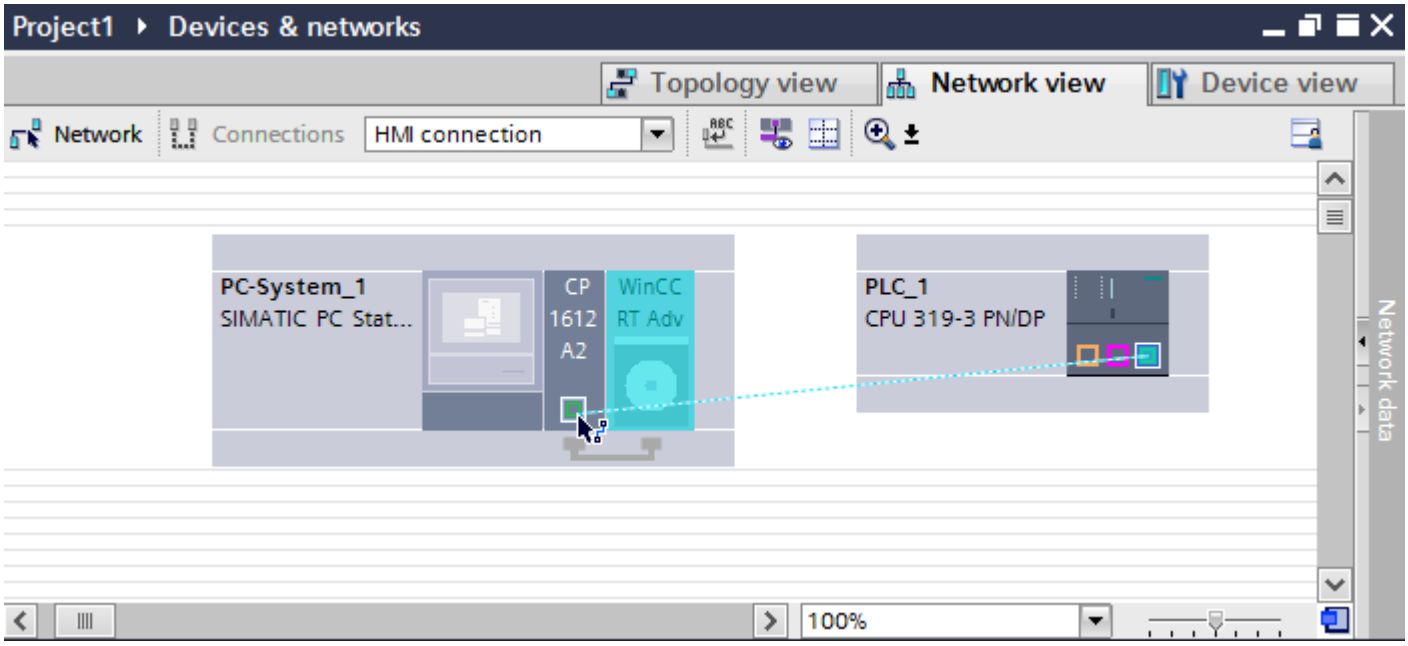
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Use a drag-and-drop operation to move a PROFINET-capable communication processor from the hardware catalog to the WinCC Runtime.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET interface of the communication processor.



5. Click the connecting line.
   The connection is displayed graphically in the Inspector window.
6. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
7. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFINET parameters (Page 6262)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 300/400. The IP address and subnet mask connection parameters are configured.

PROFINET parameters

PROFINET parameters for the HMI connection

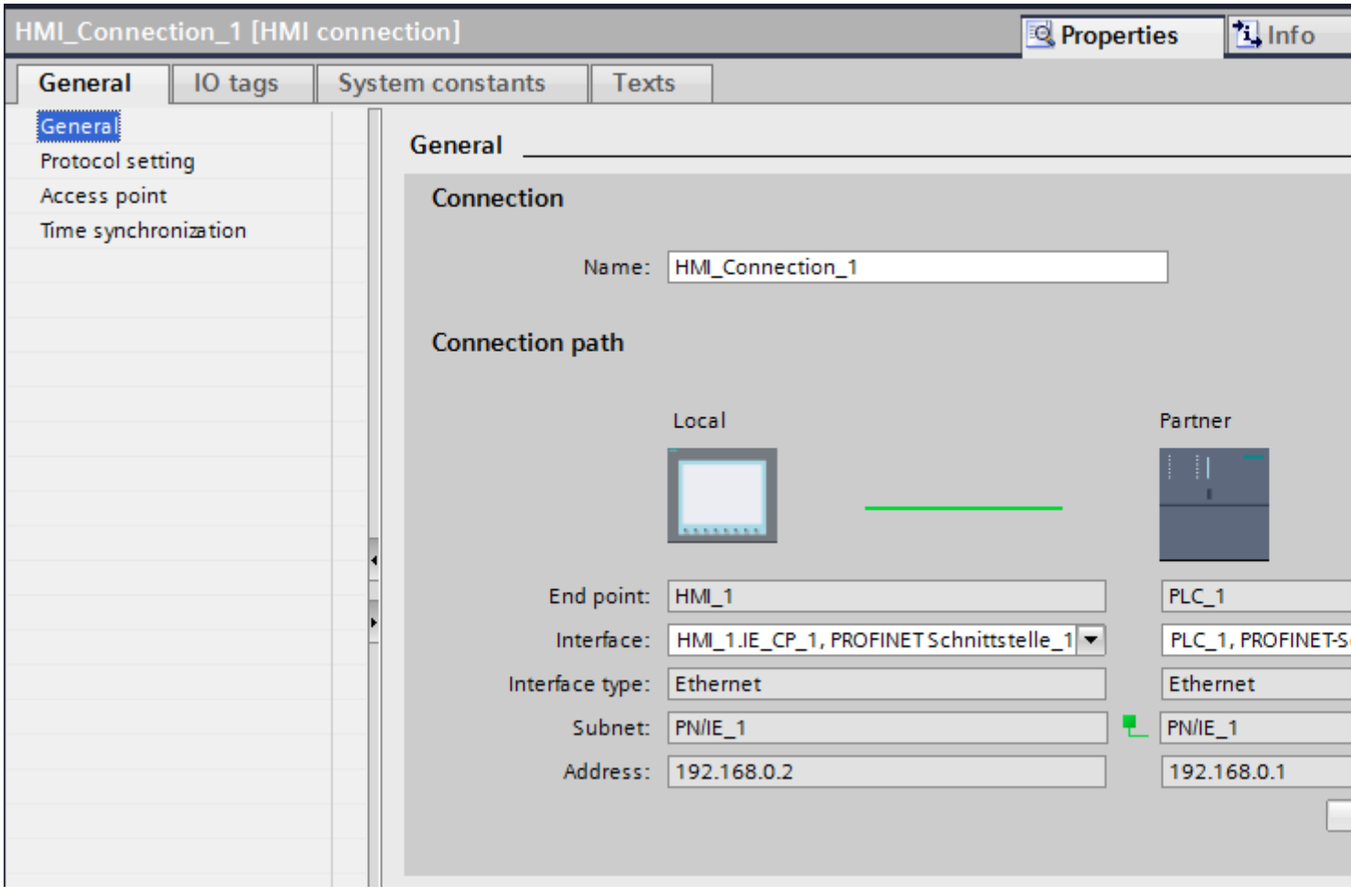
PROFINET parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Displays whether the devices are networked together.

- displayed if the devices are networked together.

- displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFINET parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area cannot be edited.
- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the selected IP address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

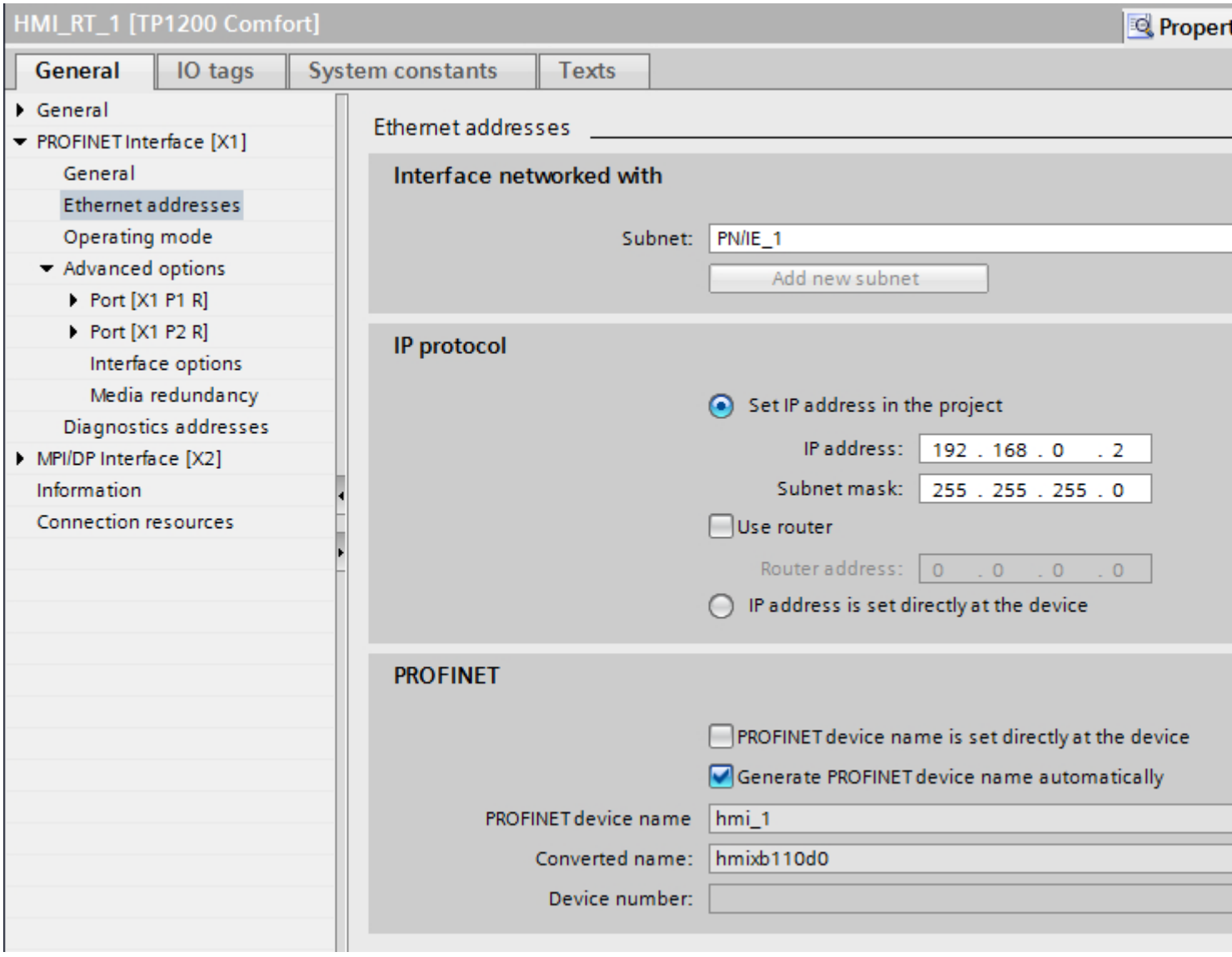
PROFINET parameters for the HMI device

PROFINET parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Set IP address in the project"
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

  Note

  The device is automatically restarted in the case of HMI devices with the Windows CE 3.0 operating system.

  HMI devices with Windows CE 3.0:
  - OP 77B
  - TP 177B color PN/DP
  - TP 177B mono DP
  - OP 177B color PN/DP
  - OP 177B mono DP
  - Mobile Panel 177 PN
  - Mobile Panel 177 DP
  - TP 277 6"
  - OP 277 6"

- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
- "Use IP router"
  If you are using an IP router, select "Use IP router" and enter the router address in the "Router address" field.
- "Set IP address using a different method"
  If the function "Set IP address using a different method" is activated, the IP address is not taken from the project. You have to enter the IP address directly in the Control Panel of the HMI device.

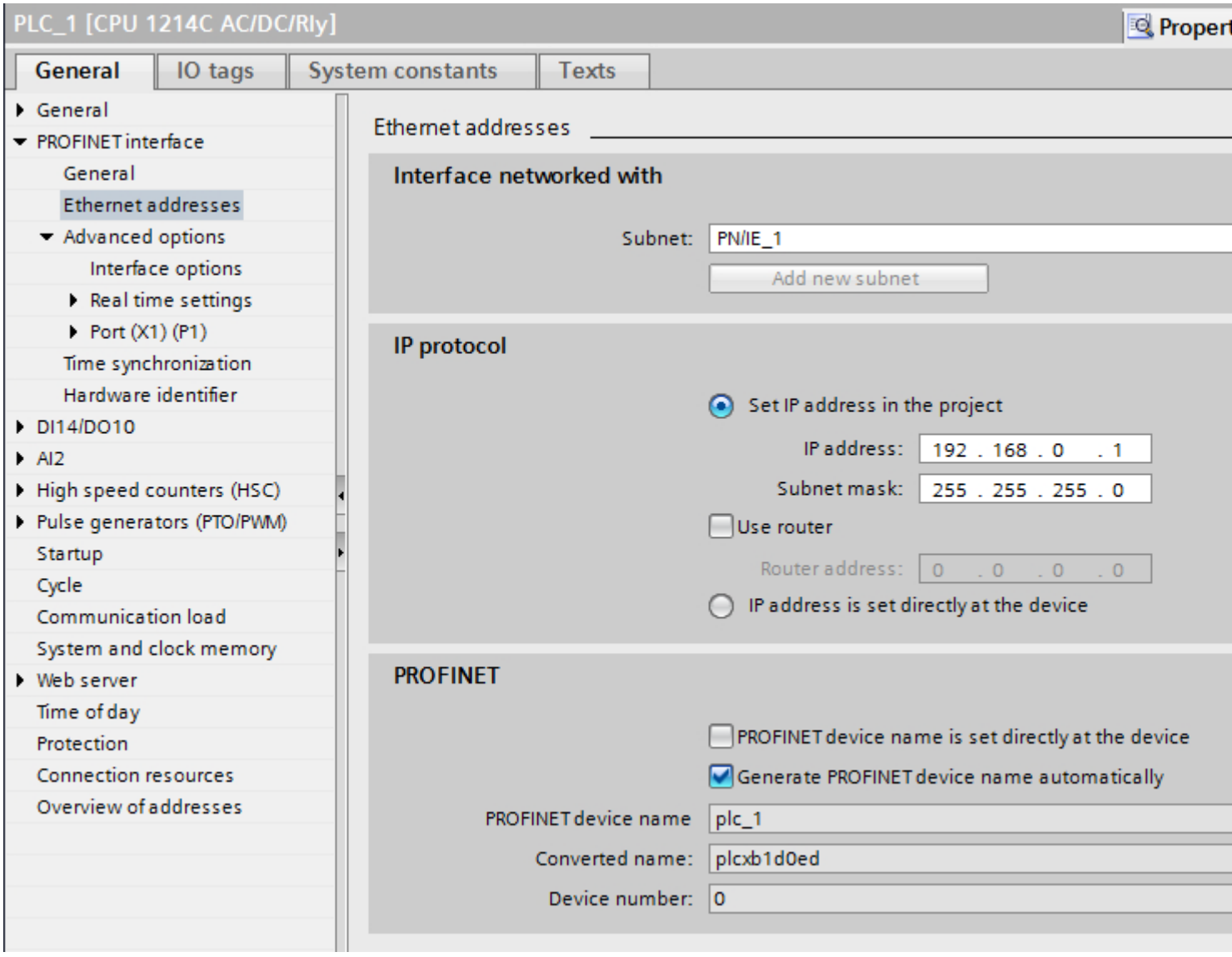
PROFINET parameters for the PLC

PROFINET parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFINET parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "IP address"
  You assign the IP address of the HMI device in the "IP address" area.
- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
  If you are using an IP router, select "Use IP router" and enter the router address in the field.

Configuring Industrial Ethernet

Rules for the network configuration

The Ethernet interfaces of the devices have a default IP address that you can change.

IP address

The IP parameters are visible if the communication-capable devices support the TCP/IP protocol.

The IP address consists of 4 decimal figures in the range of 0 to 255. The decimal figures are separated from one another by a dot.

Example: 140.80.0.2

The IP address consists of the following:

- The address of the (sub) net
- The address of the node (generally also called host or network node)

Subnet mask

The subnet mask splits these two addresses. It determines which part of the IP address addresses the network and which part of the IP address addresses the node.

The set bits of the subnet mask determine the network part of the IP address.

Example:

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

In the example given for the above IP address, the subnet mask shown here has the following meaning:

The first 2 bytes of the IP address identify the subnet - i.e. 140.80. The last two bytes address the node, thus 0.2.

It is generally true that:

- The network address results from AND linking the IP address and subnet mask.
- The node address results from AND NOT linking the IP address and subnet mask.

Relation between IP address and default subnet mask

An agreement exists relating to the assignment of IP address ranges and so-called "Default subnet masks". The first decimal number (from the left) in the IP address determines the structure of the default subnet mask. It determines the number of "1" values (binary) as follows:

IP address (decimal)	IP address (binary)	Address class	Default subnet mask
0 to 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 to 191	10xxxxxx.xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

Note

Range of values for the first decimal point

A value of between 224 and 255 is also possible for the first decimal number of the IP address (address class D etc). This is, however, not recommended because there is no address check for these values.

Masking other subnets

You can use the subnet mask to add further structures and form "private" subnets for a subnet that is assigned one of the address classes A, B or C. This is done by setting other lower points of the subnet mask to "1". For each bit set to "1", the number of "private" networks doubles and the number of nodes they contain is halved. Externally, the network functions like an individual network as it did previously.

Example:

You have a subnet of address class B (e.g. IP address 129.80.xxx.xxx) and change the default subnet mask as follows:

Masks	Decimal	Binary
Default subnet mask	255.255.0.0	11111111.11111111.00000000.00000000
Subnet mask	255.255.128.0	11111111.11111111.10000000.00000000

Result:

All nodes with addresses between 129.80.001.xxx and 129.80.127.xxx are on one subnet, all nodes with addresses between 129.80.128.xxx and 129.80.255.xxx are on another subnet.

Router

The job of the routers is to connect the subnets. If an IP datagram is to be sent to another network, it first has to be conveyed to a router. To make this possible, in this case you have to enter the address of the router for each node in the subnet.

The IP address of a node in the subnet and the address of the router may only differ at the points at which there is a "0" in the subnet mask.

Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission rate / duplex

Depending on the selected device, you can make the following settings for "Transmission rate / duplex":

- Automatic setting
  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the peer port. The "Enable autonegotiation" option is also enabled as a default, in other words, you can use cross cables or patch cables for the connection.
- TP/ITP at x Mbps full duplex (half duplex)
  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:
  - Autonegotiation enabled
    You can use both cross cable and patch cable.
  - Autonegotiation disabled
    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.
- Deactivated
  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option enables or disables port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because with this option the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can accordingly not be optimally set.

Note

When a local port is connected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not accept the setting, an error message is generated.

Wiring rules for disabled autonegotiation

Requirements

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission rate
- Autonegotiation incl. autocrossing disabled

The time for negotiating the transmission rate during startup has been saved.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

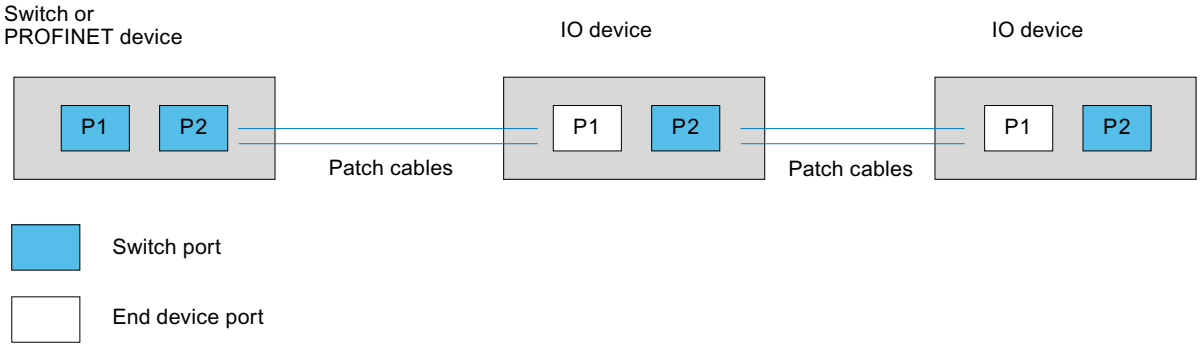
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in line using a single cable type (patch cable). To do this, you connect port 2 of the IO device (distributed I/O) with port 1 of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirements

To use boundaries, the respective device must have more than one port. If the PROFINET does not support boundary settings, they are not shown.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of discovery of accessible devices"
  No forwarding of DCP frames to identify accessible devices. Devices downstream from this port cannot be reached by the project tree under "Accessible devices". Devices downstream from this port cannot be reached by the CPU.
- "End of topology discovery"
  LLDP frames (Link Layer Discovery Protocol) are not forwarded for topology detection.
- "End of sync domain"
  No forwarding of sync frames transmitted to synchronize nodes within a sync domain.
  If you operate, for example, a PROFINET device with more than two ports in a ring, you should prevent the sync frame from being fed into the ring by setting a sync boundary (at the ports not inside the ring).
  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device from the other sync domain.

Restrictions

The following restrictions must be observed:

- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been determined for the port, the following check boxes cannot be used:
  - "End of discovery of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

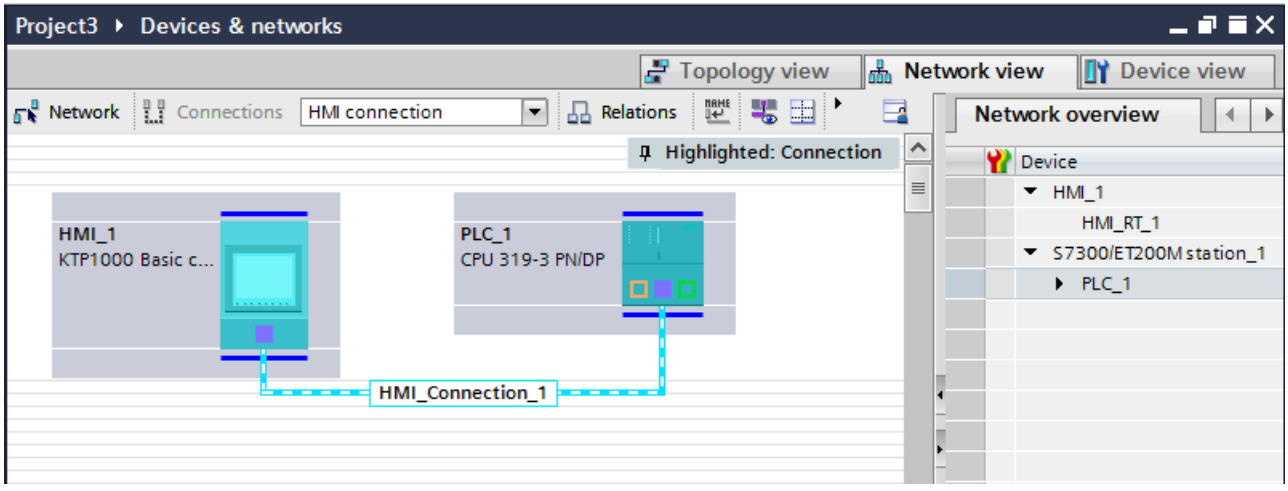
12.11.8.3 Communication via PROFIBUS

Configuring an HMI connection

Communication via PROFIBUS

HMI connections via PROFIBUS

If you have inserted an HMI device and a SIMATIC S7 300/400 into the project, you interconnect the two PROFIBUS interfaces in the "Devices & Networks" editor.



HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFIBUS in the "Devices & Networks" editor.

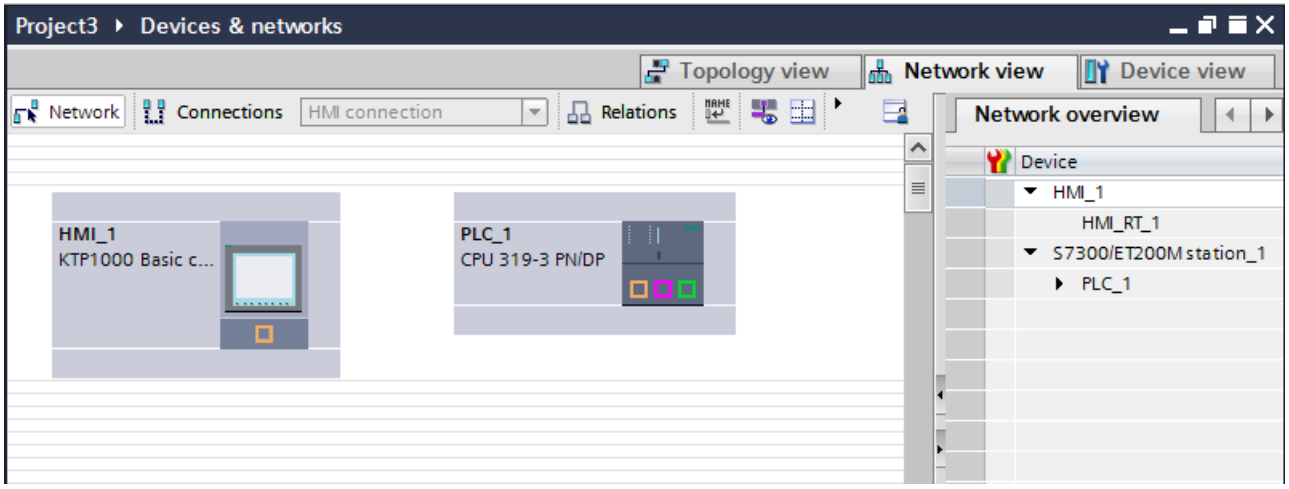
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- HMI device with MPI/DP interface
- SIMATIC S7 300/400 with PROFIBUS interface

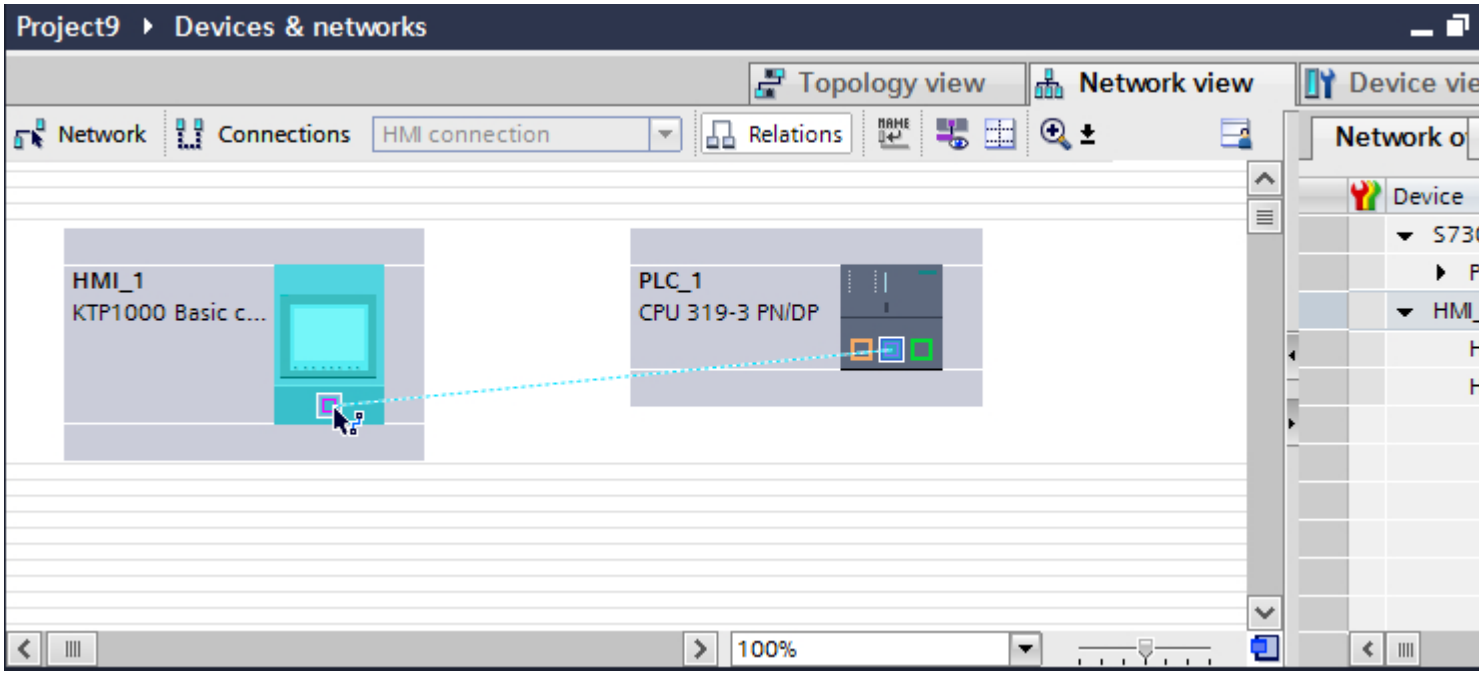
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.



2. Click the "Connections" button.
   The devices available for connection are highlighted in color.
3. Click the HMI device interface.
4. Select the "PROFIBUS" interface type in the Inspector window under "Properties > General > HMI MPIDP > Parameters".

5. Click the interface of the PLC and use a drag-and-drop operation to draw a connection to the HMI device.



6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFIBUS parameters (Page 6282)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 300/400 via PROFIBUS.

Configuring an HMI connection

Communication via PROFIBUS

Communication via PROFIBUS

This section describes the communication between a WinCC Runtime and the SIMATIC S7 300/400 PLC via PROFIBUS.

You can use the following WinCC Runtimes as an HMI device:

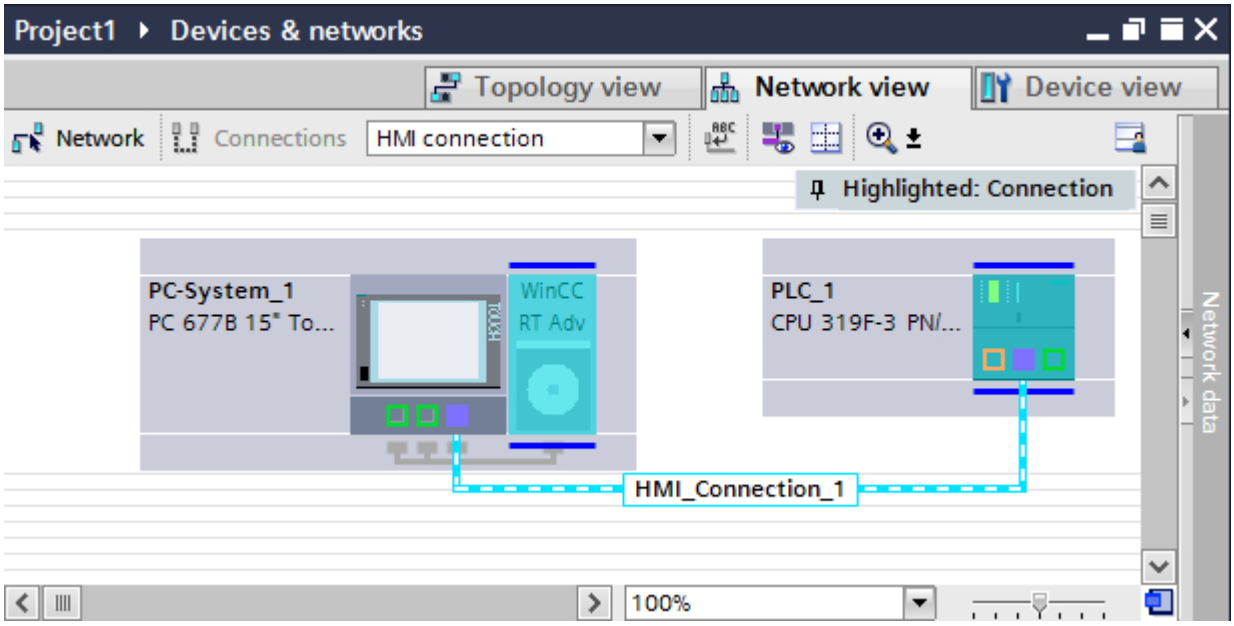
- WinCC RT Advanced
- WinCC RT Professional

WinCC Runtime as HMI device

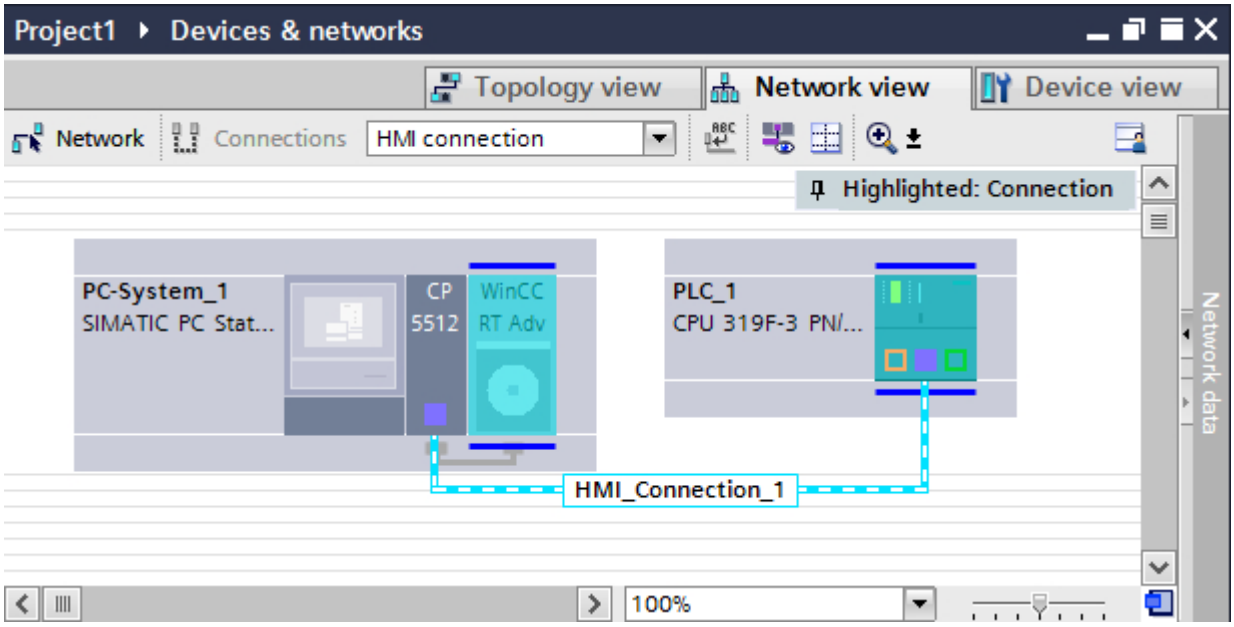
You configure the HMI connections between WinCC Runtime and SIMATIC S7 300/400 in the "Devices & Networks" editor.

If you want to use a WinCC Runtime as an HMI device, you have the following options:

1. You create a SIMATIC PC and configure a WinCC Runtime on this SIMATIC PC.
   In this way you use the SIMATIC PC with a WinCC Runtime as an HMI device.



2. You create a WinCC Runtime and configure a communication processor to the Runtime.
   In this way you use your configuration PC with a WinCC Runtime as an HMI device.



You can also connect multiple HMI devices to one SIMATIC S7 300/400 and multiple SIMATIC S7 300/400s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFIBUS in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS with a SIMATIC PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFIBUS in the "Devices & Networks" editor.

Requirements

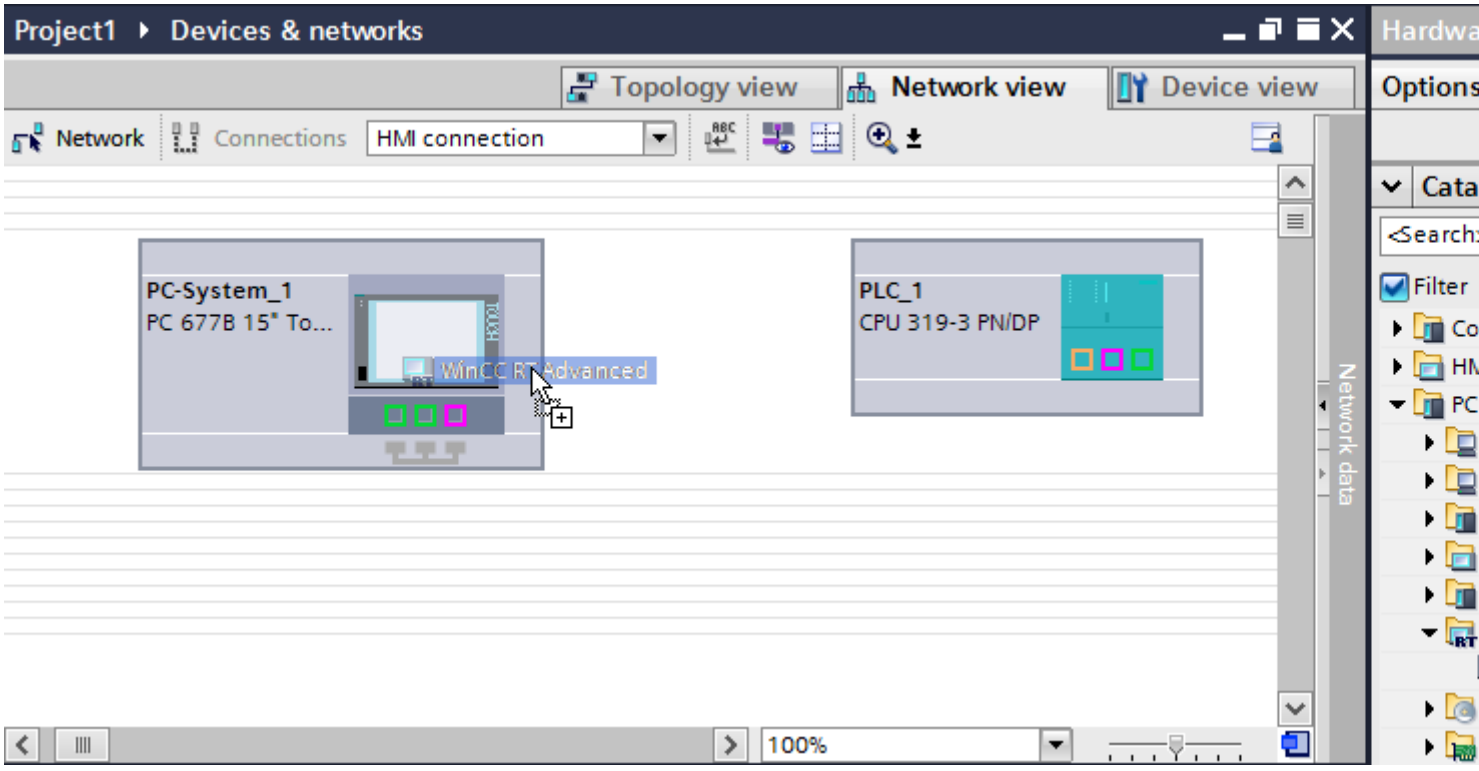
The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 300/400 with PROFIBUS interface
- SIMATIC PC with PROFIBUS interface

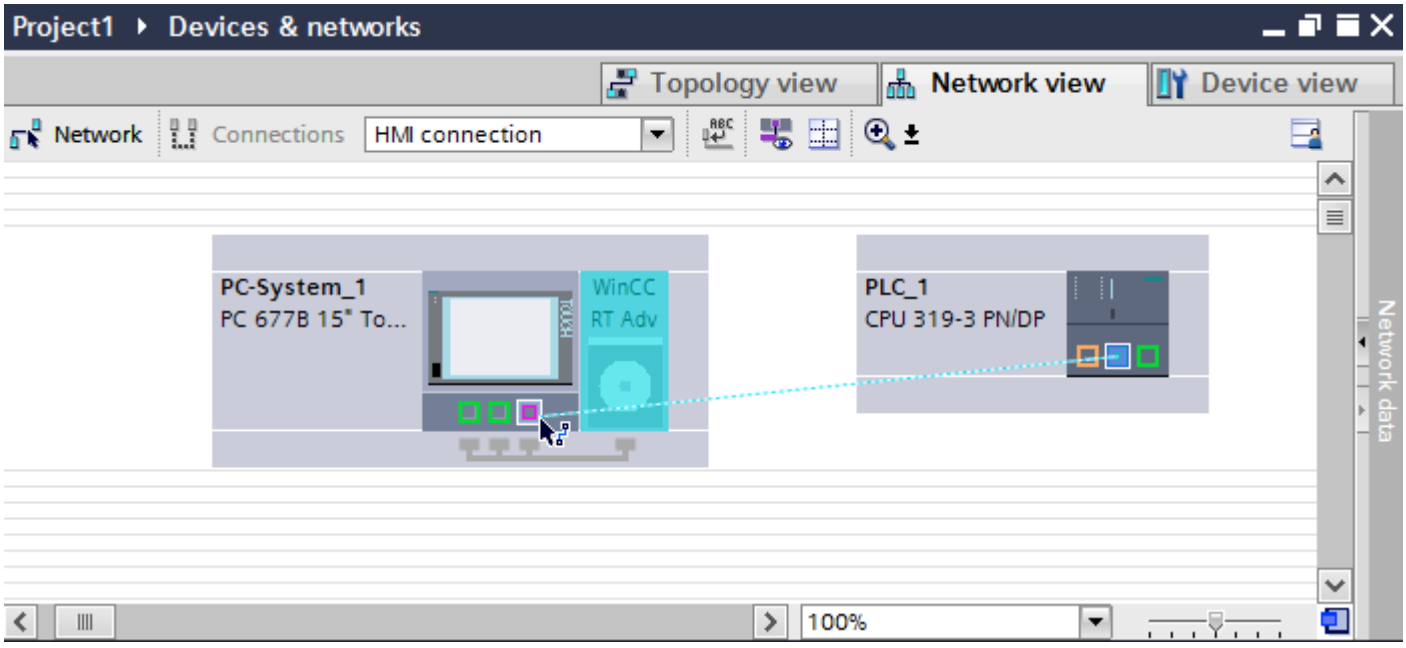
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Click the MPI interface of the PC and select "PROFIBUS" for the interface type in the Inspector window.

3. Use a drag-and-drop operation to move a WinCC Runtime from the hardware catalog to the PC.



4. Click the "Connections" button and select "HMI connection" for the connection type. The devices available for connection are highlighted in color.
5. Click the PROFIBUS interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFIBUS interface of the PC.



6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFIBUS parameters (Page 6282)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 300/400 via PROFIBUS.

Configuring an HMI connection via PROFIBUS with a PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFIBUS in the "Devices & Networks" editor.

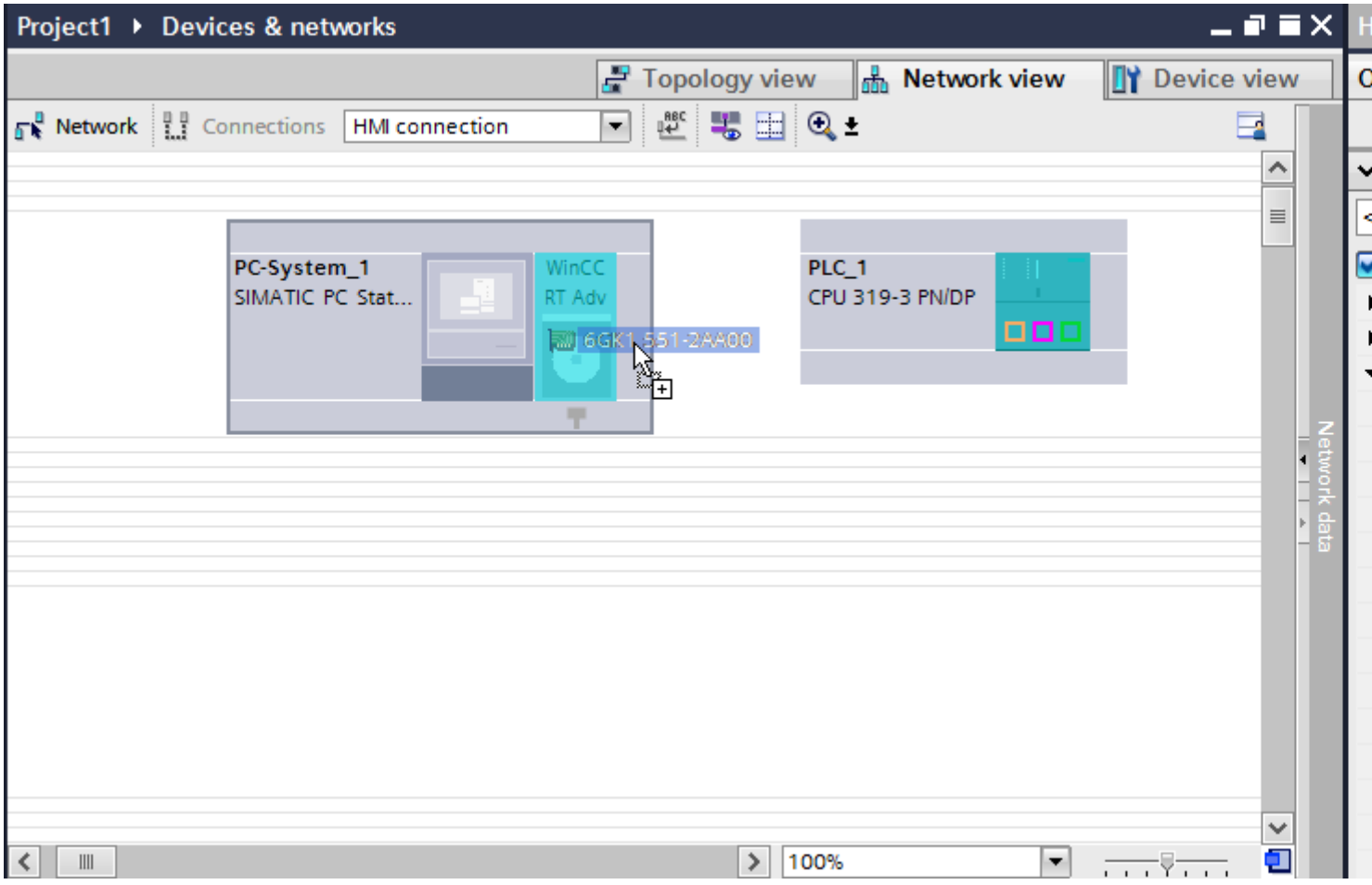
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 300/400
- PC station with WinCC RT Advanced

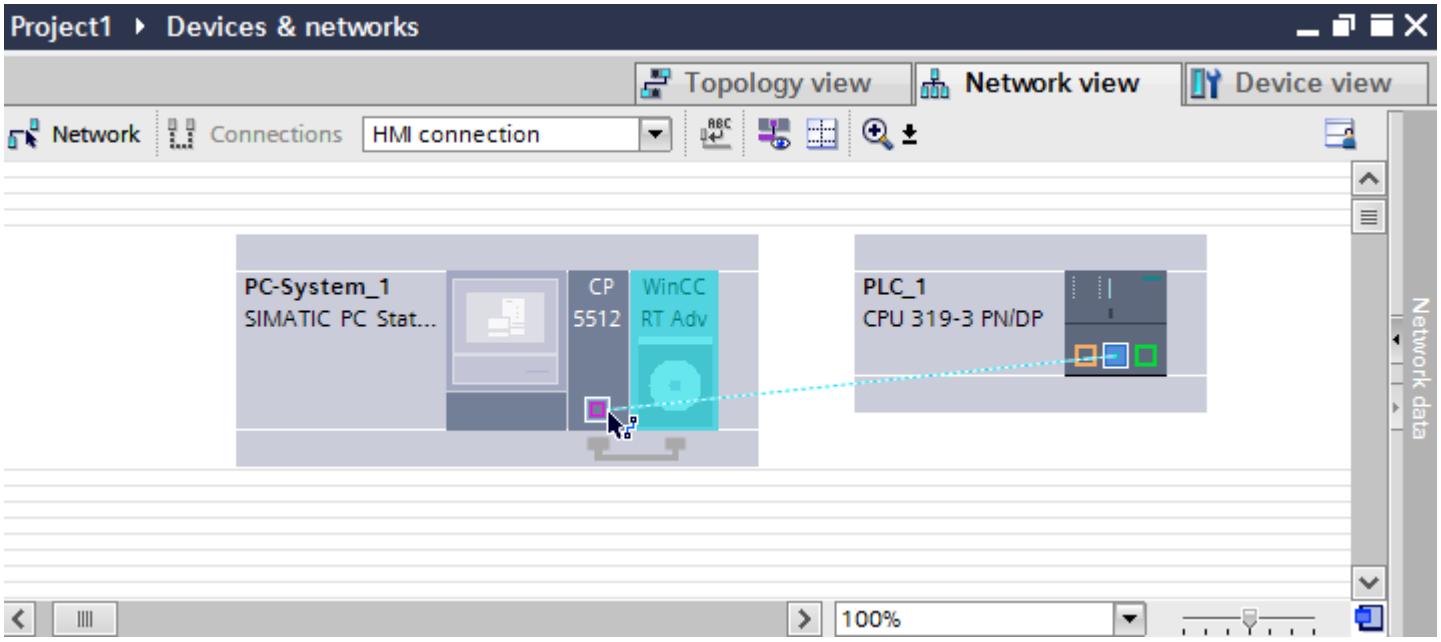
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a PROFIBUS-capable communication processor from the hardware catalog to the WinCC Runtime.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click the PROFIBUS interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFIBUS interface of the communication processor.



5. Click the connecting line.
6. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
7. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFIBUS parameters (Page 6282)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 300/400 via PROFIBUS.

PROFIBUS parameters

PROFIBUS parameters for the HMI connection

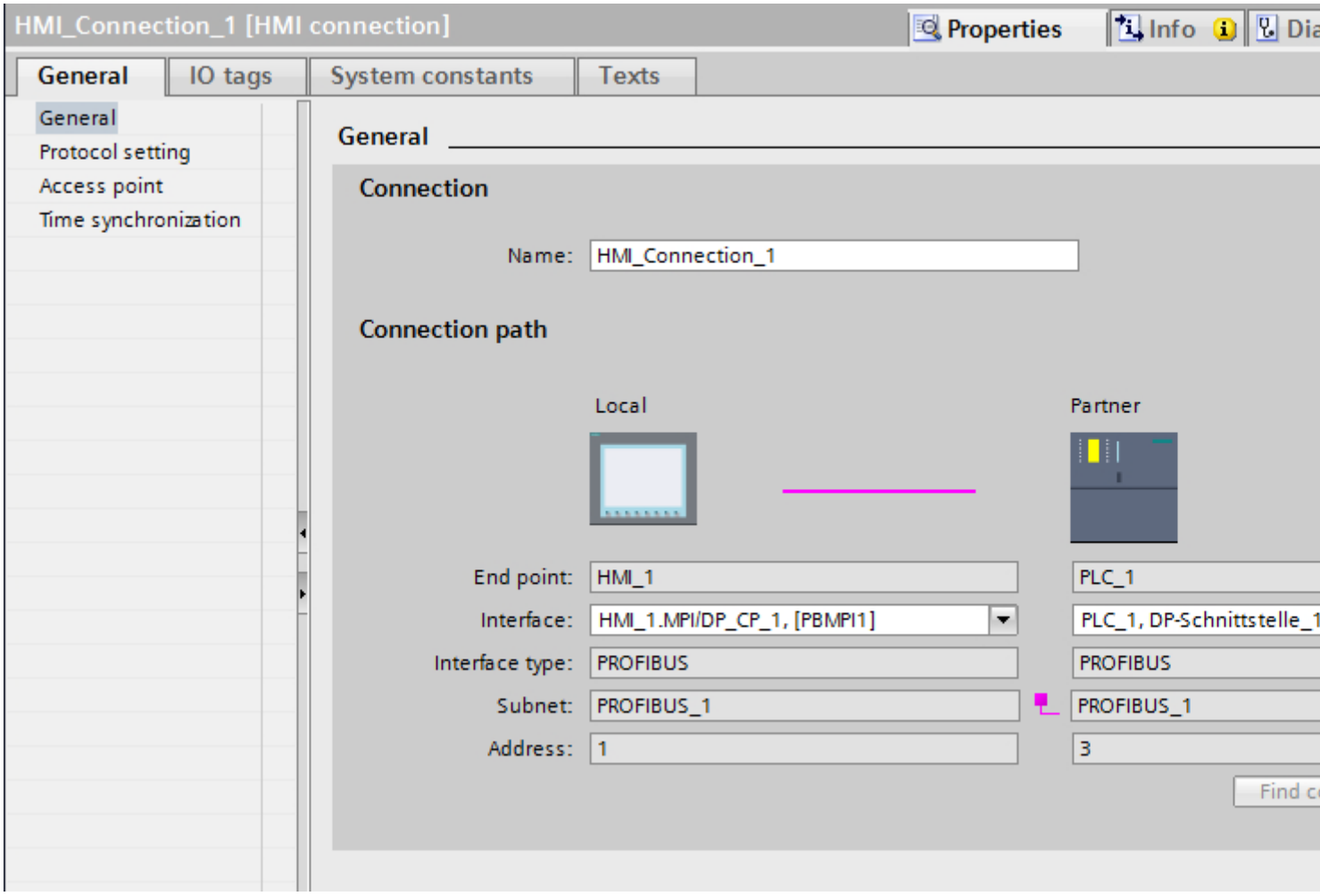
PROFIBUS parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Displays whether the devices are networked together.

- displayed if the devices are networked together.

- displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFIBUS parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area is not editable.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area is not editable.
- "Subnet"
  Displays the selected subnet. This area is not editable.
- "Address"
  Displays the PROFIBUS address of the device. This area is not editable.
- "Find connection path" button
  Enables the subsequent specification of connections.

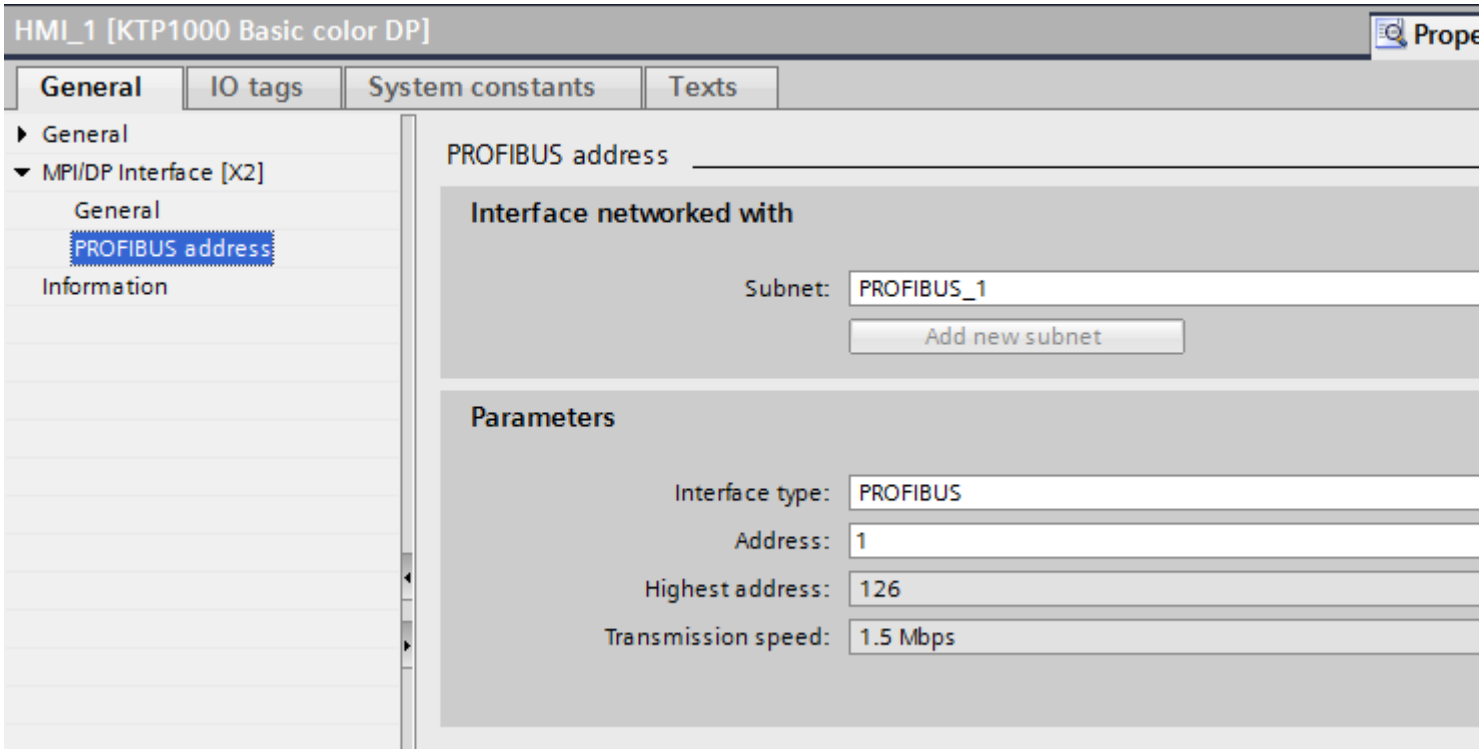
PROFIBUS parameters for the HMI device

PROFIBUS parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  You assign the interface type in the "Interface type" area. Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

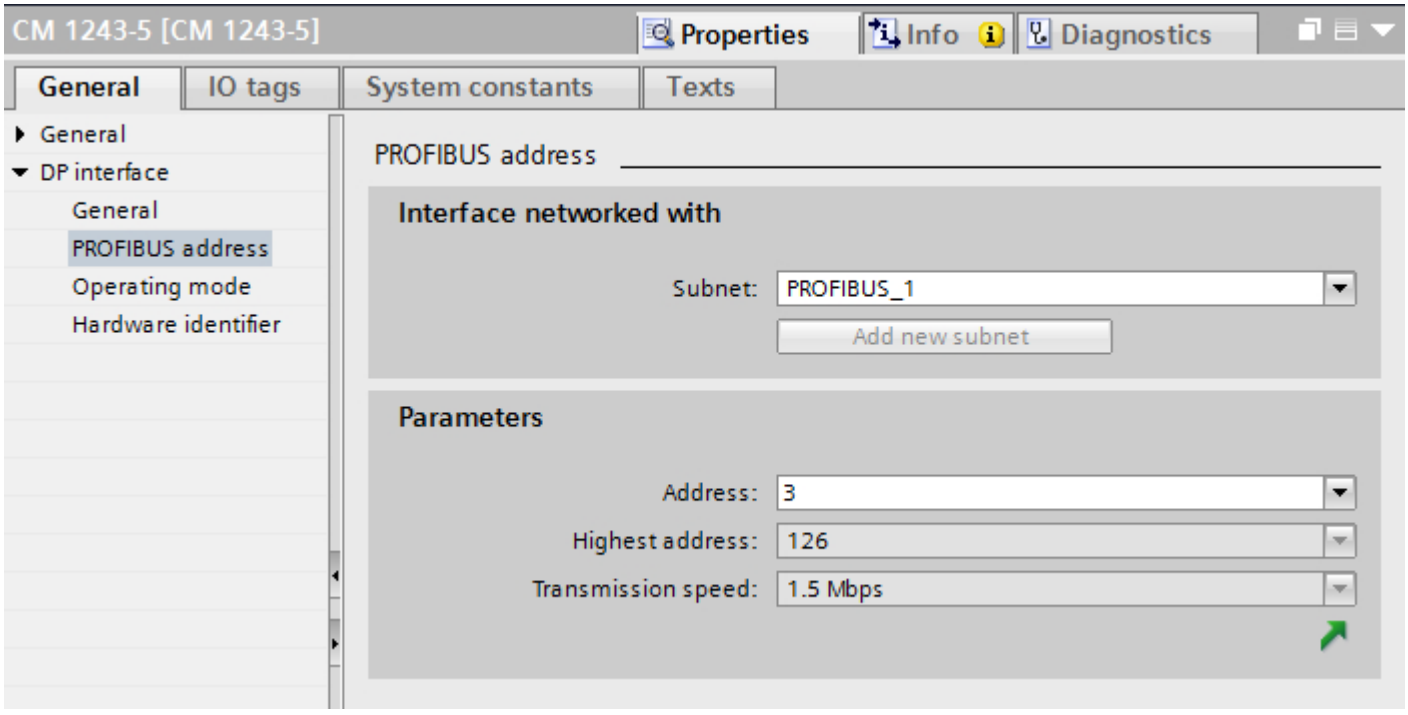
PROFIBUS parameters for the PLC

PROFIBUS parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.

- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Bus profiles with PROFIBUS

Introduction

Depending on the device types connected and protocols used on the PROFIBUS, different profiles are available. The profiles differ in terms of the setting options and calculation of bus parameters. The profiles are explained below.

Devices with different profiles on the same PROFIBUS subnet

The PROFIBUS subnet only functions without problem if the bus parameters of all devices have the same values.

Profiles and transmission rates

Profiles	Supported transmission speeds in Kbits/s
DP	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Standard	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Universal	9,6 19,2 93,75 187,5 500 1500

Meaning of profiles

Profile	Meaning
DP	Select the "DP" bus profile when the only devices connected to the PROFIBUS subnet are those which satisfy the requirements of standard EN 50170 Volume 2/3, Part 8-2 PROFIBUS. The bus parameter setting is optimized on these devices. This includes devices with DP master and DP slave interfaces of the SIMATIC S7 and distributed I/Os of other manufacturers.
Standard	Compared to the "DP" profile, the "Standard" profile also offers scope for devices of another project or devices which have not been configured here to be taken into account when calculating the bus parameters. The bus parameters are then calculated following a simple, non-optimized algorithm.
Universal	Select the "Universal" bus profile when individual devices on the PROFIBUS subnet use the PROFIBUS-FMS service. This includes the following devices for example: • CP 343-5 • PROFIBUS-FMS devices of other manufacturers As with the "Standard" profile, this profile allows you to take other devices into account when calculating the bus parameters.

12.11.8.4 Communication via MPI

Configuring an HMI connection

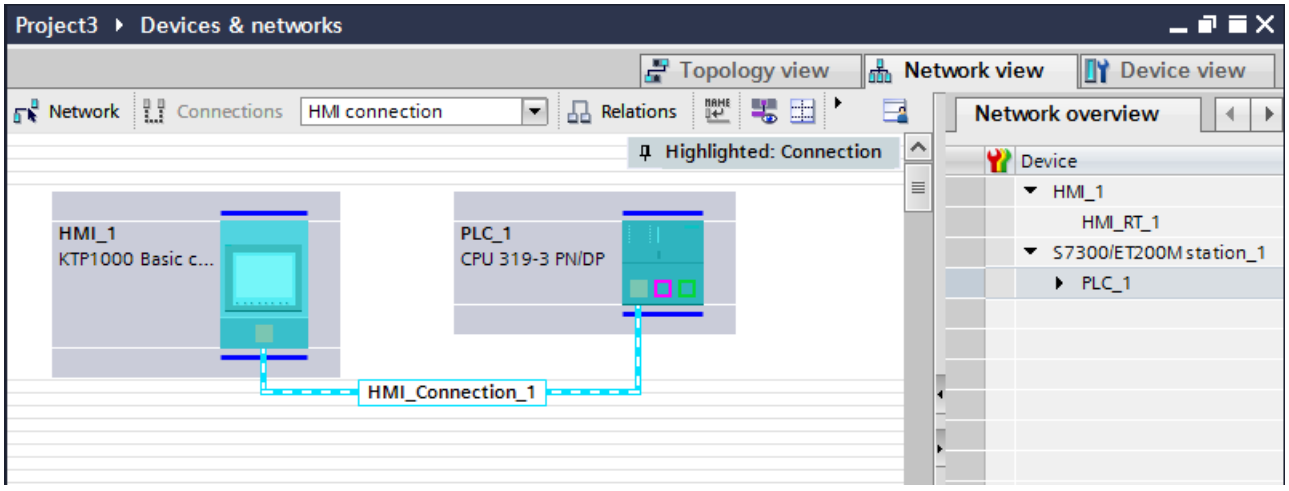
Communication via MPI

HMI connections via MPI

If you have inserted an HMI device and a SIMATIC S7 300/400 into the project, you interconnect the two MPI interfaces in the "Devices & Networks" editor.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.



Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via MPI

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via MPI in the "Devices & Networks" editor.

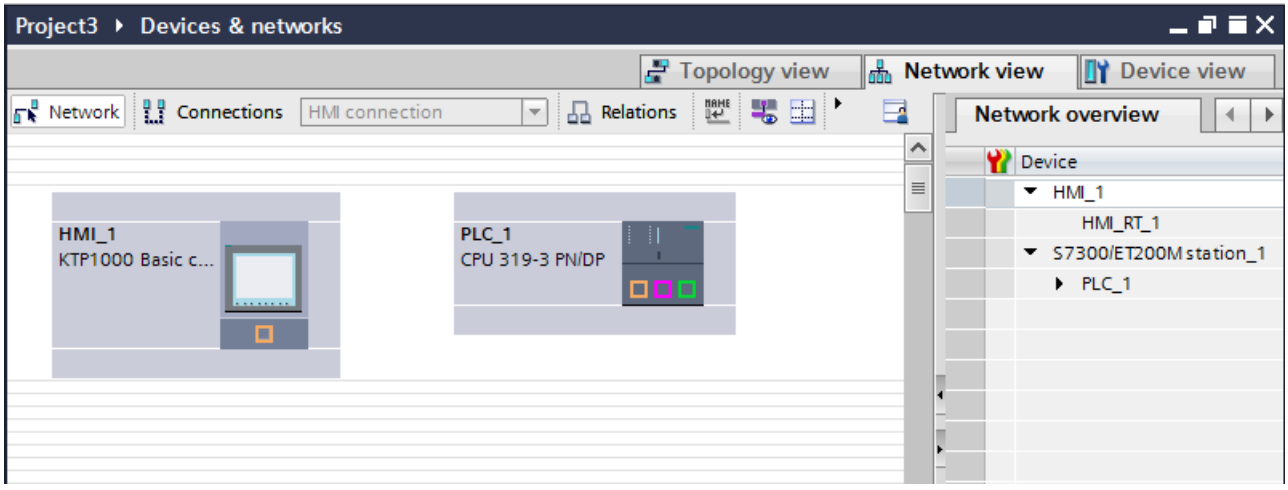
Requirements

The following communication partners are created in the "Devices & Networks" editor:

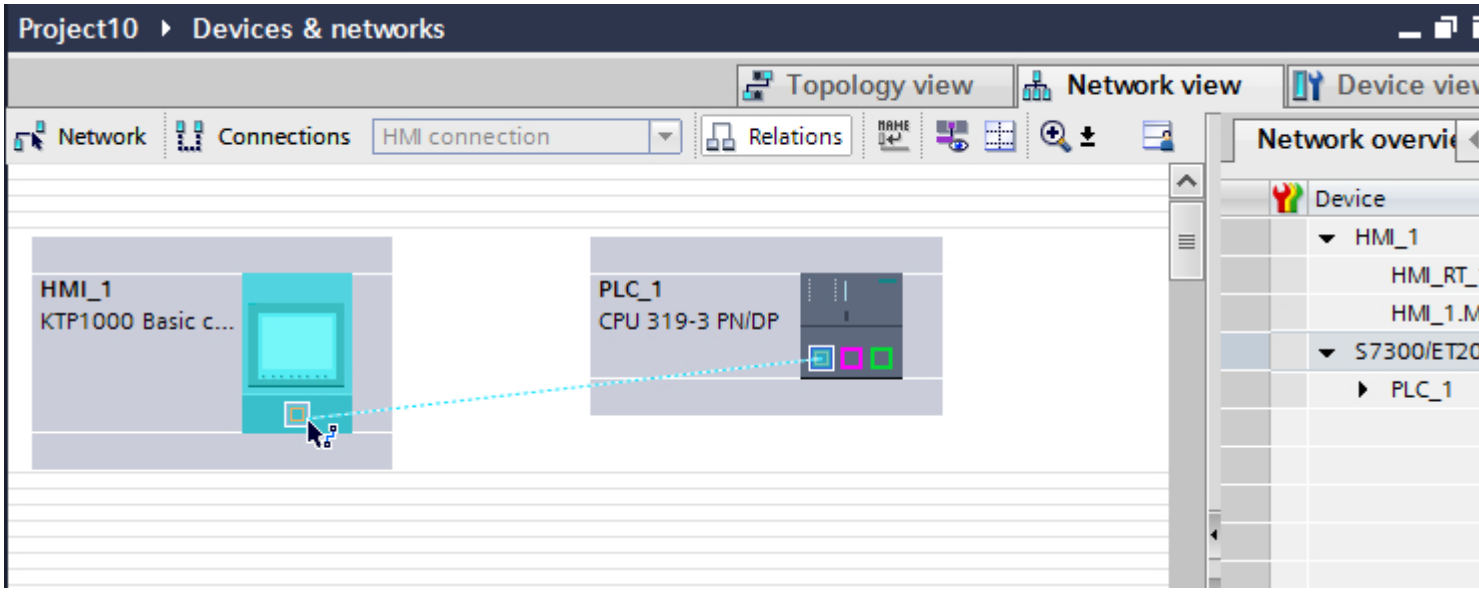
- HMI device with MPI/DP interface
- SIMATIC S7 300/400 with MPI/DP interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.



2. Click the "Connections" button.
   The devices available for connection are highlighted in color.
3. Click the interface of the PLC and use a drag-and-drop operation to draw a connection to the HMI device.



4. Click the connecting line.
   The connection is displayed graphically in the Inspector window.
5. Click "Highlight HMI connection" and select the HMI connection.
6. Click the communication partners in the "Network view" and change the MPI parameters in the Inspector window according to the requirements of your project.
   See the chapter "MPI parameters (Page 6296)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. Use the table to monitor the connection parameters and change the connection partner. You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 300/400 via MPI.

Configuring an HMI connection

Communication via MPI

Communication via MPI

This section describes the communication between a WinCC Runtime and the SIMATIC S7 300/400 PLC via MPI.

You can use the following WinCC Runtimes as an HMI device:

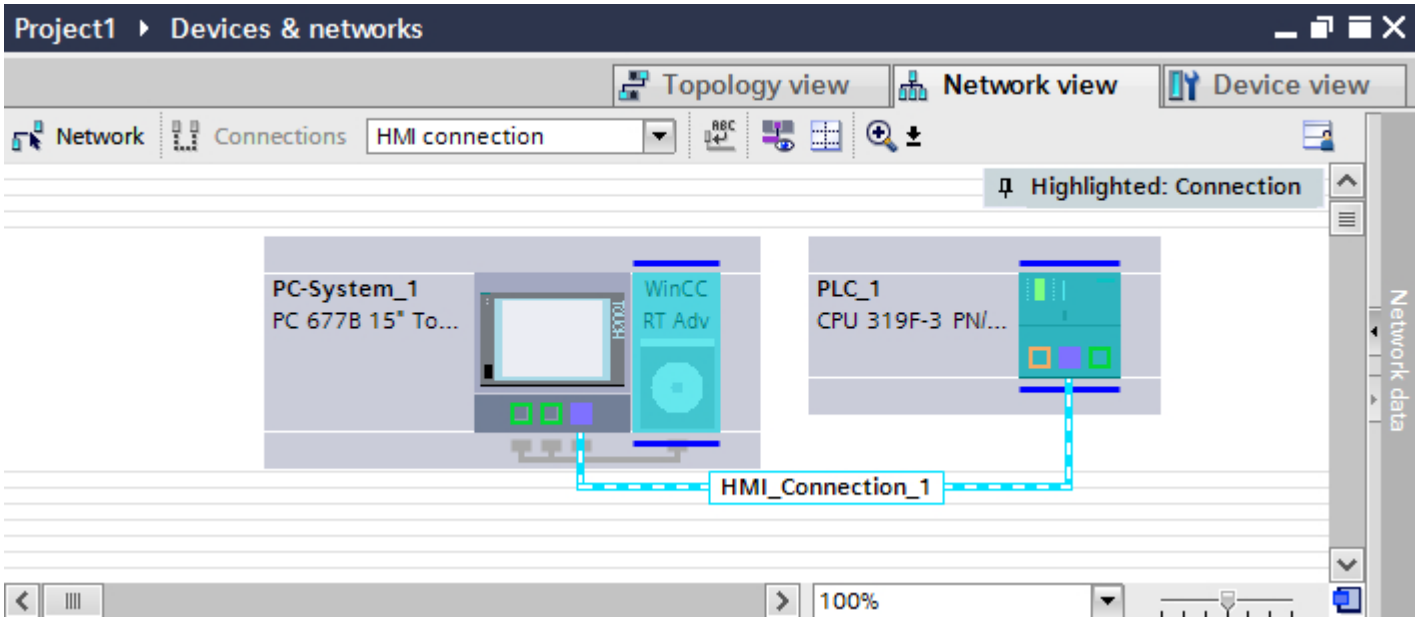
- WinCC RT Advanced

WinCC Runtime as HMI device

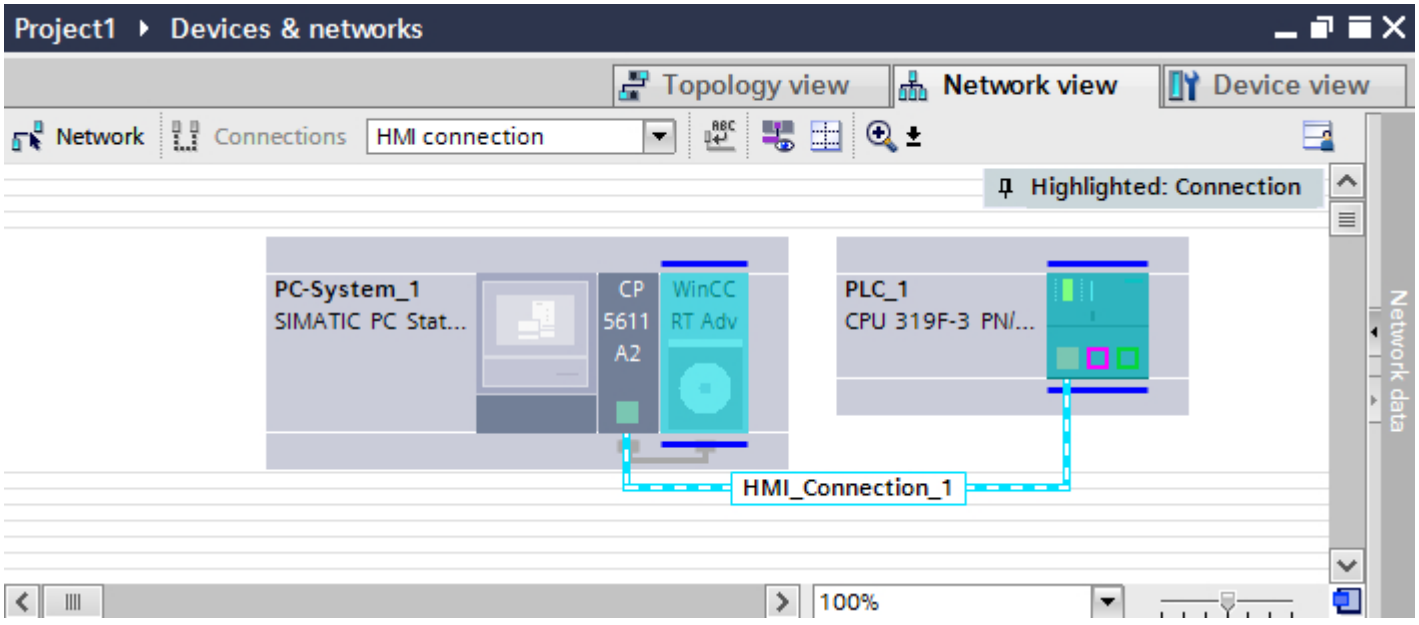
You configure the HMI connections between WinCC Runtime and SIMATIC S7 300/400 in the "Devices & Networks" editor.

If you want to use a WinCC Runtime as an HMI device, you have the following options:

1. You create a SIMATIC PC and configure a WinCC Runtime on this SIMATIC PC.
   In this way you use the SIMATIC PC with a WinCC Runtime as an HMI device.



2. You create a WinCC Runtime and configure a communication processor to the Runtime.
   In this way you use your configuration PC with a WinCC Runtime as an HMI device.



You can also connect multiple HMI devices to one SIMATIC S7 300/400 and multiple SIMATIC S7 300/400s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via MPI in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via MPI in the "Connections" editor of the HMI device.

Configuring an HMI connection via MPI with a SIMATIC PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFIBUS in the "Devices & Networks" editor.

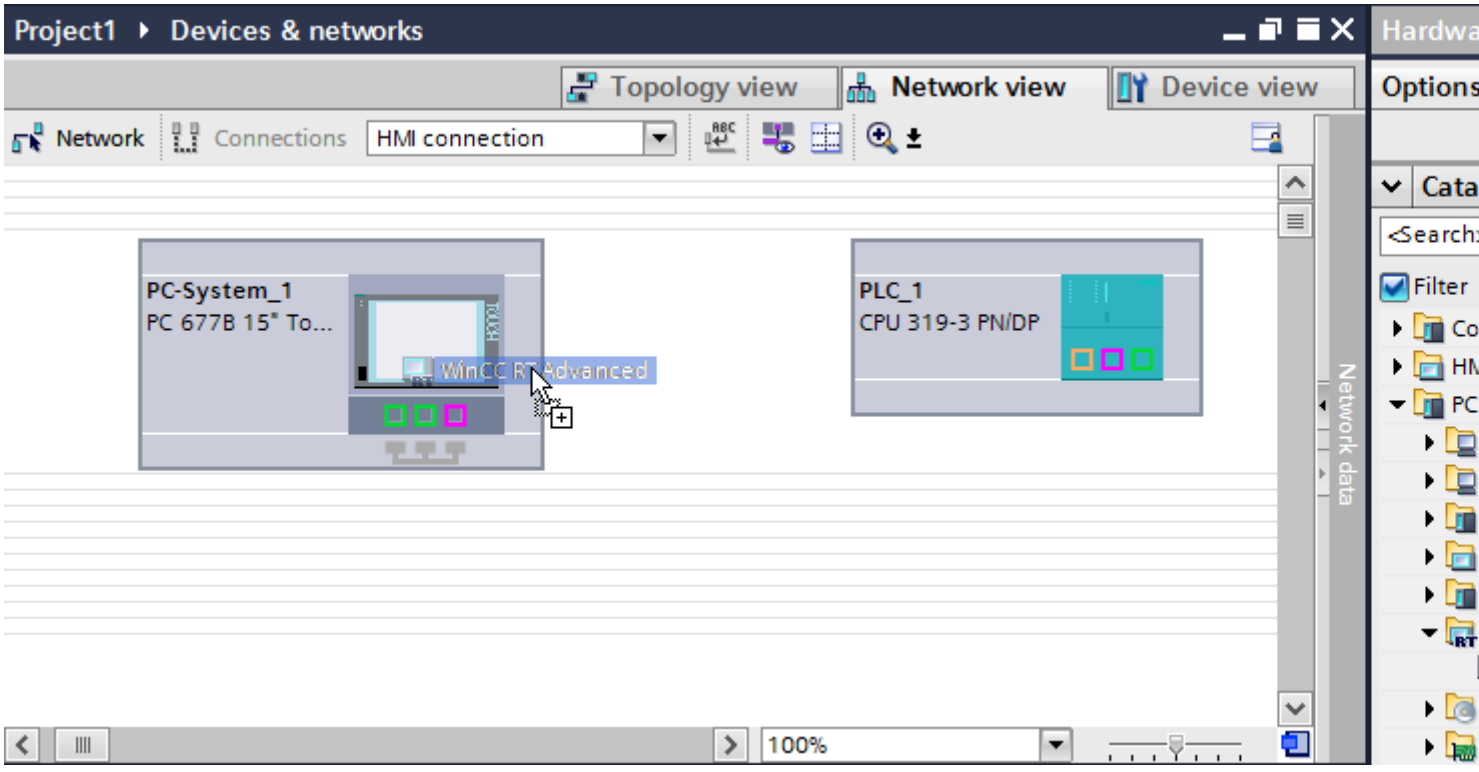
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 300/400 with MPI interface
- SIMATIC PC with MPI interface

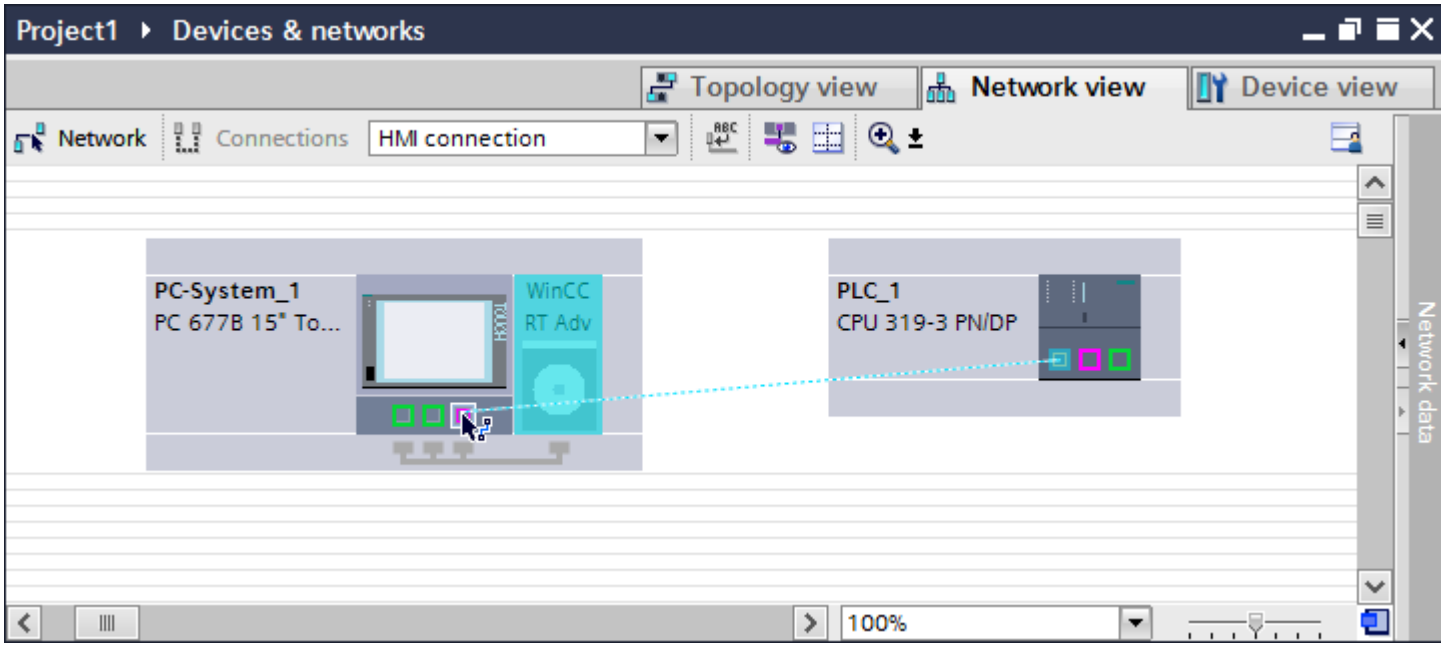
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a WinCC Runtime from the hardware catalog to the PC.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click the MPI interface of the PLC and use a drag-and-drop operation to draw a connection to the MPI interface of the PC.



5. Click the connecting line.
6. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
7. Click the communication partners in the "Network view" and change the MPI parameters in the Inspector window according to the requirements of your project.
   See the chapter "MPI parameters (Page 6296)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 300/400 via MPI.

Configuring an HMI connection via MPI with a PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via MPI in the "Devices & Networks" editor.

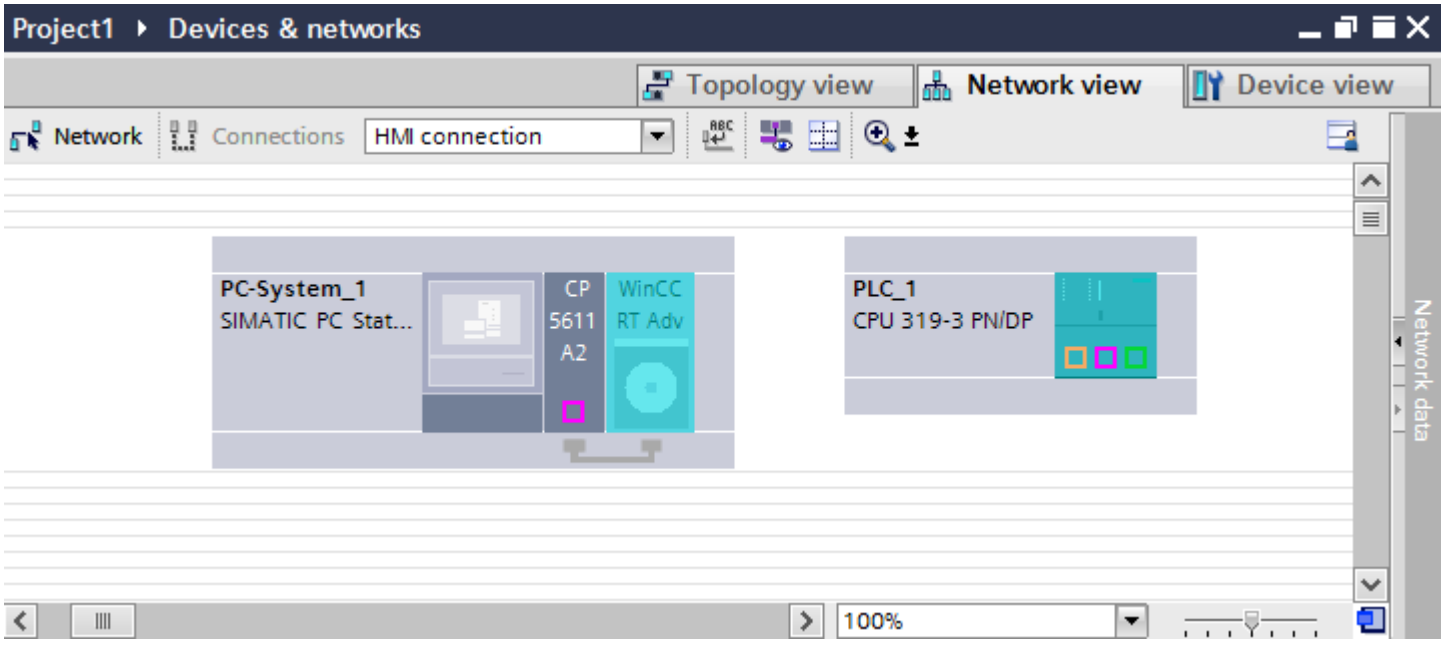
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC S7 300/400 with MPI interface
- PC station with WinCC RT Advanced

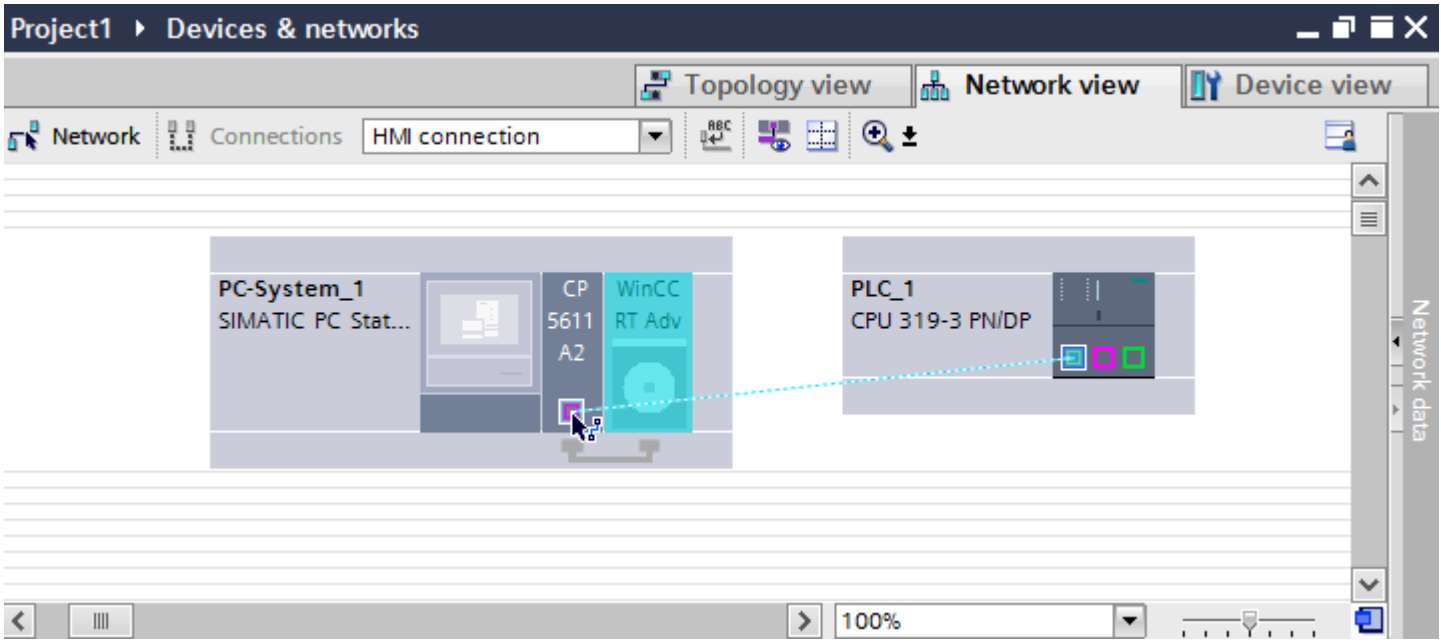
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a PROFIBUS-capable communication processor from the hardware catalog to the WinCC Runtime.



3. Click the PROFIBUS interface of the communication processor and change the interface to "MPI".
4. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

5. Click the MPI interface of the PLC and use a drag-and-drop operation to draw a connection to the MPI interface of the communication processor.



6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the MPI parameters in the Inspector window according to the requirements of your project.
   See the chapter "MPI parameters (Page 6296)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 300/400 via MPI.

MPI parameters

MPI parameters for the HMI connection

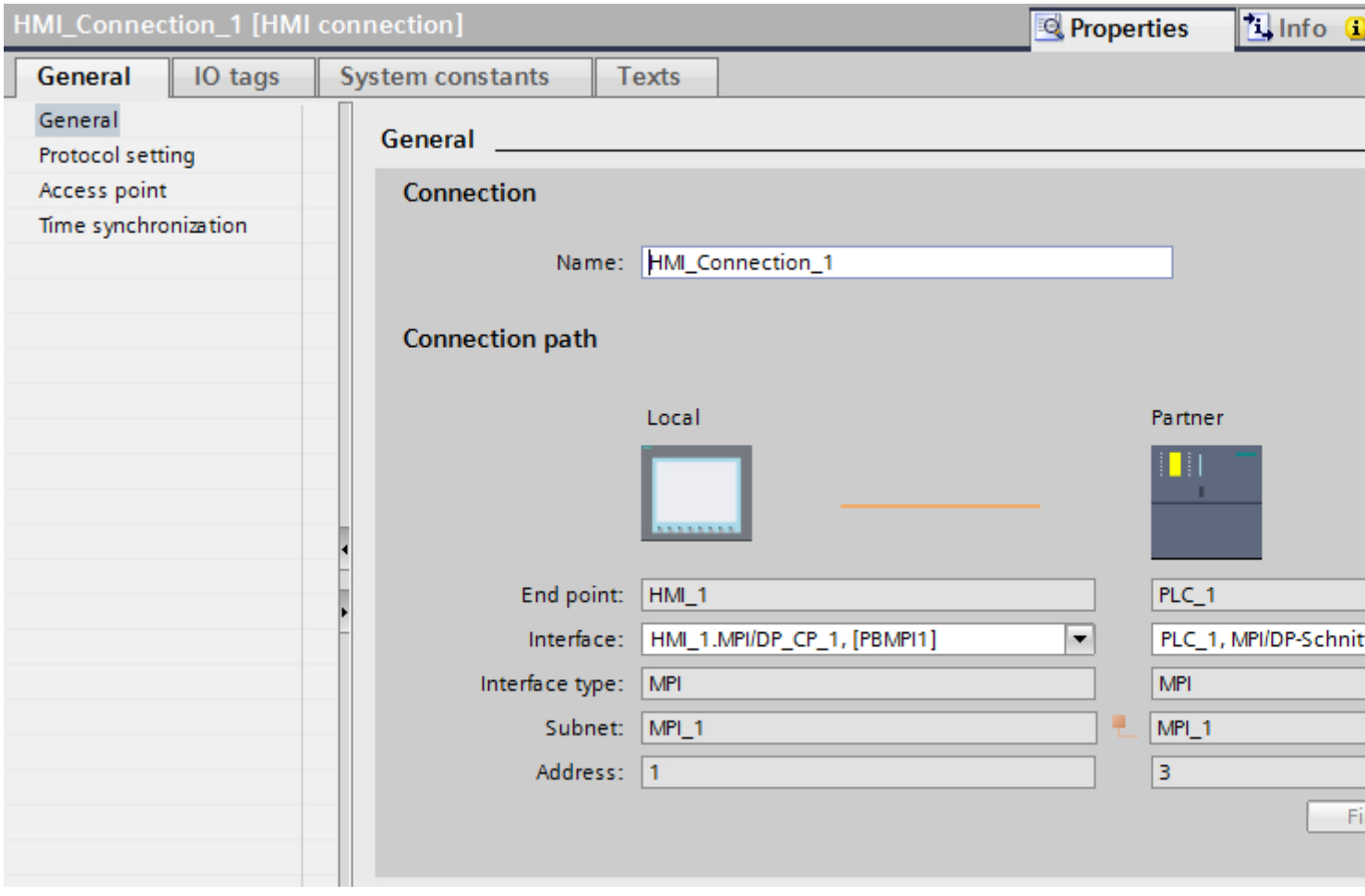
MPI parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Displays whether the devices are networked together.

- displayed if the devices are networked together.

- displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated MPI parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the name of the device. This area is not editable.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area is not editable.
- "Subnet"
  Displays the selected subnet. This area is not editable.
- "Address"
  Displays the MPI address of the device. This area is not editable.
- "Find connection path" button
  Enables the subsequent specification of connections.

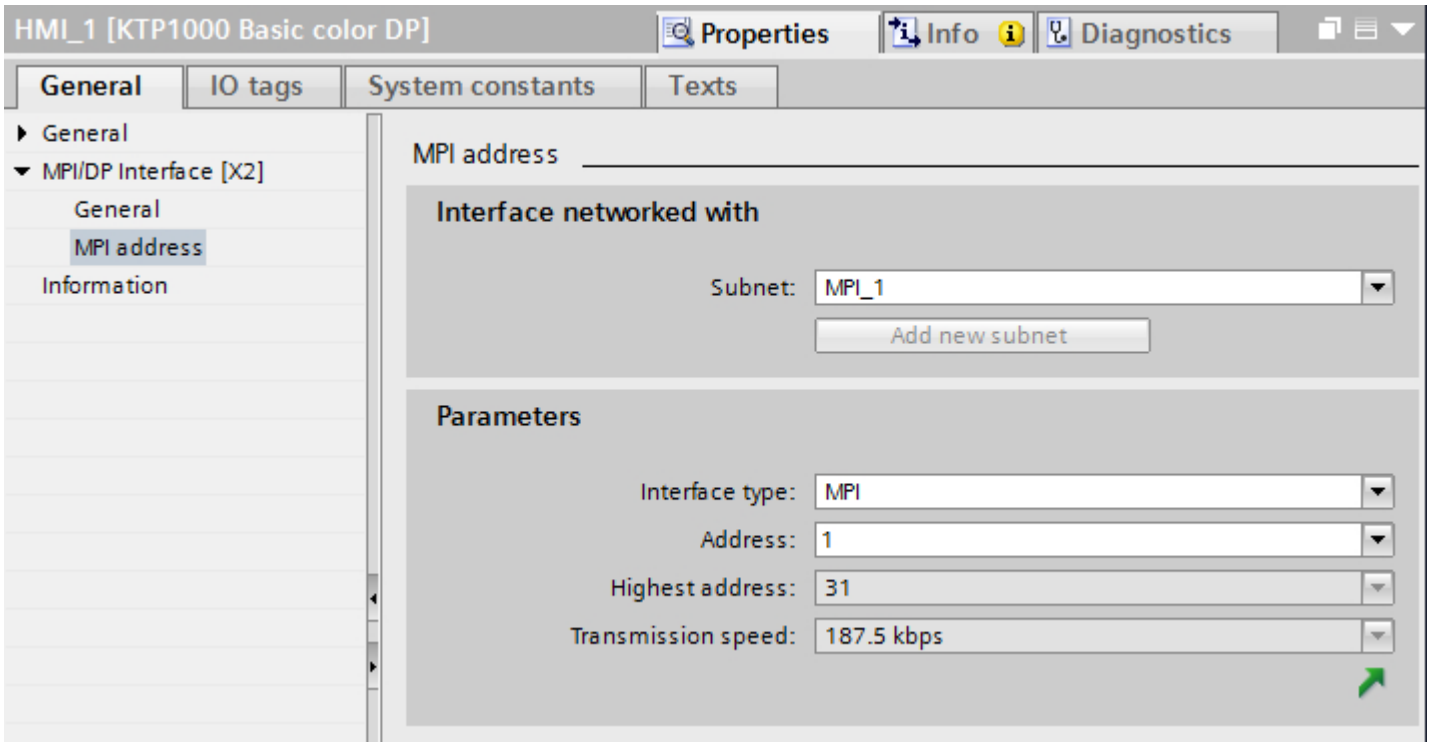
MPI parameters for the HMI device

MPI parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing MPI parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  You assign the interface type in the "Interface type" area. Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the MPI address of the HMI device in the "Address" area. The MPI address must be unique throughout the MPI network.
- "Highest address"
  The "Highest address" area displays the highest address of the MPI network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

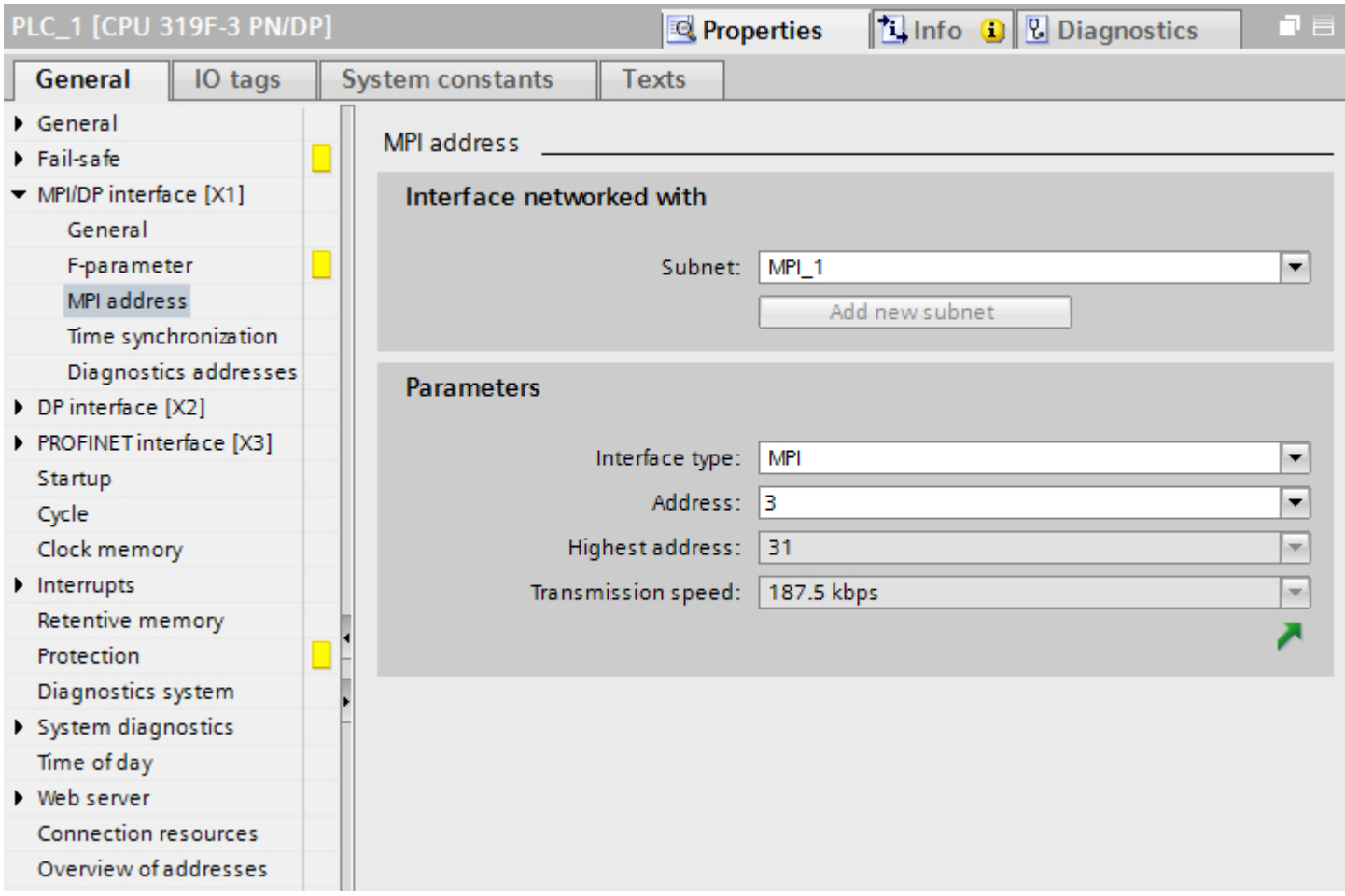
MPI parameters for the PLC

MPI parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the MPI address of the HMI device in the "Address" area. The MPI address must be unique throughout the MPI network.
- "Highest address"
  The "Highest address" area displays the highest address of the MPI network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Addressing of the PLC via MPI

Introduction

Each communication partner must be assigned an MPI network address.

Each S7 module which supports communication functions and is operated the SIMATIC S7-300/400 PLC is assigned a unique MPI address. Only one CPU may be used per rack.

Note

HMI devices cannot be operated with incorrect addressing

Always avoid redundant addressing on the MPI bus.

MPI address of the communication partner of a SIMATIC S7-300

When assigning addresses, you have to distinguish between communication partners with and without separate MPI address.

- If the communications partner has its own MPI address, you only need to define the MPI address.
- If the communication partners do not have a separate MPI address, specify the MPI address of the communications partner used for the connection. In addition, define the slot and rack of a communication partner without its own MPI address.

MPI address of the communication partner of a SIMATIC S7-400

Only S7 modules with an MPI connector are assigned an MPI address. Modules without an MPI connector are addressed indirectly:

- MPI address of the module to which the HMI is connected.
- The slot and the rack of the module with which the HMI device communicates.

12.11.8.5 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Configuring area pointers (Page 6044)

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Use

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1st word	Current screen type															
2nd word	Current screen number															
3rd word	Reserved															
4th word	Current field number															
5th word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" or "40" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

Note

Symbolic addressing is not possible if you are using the "Date/Time" area pointer.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the Date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

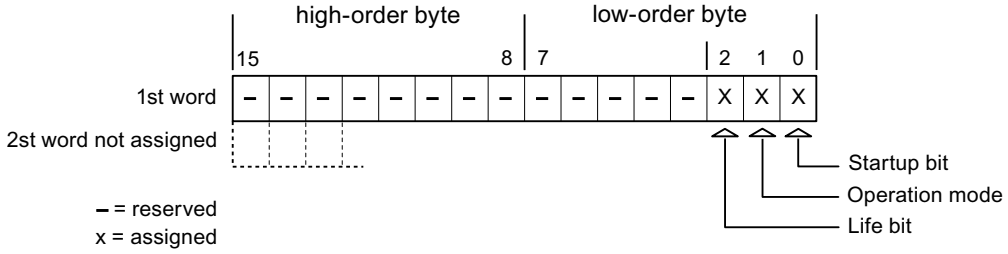
By default, the "Coordination" area pointer has the length of one word and cannot be changed.

Application

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

"Project ID" area pointer

Function

When Runtime starts, a check can be carried out as to whether the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program. If there is no concordance, a system event is given on the HMI device and Runtime is stopped.

Use

Note

HMI connections cannot be switched "online".

The HMI connection in which the "Project ID" area pointer is used must be switched "online".

To use this area pointer, set up the following during the configuration:

- Define the version of the configuration. Values between 1 and 255 are possible.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:
  You enter the data address in the editor "Communication > Connections" under "Address".

Connection failure

A connection failure to a device on which the "Project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "Project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the Engineering System.

"Job mailbox" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	**Setting the date (BCD coded)** [3) 4)]	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	**User logon**	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	**User logoff**	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	**Transfer date/time to PLC**	
	(in the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	**Transfer date/time to PLC**	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	**Update tag**	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in Parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-

No.	Function	
14	**Set time (BCD-coded)**	
50	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	**Screen selection** [2]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	**Reading data record from PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	**Writing data record to PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1]	Only for devices supporting recipes.
[2]	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
[3]	The weekday is ignored on HMI device KTP 600 BASIC PN.
[4]	The weekday is ignored when you configure the "Date/Time PLC" area pointer.

"Data record" area pointer

"Data record" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization via the data mailbox

Data records are always transferred directly, which means that the tag values are read straight from an address or written straight to an address configured for this tag without being redirected via an interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system alarm.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then, for example, process, edit, or save these values in the HMI device.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data mailbox free
2	0000 0010	Transferring
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer started by the operator in the recipe view

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data record and sets the data record number to 0.	Abort with system event.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data record.	Abort with system event.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxs No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the function from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	

Step	Action	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible
- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system alarms
- Triggered by function
  Output of system alarms
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data record.

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Displaying tags with Runtime Advanced and Panels (Page 4270)

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

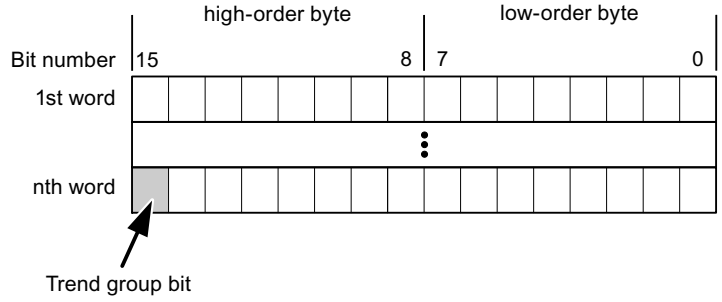
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Permitted data types for trends

For SIMATIC S7

You assign one bit to each trend during configuration. Tags and array tags of the "Word" or "Int" data type are permitted.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with Alarms (Page 4293)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 300/400	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIME

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

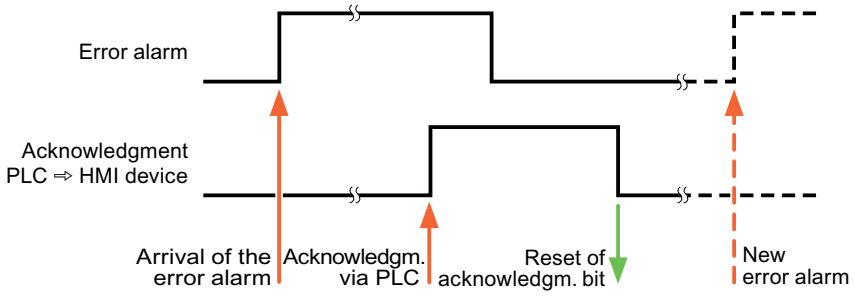
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment

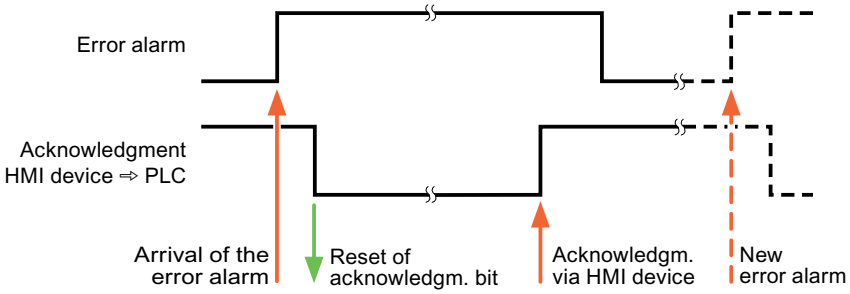
tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



LED mapping

Function

Keyboard devices have LEDs in the function keys. These LEDs can be activated from the PLC. Thus, it is possible, for example, to signal to the user which key he should press in a given situation by lighting up an LED.

Note

The LED function cannot be configured for Basic Panels.

Requirements

In order to activate an LED, an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration.

LED assignment

The assignment of the individual LEDs to the bits in the LED tags is specified when the function keys are configured. In this process, the "LED tag" and the assigned "Bit" are specified for each function key in the "General" group of the properties window.

The "Bit" bit number designates the first of two consecutive bits that control the following LED statuses.

Bit n+ 1	Bit n	LED function	
		All Mobile Panels, all Operator Panels, all Multi Panels, all Comfort Panels	Panel PCs
0	0	Off	Off
0	1	Rapid flashing	Flashing
1	0	Slow flashing	Flashing
1	1	On permanently	On permanently

12.11.8.6 Performance features of communication

S7-300/400 device dependency

Communication with the SIMATIC S7-300/400 controller

If you use devices from an earlier version of the TIA Portal with TIA Portal V13, it may not be possible to configure integrated connections to certain HMI devices.

Basic Panels V11.0

HMI devices	SIMATIC S7-300/400
KP300 Basic	Yes
KP400 Basic	Yes
KTP400 Basic PN	Yes
KTP600 Basic DP	Yes
KTP600 Basic PN	Yes
KTP1000 Basic DP	Yes
KTP1000 Basic PN	Yes
TP1500 Basic PN	Yes

Basic Panels V12.0

HMI devices	SIMATIC S7-300/400
KP300 Basic	Yes
KP400 Basic	Yes
KTP400 Basic PN	Yes
KTP600 Basic DP	Yes
KTP600 Basic PN	Yes

HMI devices	SIMATIC S7-300/400
KTP1000 Basic DP	Yes
KTP1000 Basic PN	Yes
TP1500 Basic PN	Yes

Basic Panels V13.0

HMI devices	SIMATIC S7-300/400
KTP400 Basic PN	Yes
KTP700 Basic PN	Yes
KTP700 Basic DP	Yes
KTP900 Basic PN	Yes
KTP1200 Basic PN	Yes
KTP1200 Basic DP	Yes

Basic Panels V13.0.1

HMI devices	SIMATIC S7-300/400
KTP400 Basic PN	Yes
KTP700 Basic PN	Yes
KTP700 Basic DP	Yes
KTP900 Basic PN	Yes
KTP1200 Basic PN	Yes
KTP1200 Basic DP	Yes

Panels V11.0

HMI devices	SIMATIC S7-300/400
OP 73	Yes
OP 77A	Yes
OP 77B	Yes
TP 177A	Yes
TP 177A Portrait	Yes
TP 177B 4"	Yes
TP 177B 6" mono	Yes
TP 177B 6"	Yes
OP 177B 6" mono	Yes
OP 177B 6"	Yes

HMI devices	SIMATIC S7-300/400
TP 277 6"	Yes
OP 277 6"	Yes

Multi Panels V11.0

HMI devices	SIMATIC S7-300/400
MP 177 6" Touch	Yes
MP 277 8" Key	Yes
MP 277 10" Key	Yes
MP 277 10" Touch	Yes
MP 377 12" Key	Yes
MP 377 12" Touch	Yes
MP 377 15" Touch	Yes
MP 377 19" Touch	Yes

Mobile Panels V11.0

HMI devices	SIMATIC S7-300/400
Mobile Panel 177 6" DP	Yes
Mobile Panel 177 6" PN	Yes
Mobile Panel 277 8"	Yes
Mobile Panel 277 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	Yes
Mobile Panel 277 10"	Yes

Mobile Panels V12.0

HMI devices	SIMATIC S7-300/400
Mobile Panel 177 6" DP	Yes
Mobile Panel 177 6" PN	Yes
Mobile Panel 277 8"	Yes
Mobile Panel 277 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	Yes
Mobile Panel 277 10"	Yes

Mobile Panels V13.0.1

HMI devices	SIMATIC ET 200 CPU
KTP 700 Mobile	Yes
KTP 900 Mobile	Yes

Comfort Panels V11.0

HMI devices	SIMATIC S7-300/400
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V12.0

HMI devices	SIMATIC S7-300/400
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes

HMI devices	SIMATIC S7-300/400
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V13.0

HMI devices	SIMATIC S7-300/400
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V13.0.1

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KP400 Comfort	No	Yes	Yes	Yes	Yes
KTP400 Comfort	No	Yes	Yes	Yes	Yes
KTP400 Comfort Portrait	No	Yes	Yes	Yes	Yes

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KP700 Comfort	No	Yes	Yes	Yes	Yes
TP700 Comfort	No	Yes	Yes	Yes	Yes
TP700 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP900 Comfort	No	Yes	Yes	Yes	Yes
TP900 Comfort	No	Yes	Yes	Yes	Yes
TP900 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP1200 Comfort	No	Yes	Yes	Yes	Yes
TP1200 Comfort	No	Yes	Yes	Yes	Yes
TP1200 Comfort Portrait	No	Yes	Yes	Yes	Yes
KP1500 Comfort	No	Yes	Yes	Yes	Yes
TP1500 Comfort	No	Yes	Yes	Yes	Yes
TP1500 Comfort Portrait	No	Yes	Yes	Yes	Yes
TP1900 Comfort	No	Yes	Yes	Yes	Yes
TP1900 Comfort Portrait	No	Yes	Yes	Yes	Yes
TP2200 Comfort	No	Yes	Yes	Yes	Yes
TP2200 Comfort Portrait	No	Yes	Yes	Yes	Yes

Runtime V11.0

HMI devices	SIMATIC S7-300/400
WinCC RT Advanced	Yes

Runtime V12.0

HMI devices	SIMATIC S7-300/400
WinCC RT Advanced	Yes

Runtime V13.0

HMI devices	SIMATIC S7-300/400
WinCC RT Advanced	Yes

Runtime V13.0.1

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
WinCC RT Advanced	No	Yes	Yes	Yes	Yes

Permitted data types for SIMATIC S7 300/400

Permitted data types for connections with SIMATIC S7 300/400

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
BOOL	1-bit
BYTE	1 byte
WORD	2 bytes
DWORD	4 bytes
CHAR	1 byte
INT	2 byte
DINT	4 bytes
REAL	4 bytes
TIME	4 bytes
DATE	2 bytes
TIME_OF_DAY, TOD	4 bytes
S5TIME	2 bytes
COUNTER	2 bytes
TIMER	2 bytes
DATE_AND_TIME	8 bytes
STRING	(2+n) bytes, n = 0 to 254

12.11.8.7 Creating connections in the "Connections" editor

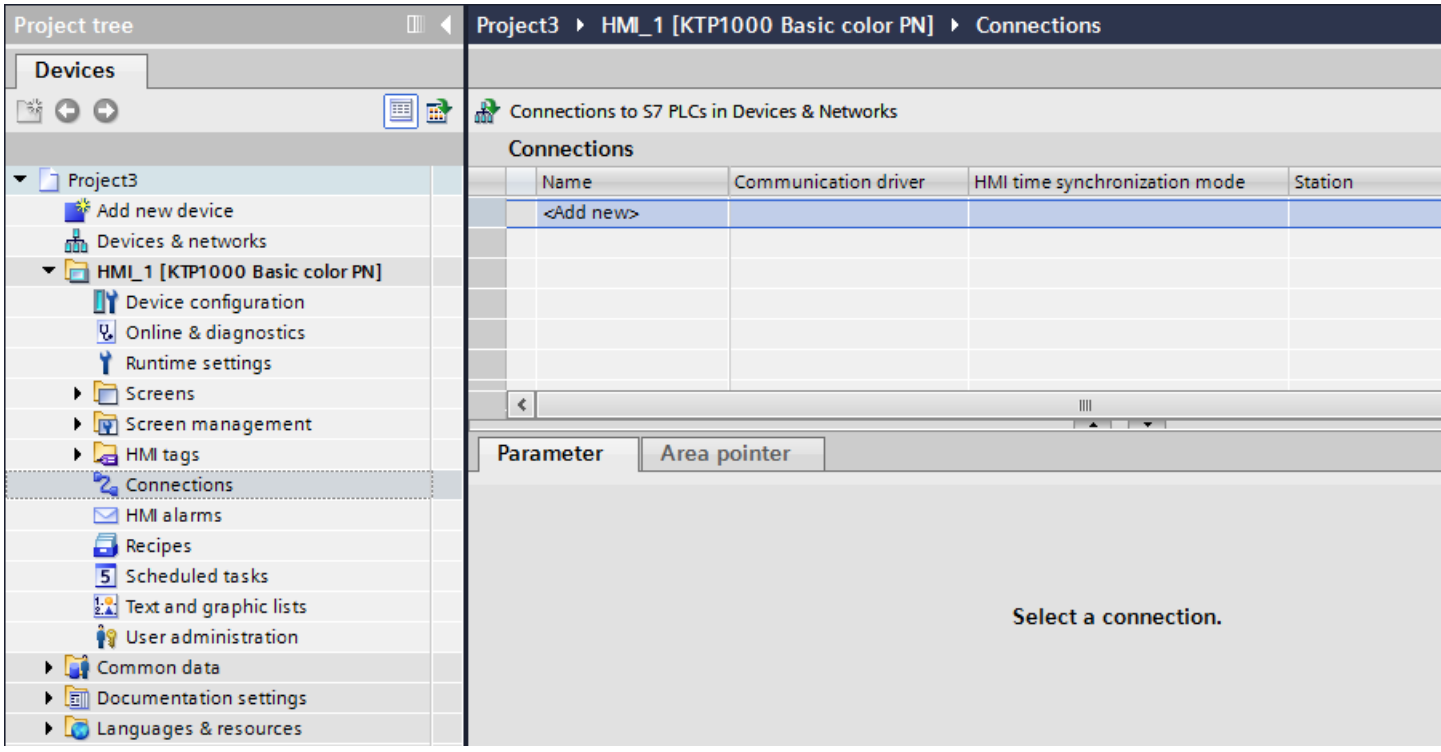
Creating a PROFINET connection

Requirements

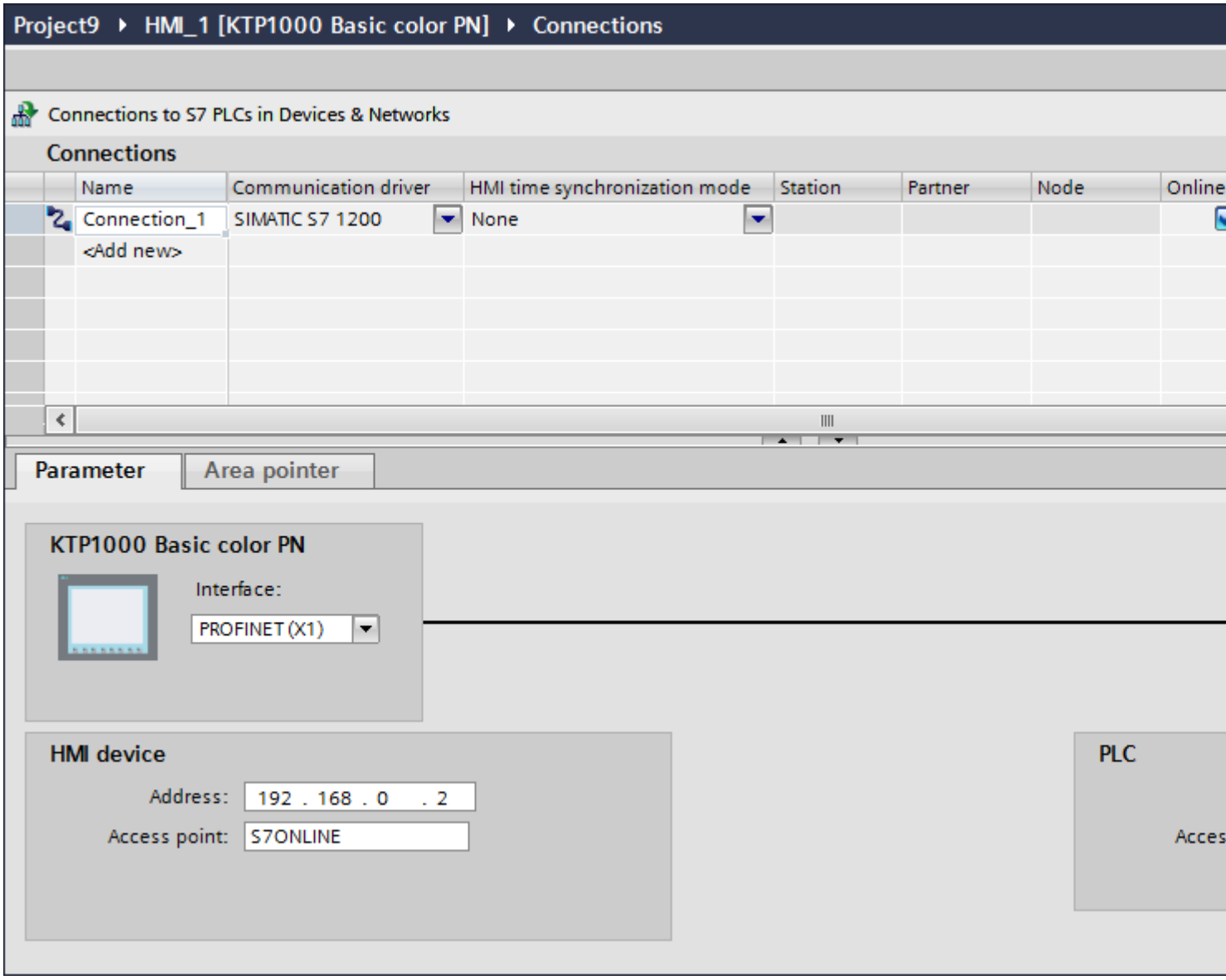
- A project is open.
- An HMI device with a PROFINET interface has been created.

Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.



4. Click the name of the connection.
5. Select a PROFINET interface of the HMI device in the Inspector window under "Parameters > Interface".
6. Set the IP addresses of the communication partners in the Inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

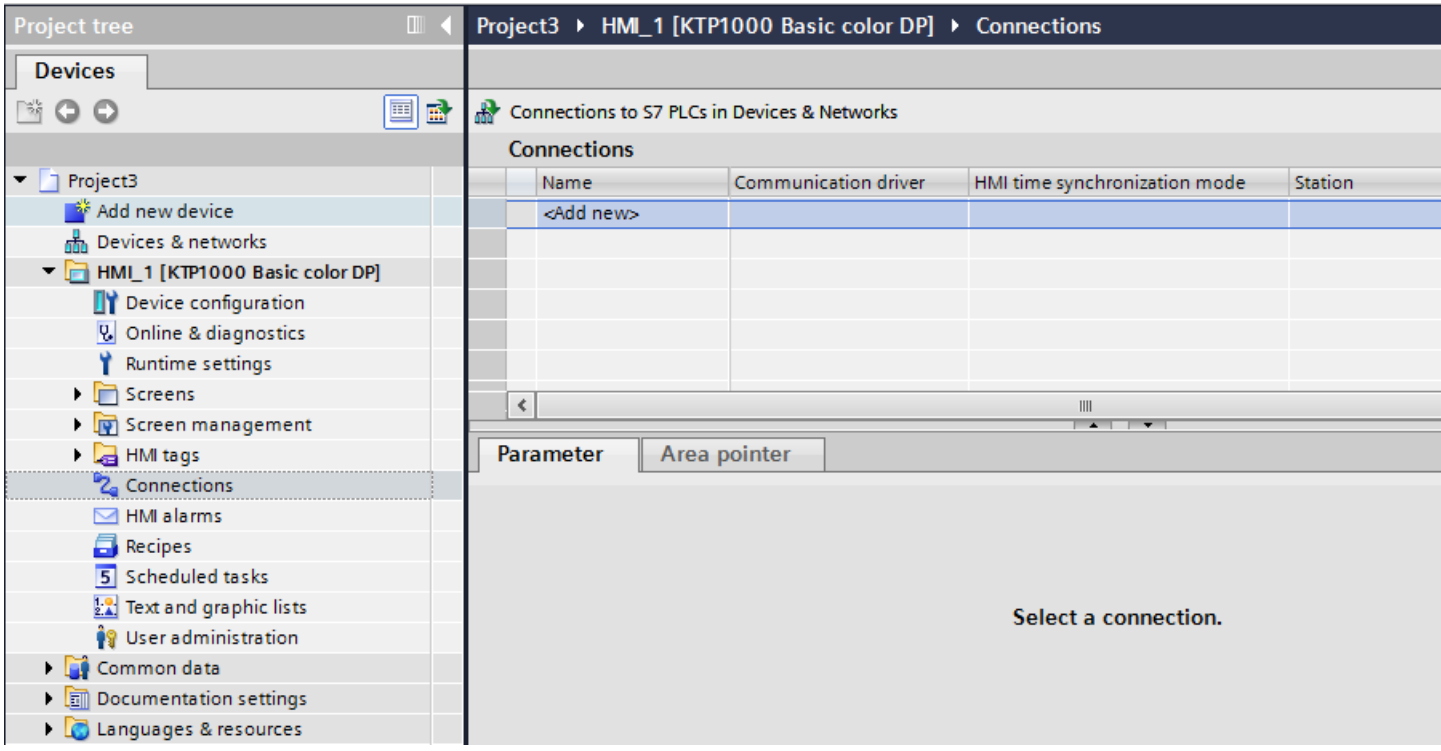
Creating a PROFIBUS connection

Requirements

- A project is open.
- An HMI device with a PROFIBUS interface has been created.

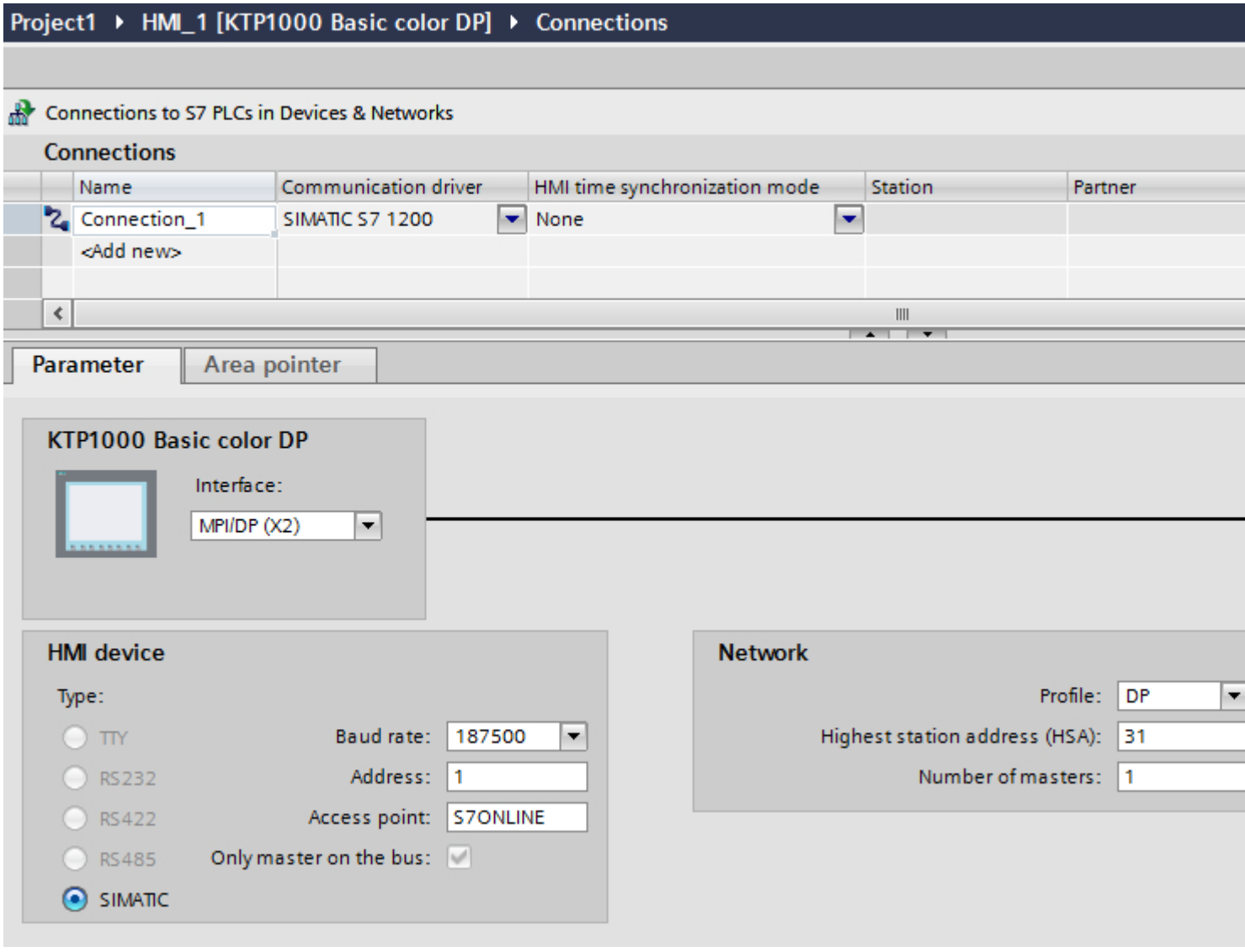
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select the "MPI/DP" interface in the Inspector window under "Parameters > Interface".

6. Select the "DP" profile in the Inspector window under "Parameters > Network".



7. Set the addresses of the communication partners in the inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

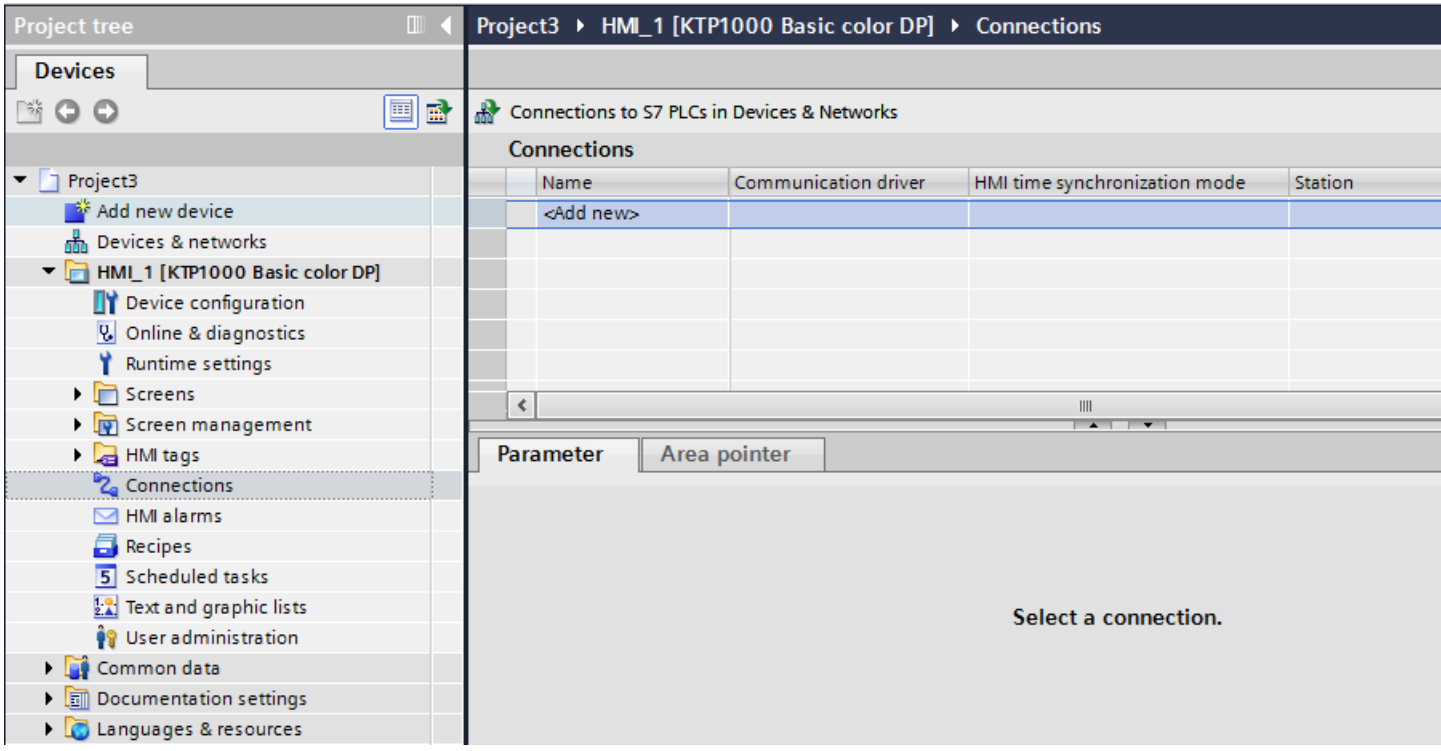
Creating an MPI connection

Requirements

- A project is open.
- An HMI device with an MPI interface has been created.

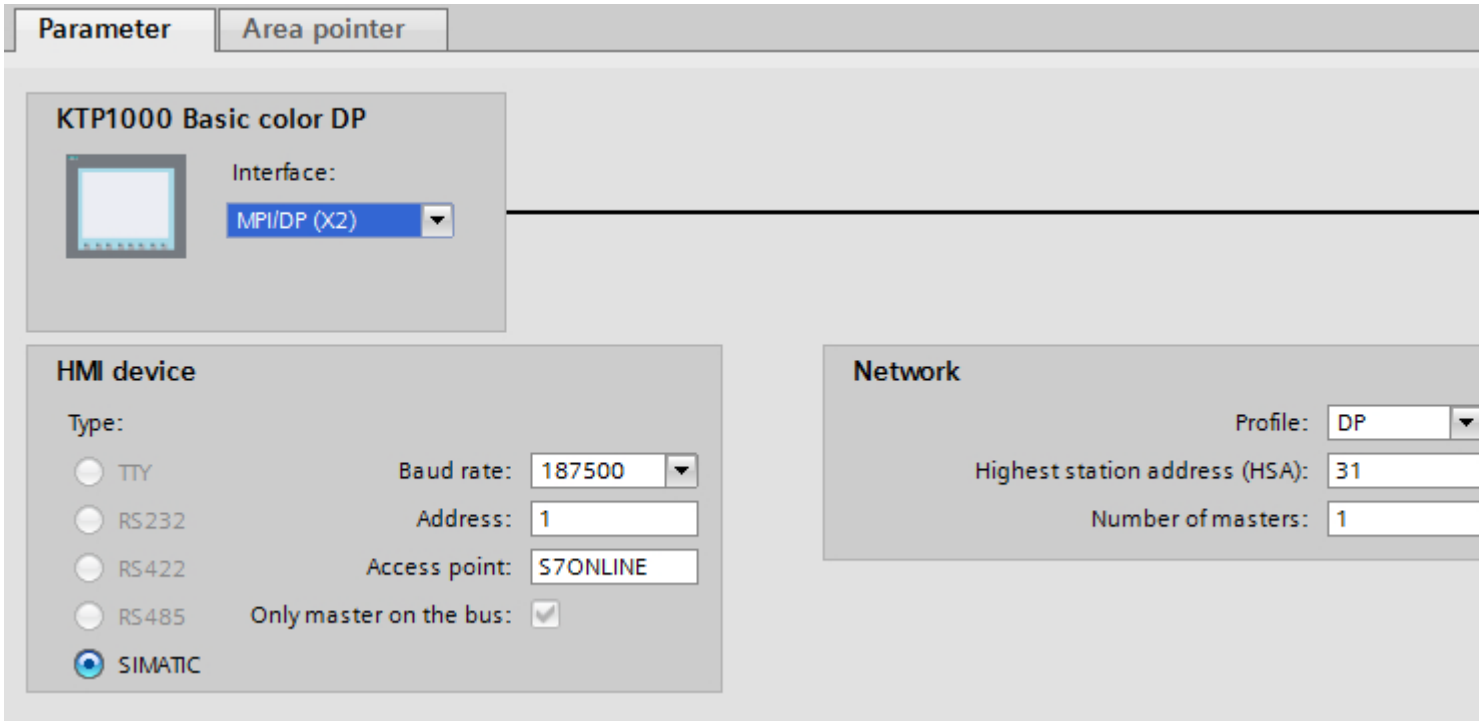
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select the "MPI/DP" interface in the Inspector window under "Parameters > Interface".

6. Select the "MPI" profile in the Inspector window under "Parameters > Network".



7. Set the addresses of the communication partners in the inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

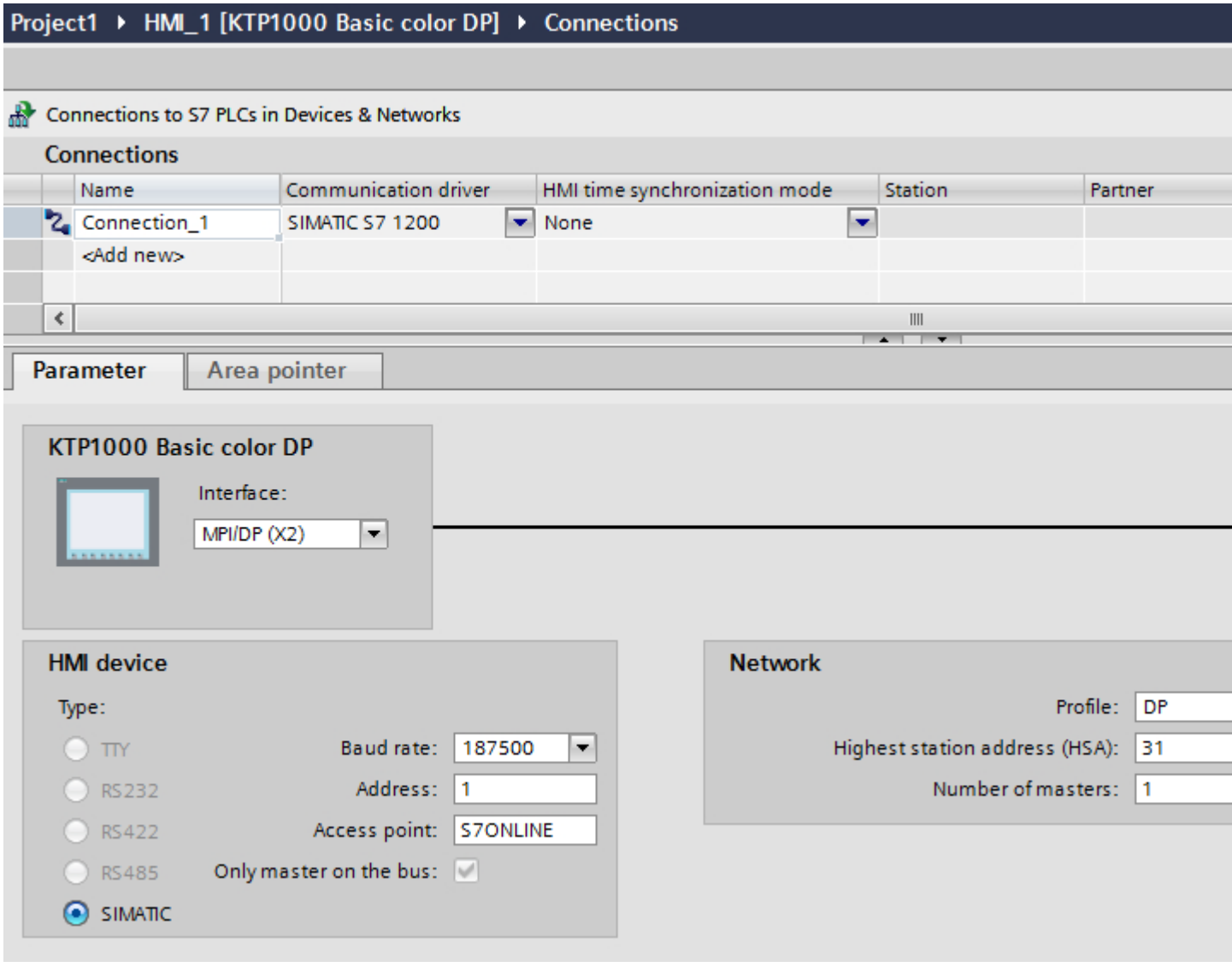
Parameters for the connection

Parameters for the connection (SIMATIC S7 300/400)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.


Project1 ▸ HMI_1 [KTP1000 Basic color DP] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Connections
Name
Communication driver
HMI time synchronization mode
Station
Partner
Connection_1
SIMATIC S7 1200
None
<Add new>
Parameter
Area pointer
KTP1000 Basic color DP
Interface:
MPI/DP (X2)
HMI device
Type:
TTY
RS232
RS422
RS485
SIMATIC
Baud rate:
187500
Address:
1
Access point:
S7ONLINE
Only master on the bus:
Network
Profile:
DP
Highest station address (HSA):
31
Number of masters:
1

Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"
- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Expansion slot"
  Defines the number of the expansion slot of the CPU to be addressed.

- "Rack"
  Defines the rack number of the CPU to be addressed.
- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance.
  Disable cyclic mode if you are operating several HMI devices in parallel.

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.
- "Sole master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).
  In S7 200, you must set an HMI device as the master.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"
  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200 PLCs.

MPI parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the MPI network.

- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  For "Address", you set the MPI address of the HMI device. The MPI address must be unique throughout the MPI network.
- "Sole master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master). If you have connected only slaves to the HMI device, you must therefore disable the "Sole master on bus" safety feature.
  In S7 200, you must set an HMI device as the master.

Parameters for the network

Under "Network", you set the parameters for the MPI network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "MPI". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual MPI address. The setting must be identical throughout the network.
- "Number of masters"
  This setting is not required for MPI.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the MPI address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Cyclic mode"
  When cyclic mode is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This improves system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200.

Cyclic operation

Handling the "Cyclic operation" selection

If "Cyclic operation" is enabled, the HMI device sends a message frame to the CPU at the beginning of communication indicating that certain tags are required on a recurring basis.

The CPU then always transmits the data at the same cyclic interval. This saves the HMI device from having to output new requests for the data.

If cyclic operation is disabled, the HMI device sends a request whenever information is required.

Additional properties:

- Cyclic operation reduces data transmission load at the HMI device. The PLC resources are used to relieve load on the HMI device.
- The PLC only supports a certain number of cyclic services. The HMI device handles the operation if the PLC cannot provide any further resources for cyclic services.
- The HMI device generates the cycle if the PLC does not support the cyclic mode.
- Screen tags are not integrated in cyclic operation.
- Cyclic mode is only set up at the restart of Runtime.
- The HMI device transfers several jobs to the PLC if cyclic mode is enabled, depending on the PLC.
- The HMI device only transfers one job to the PLC if cyclic mode is disabled.

12.11.9 Communicating with the SIMATIC S7-1500 Software Controller

12.11.9.1 Communication with SIMATIC S7-1500 Software Controller

Introduction

This section describes the communication between a HMI device and a SIMATIC S7-1500 Software Controller.

You can configure the following communication channels for the SIMATIC S7-1500 Software Controller :

- PROFINET

HMI connection for communication

You configure connections between the HMI device and SIMATIC S7-1500 Software Controller in the "Devices & Networks" editor.

12.11.9.2 Communication via PROFINET

Basics of HMI connections to an SIMATIC S7-1500 Software Controller

Introduction

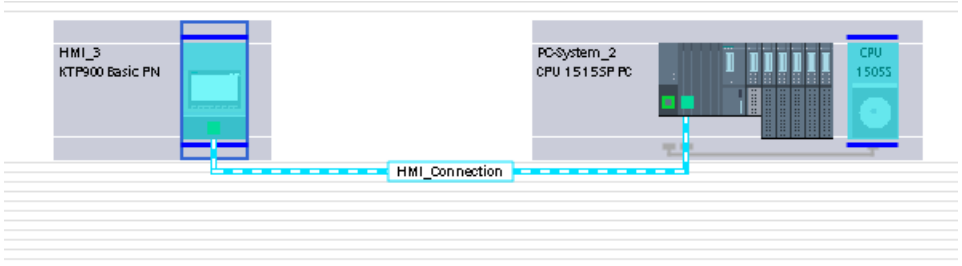
In a PC station, the SIMATIC S7-1500 Software Controller can support the following HMI connections depending on the network:

- Direct HMI connection
- Routed HMI connection to a PLC in another subnet

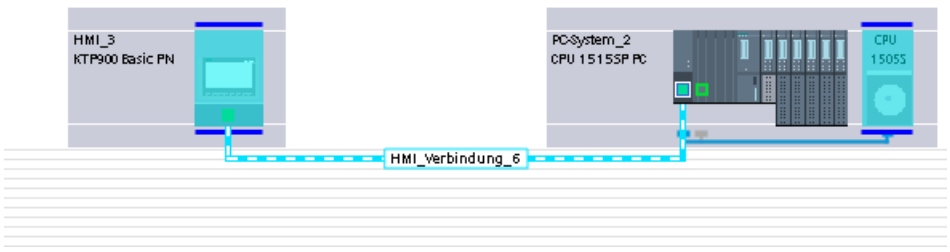
You can find additional information on networking a PC station with an SIMATIC S7-1500 Software Controller in the documentation for the SIMATIC S7-1500 Sofware Controller.

Direct HMI connection to an SIMATIC S7-1500 Software Controller

If a local interface is available on the PC station with the SIMATIC S7-1500 Software Controller, this interface is automatically assigned to the SIMATIC S7-1500 Software Controller when the HMI connection is established. An HMI connection created like this is a direct connection.

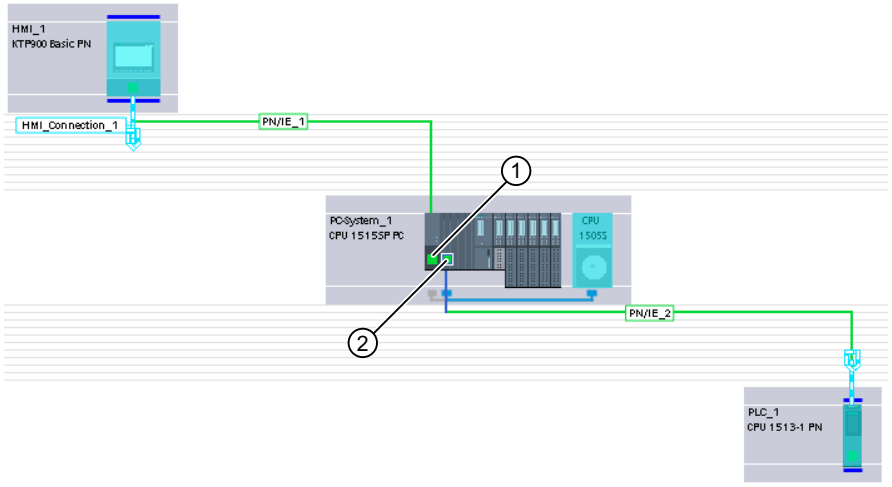


If an interface other than the local interface has been configured for communication on the PC station, the HMI connection is classified as a "routed connection". The HMI connection is, however, shown as a direct connection:



Routed HMI connection over an SIMATIC S7-1500 Software Controller

The SIMATIC S7-1500 Software Controller supports the routing functionality. The PC station must, however, have at least two PN/IE interfaces. The figure below shows the configuration of an HMI connection to an S7 CPU routed through the SIMATIC S7-1500 Software Controller.



① Local interface or interface retrofitted over a communication processor

Required settings under "Interface assignment":

- "PC station"

② "X1" local interface for process communication

Required settings under "Interface assignment":

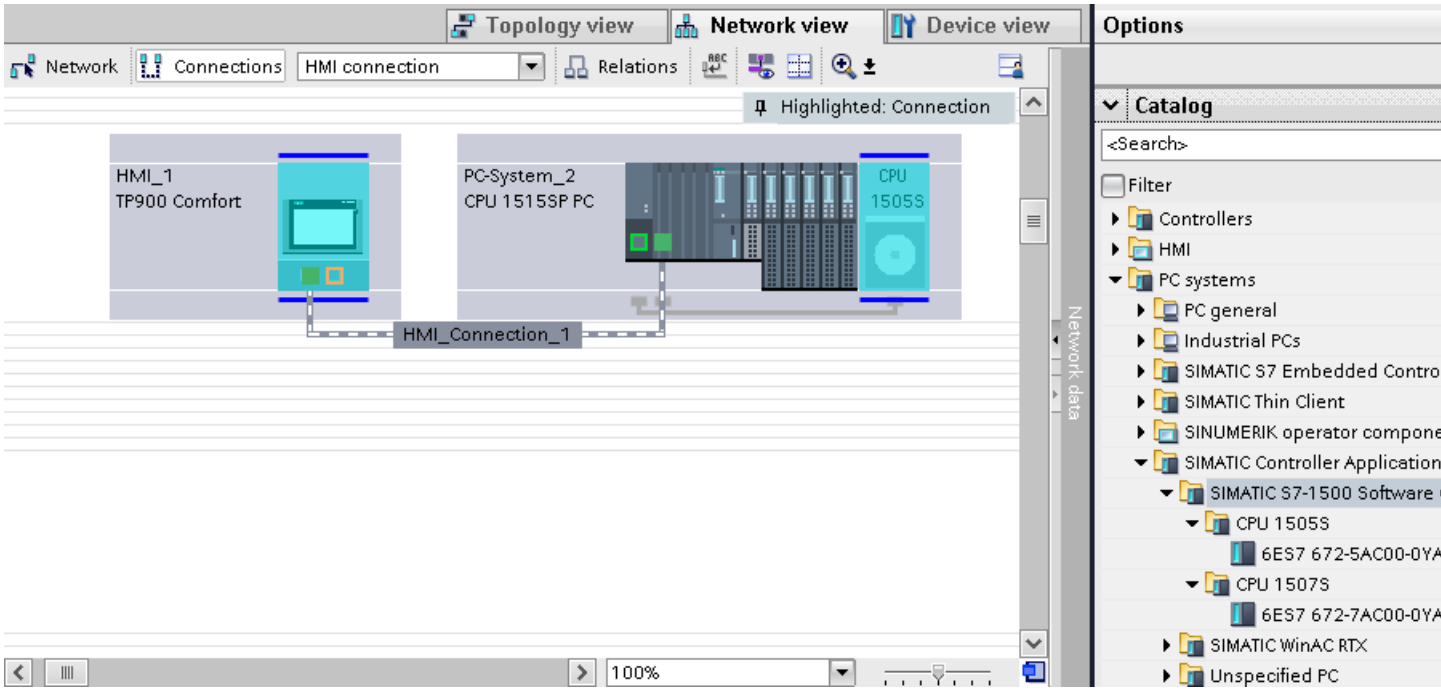
- "Software_PLC_1" (set automatically)

Configuring an HMI connection

Communication via PROFINET

HMI connections via PROFINET

If you have inserted an HMI device and a SIMATIC S7-1500 Software Controller in the project, connect the two PROFINET interfaces in the "Devices & Networks" editor.



You can also connect multiple HMI devices to a single SIMATIC S7-1500 Software Controller and multiple SIMATIC S7-1500 Software Controller to a single HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET

Introduction

You configure a HMI connection between HMI devices and a SIMATIC S7-1500 Software Controller via PROFINET or Ethernet in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

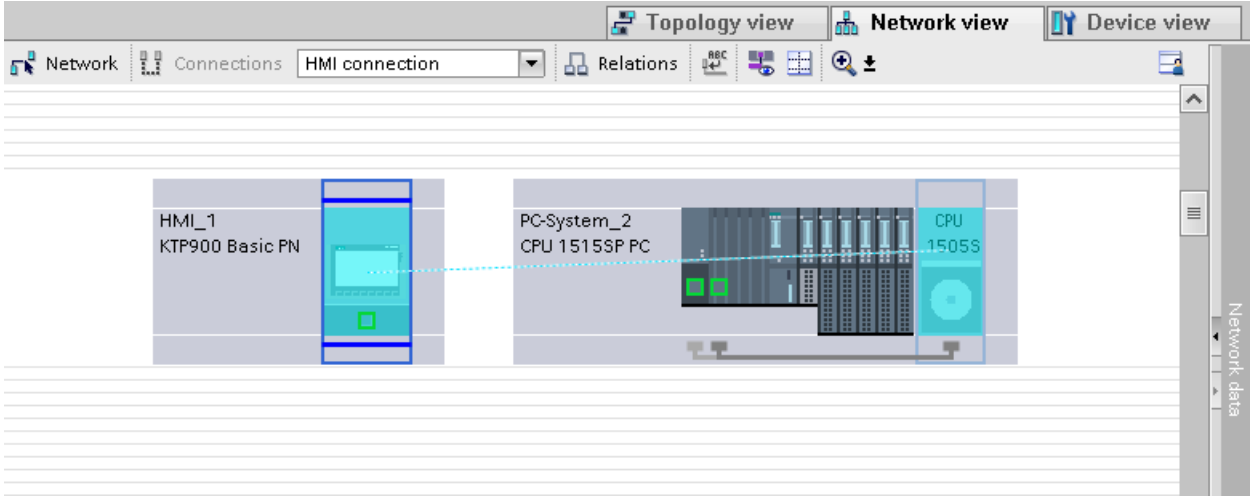
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- HMI device with PROFINET or Ethernet interface
- SIMATIC S7-1500 Software Controller on PC system with PROFINET interface.

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.
3. Click the HMI device and use a drag-and-drop operation to draw a connection to the SIMATIC S7-1500 Software Controller.



4. Click the connecting line.

5. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
6. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the section "PROFINET parameters (Page 6350)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7-1500 Software Controller. The IP address and subnet mask connection parameters are configured.

Configuring an HMI connection

Communication via PROFINET

Communication via PROFINET

This section describes communication over PROFINET between a WinCC Runtime and an SIMATIC S7-1500 Software Controller.

You can use the following WinCC Runtimes as an HMI device:

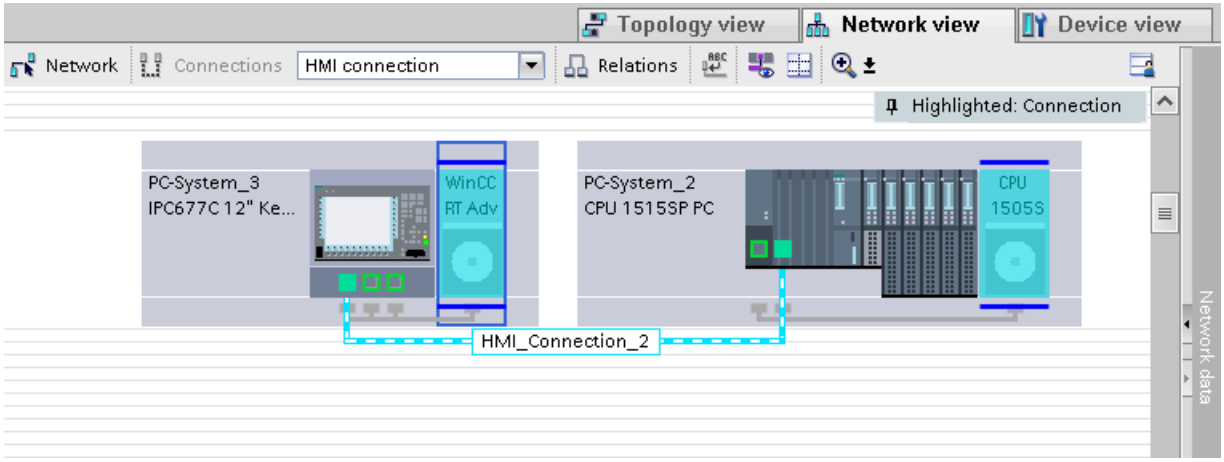
- WinCC RT Advanced
- WinCC RT Professional

WinCC Runtime as HMI device

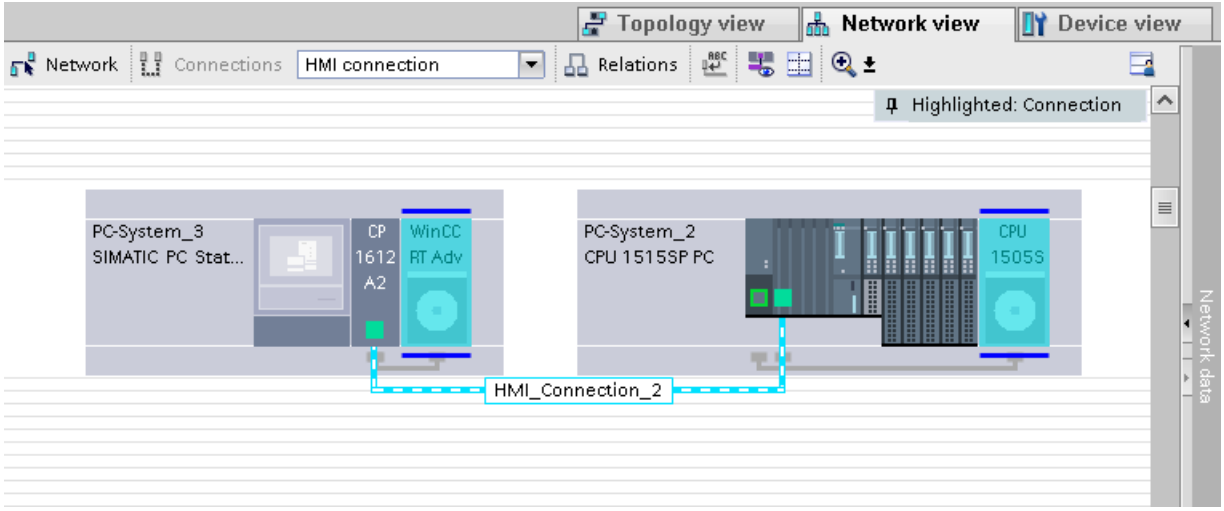
Configure the HMI connections between a WinCC Runtime and an SIMATIC S7-1500 Software Controller in the "Devices & Networks" editor.

If you want to use a WinCC Runtime as an HMI device, you have the following options:

1. You create a SIMATIC PC and configure a WinCC Runtime on this SIMATIC PC.
   In this way you use the SIMATIC PC with a WinCC Runtime as an HMI device.



2. You create a WinCC Runtime and configure a communication processor to the Runtime.
   In this way you use your configuration PC with a WinCC Runtime as an HMI device.



You can also connect multiple HMI devices to a single SIMATIC S7-1500 Software Controller and multiple SIMATIC S7-1500 Software Controller to a single HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET with a PC

Introduction

You configure an HMI connection between an HMI device and a SIMATIC S7-1500 Software Controller via PROFINET or Ethernet in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

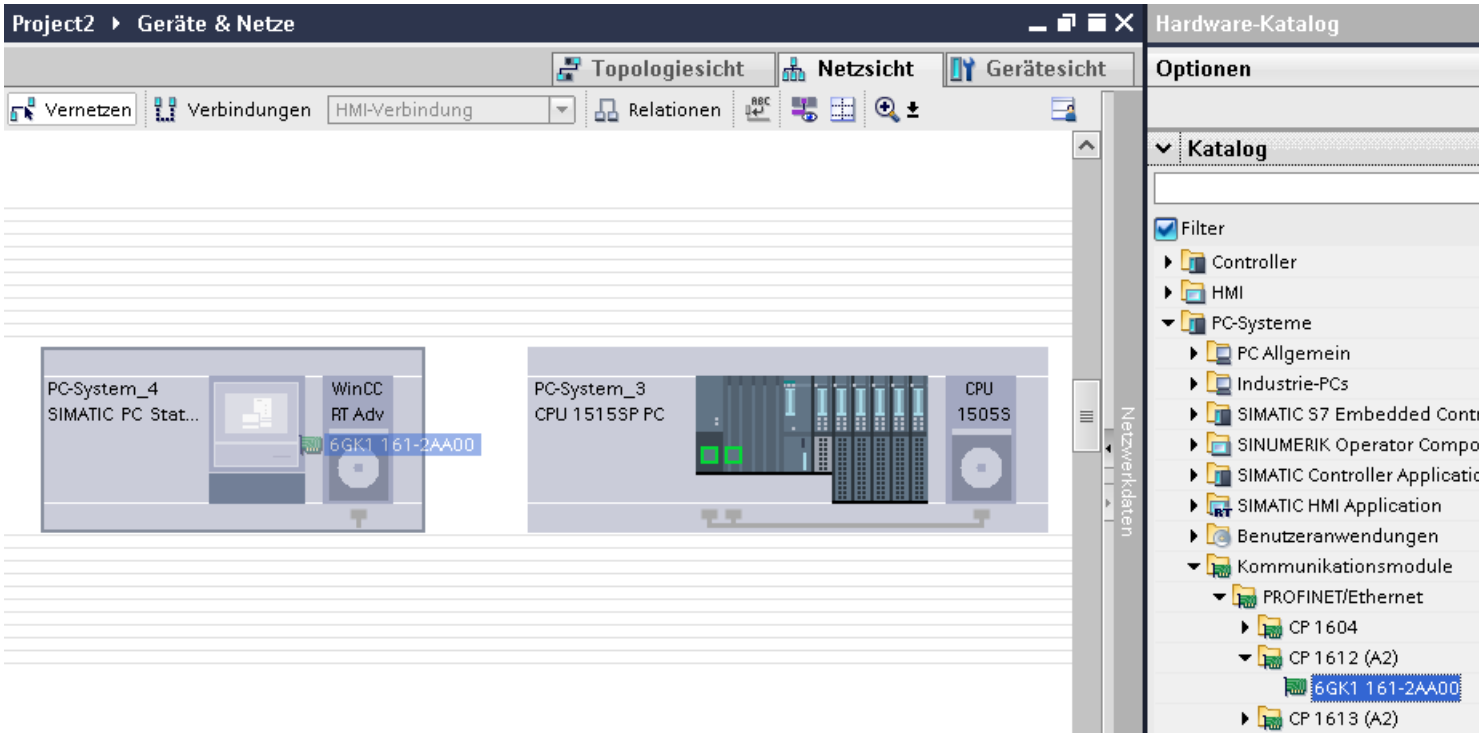
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- PC station with an SIMATIC S7-1500 Software Controller with PROFINET interface
- PC station with a "WinCC Runtime" application

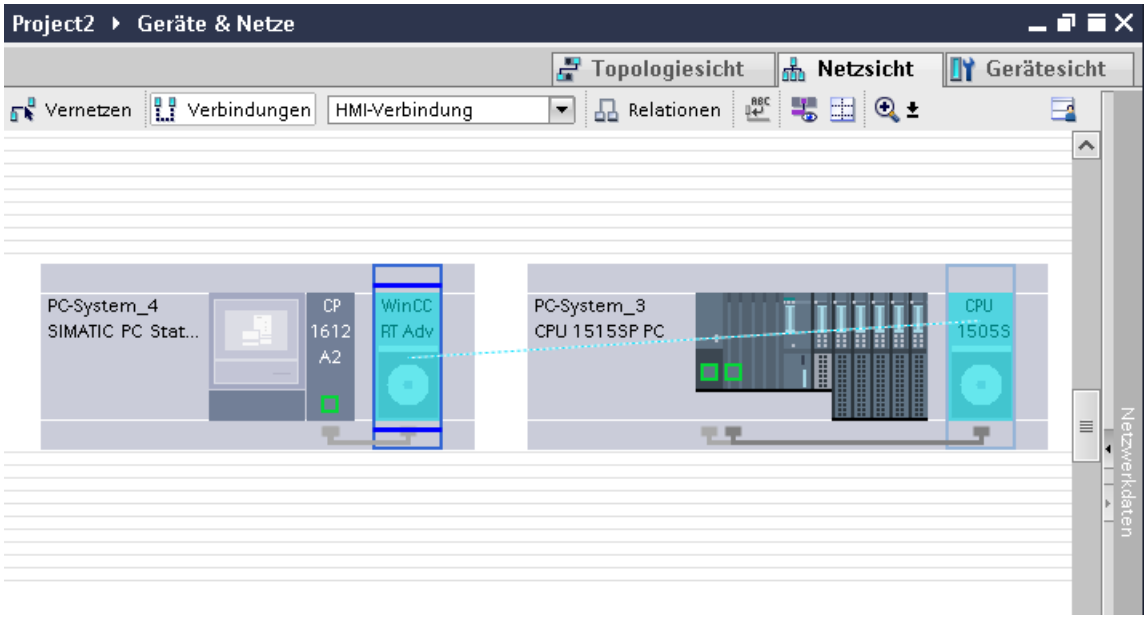
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a PROFINET-capable communication processor from the hardware catalog to the WinCC Runtime.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click on WinCC Runtime in the PC station and create a connection to the SIMATIC S7-1500 Software Controller with drag-and-drop.



5. Click on the connected interface in the PC station and select the "PC station" entry under "Interface assignment" in the Inspector window.
6. Click the connecting line.
   The connection is displayed graphically in the Inspector window.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the section "PROFINET parameters (Page 6350)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7-1500 Software Controller. The IP address and subnet mask connection parameters are configured.

Configuring an HMI connection via PROFINET with a SIMATIC PC

Introduction

You configure an HMI connection between an HMI device and a SIMATIC S7-1500 Software Controller via PROFINET or Ethernet in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

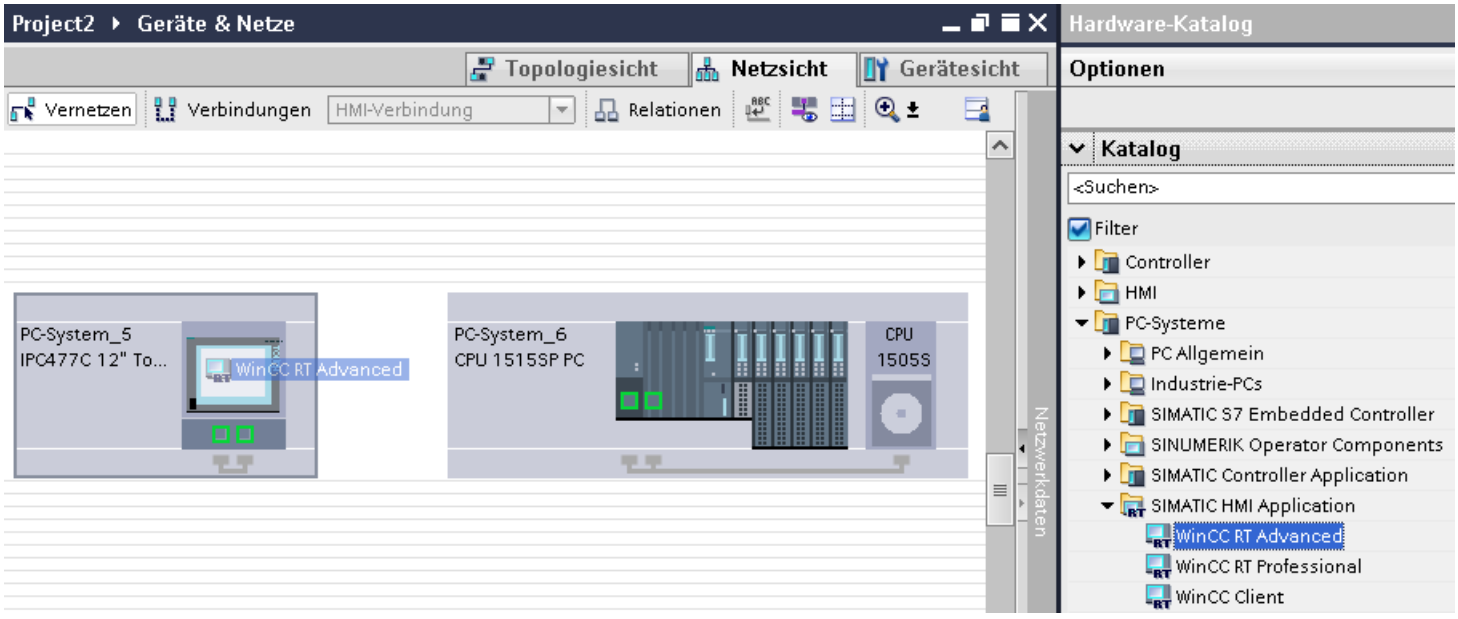
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- PC station with an SIMATIC S7-1500 Software Controller with PROFINET interface
- SIMATIC PC with PROFINET interface

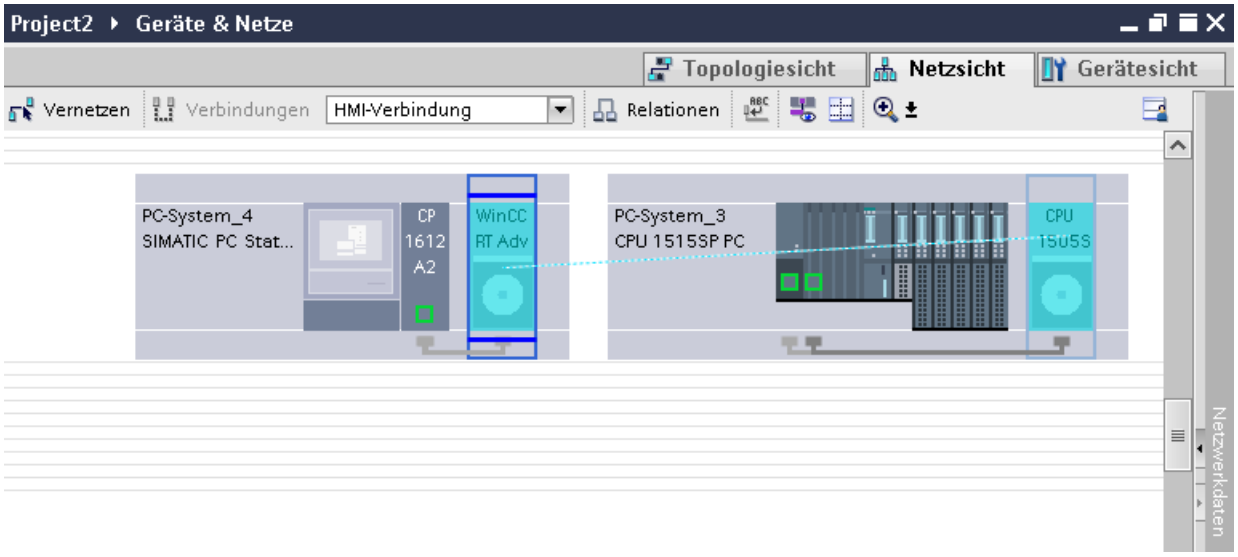
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a WinCC Runtime from the hardware catalog to the PC.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click on WinCC Runtime in the PC station and create a connection to the SIMATIC S7-1500 Software Controller with drag-and-drop.



5. Click on the connected interface in the PC station and select the "PC station" entry under "Interface assignment" in the Inspector window.
6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the section "PROFINET parameters (Page 6350)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7-1500 Software Controller. The IP address and subnet mask connection parameters are configured.

12.11.9.3 PROFINET parameters

PROFINET parameters for the HMI connection

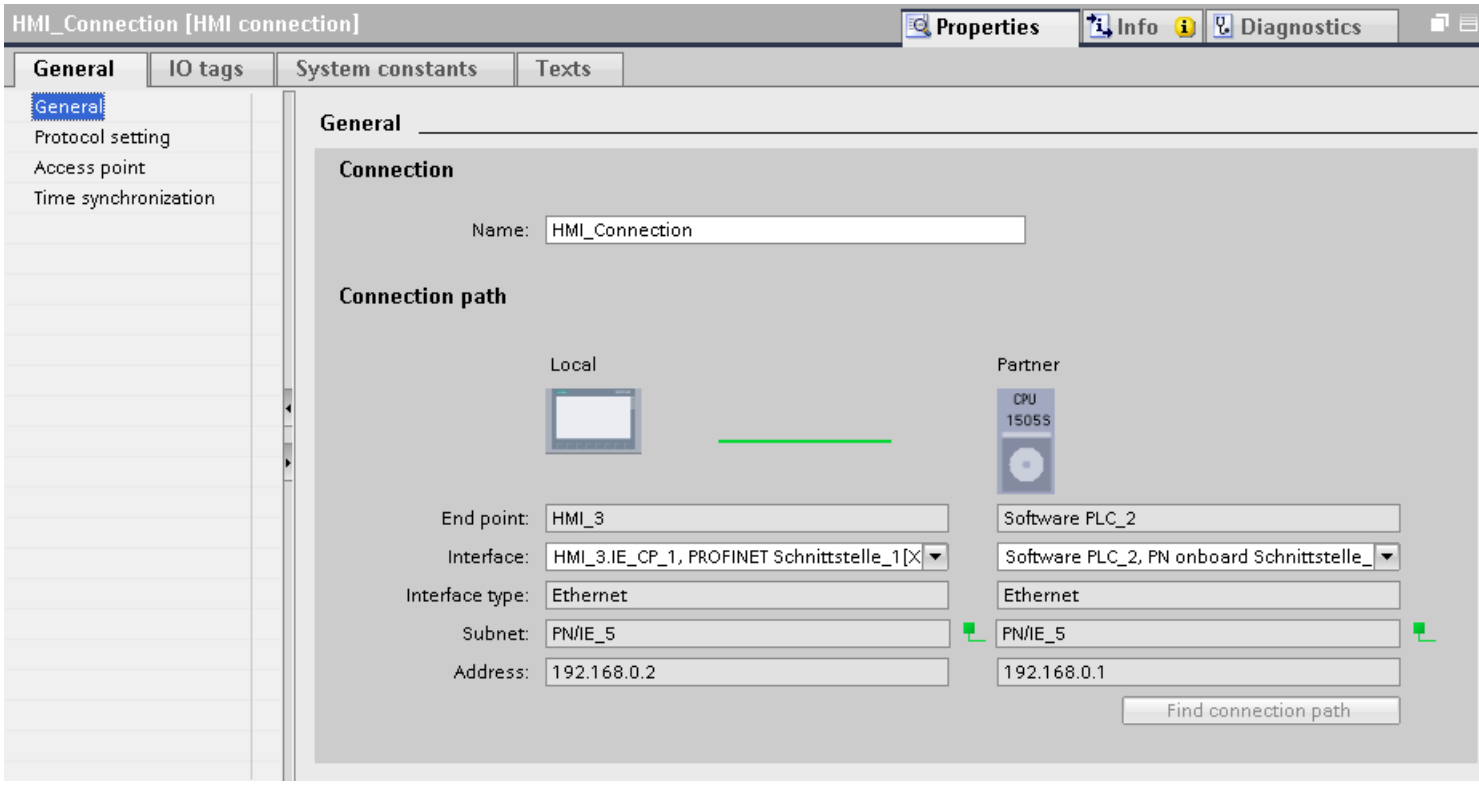
PROFINET parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Shows the name of the HMI connection.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFINET parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area cannot be edited.
- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the selected IP address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

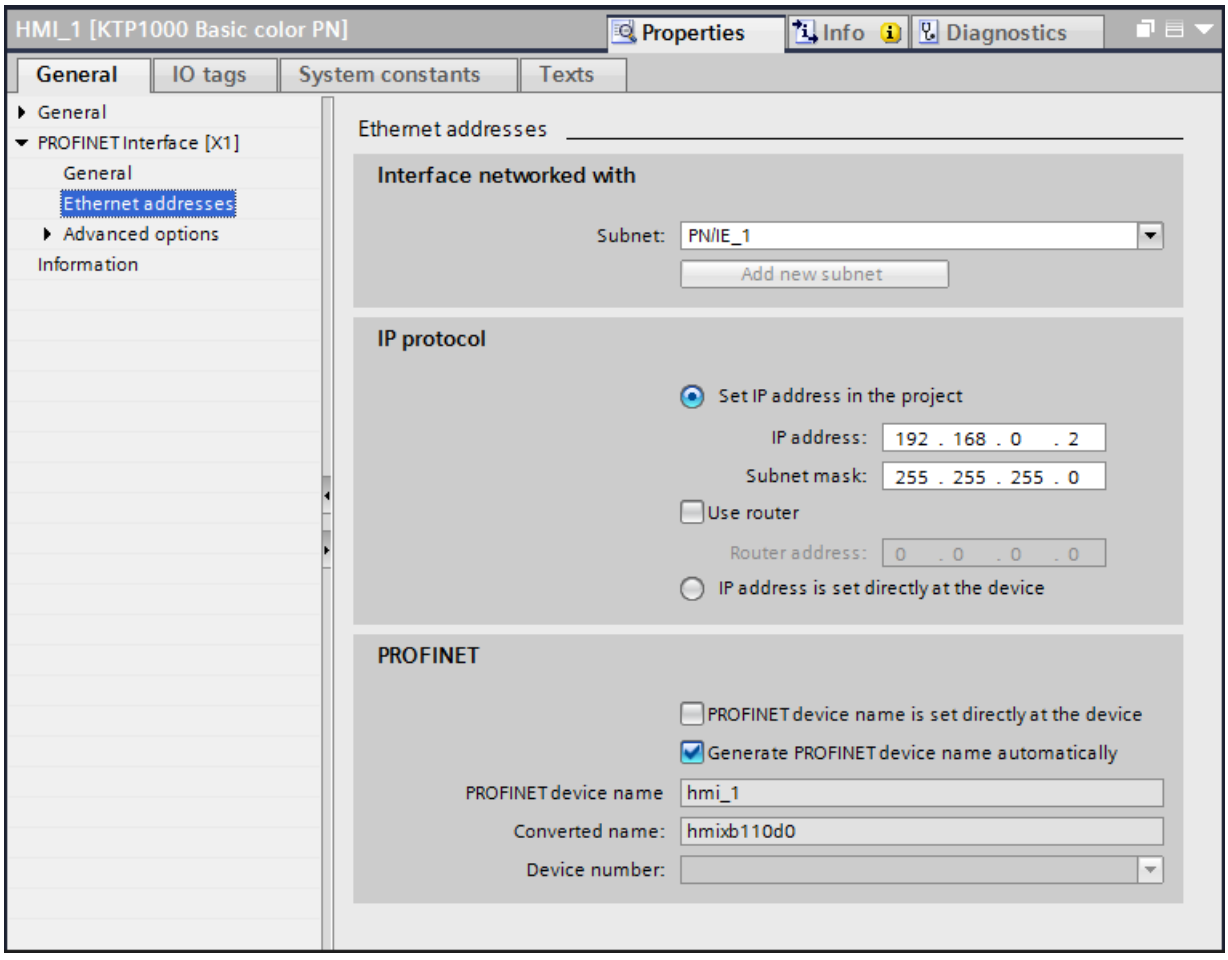
PROFINET parameters for the HMI device

PROFINET parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Set IP address in the project"
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

  Note

  The device is automatically restarted in the case of HMI devices with the Windows CE 3.0 operating system.

  HMI devices with Windows CE 3.0:
  - OP 77B
  - TP 177B color PN/DP
  - TP 177B mono DP
  - OP 177B color PN/DP
  - OP 177B mono DP
  - Mobile Panel 177 PN
  - Mobile Panel 177 DP
  - TP 277 6"
  - OP 277 6"

- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
- "Use IP router"
  If you are using an IP router, select "Use IP router" and enter the router address in the "Router address" field.
- "Set IP address using a different method"
  If the function "Set IP address using a different method" is activated, the IP address is not taken from the project. You have to enter the IP address directly in the Control Panel of the HMI device.

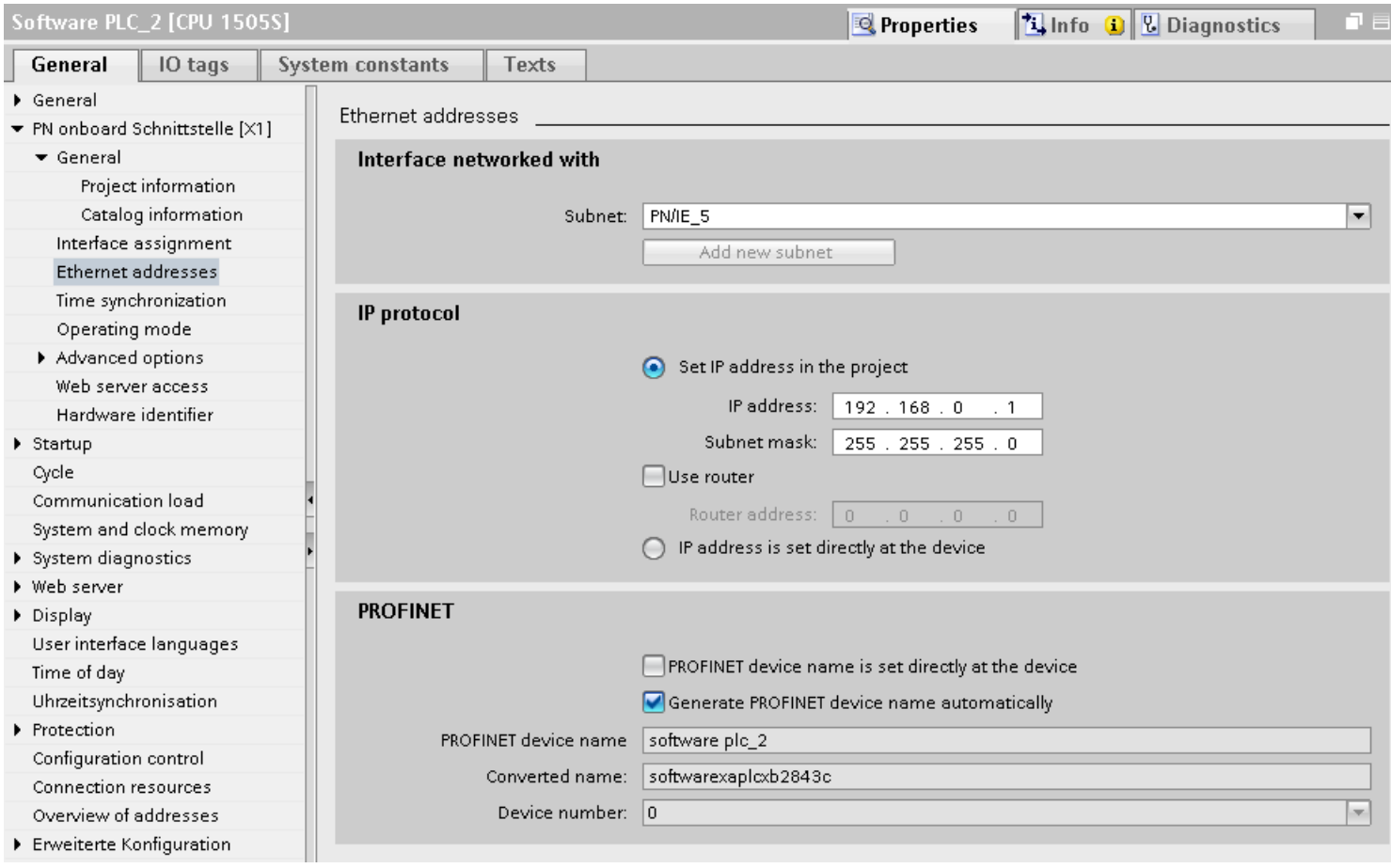
PROFINET parameters for the PLC

PROFINET parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFINET parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "IP address"
  You assign the IP address of the HMI device in the "IP address" area.
- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
  If you are using an IP router, select "Use IP router" and enter the router address in the field.

Protection of communication

Security levels

If you want to protect the PLC and HMI device communication, you can assign protection levels for the communication.

For an SIMATIC S7-1500 Software Controller, you can enter multiple passwords to set up different access rights for different user groups.

The passwords are entered in a table, so that exactly one protection level is assigned to each password.

The effect of the password is given in the "Protection" column.

Example

You select the "Complete protection" protection level for a standard CPU (i.e., not an F-CPU) when configuring it.

Afterwards, you enter a separate password for every protection level above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the assigned passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (no protection) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (write protection) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (read and write protection) allows access as if the CPU were read-protected and write-protected, so that only HMI access is possible for users who know this password.

Access password for the HMI connection

Introduction

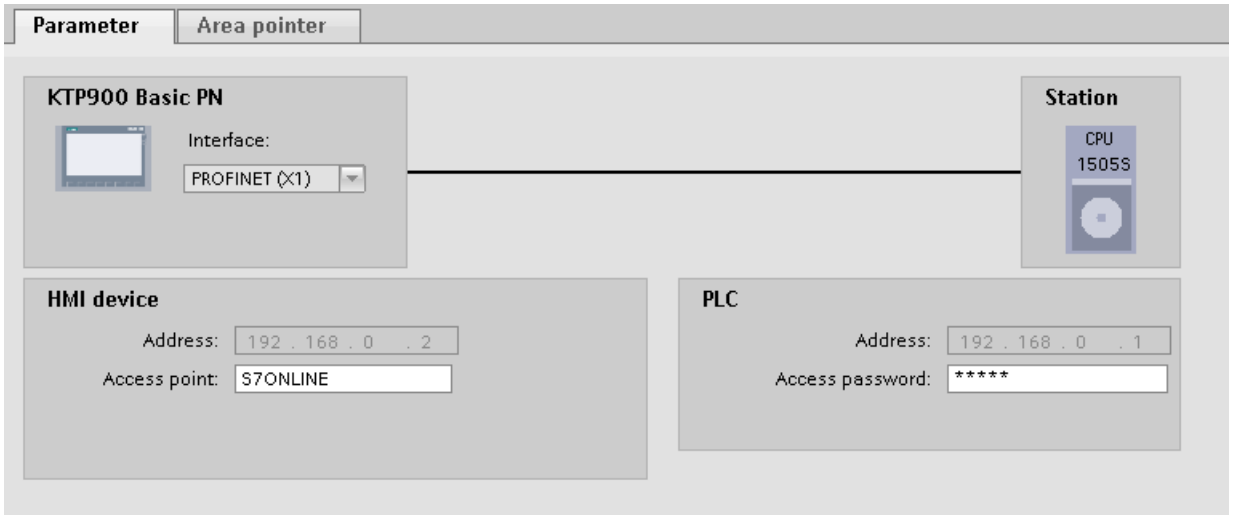
Communication with a PLC with the"Complete protection" protection level is protected by a password. The password is stored in the properties of the PLC.

Enter the password from the PLC in the "Access password" area.

Communication to the PLC is denied if an incorrect password or no password is entered.

Entering the access password

Enter the access password for the HMI connection in the "Connections" editor.



12.11.9.4 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Auto-Hotspot

Restrictions

You can only configure the following data types for communication with SIMATIC S7 1500 for data exchange using area pointers:

- UInt and array of UInt
- Word and array of Word
- Int and array of Int
- "Array[0..15] of Bool" for area pointer "Coordination"
- Date_And_Time
- DTL and LDT

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Use

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1st word	Current screen type															
2nd word	Current screen number															
3rd word	Reserved															
4th word	Current field number															
5th word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" or "40" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Using data types

The data types "Date_And_Time, DTL" and "LDT" can only be used with the "Date/time" and "Date/time PLC" area pointers.

The data format of the "Date/time" area pointer depends on job mailbox 40/41.

If there are no control tags linked to the area pointer, or a control tag is linked with the data type "Array[0..5] of UInt/Word/Int", the following applies:

The configuration of the "Date/time" area pointer is only used for job mailbox 41.

If job mailbox 40 is used, the data format "DATE_AND_TIME (BCD-encoded)" is used (shown in the next section).

If the "Date/time" and "Date/time PLC" area pointers are linked to a control tag with the data type "DATE_AND_TIME", "DTL" or "LDT", the associated data format is used in the corresponding area pointer.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the Date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 1)	Reserved			Reserved		
n+5 1)	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Using data types

The data types "Date_And_Time, DTL" and "LDT" can only be used with the "Date/time" and "Date/time PLC" area pointers.

The data format of the "Date/time" area pointer depends on job mailbox 40/41.

If there are no control tags linked to the area pointer, or a control tag is linked with the data type "Array[0..5] of UInt/Word/Int", the following applies:

The configuration of the "Date/time" area pointer is only used for job mailbox 41.

If job mailbox 40 is used, the data format "DATE_AND_TIME (BCD-encoded)" is used (shown in the next section).

If the "Date/time" and "Date/time PLC" area pointers are linked to a control tag with the data type "DATE_AND_TIME", "DTL" or "LDT", the associated data format is used in the corresponding area pointer.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

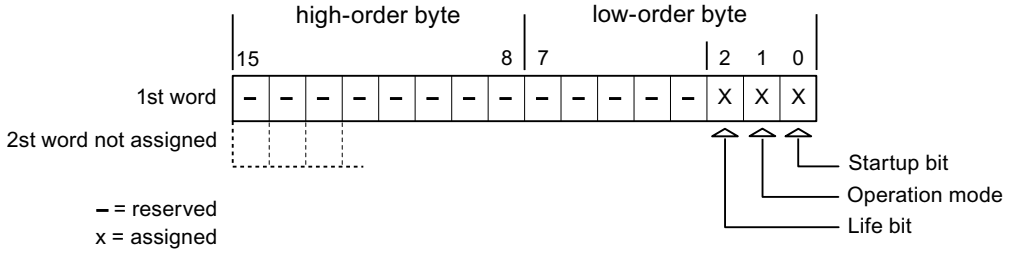
By default, the "Coordination" area pointer has the length of one word and cannot be changed.

Usage

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of the bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

"Project ID" area pointer

Function

When Runtime starts, a check can be carried out as to whether the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program. If there is no concordance, a system event is given on the HMI device and Runtime is stopped.

Use

Note

HMI connections cannot be switched "online".

The HMI connection in which the "Project ID" area pointer is used must be switched "online".

To use this area pointer, set up the following during the configuration:

- Define the version of the configuration. Values between 1 and 255 are possible.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:
  You enter the data address in the editor "Communication > Connections" under "Address".

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

"Job mailbox" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	

Word	Most significant byte	Least significant byte
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	**Setting the date (BCD coded)** [3) 4)]	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	**User logon**	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	**User logoff**	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	**Transfer date/time to PLC**	
	(in the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	**Transfer date/time to PLC**	

No.	Function	
14	**Set time (BCD-coded)**	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	**Update tag**	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in Parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	**Screen selection** [2]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	**Reading data record from PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	**Writing data record to PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1]	Only for devices supporting recipes.
[2]	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
[3]	The weekday is ignored on HMI device KTP 600 BASIC PN.
[4]	The weekday is ignored when you configure the "Date/Time PLC" area pointer.

"Data record" area pointer

"Data record" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization via the data mailbox

Data records are always transferred directly, which means that the tag values are read straight from an address or written straight to an address configured for this tag without being redirected via an interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system alarm.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then, for example, process, edit, or save these values in the HMI device.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data mailbox free
2	0000 0010	Transferring
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer started by the operator in the recipe view

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data record and sets the data record number to 0.	Abort with system event.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data record.	Abort with system event.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of a transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxs No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the function from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible

- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system alarms
- Triggered by function
  Output of system alarms
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data record.

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Auto-Hotspot

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

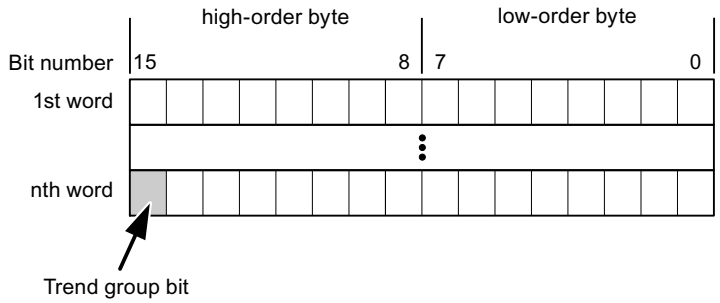
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Permitted data types for trends

For SIMATIC S7

You assign one bit to each trend during configuration. Tags and array tags of the "Word" or "Int" data type are permitted.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Auto-Hotspot

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

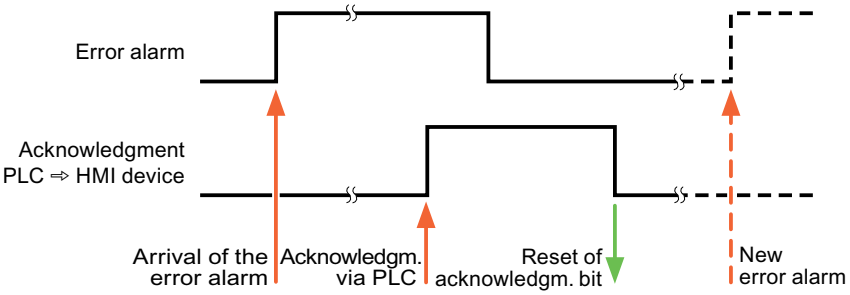
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

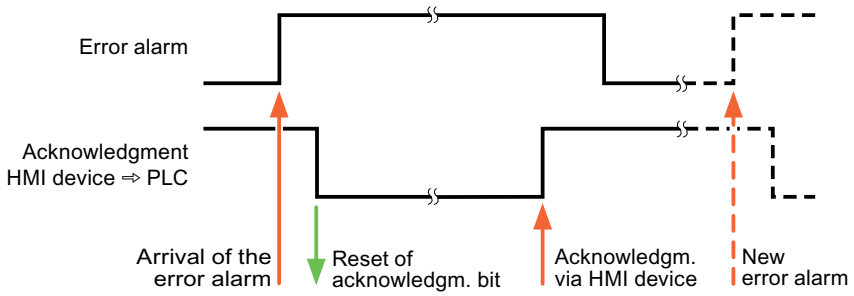
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



LED mapping

Function

Keyboard devices have LEDs in the function keys. These LEDs can be activated from the PLC. Thus, it is possible, for example, to signal to the user which key he should press in a given situation by lighting up an LED.

Note

The LED function cannot be configured for Basic Panels.

Requirements

In order to activate an LED, an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration.

LED assignment

The assignment of the individual LEDs to the bits in the LED tags is specified when the function keys are configured. In this process, the "LED tag" and the assigned "Bit" are specified for each function key in the "General" group of the properties window.

The "Bit" bit number designates the first of two consecutive bits that control the following LED statuses.

Bit n+ 1	Bit n	LED function	
		All Mobile Panels, all Operator Panels, all Multi Panels, all Comfort Panels	Panel PCs
0	0	Off	Off
0	1	Rapid flashing	Flashing
1	0	Slow flashing	Flashing
1	1	On permanently	On permanently

12.11.9.5 Performance features of communication

Device dependencies of the SIMATIC S7-1500 software controller

Device dependency

If you use devices from an earlier version of the TIA Portal with TIA Portal V13, it may not be possible to configure connections to certain HMI devices.

Basic Panels V11.0

HMI devices	SIMATIC S7-1500 Software Controller
KP300 Basic	No
KP400 Basic	No
KTP400 Basic PN	No
KTP600 Basic DP	No
KTP600 Basic PN	No
KTP1000 Basic DP	No
KTP1000 Basic PN	No
TP1500 Basic PN	No

Basic Panels V12.0

HMI devices	SIMATIC S7-1500 Software Controller
KP300 Basic	Yes
KP400 Basic	Yes
KTP400 Basic PN	Yes
KTP600 Basic DP	Yes
KTP600 Basic PN	Yes
KTP1000 Basic DP	Yes
KTP1000 Basic PN	Yes
TP1500 Basic PN	Yes

Basic Panels V13.0

HMI devices	SIMATIC S7-1500 Software Controller
KTP400 Basic	Yes
KTP700 Basic	Yes
KTP900 Basic	Yes
KTP1200 Basic	Yes

Panels V11.0

HMI devices	SIMATIC S7-1500 Software Controller
OP 73	No
OP 77A	No
OP 77B	No
TP 177A	No
TP 177A Portrait	No
TP 177B 4"	No
TP 177B 6" mono	No
TP 177B 6"	No
OP 177B 6" mono	No
OP 177B 6"	No
TP 277 6"	No
OP 277 6"	No

Panels V12.0

HMI devices	SIMATIC S7-1500 Software Controller
OP 73	No
OP 77A	No
OP 77B	No
TP 177A	No
TP 177A Portrait	No
TP 177B 4"	No
TP 177B 6" mono	No
TP 177B 6"	No
OP 177B 6" mono	No
OP 177B 6"	No
TP 277 6"	No
OP 277 6"	No

Multi Panels V11.0

HMI devices	SIMATIC S7-1500 Software Controller
MP 177 6" Touch	No
MP 277 8" Key	No
MP 277 10" Key	No
MP 277 10" Touch	No
MP 377 12" Key	No
MP 377 12" Touch	No

HMI devices	SIMATIC S7-1500 Software Controller
MP 377 15" Touch	No
MP 377 19" Touch	No

Multi Panels V12.0

HMI devices	SIMATIC S7-1500 Software Controller
MP 177 6" Touch	Yes
MP 277 8" Key	Yes
MP 277 10" Key	Yes
MP 277 10" Touch	Yes
MP 377 12" Key	Yes
MP 377 12" Touch	Yes
MP 377 15" Touch	Yes
MP 377 19" Touch	Yes

Mobile Panels V11.0

HMI devices	SIMATIC S7-1500 Software Controller
Mobile Panel 177 6" DP	No
Mobile Panel 177 6" PN	No
Mobile Panel 277 8"	No
Mobile Panel 277 8" IWLAN V2	No
Mobile Panel 277F 8" IWLAN V2	No
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	No
Mobile Panel 277 10"	No

Mobile Panels V12.0

HMI devices	SIMATIC S7-1500 Software Controller
Mobile Panel 177 6" DP	Yes
Mobile Panel 177 6" PN	Yes
Mobile Panel 277 8"	Yes
Mobile Panel 277 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	Yes
Mobile Panel 277 10"	Yes

Comfort Panels V11.0

HMI devices	SIMATIC S7-1500 Software Controller
KP400 Comfort	No
KTP400 Comfort	No
KTP400 Comfort Portrait	No
KP700 Comfort	No
TP700 Comfort	No
TP700 Comfort Portrait	No
KP900 Comfort	No
TP900 Comfort	No
TP900 Comfort Portrait	No
KP1200 Comfort	No
TP1200 Comfort	No
TP1200 Comfort Portrait	No
KP1500 Comfort	No
TP1500 Comfort	No
TP1500 Comfort Portrait	No
TP1900 Comfort	No
TP1900 Comfort Portrait	No
TP2200 Comfort	No
TP2200 Comfort Portrait	No

Comfort Panels V12.0

HMI devices	SIMATIC S7-1500 Software Controller
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes

HMI devices	SIMATIC S7-1500 Software Controller
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V13.0

HMI devices	SIMATIC S7-1500 Software Controller
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Runtime V11.0

HMI devices	SIMATIC S7-1500 Software Controller
WinCC RT Advanced	No

Runtime V12.0

HMI devices	SIMATIC S7-1500 Software Controller
WinCC RT Advanced	Yes

Runtime V13.0

HMI devices	SIMATIC S7-1500 Software Controller
WinCC RT Advanced	Yes

Permitted data types

Valid data types for connections with SIMATIC S7 1500

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length	
BOOL	1 bit	
BYTE	1 byte	
WORD	2 bytes	
DWORD	4 bytes	
CHAR	1 byte	
WCHAR	2 bytes	RT Professional
ARRAY of WCHAR	--	
INT	2 bytes	
DINT	4 bytes	
REAL	4 bytes	
TIME	4 bytes	
DATE	2 bytes	
TIME_OF_DAY	4 bytes	
S5TIME	2 bytes	
COUNTER	2 bytes	
TIMER	2 bytes	
DATE_AND_TIME	8 bytes	
STRING	(2+n) bytes, n = 0 to 254	
WSTRING	(4+2*n) bytes, n = 0 to 254	Basic Panels
	(4+2*n) bytes, n = 0 to 4094	Panels, RT Advanced
	(4+2*n) bytes, n = 0 to 65534	RT Professional
DTL	12 bytes	
LDT	8 bytes	
LINT	8 bytes	
LREAL	8 bytes	
LTIME	8 bytes	
LTIME_OF_DAY	8 bytes	
SINT	1 byte	
UDINT	4 bytes	
UINT	2 bytes	

Data type	Length
ULINT	8 bytes
USINT	1 byte

12.11.9.6 Configuring connections in the "Connections" editor

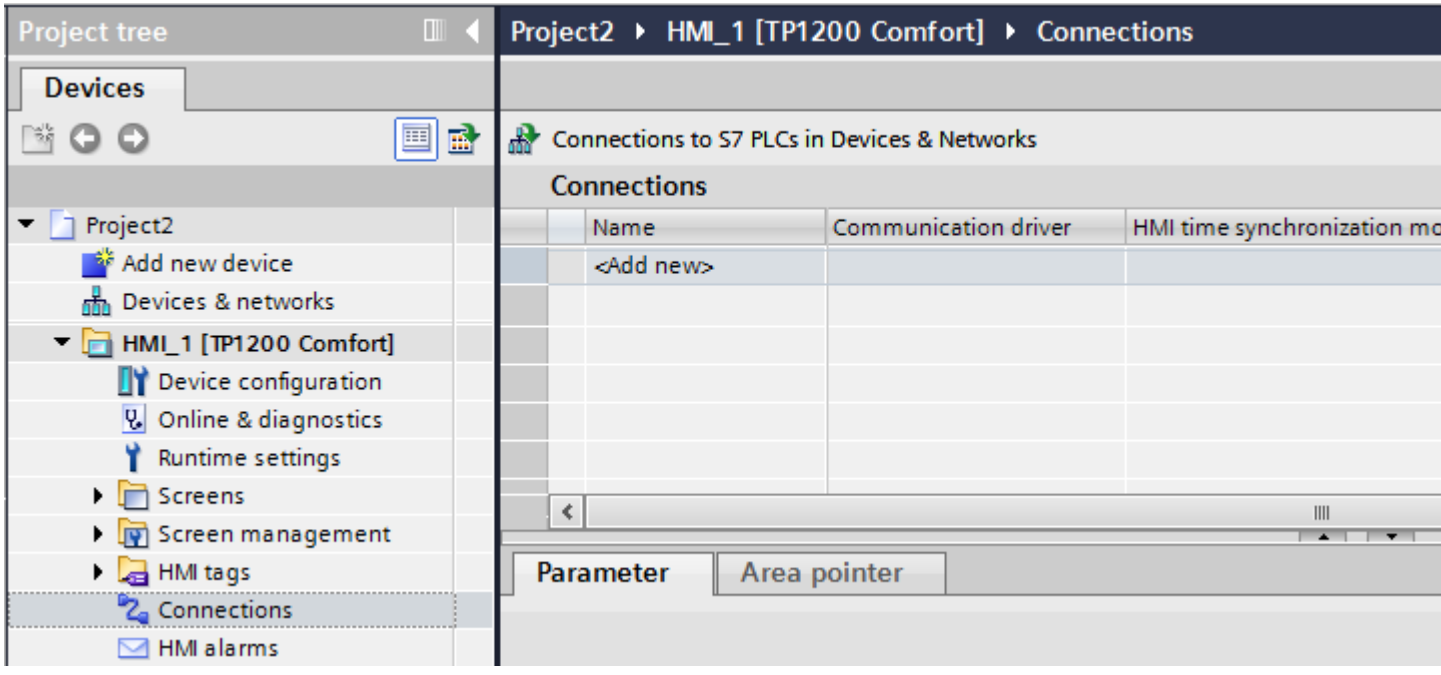
Creating a PROFINET connection

Requirements

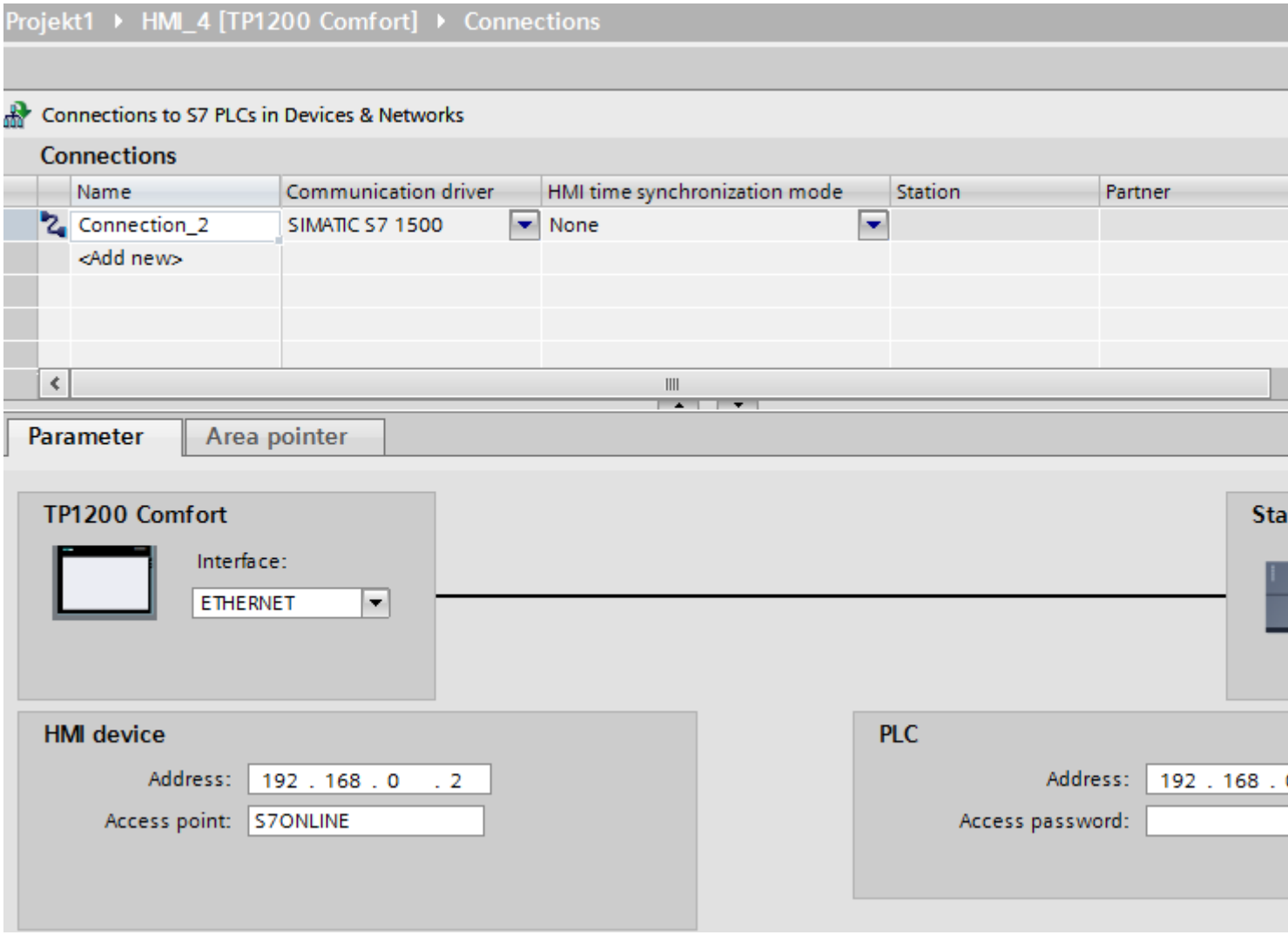
- A project is open.
- An HMI device with a PROFINET interface has been created.

Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.



4. Click the name of the connection.
5. Select a PROFINET interface of the HMI device in the Inspector window under "Parameters > Interface".
6. Set the IP addresses of the communication partners in the Inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

12.11.9.7 Configuring connections in the "Connections" editor

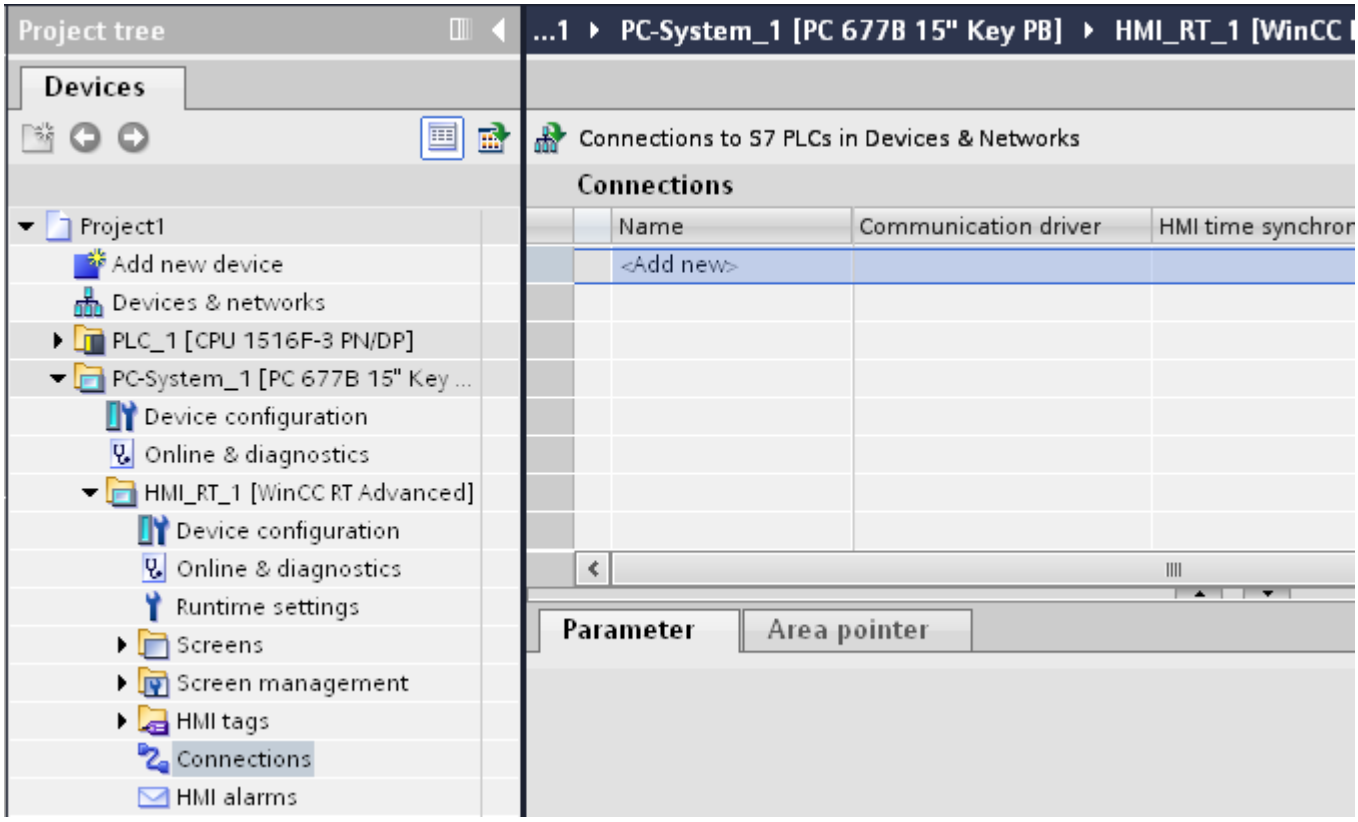
Creating a connection

Requirements

- A project is open.
- WinCC RT Advanced is created.

Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select an interface of the HMI device in the Inspector window under "Parameters > Interface".
6. Set the parameters for the connection in the Inspector window.

Interfaces

Select one of the following interfaces in the Inspector window under "Parameters > WinCC RT Advanced > Interfaces".:

- Industrial Ethernet
- MPI
- PROFIBUS

For additional information on the parameters of the interfaces see:

Auto-Hotspot

Connection parameters

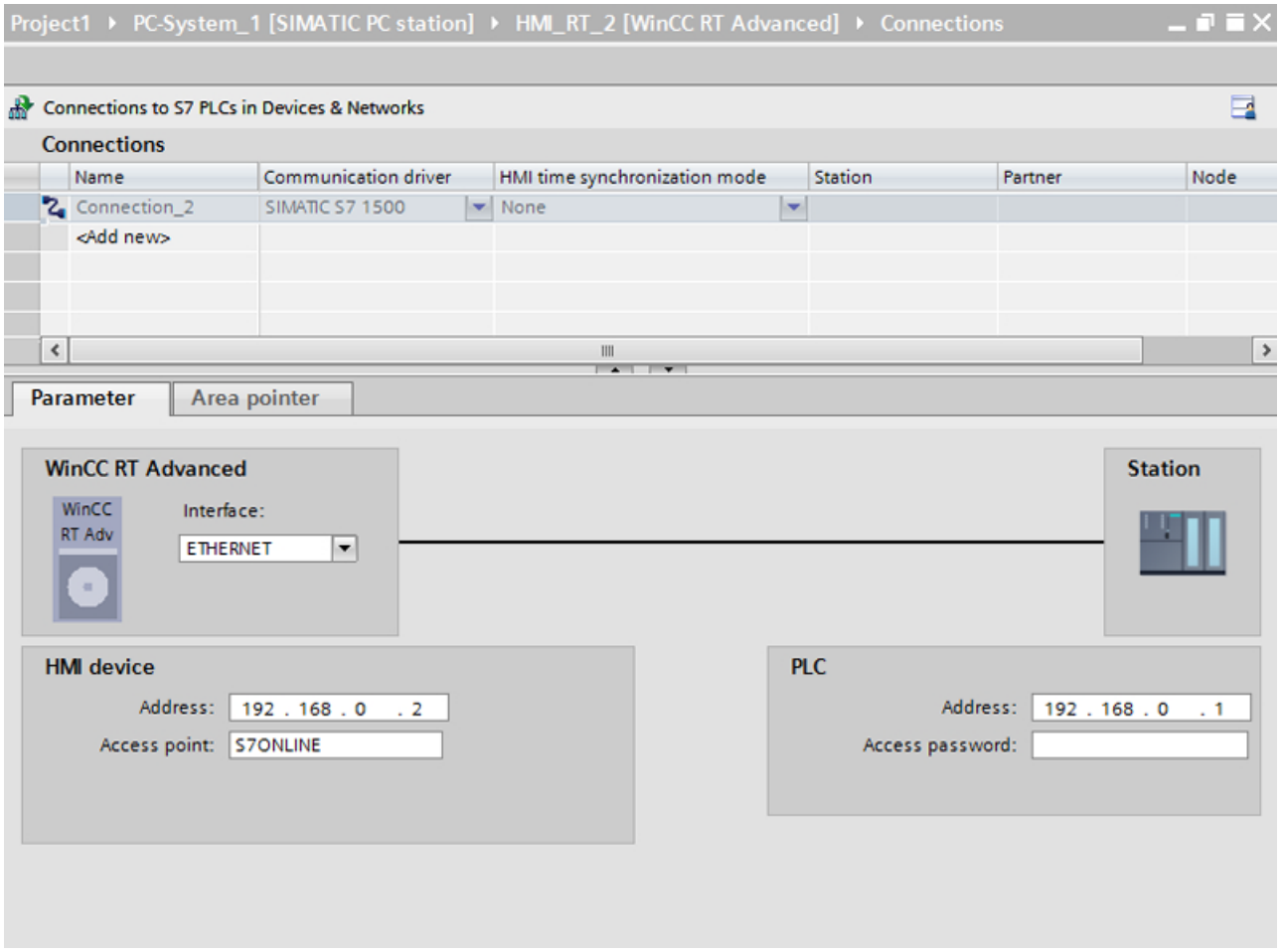
ETHERNET

Introduction

You assign the parameters for the HMI device and the controller in the network under "Parameters".

The parameters described in the following apply to the following interfaces:

- TCP/IP



HMI device

- "Address"
  Enter the IP address of the HMI device under "Address".
- "Access point"
  Specifies the access point for the PG/PC interface that can be used to reach the communication partner.

Controller

- "Address"
  Enter the IP address of the controller under "Address".
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the controller.

Note

You only need a password if you have set the "Complete protection" protection level at the controller.

No connection to the controller is established if the "Complete protection" protection level is stored on the controller and you do not enter a password.

12.11.9.8 Configuring time synchronization

Time synchronization

Introduction

To have the same time of day throughout the plant, you can synchronize the time on various plant components using time synchronization. WinCC time synchronization is operated as a master-slave system.

One system component must be a clock for all components of a plant to work with identical time. The component functioning as the clock is referred to as the time master. The components that receive the time are time slaves.

Properties of time synchronization

- The HMI device can define the time as master or can accept the time of the PLC as slave.
- In "master mode", the time is synchronized at each connection setup.
- In "slave mode", the time is synchronized at each connection setup and than at cyclic intervals of 10 minutes.
- The first time synchronization is performed on the HMI device immediately after the start of runtime.
- Time synchronization is only performed on the HMI device during operation of runtime.

Time synchronization restrictions

Approved HMI devices

You can configure the time synchronization between a SIMATIC S7-1200 or SIMATIC S7-1500 and an HMI device with the following HMI devices:

Device	Operating system
Basic Panels	-
TP177 4"	Windows CE 5.0

Device	Operating system
Multi Panel 177	Windows CE 5.0
Multi Panel 277	Windows CE 5.0
Multi Panel 377	Windows CE 5.0
Mobile Panel 277	Windows CE 5.0
Mobile 277 IWLAN V2	Windows CE 5.0
Comfort Panels	Windows CE 6.0
PC systems with WinCC RT Advanced	Microsoft Windows XP Microsoft Windows 7

Configuration restrictions

- If an HMI device has several connections to SIMATIC S7-1200 or SIMATIC S7-1500, you can only configure one connection as "slave".
- If you have enabled time synchronization for the HMI device as "slave", you can no longer use the global area pointer "Date/time PLC".
- An HMI device can only request the time from a PLC with "Complete protection" security type configuration if the correct "Access password" is configured.
  Configure the "Access password" for communication with a PLC with "Complete protection" security type in the "Connections" editor of the HMI device.
  This "Access password" must match the password configured on the PLC. The PLC password is assigned in the PLC properties at: "General > Security"
- Basic Panels can only be configured as "Slave".
- If you use Basic Panels for the configuration, it is not possible to use time synchronization via NTP and the "Date/time PLC" area pointer simultaneously.
- Time synchronization with SIMATIC S7-1200 (V1.0) controllers is not possible.
- Time synchronization between the HMI device TP177 4" and SIMATIC S7-1200 (V4.0) controllers is not possible.
- Time synchronization between the HMI device TP177 4" and SIMATIC S7-1500 is not possible.

Configuring time synchronization for integrated connections

Introduction

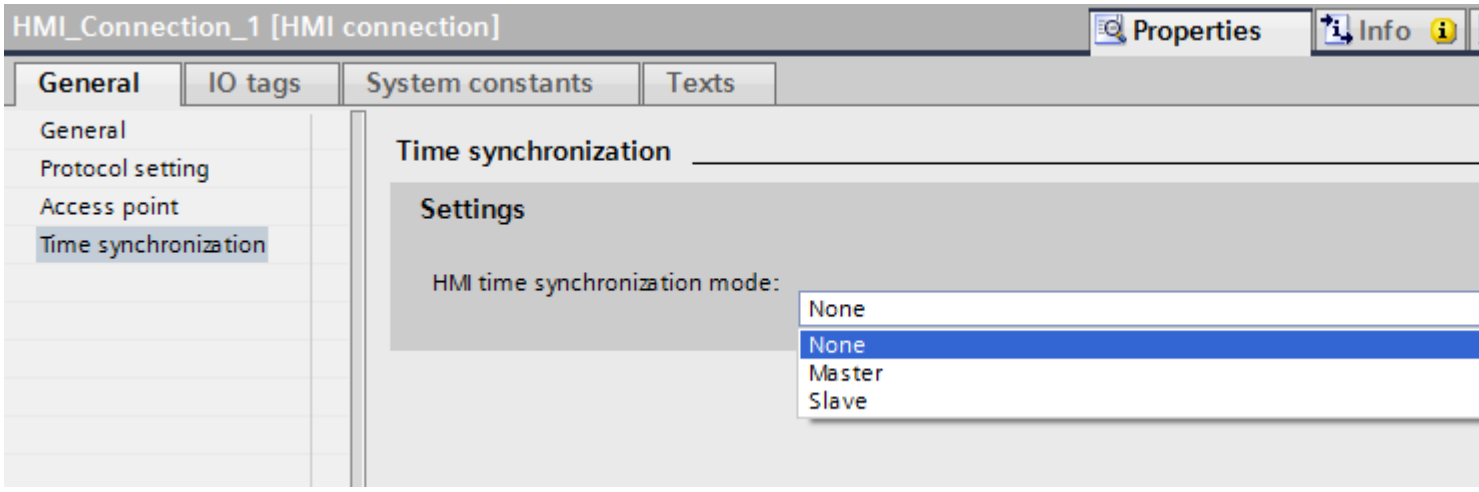
You configure time synchronization for an integrated connection in the "Devices & Networks" editor.

Requirements

- An HMI connection between an HMI device and a SIMATIC S7 1200 or SIMATIC S7 1500 has been configured.
- The HMI device must support the "time synchronization" function.
- The "Devices & Networks" editor is open.

Procedure

1. Click the line of the HMI connection in the "Devices & networks" editor.
2. Select the following in the inspector window under "General > Time synchronization > Settings":
   - None: No time synchronization is used.
   - Master: The HMI device sets the time.
   - Slave: The PLC sets the time.



Configuring time synchronization for non-integrated connections

Introduction

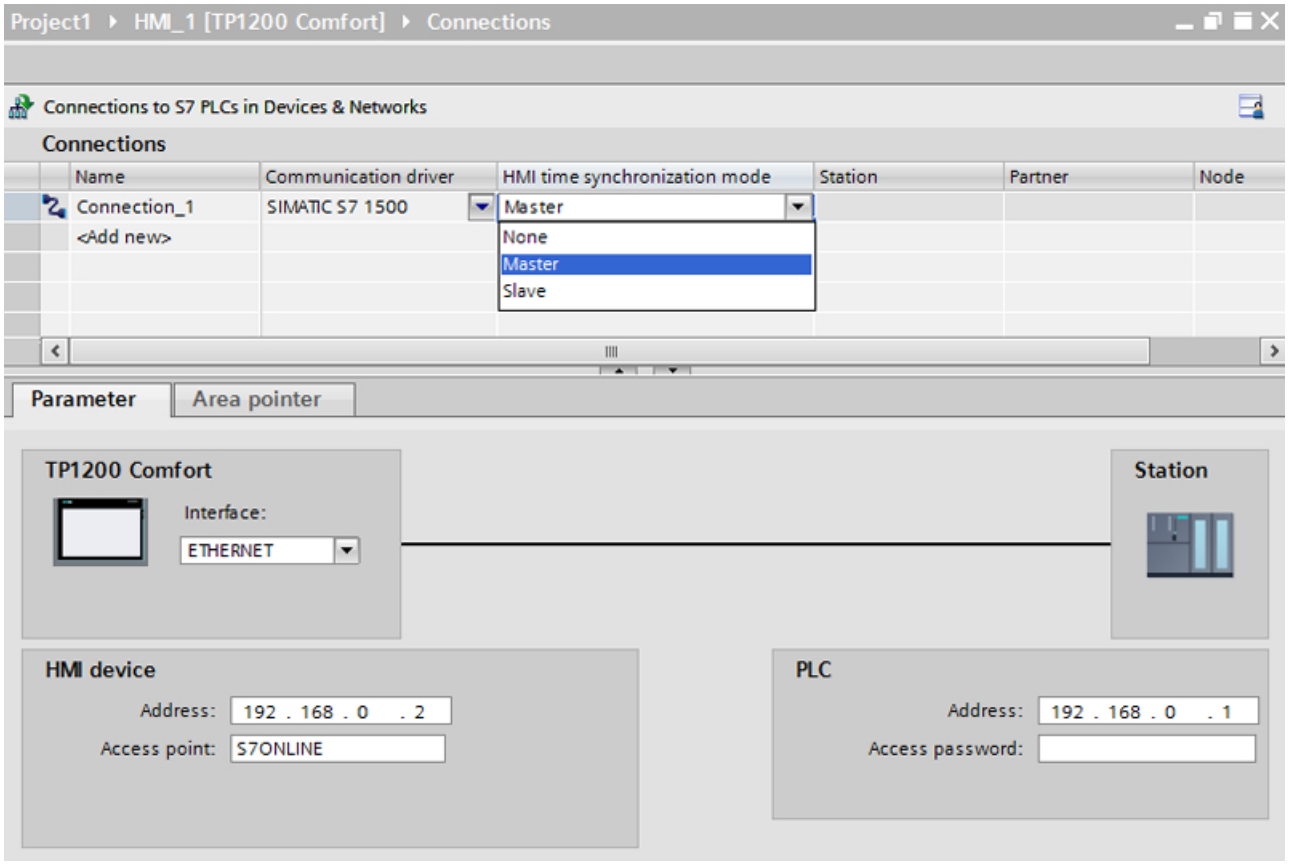
You configure time synchronization for a non-integrated connection in the "Connections" editor.

Requirements

- An HMI device which supports the "time synchronization" function has been created.
- "Connections" editor is open.

Procedure

1. Double-click "<Add>".
2. In the "Communication drivers" column, select the "SIMATIC S7 1200" PLC.
3. Select the following in the "HMI time synchronization mode" column:
   - None: No time synchronization is used.
   - Master: The HMI device sets the time.
   - Slave: The PLC sets the time.



12.11.10 Communicating with SIMATIC ET 200 CPU

12.11.10.1 Communication with SIMATIC ET 200 CPU

Introduction

This section describes the communication between an HMI device and the SIMATIC ET 200 CPU controller.

You can configure the following communication channels for the SIMATIC ET 200 CPU:

- PROFINET
- PROFIBUS

HMI connection for communication

Configure connections between the HMI device and a SIMATIC ET 200 CPU in the "Devices & Networks" editor. If you have configured a HMI device with a serial port, you must configure a PROFIBUS-capable communication module to the ET 200 CPU.

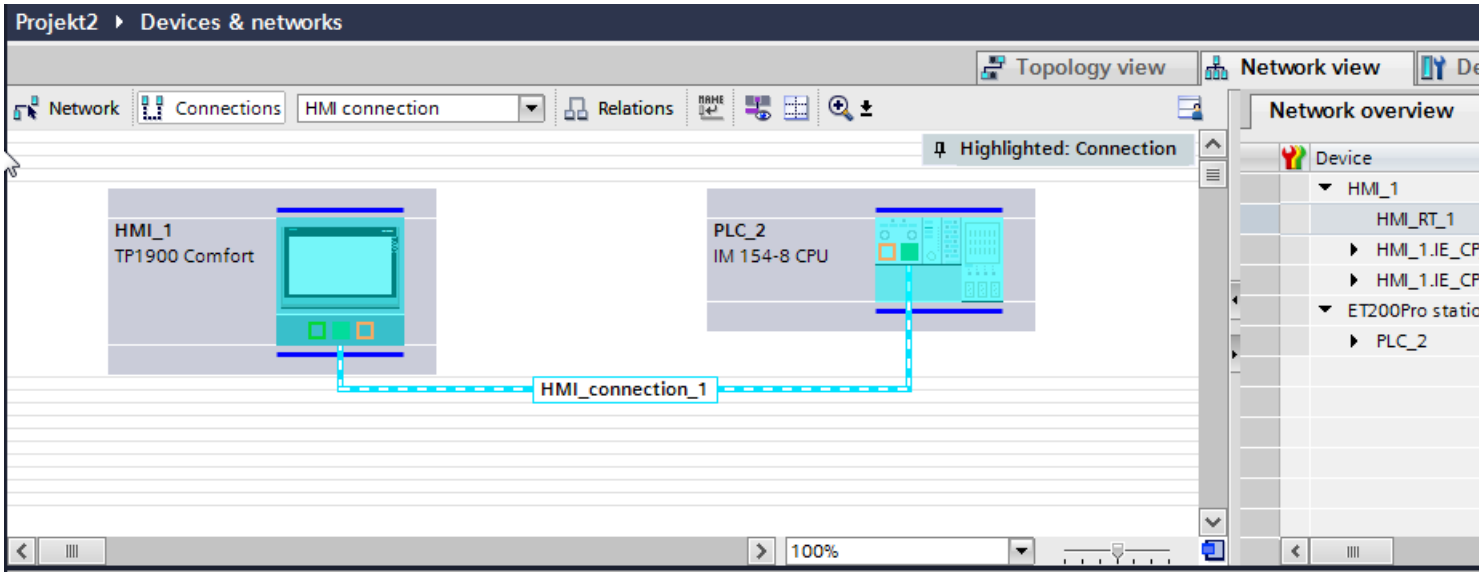
12.11.10.2 Communication via PROFINET

Configuring an HMI connection

Communication via PROFINET

HMI connections via PROFINET

If you have inserted an HMI device and a SIMATIC ET 200 CPU into the project, you interconnect the two PROFINET interfaces in the "Devices & Networks" editor.



You can also connect multiple HMI devices to a single SIMATIC ET 200 CPU and multiple SIMATIC ET 200 CPU to a single HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET

Introduction

You configure an HMI connection over PROFINET or Ethernet between HMI devices and a SIMATIC ET 200 CPU in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

Requirements

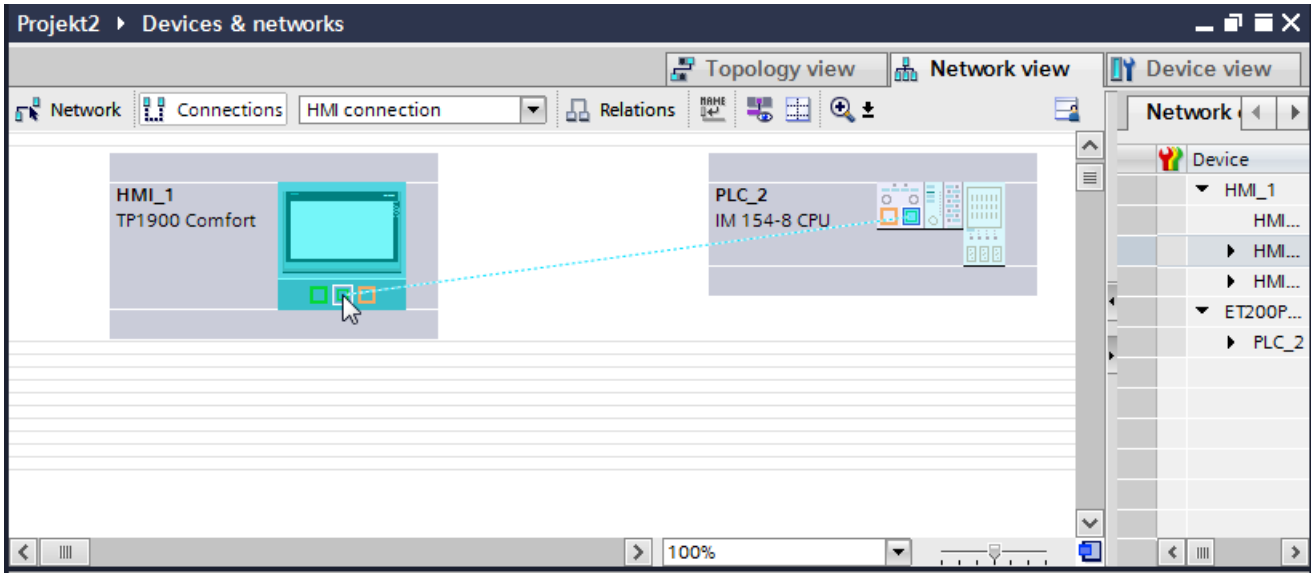
The following communication partners are created in the "Devices & Networks" editor:

- HMI device with PROFINET or Ethernet interface
- SIMATIC ET 200 CPU with PROFINET interface.

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

3. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the HMI device.



4. Click the connecting line.
5. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
6. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the section "PROFINET parameters (Page 6401)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC ET 200 CPU. The IP address and subnet mask connection parameters are configured.

Configuring an HMI connection

Communication via PROFINET

Communication via PROFINET

This section describes communication over PROFINET between a WinCC Runtime and the SIMATIC ET 200 CPU.

You can use the following WinCC Runtimes as an HMI device:

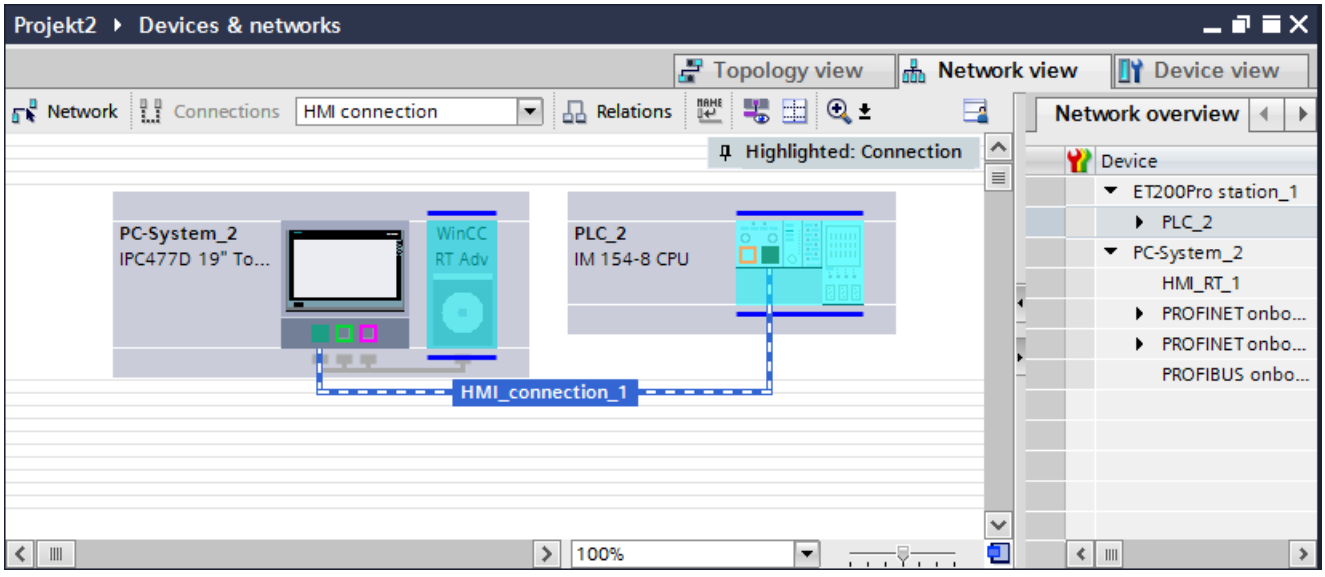
- WinCC RT Advanced
- WinCC RT Professional

WinCC Runtime as HMI device

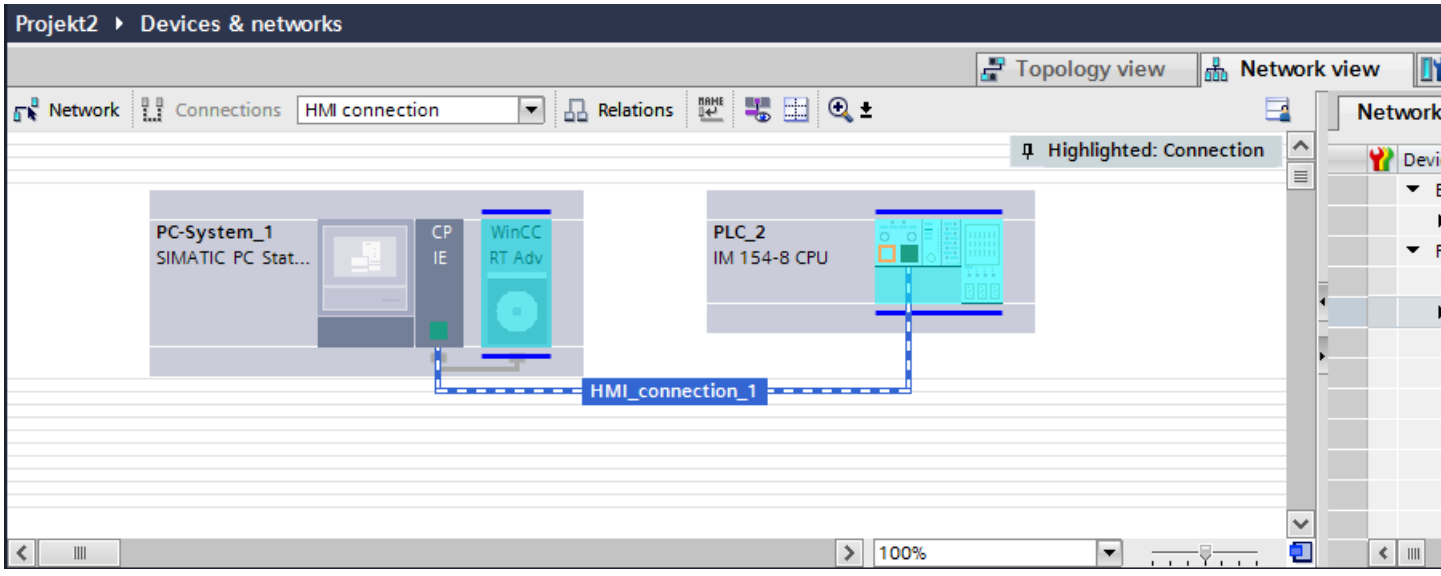
You configure the HMI connections between a WinCC Runtime and SIMATIC ET 200 CPU in the "Devices & Networks" editor.

If you want to use a WinCC Runtime as an HMI device, you have the following options:

1. You create a SIMATIC PC and configure a WinCC Runtime on this SIMATIC PC.
   In this way you use the SIMATIC PC with a WinCC Runtime as an HMI device.



2. You create a WinCC Runtime and configure a communication processor to the Runtime.
   In this way you use your configuration PC with a WinCC Runtime as an HMI device.



You can also connect multiple HMI devices to a single SIMATIC ET 200 CPU and multiple SIMATIC ET 200 CPU to a single HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET with a PC

Introduction

You configure an HMI connection over PROFINET or Ethernet between a HMI device and a SIMATIC ET 200 CPU in the "Devices & Networks" editor.

Caution
Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

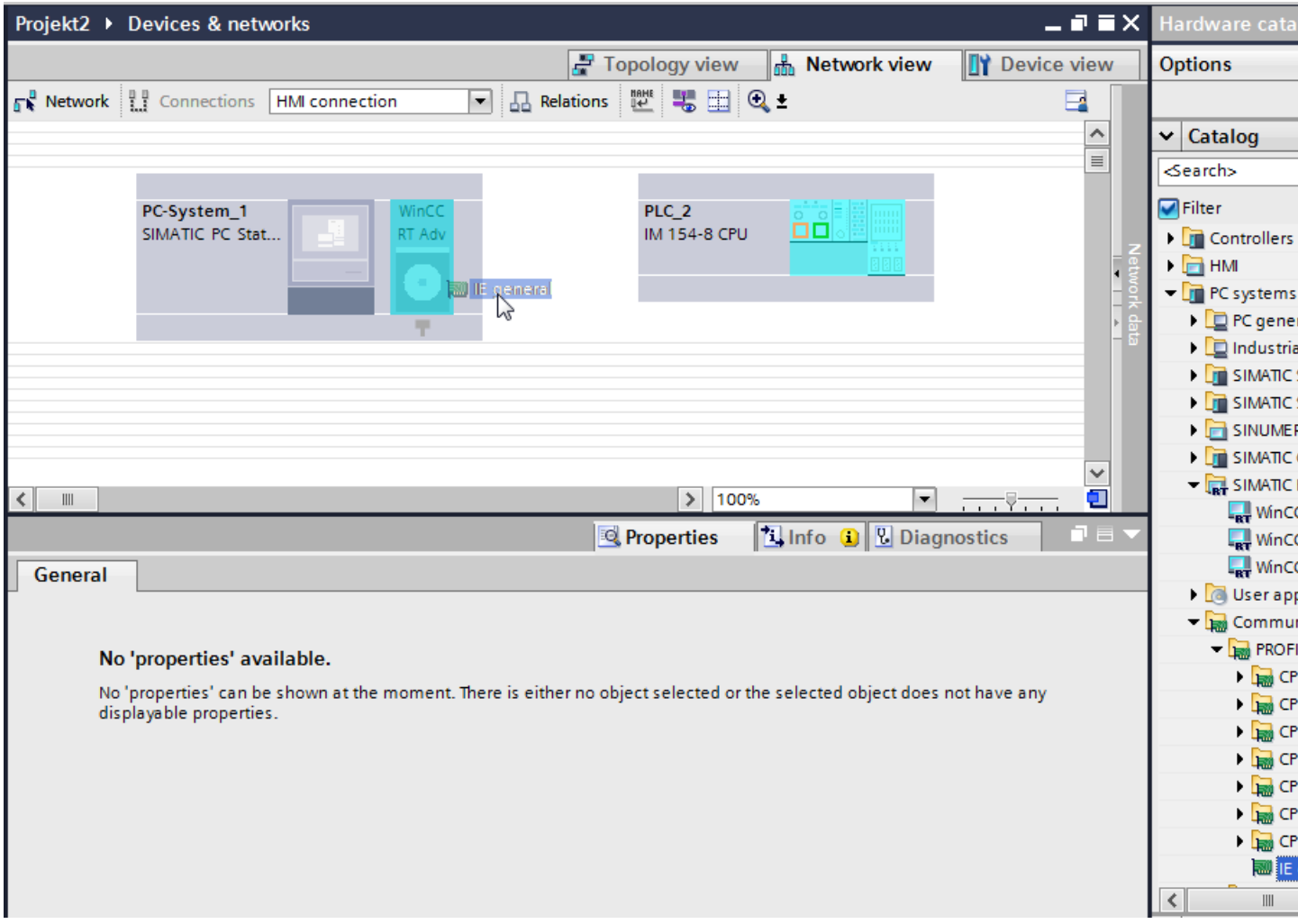
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC ET 200 CPU with PROFINET interface
- PC station with a "WinCC Runtime" application

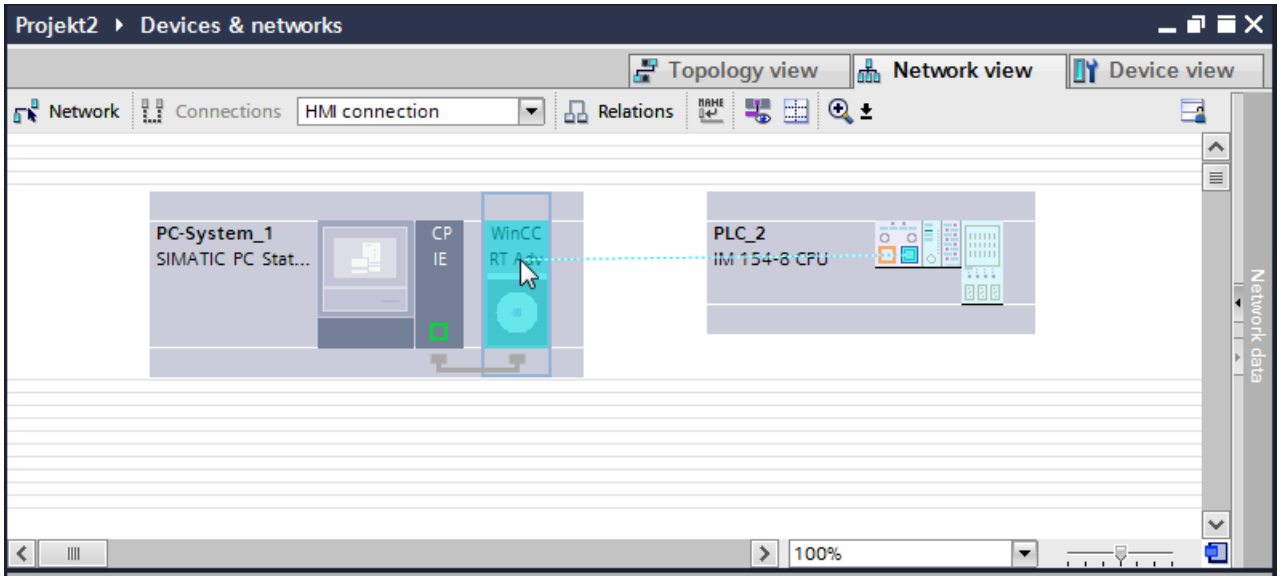
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a PROFINET-capable communication processor from the hardware catalog to the WinCC Runtime.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

4. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET interface of the communication processor.



5. Click the connected interface on the PC station and select the entry "PC station" under "Interface assignment" in the Inspector window.
6. Click the connecting line.
   The connection is displayed graphically in the Inspector window.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the section "PROFINET parameters (Page 6401)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC ET 200 CPU. The IP address and subnet mask connection parameters are configured.

Configuring an HMI connection via PROFINET with a SIMATIC PC

Introduction

You configure an HMI connection over PROFINET or Ethernet between a HMI device and SIMATIC ET 200 CPU in the "Devices & Networks" editor.

Caution

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

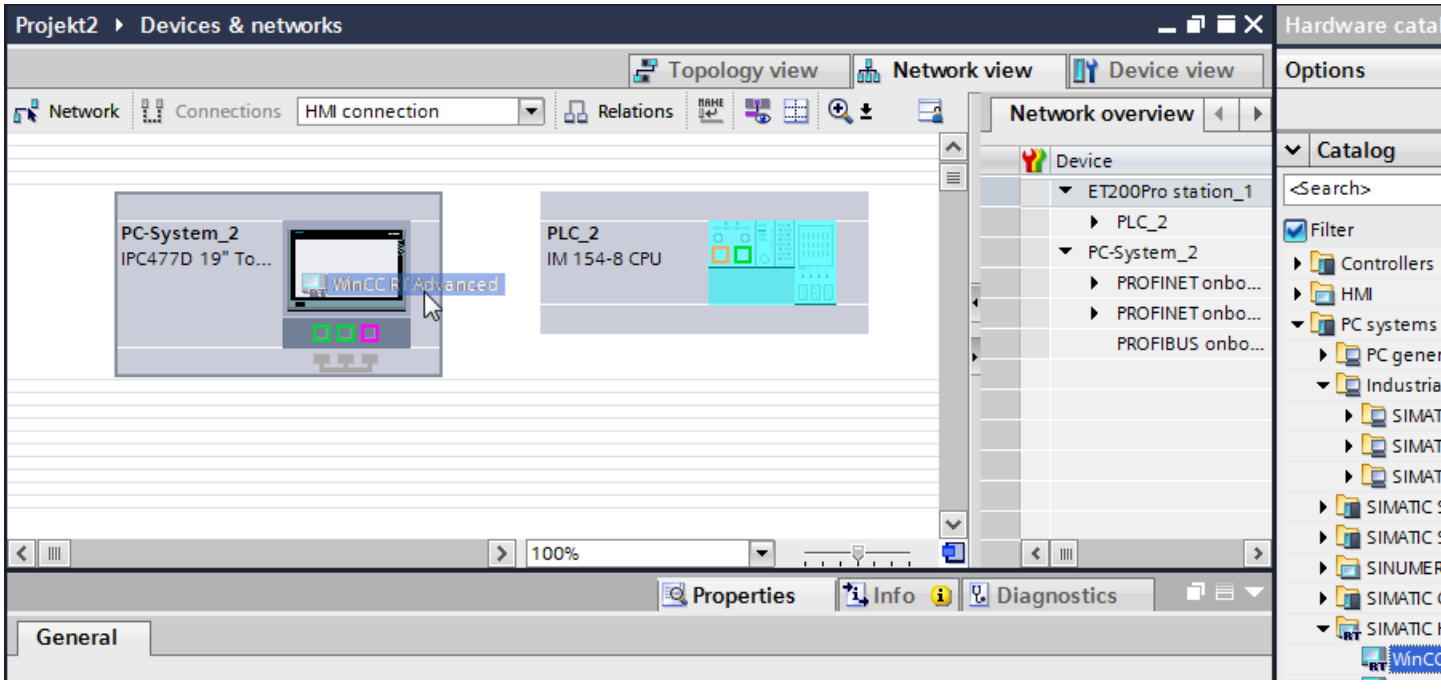
Requirements

The following communication partners are created in the "Devices & Networks" editor:

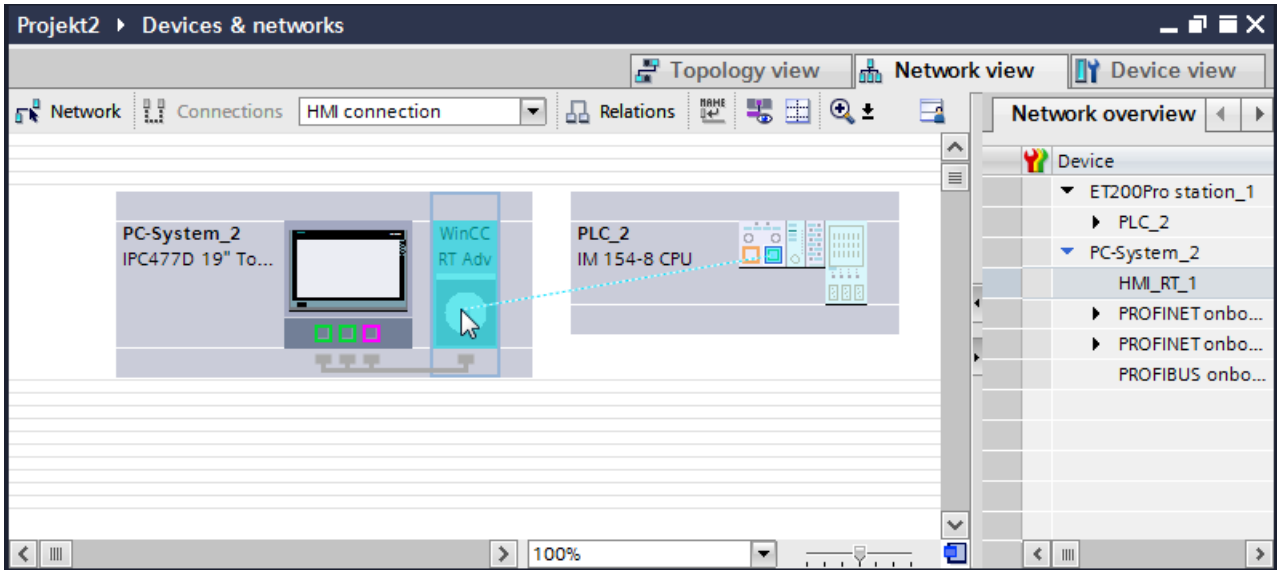
- SIMATIC ET 200 CPU with PROFINET interface
- SIMATIC PC with PROFINET interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a WinCC Runtime from the hardware catalog to the PC.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.
4. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the PC.



5. Click the connected interface on the PC station and select the entry "PC station" under "Interface assignment" in the Inspector window.
6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the section "PROFINET parameters (Page 6401)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC ET 200 CPU.

The IP address and subnet mask connection parameters are configured.

PROFINET parameters

PROFINET parameters for the HMI connection

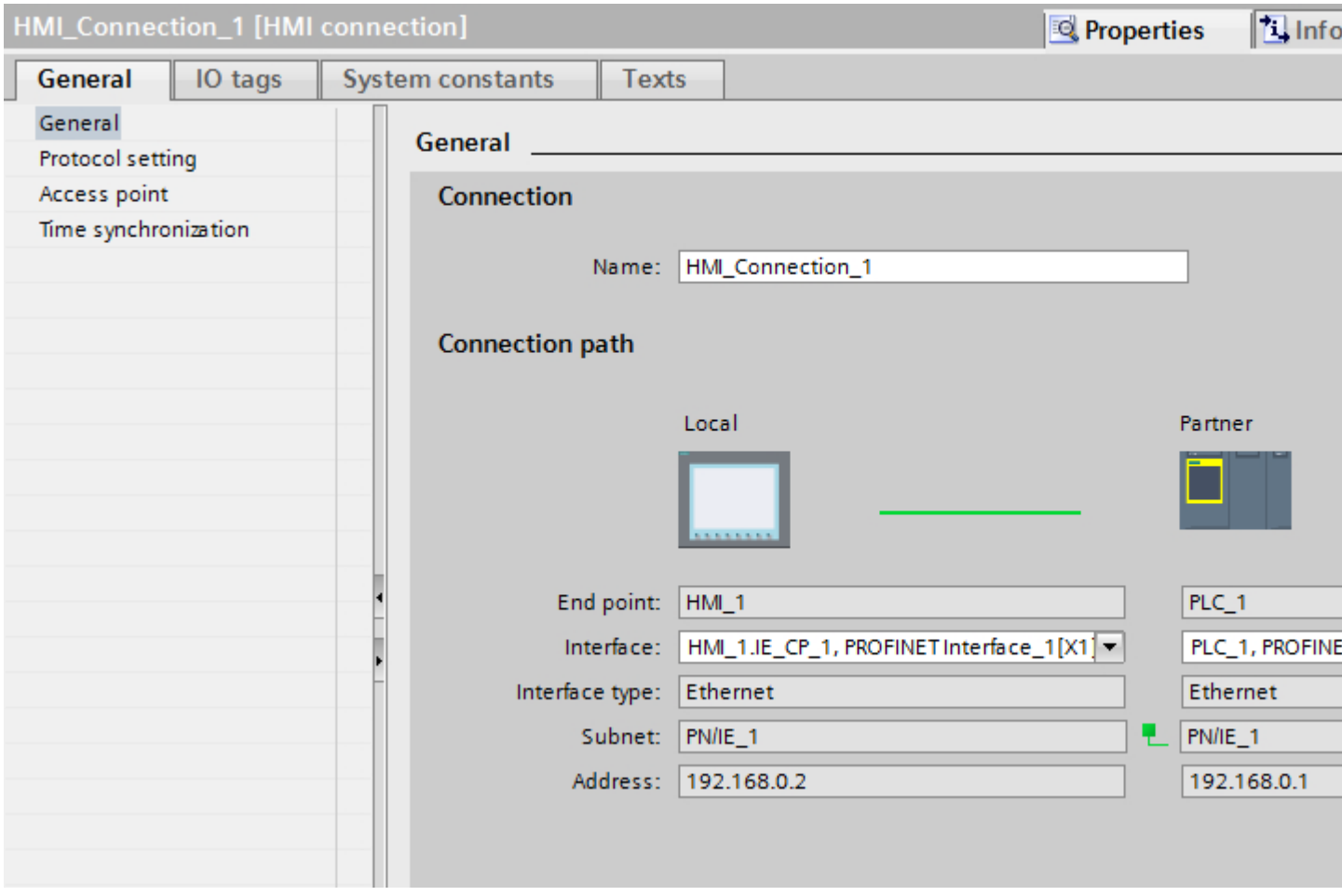
PROFINET parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Shows the name of the HMI connection.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFINET parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area cannot be edited.
- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the selected IP address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

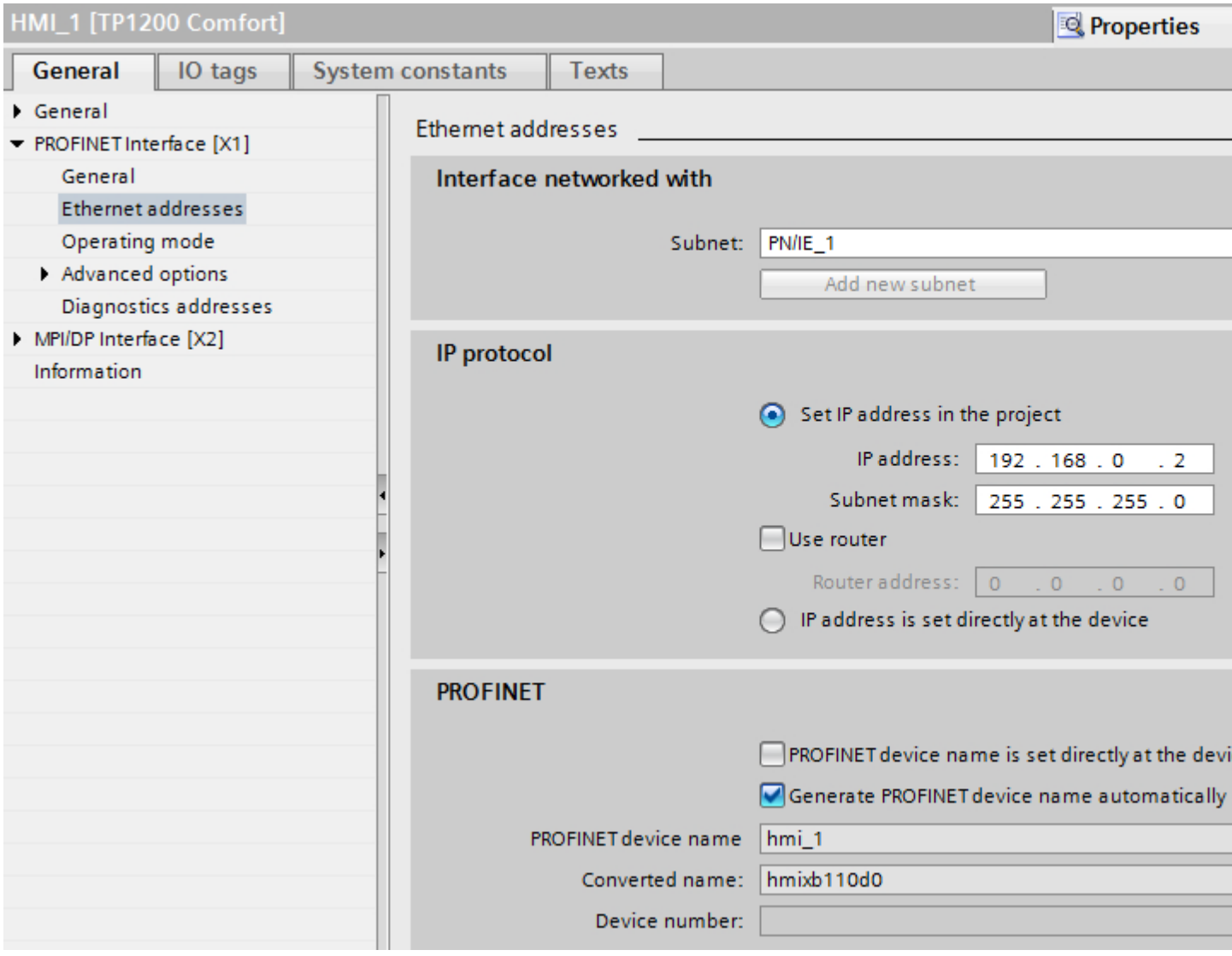
PROFINET parameters for the HMI device

PROFINET parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Set IP address in the project"
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

  Note

  The device is automatically restarted in the case of HMI devices with the Windows CE 3.0 operating system.

  HMI devices with Windows CE 3.0:
  - Mobile Panel 177 PN
  - Mobile Panel 177 DP

- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
- "Use IP router"
  If you are using an IP router, select "Use IP router" and enter the router address in the "Router address" field.
- "Set IP address directly on the device"
  If the "Set IP address on the device" function is enabled, the IP address is not taken from the project. You have to enter the IP address directly in the Control Panel of the HMI device.

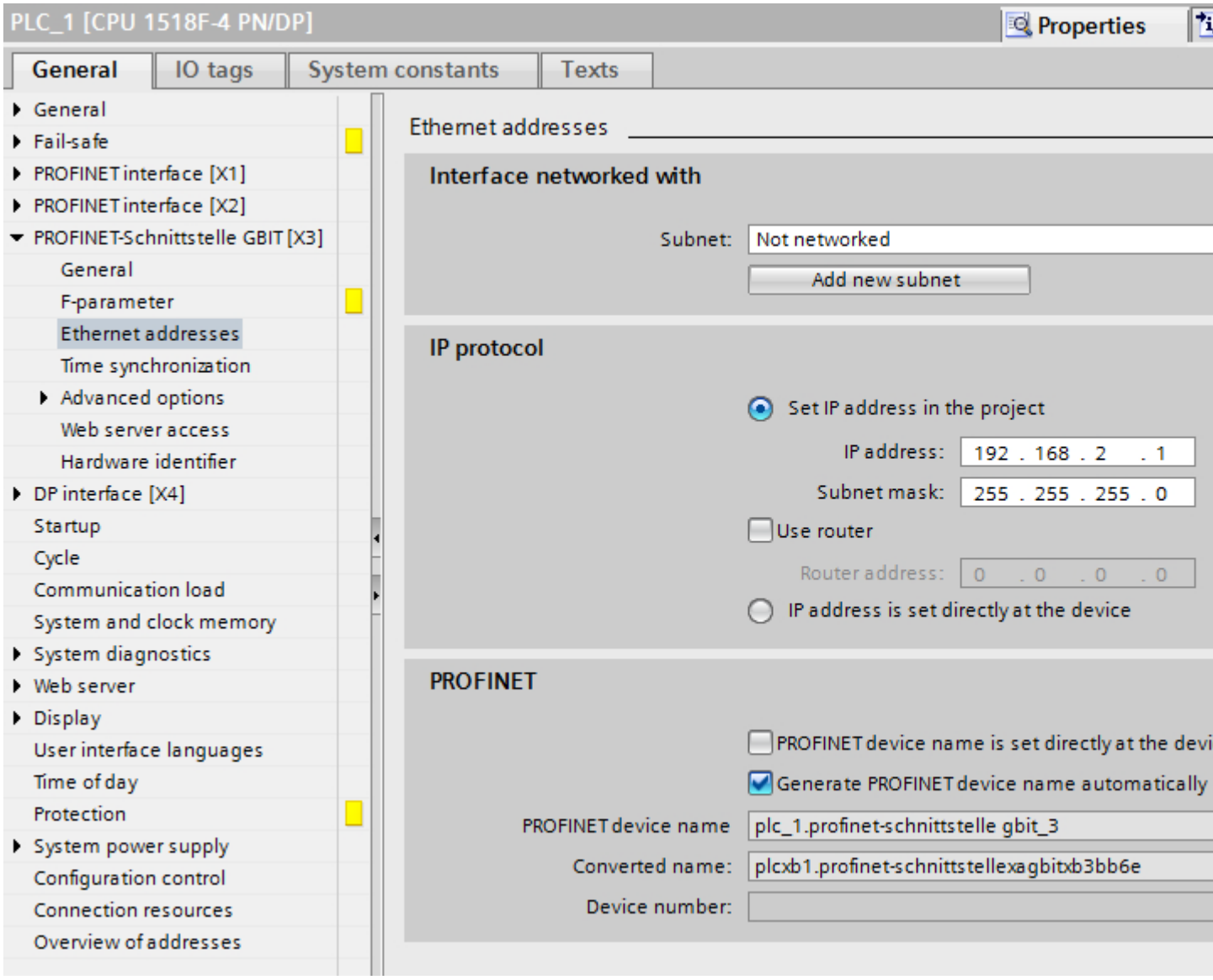
PROFINET parameters for the PLC

PROFINET parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFINET parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "IP address"
  You assign the IP address of the HMI device in the "IP address" area.
- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
  If you are using an IP router, select "Use IP router" and enter the router address in the field.

Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission rate / duplex

Depending on the selected device, you can make the following settings for "Transmission rate / duplex":

- Automatic setting
  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the peer port. The "Enable autonegotiation" option is also enabled as a default, in other words, you can use cross cables or patch cables for the connection.
- TP/ITP at x Mbps full duplex (half duplex)
  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:
  - Autonegotiation enabled
    You can use both cross cable and patch cable.
  - Autonegotiation disabled
    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.
- Deactivated
  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option enables or disables port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because with this option the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can accordingly not be optimally set.

Note

When a local port is connected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not accept the setting, an error message is generated.

Wiring rules for disabled autonegotiation

Requirements

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission rate
- Autonegotiation incl. autocrossing disabled

The time for negotiating the transmission rate during startup has been saved.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

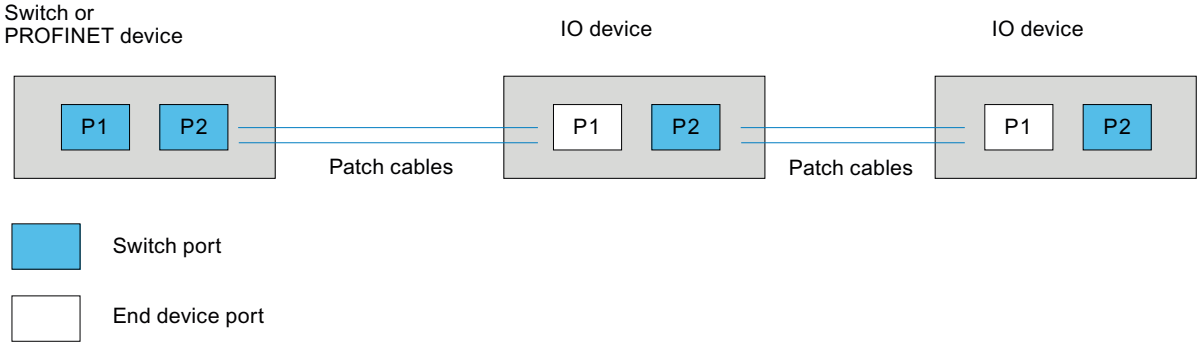
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in line using a single cable type (patch cable). To do this, you connect port 2 of the IO device (distributed I/O) with port 1 of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirements

To use boundaries, the respective device must have more than one port. If the PROFINET does not support boundary settings, they are not shown.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of discovery of accessible devices"
  No forwarding of DCP frames to identify accessible devices. Devices downstream from this port cannot be reached by the project tree under "Accessible devices". Devices downstream from this port cannot be reached by the CPU.
- "End of topology discovery"
  LLDP frames (Link Layer Discovery Protocol) are not forwarded for topology detection.
- "End of sync domain"
  No forwarding of sync frames transmitted to synchronize nodes within a sync domain.
  If you operate, for example, a PROFINET device with more than two ports in a ring, you should prevent the sync frame from being fed into the ring by setting a sync boundary (at the ports not inside the ring).
  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device from the other sync domain.

Restrictions

The following restrictions must be observed:

- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been determined for the port, the following check boxes cannot be used:
  - "End of discovery of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

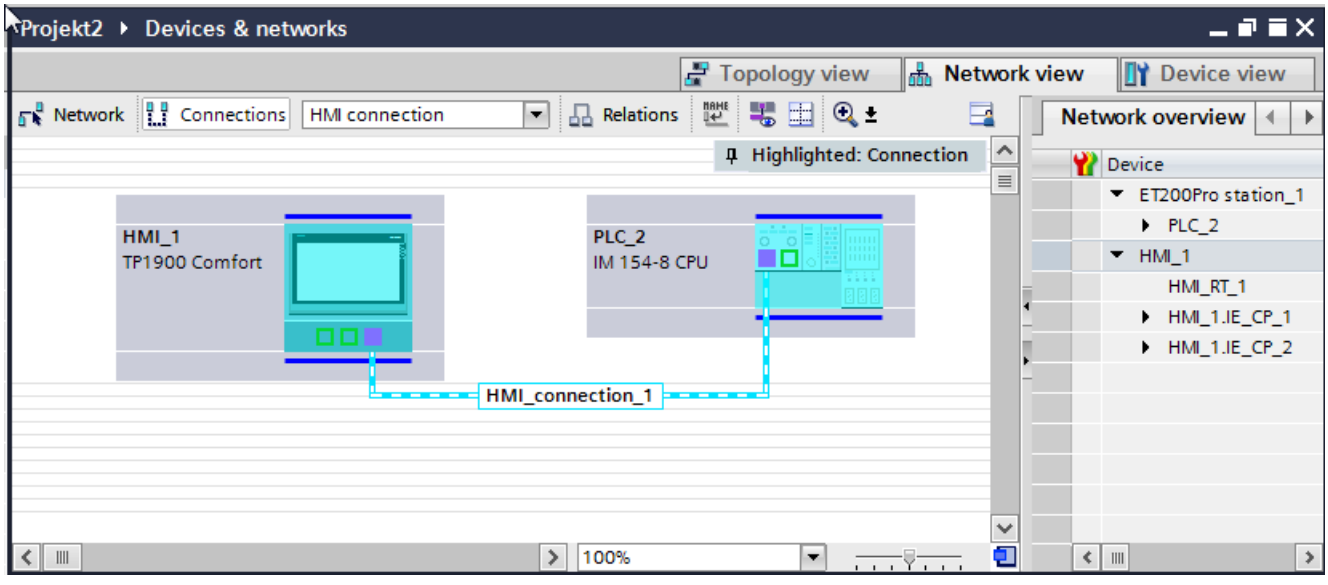
12.11.10.3 Communication via PROFIBUS

Configuring an HMI connection

Communication via PROFIBUS

HMI connections via PROFIBUS

If you want to connect a SIMATIC ET 200 CPU to a HMI device via PROFIBUS, you must configure a PROFIBUS-capable communication module to a slot of the controller first.



HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS

Introduction

You configure an HMI connection between HMI devices and a SIMATIC ET 200 CPU via PROFIBUS in the "Devices & Networks" editor.

Requirements

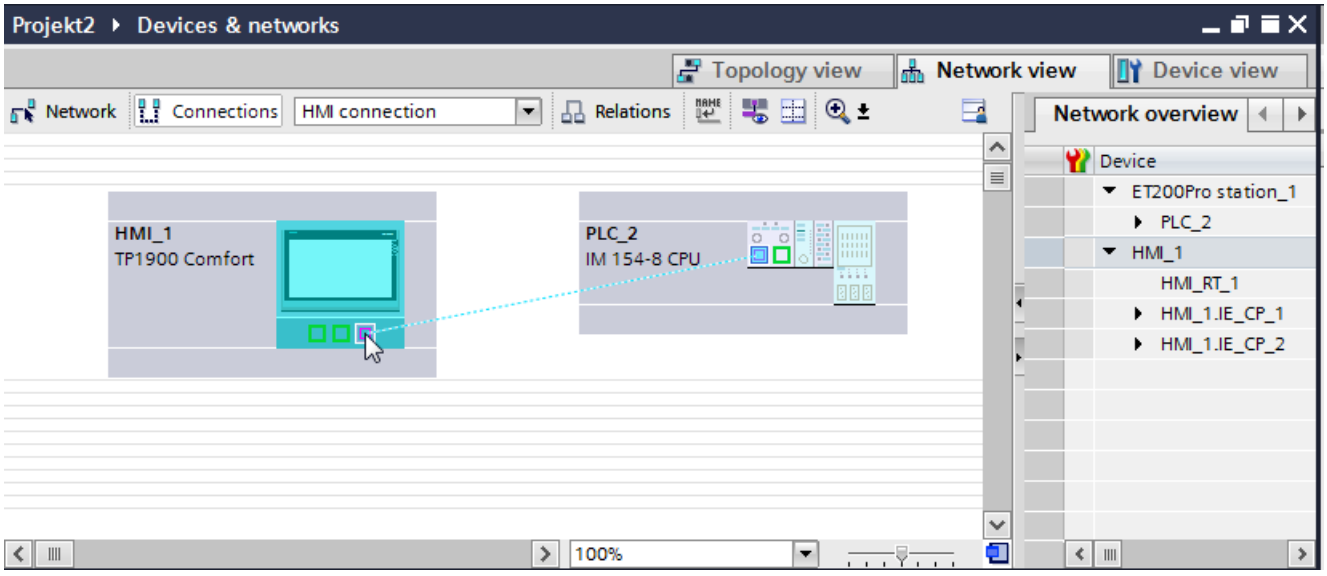
The following communication partners are created in the "Devices & Networks" editor:

- HMI device with MPI/DP interface
- SIMATIC ET 200 CPU

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button.
   The devices available for connection are highlighted in color.
3. Click the HMI device interface.
4. Select the "PROFIBUS" interface type in the Inspector window under "Properties > General > PROFIBUS address/ MPI address > Parameters".

5. Click the interface of the CPU and use a drag-and-drop operation to draw a connection to the HMI device.



6. Click the name of the connection.
   The connection is displayed graphically in the Inspector window.
7. Click "Highlight HMI connection" and select the HMI connection.
8. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the section "PROFIBUS parameters (Page 6417)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC ET 200 CPU via PROFIBUS.

Configuring an HMI connection

Communication via PROFIBUS

Communication via PROFIBUS

This section describes the communication between a WinCC Runtime and the SIMATIC ET 200 CPU controller via PROFIBUS.

You can use the following WinCC Runtimes as an HMI device:

- WinCC RT Advanced
- WinCC RT Professional

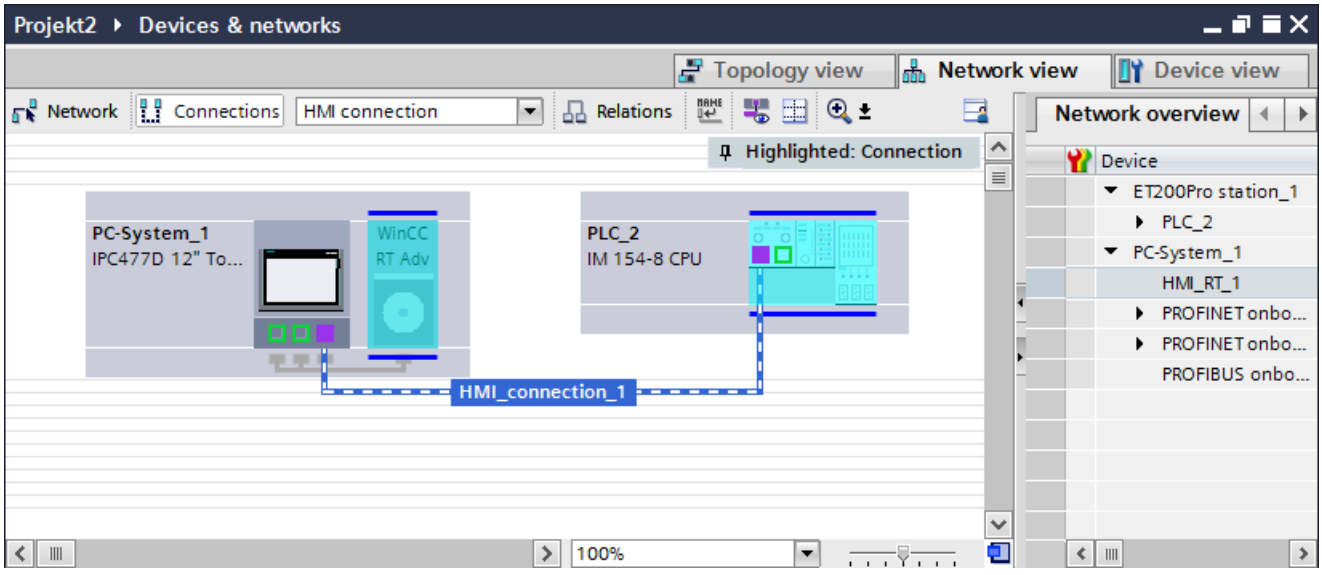
If you want to connect a SIMATIC ET 200 CPU to a HMI device via PROFIBUS, you must configure a PROFIBUS-capable communication module to a slot of the controller.

WinCC Runtime as HMI device

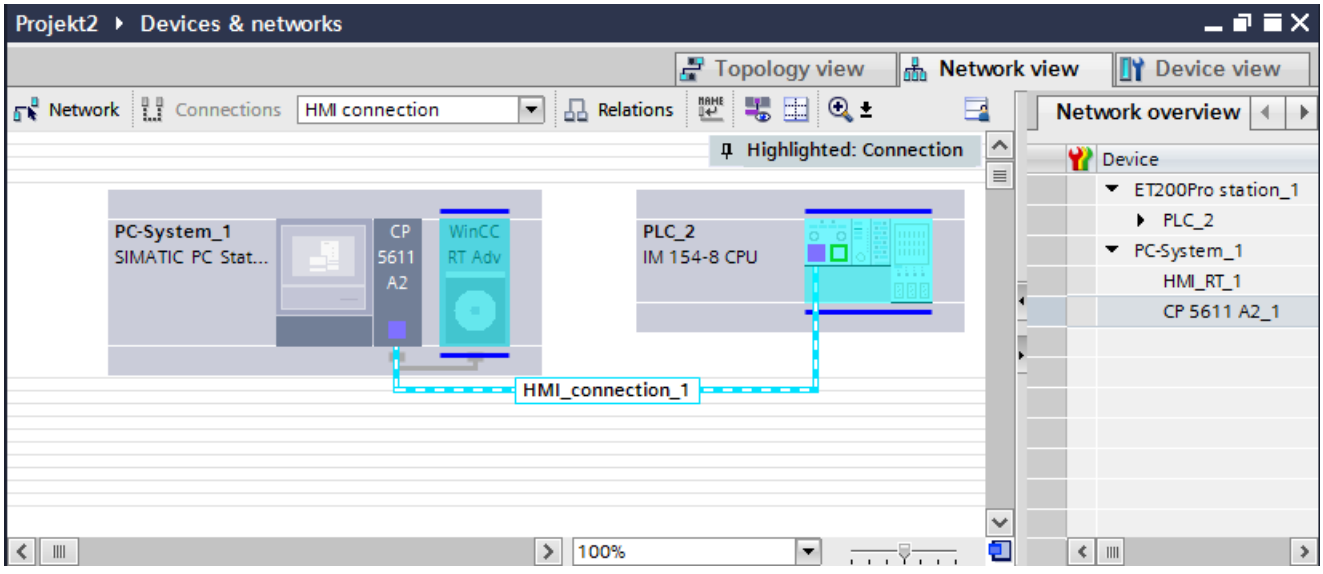
Configure the HMI connections between WinCC Runtime and SIMATIC ET 200 CPU in the "Devices & Networks" editor.

If you want to use a WinCC Runtime as an HMI device, you have the following options:

1. You create a SIMATIC PC and configure a WinCC Runtime on this SIMATIC PC.
   In this way you use the SIMATIC PC with a WinCC Runtime as an HMI device.



2. You create a WinCC Runtime and configure a communication processor to the Runtime.
   In this way you use your configuration PC with a WinCC Runtime as an HMI device.



You can also connect multiple HMI devices to a single SIMATIC ET 200 CPU and multiple SIMATIC ET 200 CPU to a single HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFIBUS in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS with a SIMATIC PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC ET 200 CPU via PROFIBUS in the "Devices & Networks" editor.

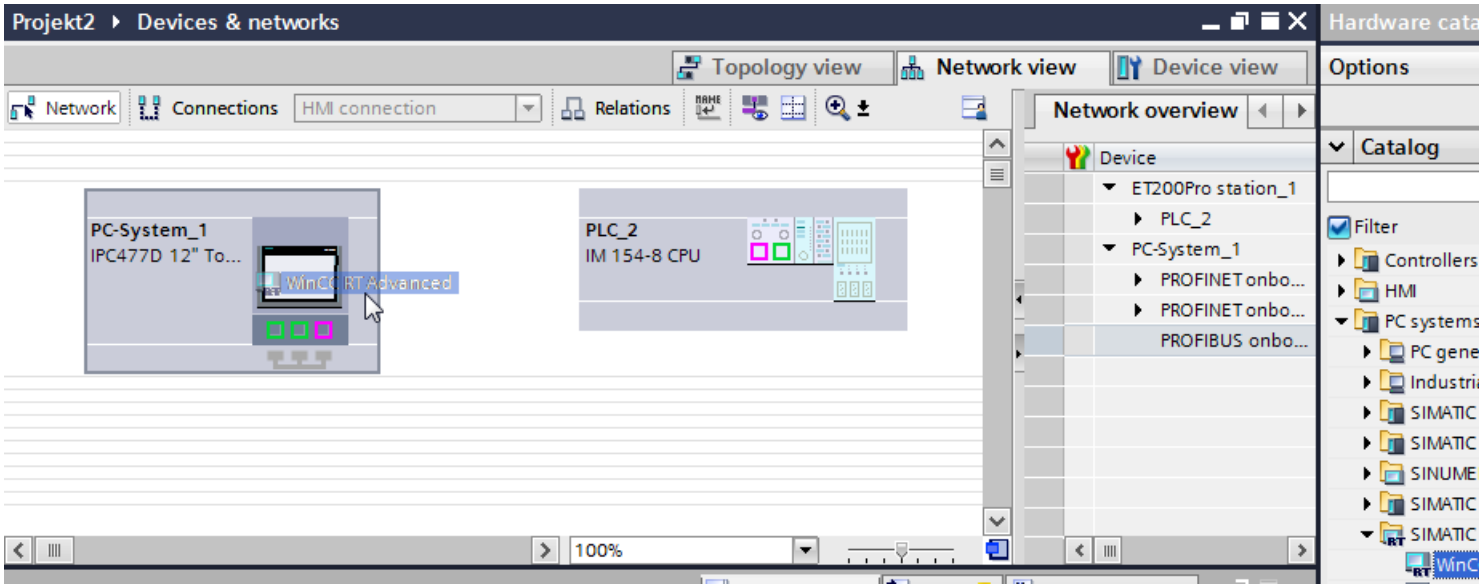
Requirements

The following communication partners are created in the "Devices & Networks" editor:

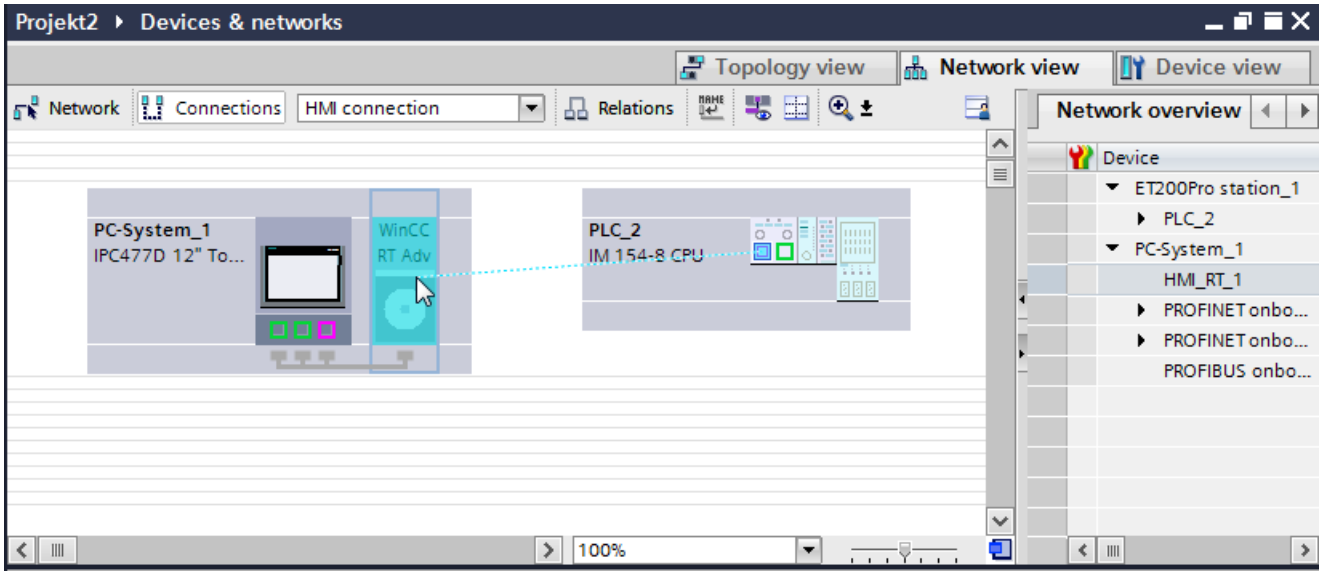
- SIMATIC ET 200 CPU
- SIMATIC PC with PROFIBUS interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a WinCC Runtime from the hardware catalog to the PC.



3. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.
4. Click the PROFIBUS interface of the CPU and use a drag-and-drop operation to draw a connection to the PROFIBUS interface of the PC.



5. Click the connecting line.
6. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
7. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the section "PROFIBUS parameters (Page 6417)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab.

You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC ET 200 CPU via PROFIBUS.

Configuring an HMI connection via PROFIBUS with a PC

Introduction

You configure an HMI connection between HMI devices and a SIMATIC ET 200 CPU via PROFIBUS in the "Devices & Networks" editor.

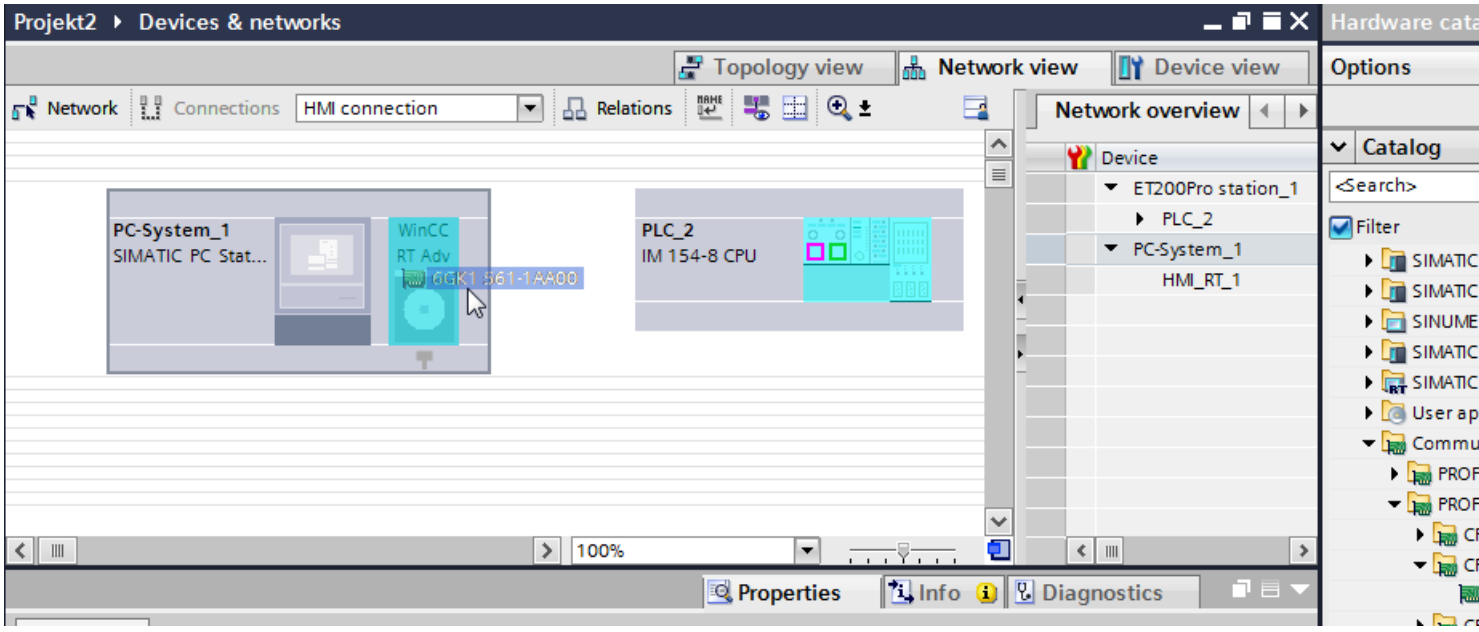
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- SIMATIC ET 200 CPU
- PC station with a WinCC RT Advanced or WinCC RT Professional

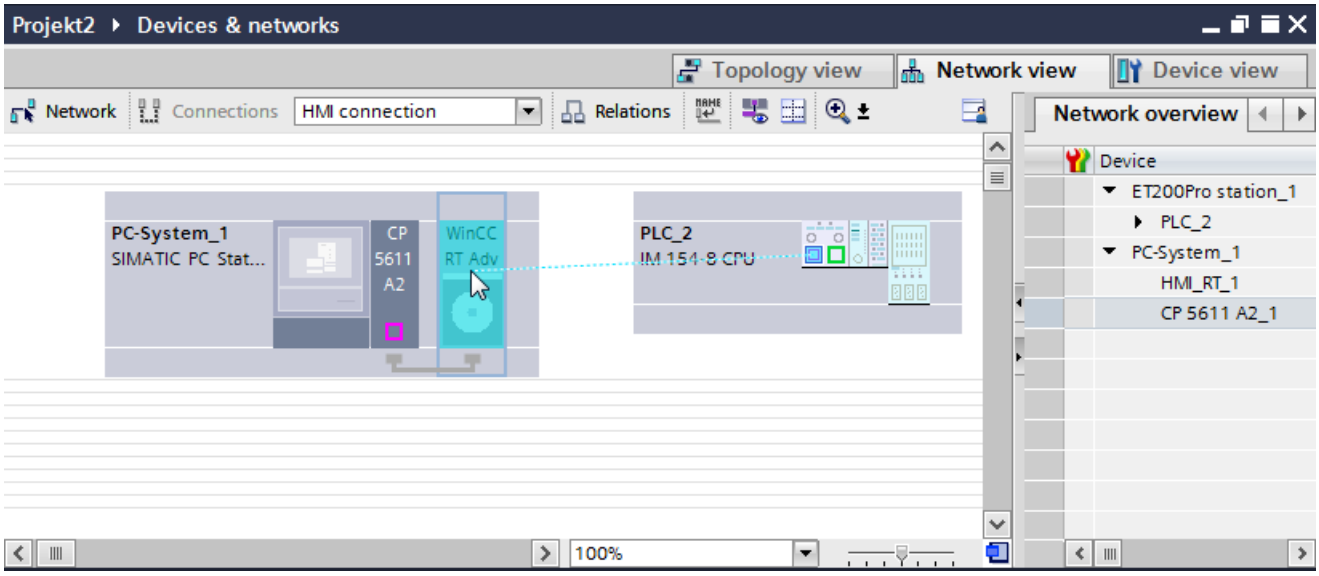
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed in the network view.
2. Use a drag-and-drop operation to move a PROFIBUS-capable communication processor from the hardware catalog to the WinCC Runtime.
3. Use a drag-and-drop operation to move a PROFIBUS-capable communication module from the hardware catalog to the PLC.



4. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

5. Click the PROFIBUS interface of the communication module and use a drag-and-drop operation to draw a connection to the PROFIBUS interface of the communication processor.



6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the section "PROFIBUS parameters (Page 6417)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab.

You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC ET 200 CPU via PROFIBUS.

PROFIBUS parameters

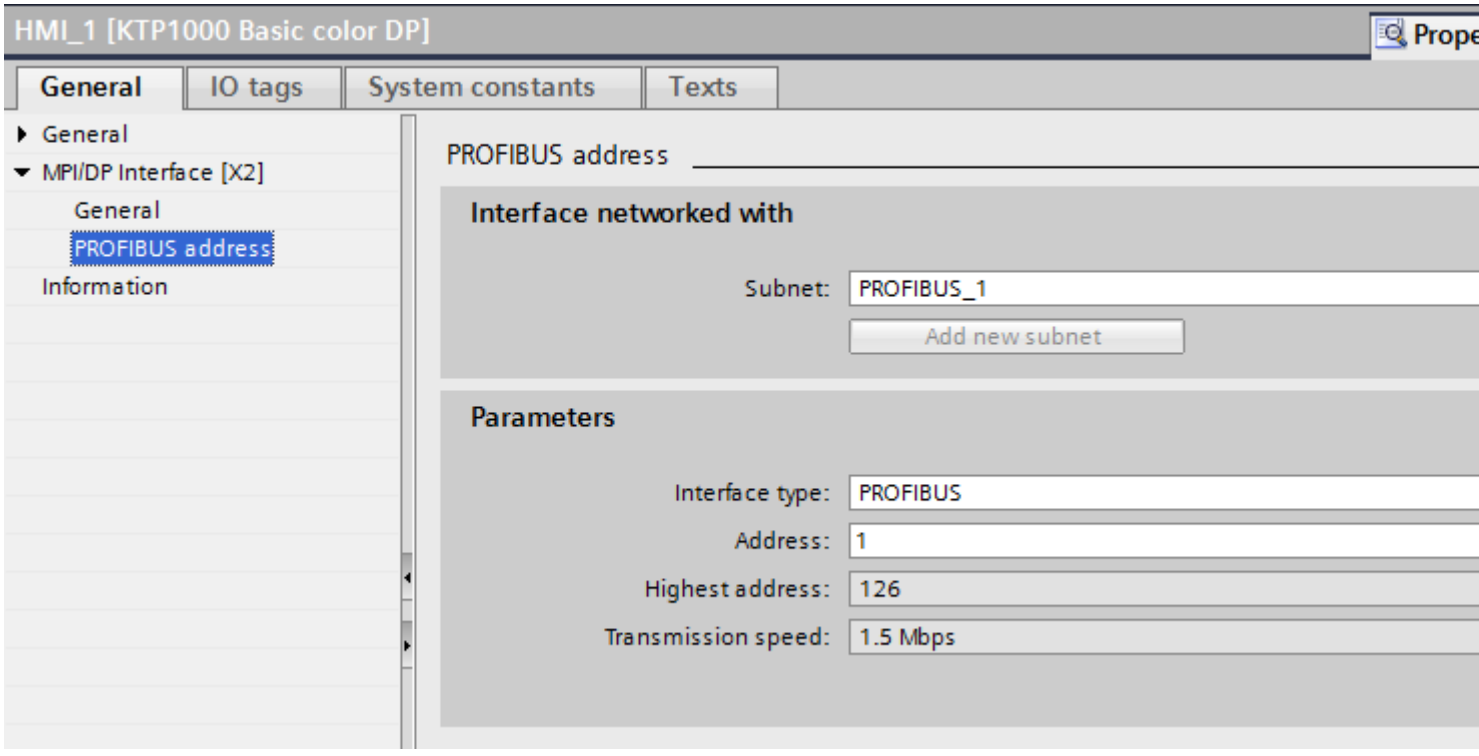
PROFIBUS parameters for the HMI device

PROFIBUS parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFIBUS parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

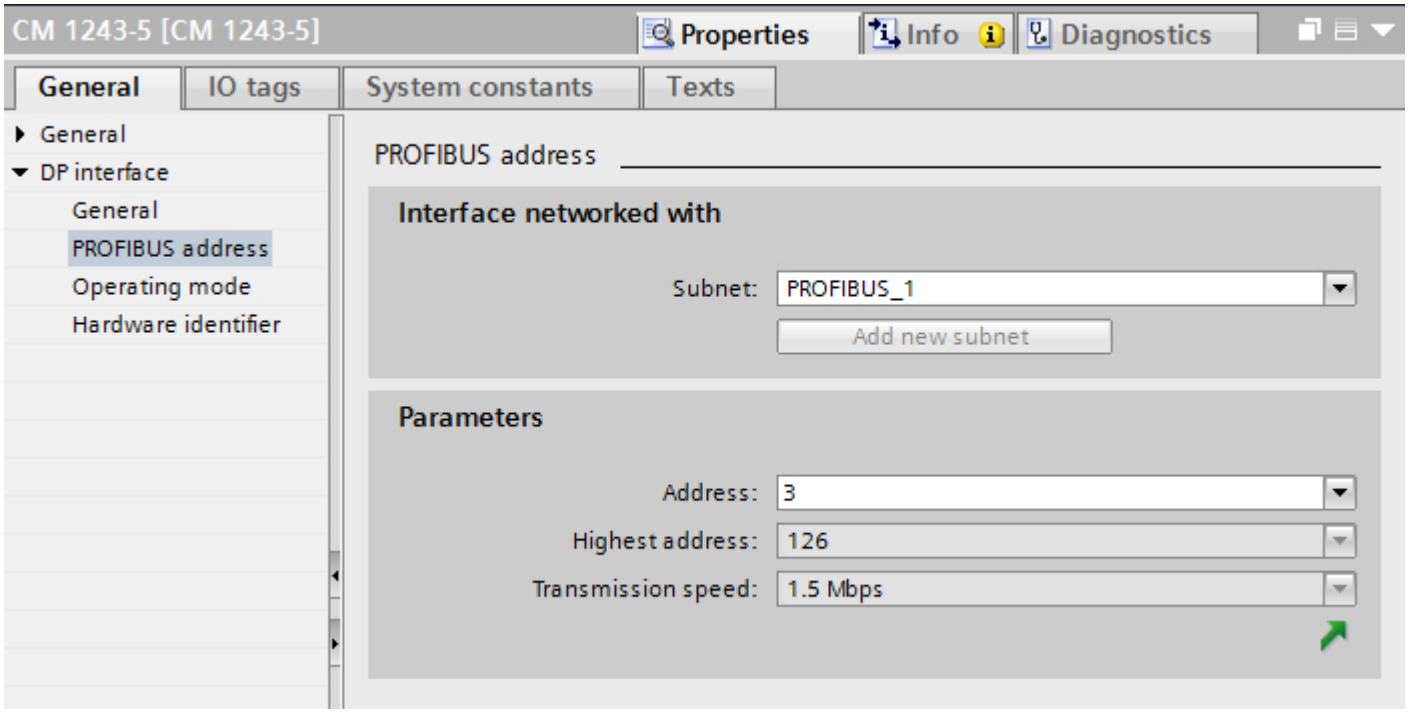
PROFIBUS parameters for the PLC

PROFIBUS parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.

- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

PROFIBUS parameters for the HMI connection

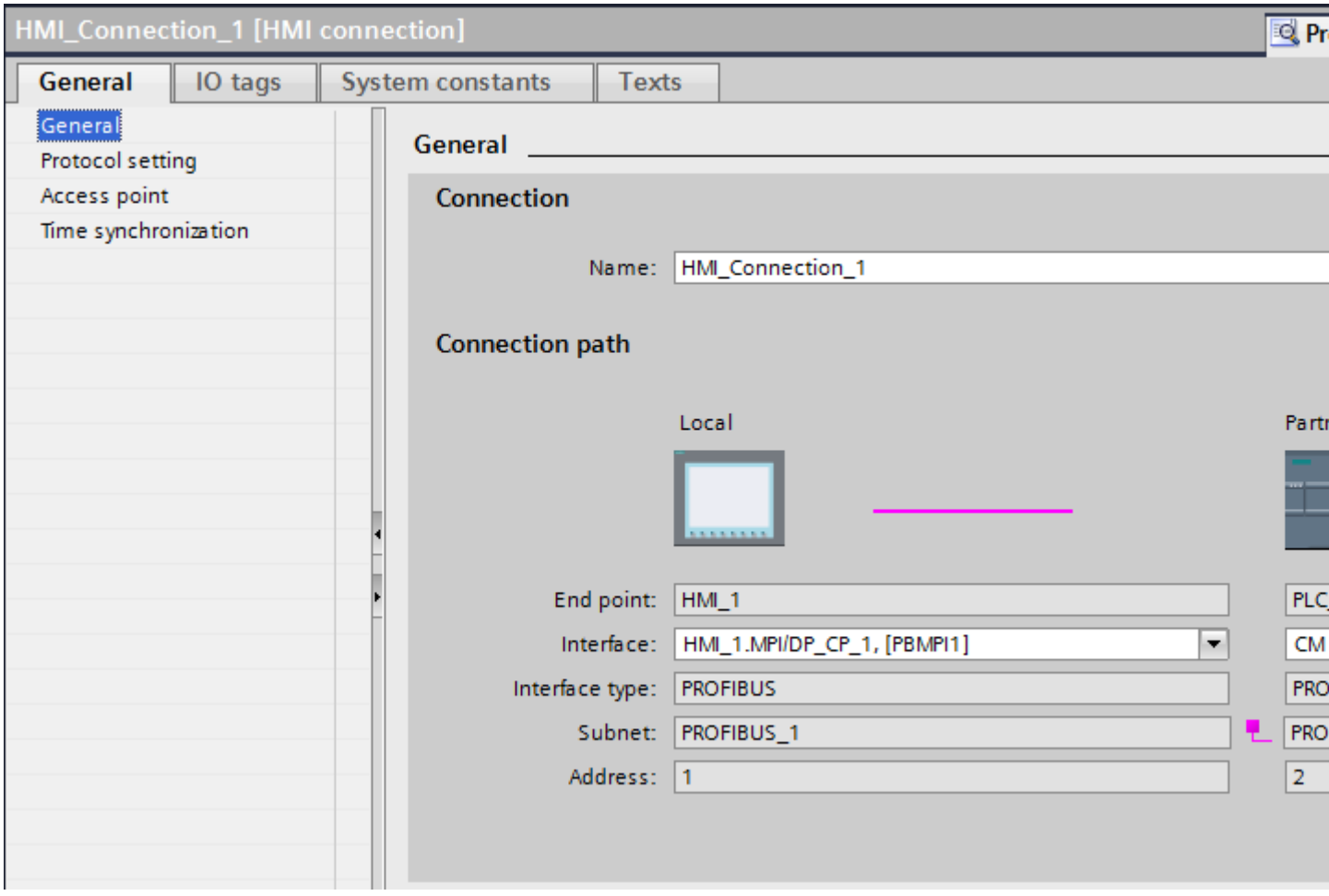
PROFIBUS parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and editing HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

The "Connection" area displays the HMI connection created for communication between the devices.

You can edit the name of the HMI connection in this area.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFIBUS parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"

Displays the selected interface type. This area cannot be edited.

- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the PROFIBUS address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

12.11.10.4 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Auto-Hotspot

Area pointer "Date/time"

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Permitted data types

You can use the following data types when you configure the "Date/Time" area pointer:

- Int
- UInt
- Word
- DTL

Use of the "DTL" data type

Use of data type "DTL" with communication driver S7-1200.

A tag of the "DTL" data type has a length of 12 bytes and saves information on date and time in a predefined structure.

The "DTL" data type has the following structure:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2554
1			
2	Month	USINT	0 to 12
3	Day	USINT	1 to 31
4	Day of week	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanoseconds	UDINT	0 to 999 999 999
9			
10			
11			

The "DTL" data type supports time information down to the nanosecond range. Because panels only support time information down to the millisecond, you may encounter the following restriction when using the area pointers:

For the transmission of time information from a panel to the controller, the smallest unit of time is 1 millisecond. The value range from microseconds to nanoseconds of the "DTL" data type is filled with zeros.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the date/time area pointer PLC to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Permitted data types

You can use the following data types when you configure the "Date/Time PLC" area pointer:

- DTL

Use of the "DTL" data type

Use of data type "DTL" with communication driver S7-1200. A tag of the "DTL" data type has a length of 12 bytes and saves information on date and time in a predefined structure.

The "DTL" data type has the following structure:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2554
1			

Byte	Component	Data type	Value range
2	Month	USINT	0 to 12
3	Day	USINT	1 to 31
4	Day of week	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanoseconds	UDINT	0 to 999 999 999
9			
10			
11			

The HMI devices do not support the use of nanoseconds. Values in the nanosecond range will be ignored during processing in Runtime.

The "DTL" data type supports time information down to the nanosecond range. Because panels only support time information down to the millisecond, you may encounter the following restriction when using the area pointers:

For the transmission of time information from a controller to a panel, the range from microseconds to nanoseconds is ignored. The time information is processed on the panel down to milliseconds.

Area pointer "Coordination"

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

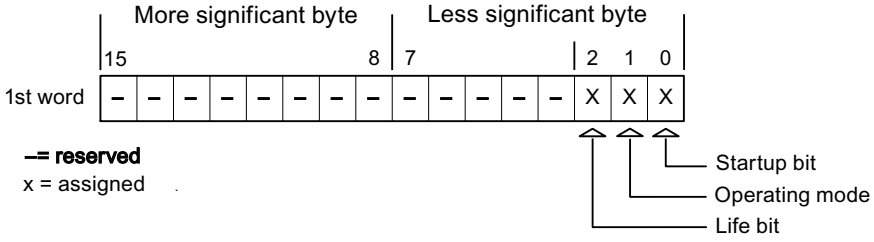
By default, the "Coordination" area pointer has the length of one word and cannot be changed.

Usage

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of the bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

Processing in the PLC

For a simpler evaluation in the PLC program, use a Bool array for this area pointer when using the SIMATIC S7 1200 communication driver. You will have to map the complete 16-bit word of the area pointer. Configure a tag of the data type "Array [0 .. 15] of bool" for this purpose.

Permitted data types

You can use the following data types when you configure the "Coordination" area pointer.

- Word
- UInt
- Bool

Area pointer "Screen number"

Function

The HMI devices store information about the screen called up on the HMI device in the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. Certain reactions can be triggered in the PLC, such as the call of a different screen.

Use

Before the "Screen number" area pointer can be used, it must be set up and activated by selecting "Communication ▸ Area pointer". You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1. Word	Current screen type															
2. Word	Current screen number															
3. Word	Reserved															
4th word	Current field number															
5. Word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

Note

Device dependency

Permanent windows are not available on Basic Panels.

Permitted data types

You can use the following data types when you configure the "Screen number" area pointer.

- Word
- UInt

Area pointer "Project ID"

Function

When Runtime starts, a check can be carried out as to whether the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program.

A missing compatibility results in a corresponding alarm and Runtime will not be started.

Use

Note

HMI connections cannot be switched "online".

The HMI connection in which the "Project ID" area pointer is used must be switched "online".

To use this area pointer, set up the following during the configuration:

- Define the version of the configuration. Values between 1 and 255 are possible.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- This is where you select the PLC tag or the tag array that you have configured as the data area for the area pointer.

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

Permitted data types

You can use the following data types when you configure the "Project ID" area pointer.

- Word
- UInt

Area pointer "Job mailbox"

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	**Set date (BCD-coded)**	

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	**User logon**	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	**User logoff**	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	**Transfer date/time to PLC**	
	An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	**Transfer date/time to PLC**	
	An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	**Update tag**	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in Parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	**Clear event buffer**	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	**Clear error alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	**Display selection**	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	**Reading data record from PLC** [1)]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	**Writing data record to PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1]	Only devices supporting recipes
[2]	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
[3]	The weekday is ignored on HMI device KTP 600 BASIC PN.

Permitted data types

You can use the following data types when you configure the "Screen number" area pointer:

- Word
- UInt

"Data record" area pointer

"Data mailbox" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization over the data mailbox

Data records are always transferred directly. That is, the tag values are read from an address or written to an address configured for this tag directly, without redirecting the values by means of interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a job mailbox, the data in the recipe view will be updated as well. Avoid operating the recipe view while job mailboxes for transfer of data records are being triggered. If you have already started editing a data record and a job mailbox is triggered for transfer of data records, then this job mailbox will be rejected.

Permitted data types

You can use the following data types when you configure the "Data record" area pointer.

- Word
- UInt

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example, when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then process, edit, or save these values, for example.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. Use this mechanism to prevent uncontrolled overwriting of data in either direction of your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Connections ▸ Area pointer" editor.
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data record free
2	0000 0010	Transfer is busy
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer started by the operator in the recipe display

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data mailbox and sets the data record number to 0.	Abort with system alarm.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data mailbox.	Abort with system alarm.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxes No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data mailboxes from the PLC to the HMI device. The job mailbox is structured as follows:

	Most significant byte	Least significant byte
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox no. 70 transfers data mailboxes from the HMI device to the PLC. The job mailbox is structured as follows:

	Most significant byte	Least significant byte
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device reads the values and stores the values in the data record specified in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed". • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the job from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The PLC program can now evaluate the transferred data. To allow further transfers, the PLC program must set the status word to 0 again.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system alarm.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	

Step	Action	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system alarm.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible

- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system alarms
- Triggered by function
  Output of system alarms
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data mailbox.

Note

Availability for specific devices

Notes in the status bar of the recipe view are not available in Basic Panels.

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Auto-Hotspot

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

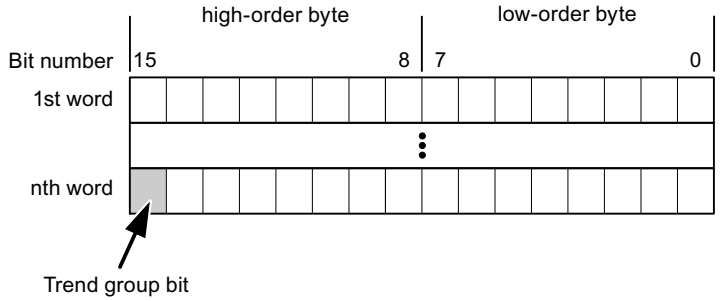
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Permitted data types for trends

For SIMATIC S7

You assign one bit to each trend during configuration. Tags and array tags of the "Word" or "Int" data type are permitted.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Auto-Hotspot

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

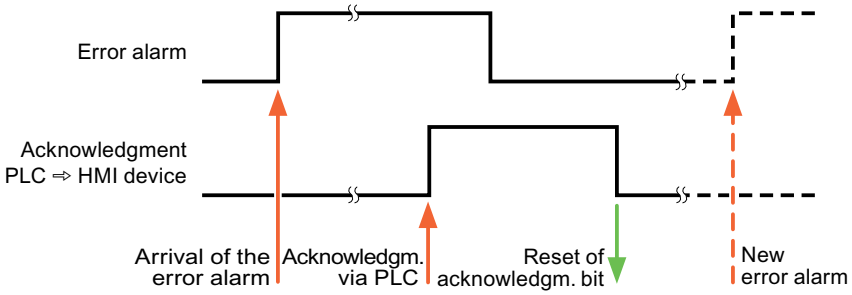
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

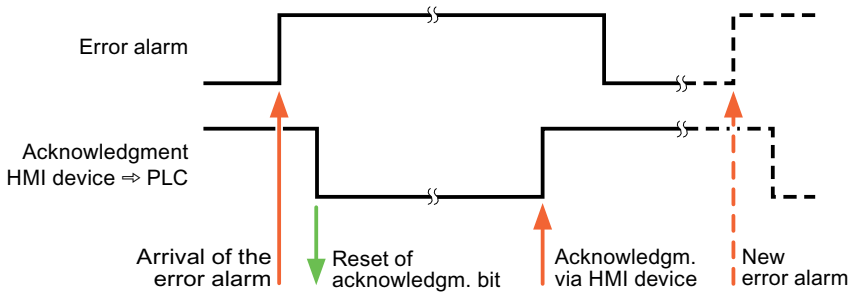
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



LED mapping

Function

Keyboard devices have LEDs in the function keys. These LEDs can be activated from the PLC. Thus, it is possible, for example, to signal to the user which key he should press in a given situation by lighting up an LED.

Note

The LED function cannot be configured for Basic Panels.

Requirements

In order to activate an LED, an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration.

LED assignment

The assignment of the individual LEDs to the bits in the LED tags is specified when the function keys are configured. In this process, the "LED tag" and the assigned "Bit" are specified for each function key in the "General" group of the properties window.

The "Bit" bit number designates the first of two consecutive bits that control the following LED statuses.

Bit n+ 1	Bit n	LED function	
		All Mobile Panels, all Operator Panels, all Multi Panels, all Comfort Panels	Panel PCs
0	0	Off	Off
0	1	Rapid flashing	Flashing
1	0	Slow flashing	Flashing
1	1	On permanently	On permanently

12.11.10.5 Performance features of communication

Device dependencies of SIMATIC ET 200 CPU

Communication with SIMATIC ET 200 CPU controller

If you use devices from an earlier version of TIA Portal with TIA Portal V12 SP1, it may not be possible to configure integrated connections to certain HMI devices.

Basic Panels V11.0

HMI devices	SIMATIC ET 200 CPU
KP300 Basic	Yes
KP400 Basic	Yes
KTP400 Basic PN	Yes
KTP600 Basic DP	Yes
KTP600 Basic PN	Yes
KTP1000 Basic DP	Yes
KTP1000 Basic PN	Yes
TP1500 Basic PN	Yes

Basic Panels V12.0

HMI devices	SIMATIC ET 200 CPU
KP300 Basic	Yes
KP400 Basic	Yes
KTP400 Basic PN	Yes
KTP600 Basic DP	Yes
KTP600 Basic PN	Yes
KTP1000 Basic DP	Yes
KTP1000 Basic PN	Yes
TP1500 Basic PN	Yes

Basic Panels V13.0

HMI devices	SIMATIC ET 200 CPU
KTP400 Basic PN	Yes
KTP700 Basic PN / DP	Yes

HMI devices	SIMATIC ET 200 CPU
KTP900 Basic PN	Yes
KTP1200 Basic PN / DP	Yes

Basic Panels V13.0.1

HMI devices	SIMATIC ET 200 CPU
KTP400 Basic PN	Yes
KTP700 Basic PN / DP	Yes
KTP900 Basic PN	Yes
KTP1200 Basic PN / DP	Yes

Panels V11.0

HMI devices	SIMATIC ET 200 CPU
OP 73	Yes
OP 77A	Yes
OP 77B	Yes
TP 177A	Yes
TP 177A Portrait	Yes
TP 177B 4"	Yes
TP 177B 6" mono	Yes
TP 177B 6"	Yes
OP 177B 6" mono	Yes
OP 177B 6"	Yes
TP 277 6"	Yes
OP 277 6"	Yes

Multi Panels V11.0

HMI devices	SIMATIC ET 200 CPU
MP 177 6" Touch	Yes
MP 277 8" Key	Yes
MP 277 10" Key	Yes
MP 277 10" Touch	Yes
MP 377 12" Key	Yes
MP 377 12" Touch	Yes
MP 377 15" Touch	Yes
MP 377 19" Touch	Yes

Multi Panels V12.0

HMI devices	SIMATIC ET 200 CPU
MP 177 6" Touch	Yes
MP 277 8" Key	Yes
MP 277 10" Key	Yes
MP 277 10" Touch	Yes
MP 377 12" Key	Yes
MP 377 12" Touch	Yes
MP 377 15" Touch	Yes
MP 377 19" Touch	Yes

Mobile Panels V11.0

HMI devices	SIMATIC ET 200 CPU
Mobile Panel 177 6" DP	Yes
Mobile Panel 177 6" PN	Yes
Mobile Panel 277 8"	Yes
Mobile Panel 277 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	Yes
Mobile Panel 277 10"	Yes

Mobile Panels V12.0

HMI devices	SIMATIC ET 200 CPU
Mobile Panel 177 6" DP	Yes
Mobile Panel 177 6" PN	Yes
Mobile Panel 277 8"	Yes
Mobile Panel 277 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	Yes
Mobile Panel 277 10"	Yes

Mobile Panels V13.0.1

HMI devices	SIMATIC ET 200 CPU
KTP 700 Mobile	Yes
KTP 900 Mobile	Yes

Comfort Panels V11.0

HMI devices	SIMATIC ET 200 CPU
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V12.0

HMI devices	SIMATIC ET 200 CPU
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes

HMI devices	SIMATIC ET 200 CPU
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V13.0

HMI devices	SIMATIC ET 200 CPU
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V13.0.1

HMI devices	SIMATIC ET 200 CPU
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes

HMI devices	SIMATIC ET 200 CPU
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Runtime V11.0

HMI devices	SIMATIC ET 200 CPU
WinCC RT Advanced	Yes
WinCC RT Professional	Yes

Runtime V12.0

HMI devices	SIMATIC ET 200 CPU
WinCC RT Advanced	Yes
WinCC RT Professional	Yes

Runtime V13.0

HMI devices	SIMATIC ET 200 CPU
WinCC RT Advanced	Yes
WinCC RT Professional	Yes

Runtime V13.0.1

HMI devices	SIMATIC ET 200 CPU
WinCC RT Advanced	Yes
WinCC RT Professional	Yes

Valid data types for SIMATIC ET 200 CPU

Valid data types for connections with SIMATIC ET 200 CPU

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
BOOL	1 bit
SINT	1 byte
INT	2 bytes
DINT	4 bytes
USINT	1 byte
UINT	2 bytes
UDINT	4 bytes
REAL	4 bytes
LREAL	8 bytes
TIME	4 bytes
DATE	2 bytes
TIME_OF_DAY, TOD	4 bytes
STRING	(2+n) bytes, n = 0 to 254
CHAR	1 byte
Array of CHAR	--
BYTE	1 byte
WORD	2 bytes
DWORD	4 bytes
Date_And_Time	8 bytes
DTL	8 bytes
LDT	8 bytes

12.11.10.6 Configuring connections in the "Connections" editor

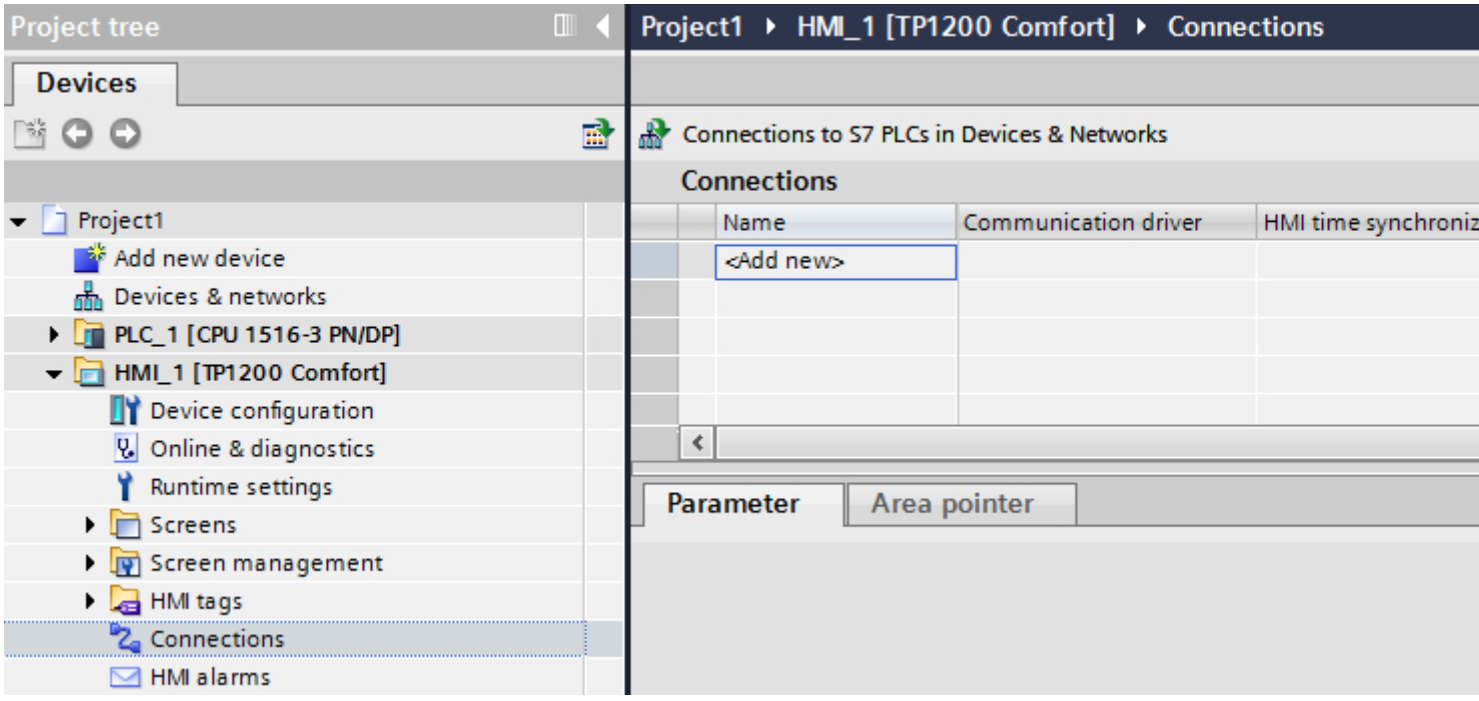
Creating a PROFINET connection

Requirements

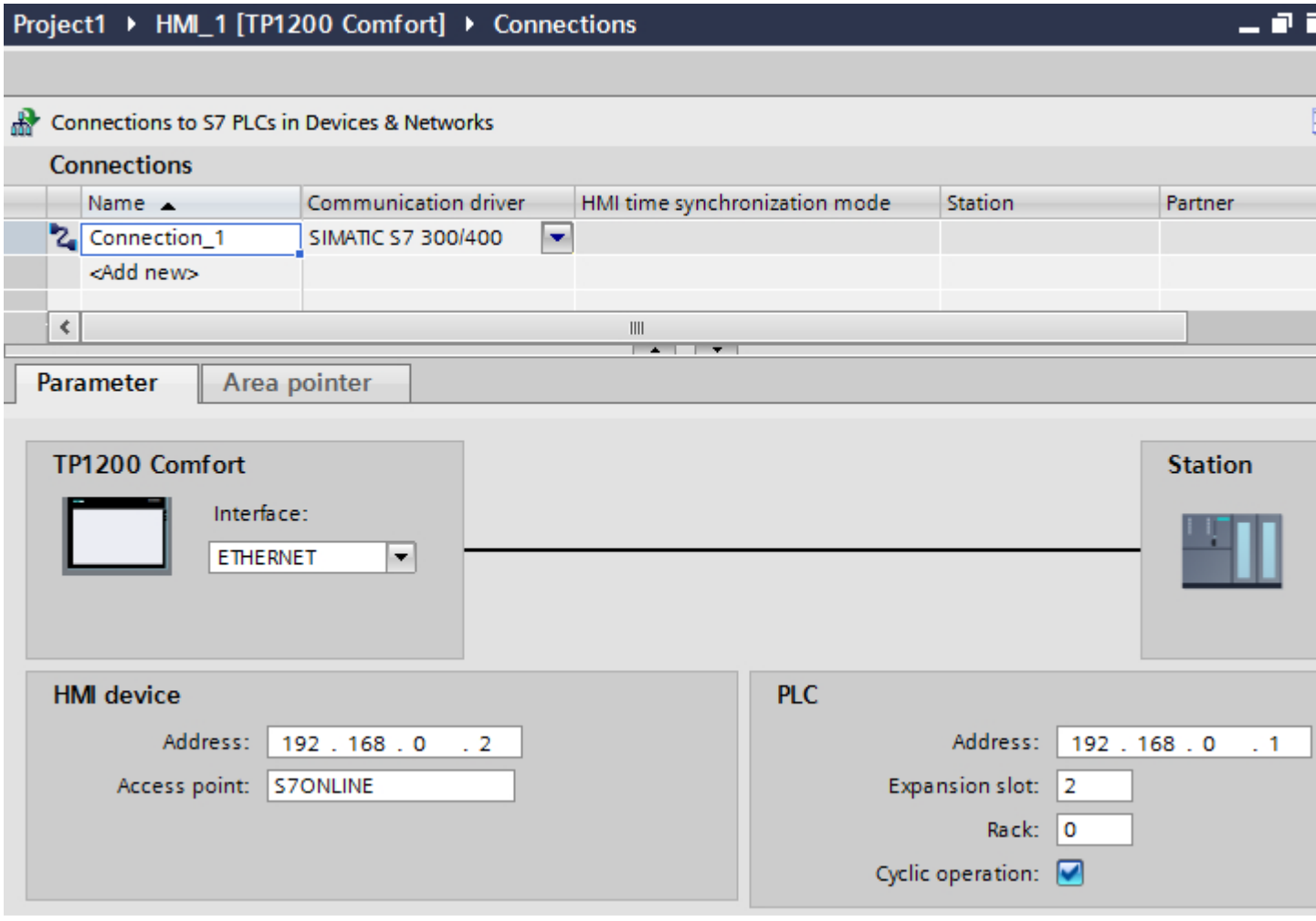
- A project is open.
- An HMI device with a PROFINET interface has been created.

Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.



4. Click the name of the connection.
5. Select a PROFINET interface of the HMI device in the Inspector window under "Parameters > Interface".
6. Set the IP addresses of the communication partners in the Inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

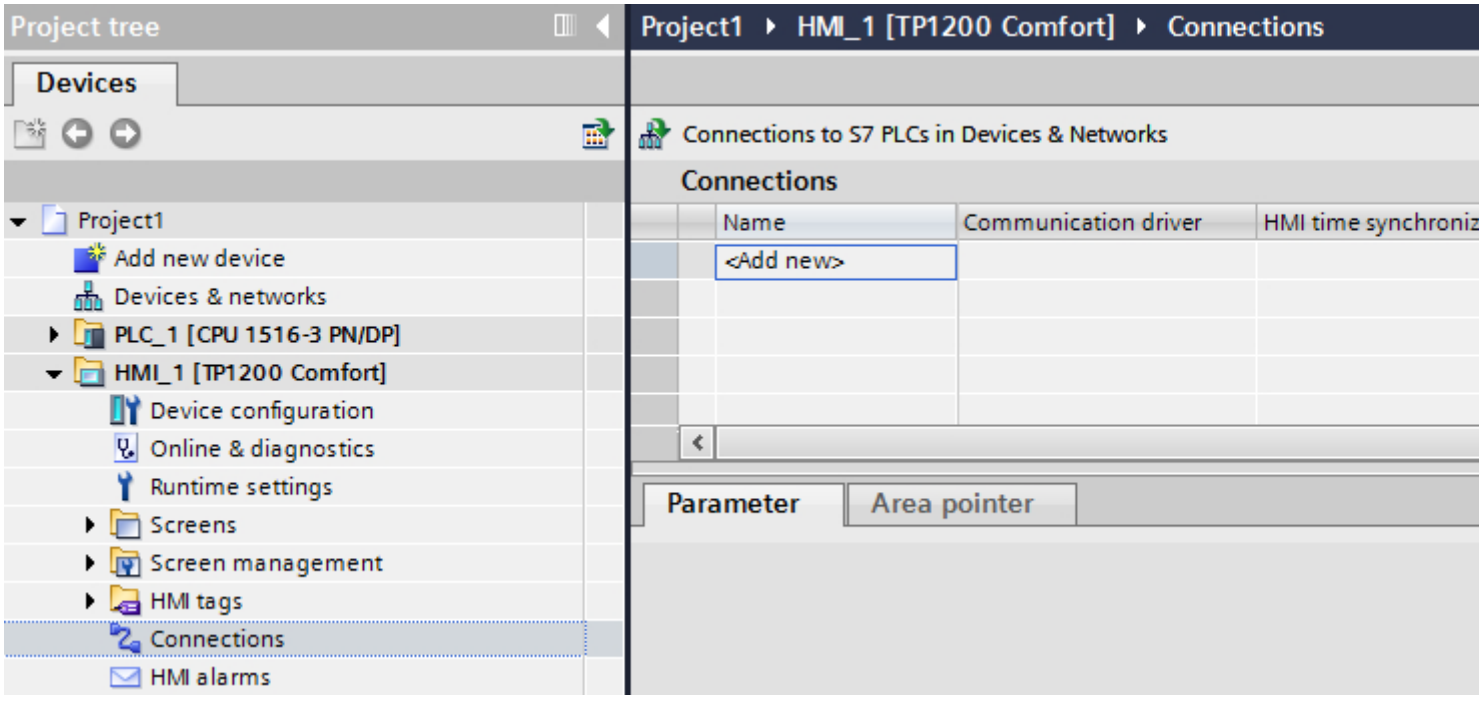
Creating a PROFIBUS connection

Requirements

- A project is open.
- An HMI device with a PROFIBUS interface has been created.

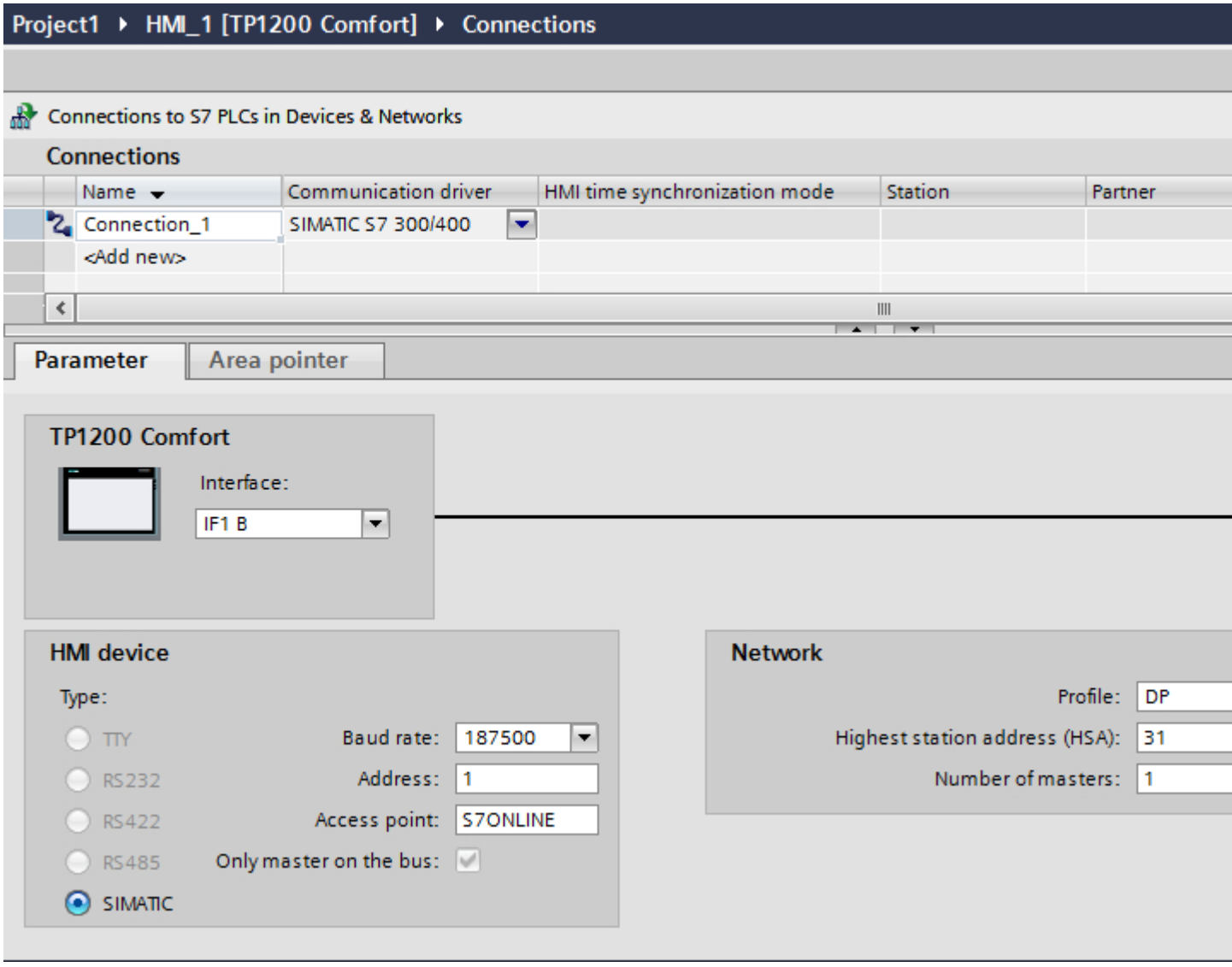
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select the "IF 1 B" interface from "Parameters > interface" in the Inspector window.

6. Select the "DP" profile in the Inspector window under "Parameters > Network".



7. Set the addresses of the communication partners in the inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

Connection parameters

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.
- "Only master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).
  In S7-200, you must set an HMI device as the master.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"
  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the PLC.

  Note

  You only need a password if you have set "Complete protection" at the PLC.

  No connection is set up to the PLC if the "Complete protection" security level is stored on the PLC and you do not enter a password.

Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"

- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

- "Access point"
  Specifies the access point for the PG/PC interface that can be used to reach the communication partner.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the PLC.

Note

You only need a password if you have set "Complete protection" at the PLC.

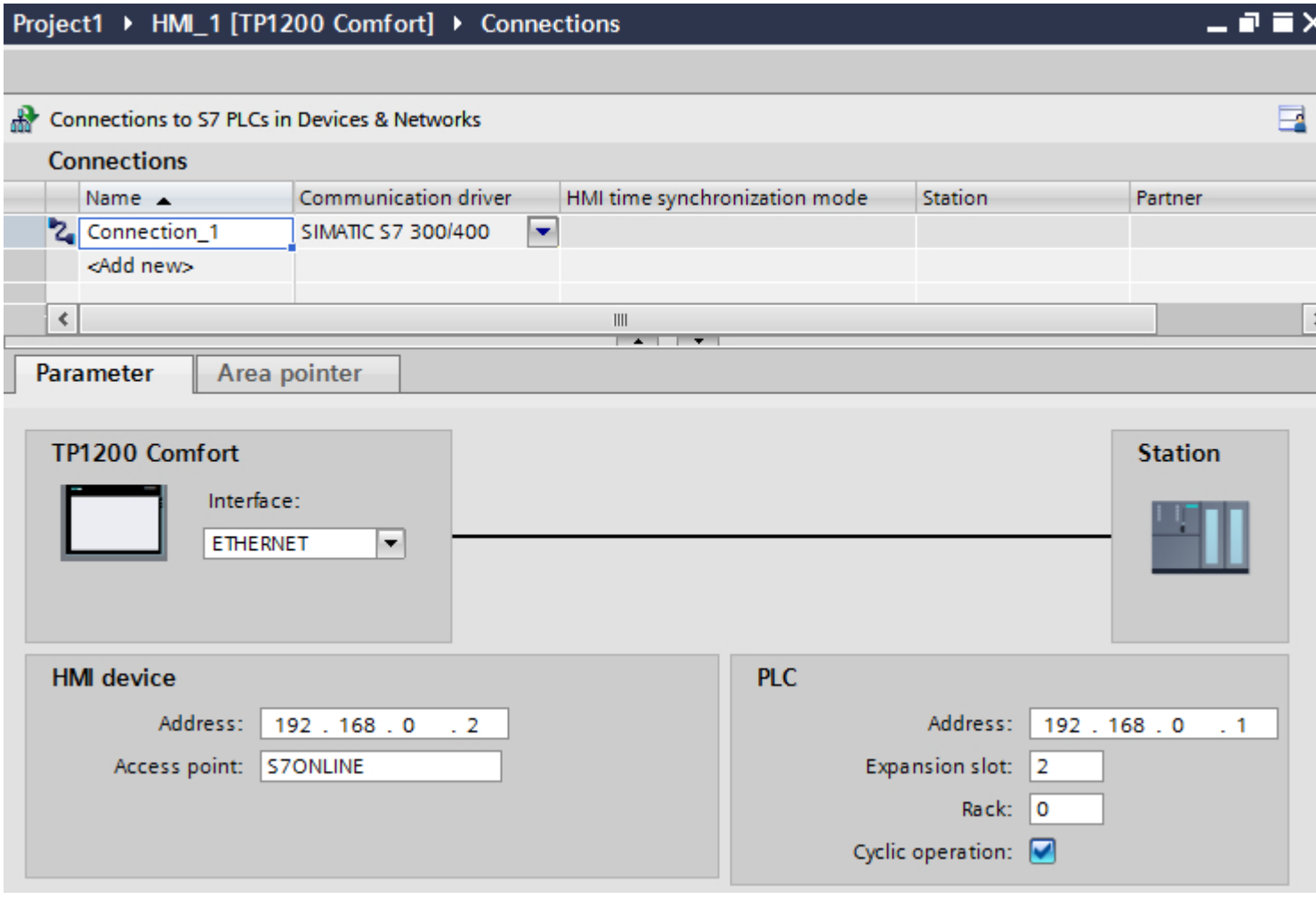
No connection is set up to the PLC if the "Complete protection" security level is stored on the PLC and you do not enter a password.

Parameters for the connection (SIMATIC S7 1500)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



12.11.10.7 Configuring connections in the "Connections" editor

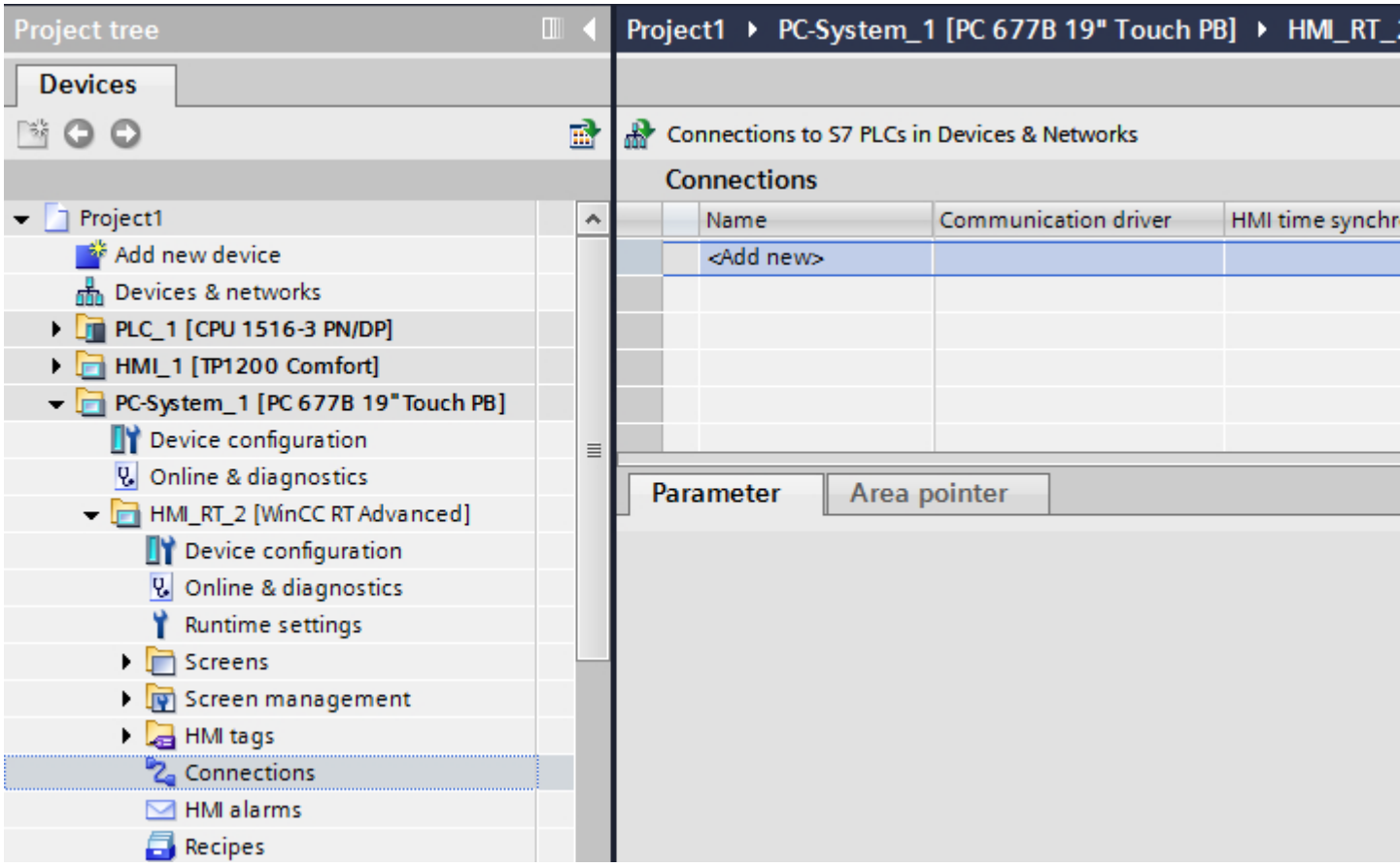
Creating a connection

Requirements

- A project is open.
- WinCC RT Advanced is created.

Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select an interface of the HMI device in the Inspector window under "Parameters > Interface".
6. Set the parameters for the connection in the Inspector window.

Interfaces

Select one of the following interfaces in the Inspector window under "Parameters > WinCC RT Advanced > Interfaces".:

- Industrial Ethernet
- MPI
- PROFIBUS

For additional information on the parameters of the interfaces see:

Auto-Hotspot

Connection parameters

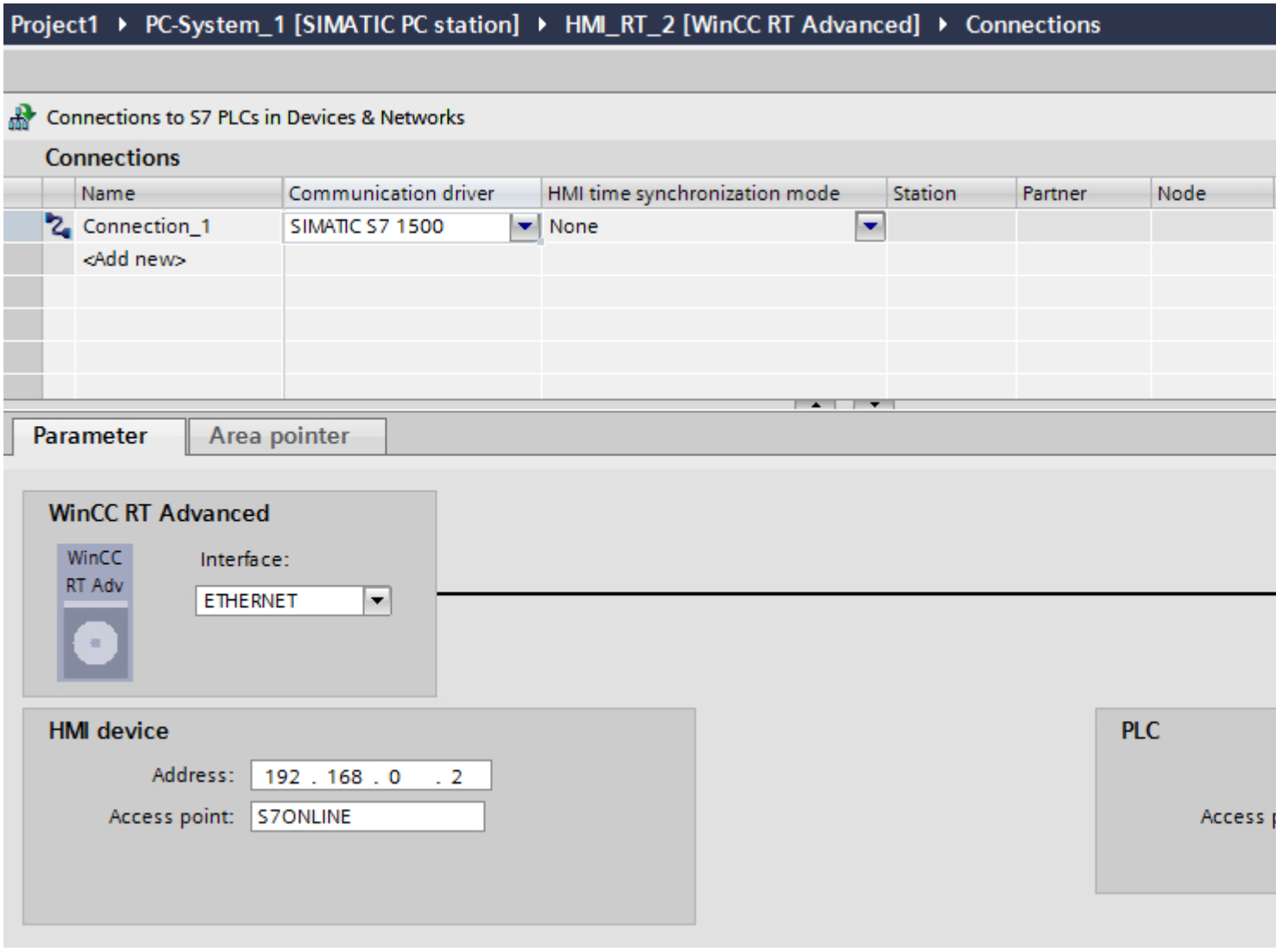
ETHERNET

Introduction

You assign the parameters for the HMI device and the controller in the network under "Parameters".

The parameters described in the following apply to the following interfaces:

- TCP/IP



HMI device

- "Address"
  Enter the IP address of the HMI device under "Address".
- "Access point"
  Specifies the access point for the PG/PC interface that can be used to reach the communication partner.

Controller

- "Address"
  Enter the IP address of the controller under "Address".
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the controller.

Note

You only need a password if you have set the "Complete protection" protection level at the controller.

No connection to the controller is established if the "Complete protection" protection level is stored on the controller and you do not enter a password.

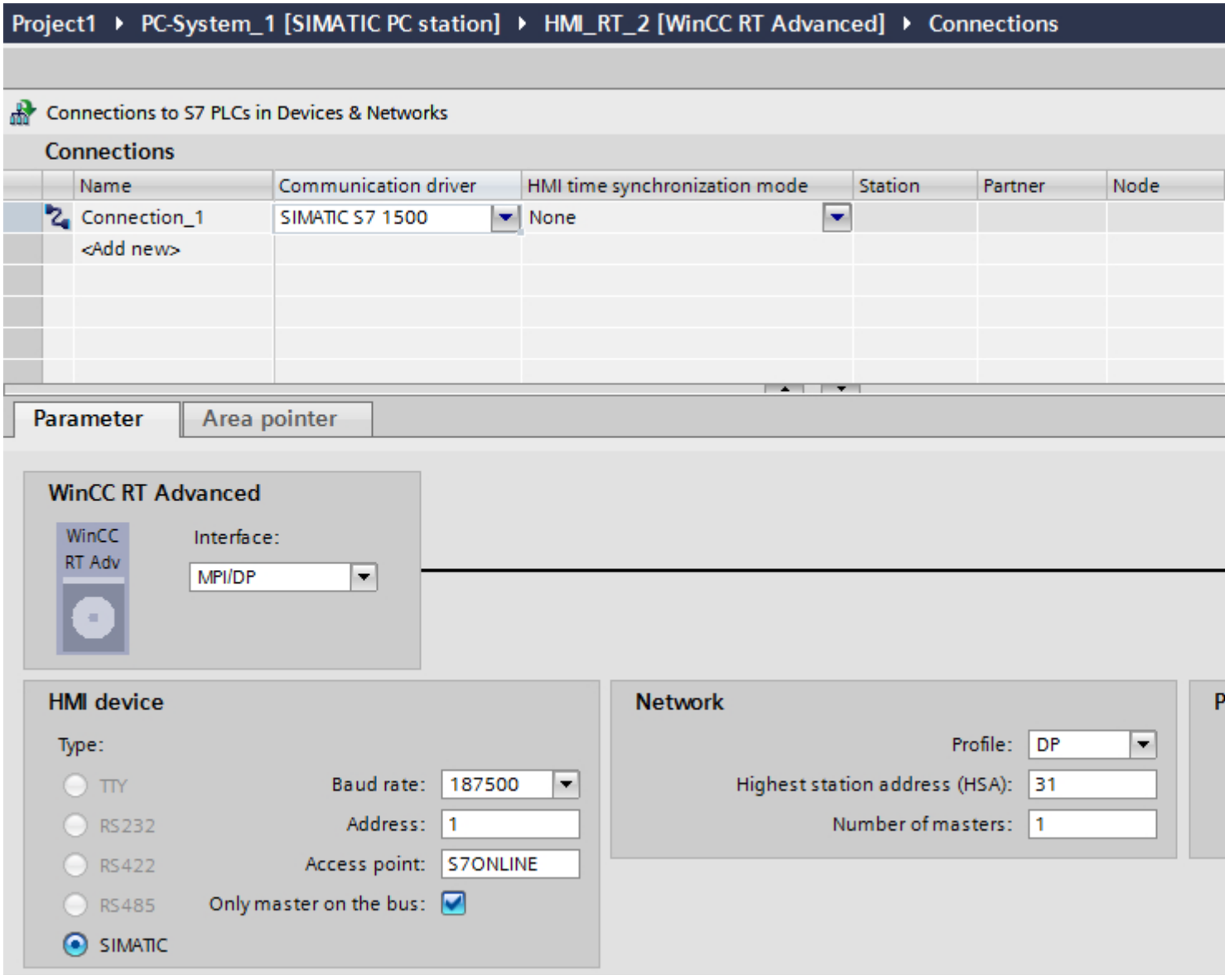
MPI/DP

Introduction

You assign the parameters for the HMI device and the controller in the network under "Parameters".

The parameters described in the following apply to the following interfaces:

- PROFIBUS



HMI device

- "Baud rate"
  Select the baud rate of the HMI device under "Baud rate".
- "Address"
  Enter the address of the HMI device under "Address".
- "Access point"
  Enter the logical device name under "Access point".

Controller

- "Address"
  Enter the address of the controller under "Address".
- "Access password"
  If you have configured the degree of protection "Complete protection" for the controller, enter the password for the controller under "Access password".

12.11.10.8 Configuring time synchronization

Time synchronization

Introduction

To have the same time of day throughout the plant, you can synchronize the time on various plant components using time synchronization. WinCC time synchronization is operated as a master-slave system.

One system component must be a clock for all components of a plant to work with identical time. The component functioning as the clock is referred to as the time master. The components that receive the time are time slaves.

Properties of time synchronization

- The HMI device can define the time as master or can accept the time of the PLC as slave.
- In "master mode", the time is synchronized at each connection setup.
- In "slave mode", the time is synchronized at each connection setup and than at cyclic intervals of 10 minutes.
- The first time synchronization is performed on the HMI device immediately after the start of runtime.
- Time synchronization is only performed on the HMI device during operation of runtime.

Time synchronization restrictions

Approved HMI devices

You can configure the time synchronization between a SIMATIC S7-1200 or SIMATIC S7-1500 and an HMI device with the following HMI devices:

Device	Operating system
Basic Panels	-
TP177 4"	Windows CE 5.0
Multi Panel 177	Windows CE 5.0

Device	Operating system
Multi Panel 277	Windows CE 5.0
Multi Panel 377	Windows CE 5.0
Mobile Panel 277	Windows CE 5.0
Mobile 277 IWLAN V2	Windows CE 5.0
Comfort Panels	Windows CE 6.0
PC systems with WinCC RT Advanced	Microsoft Windows XP Microsoft Windows 7

Configuration restrictions

- If an HMI device has several connections to SIMATIC S7-1200 or SIMATIC S7-1500, you can only configure one connection as "slave".
- If you have enabled time synchronization for the HMI device as "slave", you can no longer use the global area pointer "Date/time PLC".
- An HMI device can only request the time from a PLC with "Complete protection" security type configuration if the correct "Access password" is configured.
  Configure the "Access password" for communication with a PLC with "Complete protection" security type in the "Connections" editor of the HMI device.
  This "Access password" must match the password configured on the PLC. The PLC password is assigned in the PLC properties at: "General > Security"
- Basic Panels can only be configured as "Slave".
- If you use Basic Panels for the configuration, it is not possible to use time synchronization via NTP and the "Date/time PLC" area pointer simultaneously.
- Time synchronization with SIMATIC S7-1200 (V1.0) controllers is not possible.
- Time synchronization between the HMI device TP177 4" and SIMATIC S7-1200 (V4.0) controllers is not possible.
- Time synchronization between the HMI device TP177 4" and SIMATIC S7-1500 is not possible.

Configuring time synchronization for integrated connections

Introduction

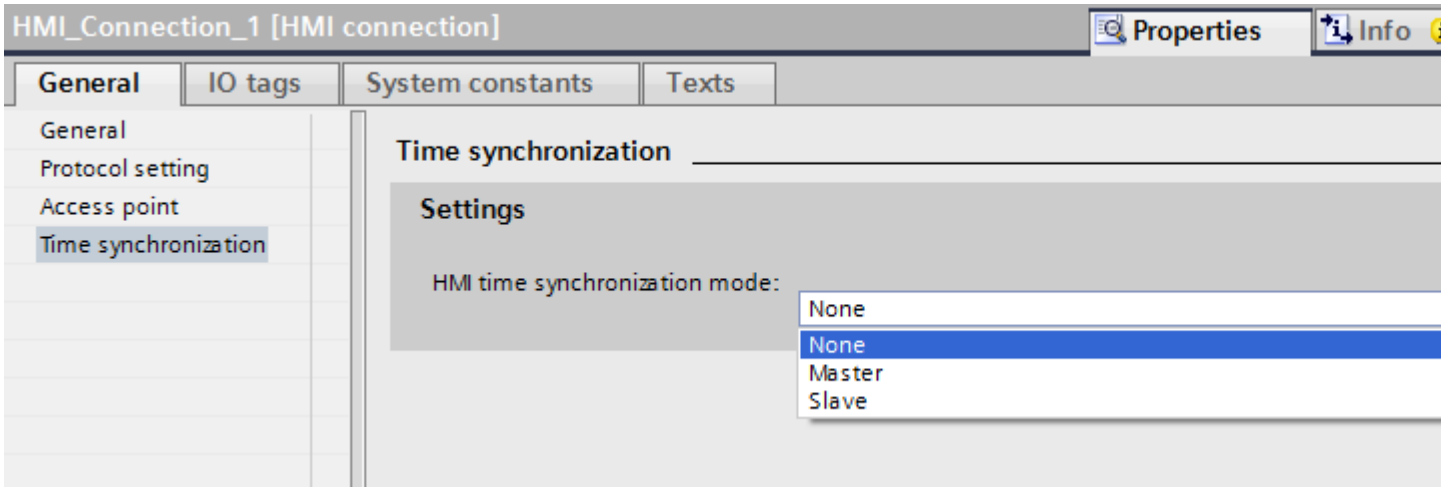
You configure time synchronization for an integrated connection in the "Devices & Networks" editor.

Requirements

- An HMI connection between an HMI device and a SIMATIC S7 1200 or SIMATIC S7 1500 has been configured.
- The HMI device must support the "time synchronization" function.
- The "Devices & Networks" editor is open.

Procedure

1. Click the line of the HMI connection in the "Devices & networks" editor.
2. Select the following in the inspector window under "General > Time synchronization > Settings":
   - None: No time synchronization is used.
   - Master: The HMI device sets the time.
   - Slave: The PLC sets the time.



Configuring time synchronization for non-integrated connections

Introduction

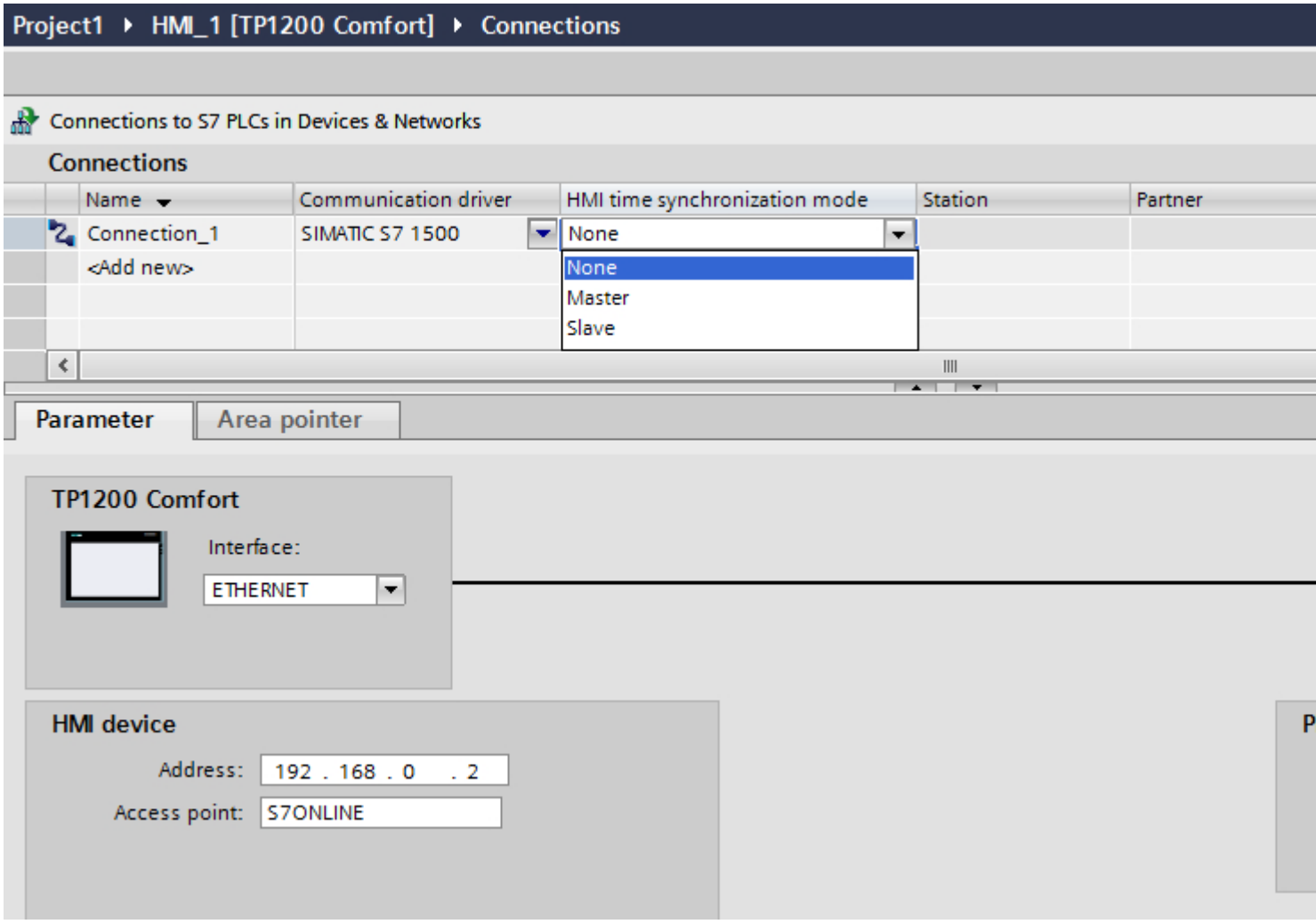
You configure time synchronization for a non-integrated connection in the "Connections" editor.

Requirements

- An HMI device which supports the "time synchronization" function has been created.
- "Connections" editor is open.

Procedure

1. Double-click "<Add>".
2. In the "Communication drivers" column, select the "SIMATIC S7 1500" PLC.
3. Select the following in the "HMI time synchronization mode" column:
   - None: No time synchronization is used.
   - Master: The HMI device sets the time.
   - Slave: The PLC sets the time.



12.11.11 Communicating with SIMATIC S7 200

12.11.11.1 Communication with SIMATIC S7 200

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 200 PLC.

You can configure the following communication channels for the SIMATIC S7 200 PLC:

- PROFINET and Ethernet
- PROFIBUS
- MPI
- PPI

HMI connection for communication

You configure connections between the HMI device and a SIMATIC S7 200 in the "Connections" editor of the HMI device.

12.11.11.2 Creating a connection to SIMATIC S7 200

Introduction

You configure a connection to the SIMATIC S7 200 PLC in the "Connections" editor of the HMI device. The interfaces are named differently depending on the HMI device.

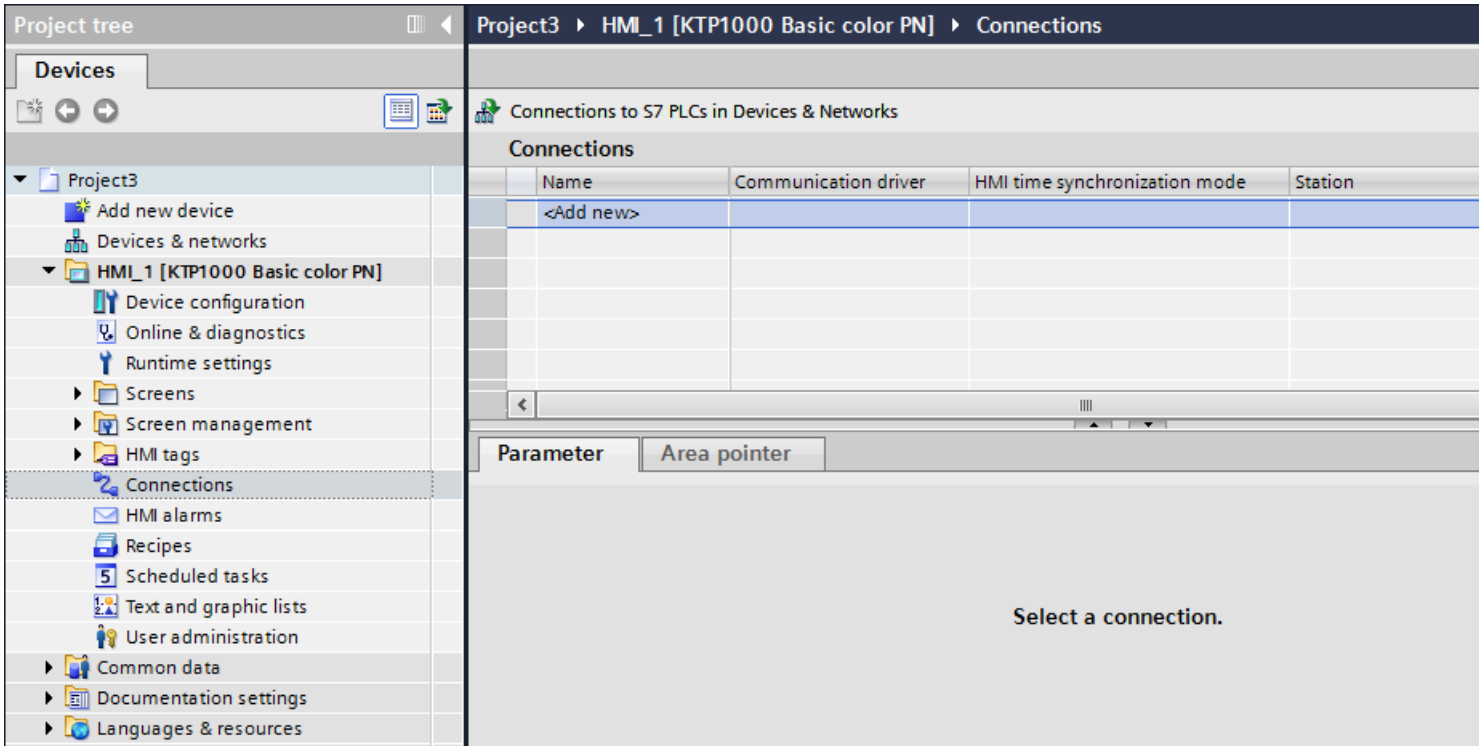
Requirements

- A project is open.
- An HMI device has been created.

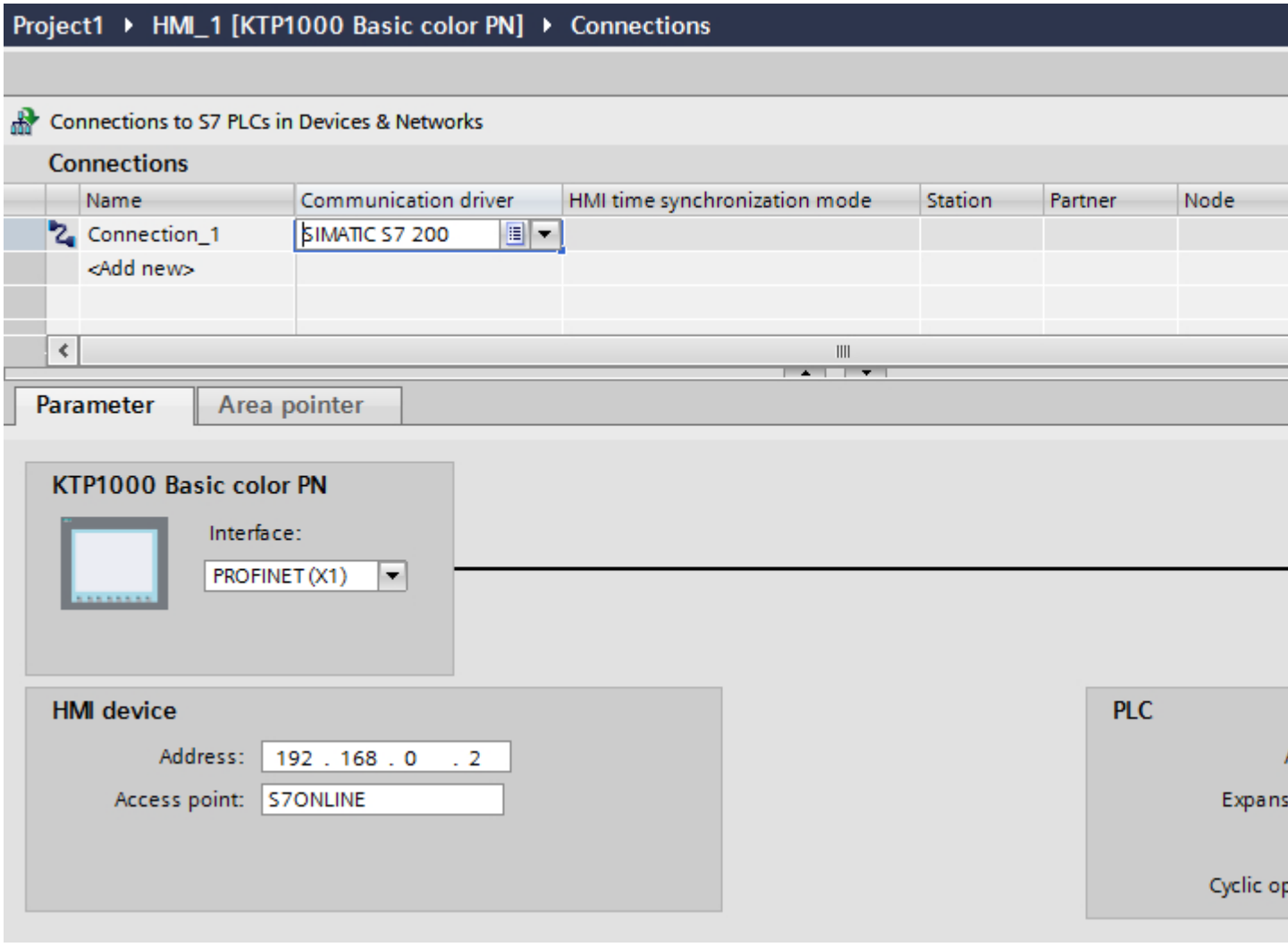
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "SIMATIC S7 200" driver.
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".



See the chapter "Parameters for the connection (Page 6471)" for additional details.

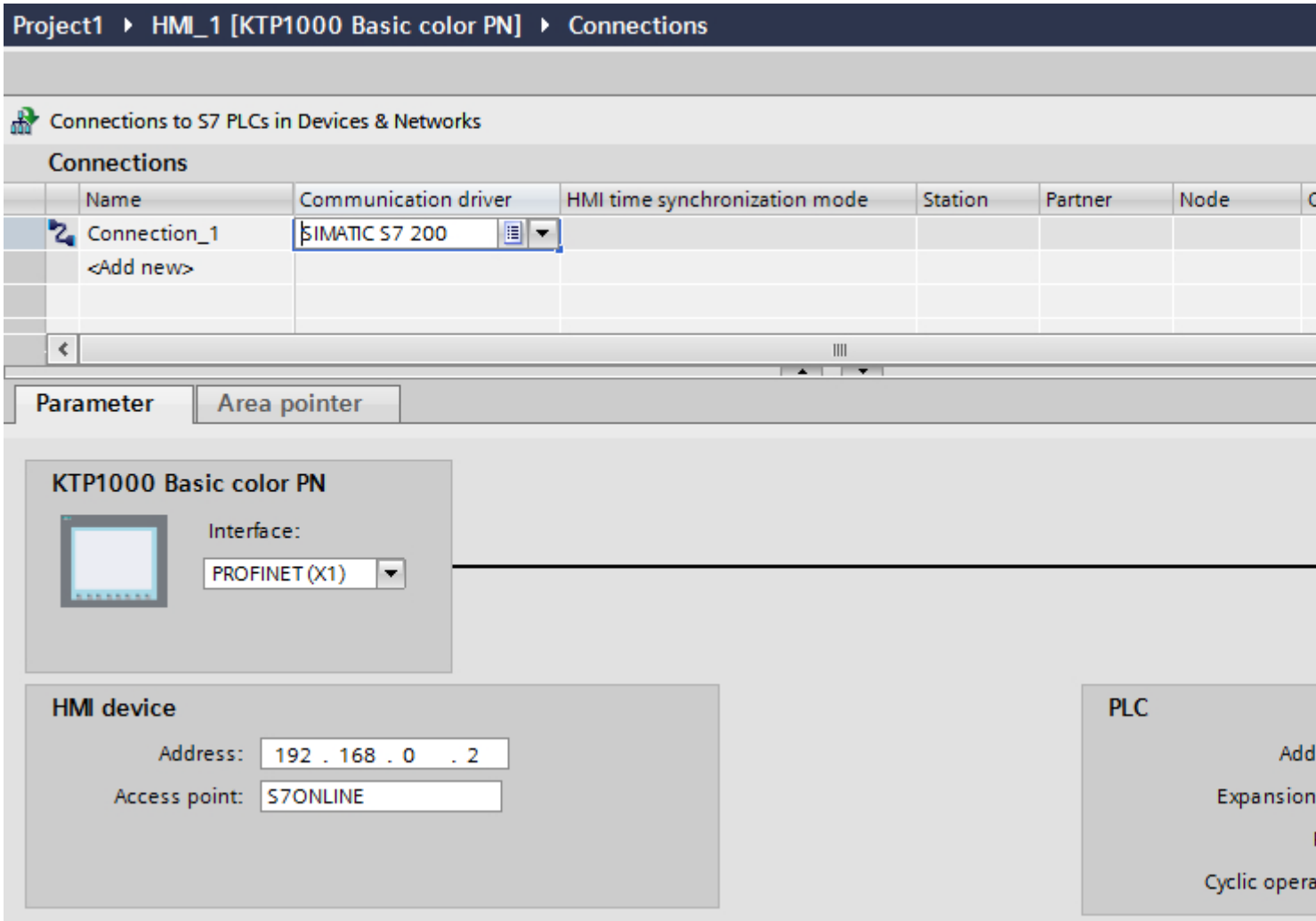
12.11.11.3 Parameters for the connection

Parameters for the connection (SIMATIC S7 200)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"
- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Expansion slot"
  Defines the number of the expansion slot of the CPU to be addressed.

- "Rack"
  Defines the rack number of the CPU to be addressed.
- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance.
  Disable cyclic mode if you are operating several HMI devices in parallel.

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.
- "Sole master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).
  In S7 200, you must set an HMI device as the master.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"
  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200 PLCs.

MPI parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the MPI network.

- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  For "Address", you set the MPI address of the HMI device. The MPI address must be unique throughout the MPI network.
- "Sole master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master). If you have connected only slaves to the HMI device, you must therefore disable the "Sole master on bus" safety feature.
  In S7 200, you must set an HMI device as the master.

Parameters for the network

Under "Network", you set the parameters for the MPI network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "MPI". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual MPI address. The setting must be identical throughout the network.
- "Number of masters"
  This setting is not required for MPI.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the MPI address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Cyclic mode"
  When cyclic mode is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This improves system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200.

PPI parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PP network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  For "Address", you set the PPI address of the HMI device. The PPI address must be unique throughout the PPI network.
- "Access point"
  For "Access point", you set the access point via which the communication partner is reached.
- "Sole master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master). If you have connected only slaves to the HMI device, you must therefore disable the "Sole master on bus" safety feature.
  In S7 200, you must set an HMI device as the master.

Parameters for the network

Under "Network", you set the parameters for the network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "PPI". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual MPI address. The setting must be identical throughout the network.
- "Number of masters"
  Set the number of the master on the network to "1".

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PPI address of the S7 module to which the HMI device is connected.
- "Cyclic operation"
  This parameter is not required for communication via PPI.

Cyclic operation

Handling the "Cyclic operation" selection

If "Cyclic operation" is enabled, the HMI device sends a message frame to the CPU at the beginning of communication indicating that certain tags are required on a recurring basis.

The CPU then always transmits the data at the same cyclic interval. This saves the HMI device from having to output new requests for the data.

If cyclic operation is disabled, the HMI device sends a request whenever information is required.

Additional properties:

- Cyclic operation reduces data transmission load at the HMI device. The PLC resources are used to relieve load on the HMI device.
- The PLC only supports a certain number of cyclic services. The HMI device handles the operation if the PLC cannot provide any further resources for cyclic services.
- The HMI device generates the cycle if the PLC does not support the cyclic mode.
- Screen tags are not integrated in cyclic operation.
- Cyclic mode is only set up at the restart of Runtime.
- The HMI device transfers several jobs to the PLC if cyclic mode is enabled, depending on the PLC.
- The HMI device only transfers one job to the PLC if cyclic mode is disabled.

12.11.11.4 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Configuring area pointers (Page 6044)

See also

Data exchange using area pointers (Page 6039)

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Application

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1st word	Current screen type															
2nd word	Current screen number															
3rd word	Reserved															
4th word	Current field number															
5th word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

Note

Symbolic addressing is not possible if you are using the "Date/Time" area pointer.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the Date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

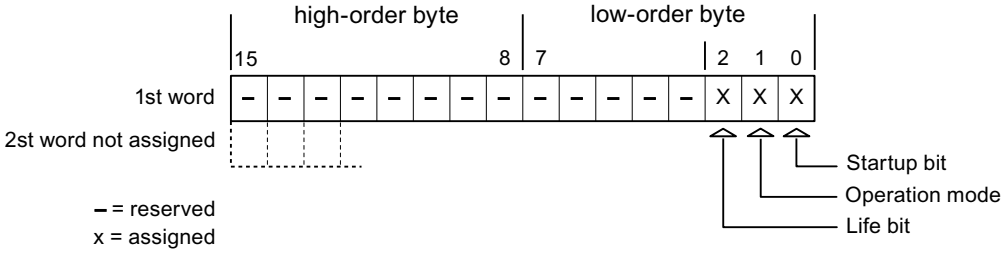
By default, the "Coordination" area pointer has the length of one word and cannot be changed.

Usage

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of the bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

"Project ID" area pointer

Function

When Runtime starts, a check can be carried out as to whether the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program. If there is no concordance, a system event is given on the HMI device and Runtime is stopped.

Use

Note

HMI connections cannot be switched "online".

The HMI connection in which the "Project ID" area pointer is used must be switched "online".

To use this area pointer, set up the following during the configuration:

- Define the version of the configuration. Values between 1 and 255 are possible.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:
  You enter the data address in the editor "Communication > Connections" under "Address".

Connection failure

A connection failure to a device on which the "Project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "Project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the Engineering System.

"Job mailbox" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	**Setting the date (BCD coded)** [3]	

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1	Left byte: - Right byte: Weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	**User logon**	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	**User logoff**	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	**Transfer date/time to PLC**	
	(In the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	**Transfer date/time to PLC**	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	**Update tag**	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in Parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	**Screen selection** [2]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	**Reading data record from PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	**Writing data record to PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1]	Only devices supporting recipes
[2]	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
[3]	The weekday is ignored on HMI device KTP 600 BASIC PN.

"Data record" area pointer

"Data record" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization over the data record

Data records are always transferred directly. That is, the tag values are read from an address or written to an address configured for this tag directly, without redirecting the values by means of interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system event.

Sequence of a transfer started by the operator in the recipe view

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data record and sets the data record number to 0.	Abort with system event.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data mailbox.	Abort with system event.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized in the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by the HMI device or by the PLC.

The two job mailboxes No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	

Step	Action	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC using job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the job from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of faults

The section below shows possible causes of errors which lead to a data record transfer being terminated with errors:

- Tag address not set up on the PLC
- Overwriting data records not possible
- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system events
- Triggered by function
  Output of system events
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data mailbox.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example, when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then process, edit, or save these values, for example.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the Inspector window the "Coordinated transfer of data records" option under "General > Synchronization > Settings".

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data record free
2	0000 0010	Transferring
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Displaying tags with Runtime Advanced and Panels (Page 4270)

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

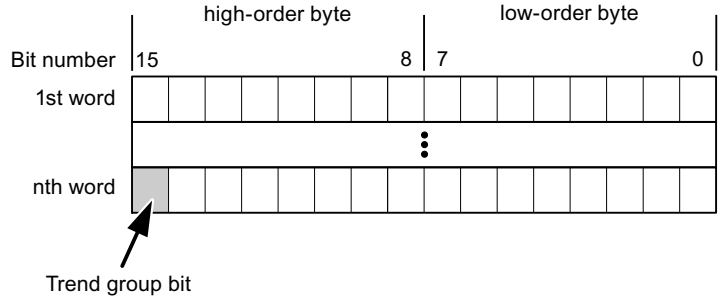
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Permitted data types for trends

For SIMATIC S7

You assign one bit to each trend during configuration. Tags and array tags of the "Word" or "Int" data type are permitted.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with Alarms (Page 4293)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

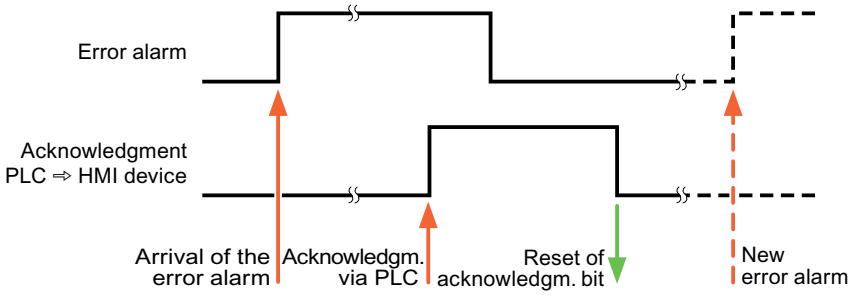
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment

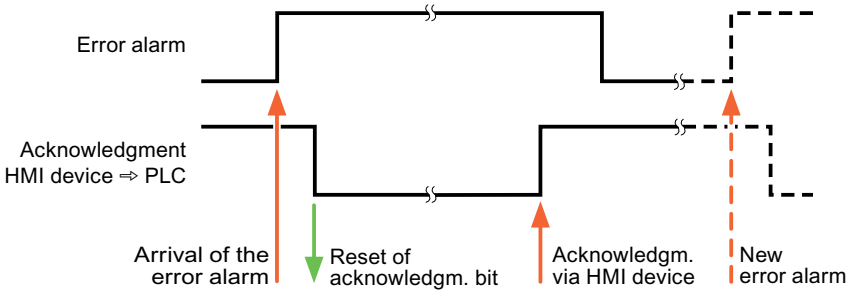
tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



LED mapping

Function

Keyboard devices have LEDs in the function keys. These LEDs can be activated from the PLC. Thus, it is possible, for example, to signal to the user which key he should press in a given situation by lighting up an LED.

Note

The LED function cannot be configured for Basic Panels.

Requirements

In order to activate an LED, an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration.

LED assignment

The assignment of the individual LEDs to the bits in the LED tags is specified when the function keys are configured. In this process, the "LED tag" and the assigned "Bit" are specified for each function key in the "General" group of the properties window.

The "Bit" bit number designates the first of two consecutive bits that control the following LED statuses.

Bit n+ 1	Bit n	LED function	
		All Mobile Panels, all Operator Panels, all Multi Panels, all Comfort Panels	Panel PCs
0	0	Off	Off
0	1	Rapid flashing	Flashing
1	0	Slow flashing	Flashing
1	1	On permanently	On permanently

12.11.11.5 Performance features of communication

Device dependency S7 200

Communication with the SIMATIC S7-200 controller

If you use devices from an earlier version of the TIA Portal with TIA Portal V13, it may not be possible to configure integrated connections to certain HMI devices.

Basic Panels V11.0

HMI devices	SIMATIC S7-200
KP300 Basic	Yes
KP400 Basic	Yes
KTP400 Basic PN	Yes
KTP600 Basic DP	Yes
KTP600 Basic PN	Yes
KTP1000 Basic DP	Yes
KTP1000 Basic PN	Yes
TP1500 Basic PN	Yes

Basic Panels V12.0

HMI devices	SIMATIC S7-200
KP300 Basic	Yes
KP400 Basic	Yes
KTP400 Basic PN	Yes
KTP600 Basic DP	Yes
KTP600 Basic PN	Yes

HMI devices	SIMATIC S7-200
KTP1000 Basic DP	Yes
KTP1000 Basic PN	Yes
TP1500 Basic PN	Yes

Basic Panels V13.0

HMI devices	SIMATIC S7-200
KTP400 Basic PN	Yes
KTP700 Basic PN	Yes
KTP700 Basic DP	Yes
KTP900 Basic PN	Yes
KTP1200 Basic PN	Yes
KTP1200 Basic DP	Yes

Panels V11.0

HMI devices	SIMATIC S7-200
OP 73	Yes
OP 77A	Yes
OP 77B	Yes
TP 177A	Yes
TP 177A Portrait	Yes
TP 177B 4"	Yes
TP 177B 6" mono	Yes
TP 177B 6"	Yes
OP 177B 6" mono	Yes
OP 177B 6"	Yes
TP 277 6"	Yes
OP 277 6"	Yes

Multi Panels V11.0

HMI devices	SIMATIC S7-200
MP 177 6" Touch	Yes
MP 277 8" Key	Yes
MP 277 10" Key	Yes
MP 277 10" Touch	Yes
MP 377 12" Key	Yes

HMI devices	SIMATIC S7-200
MP 377 12" Touch	Yes
MP 377 15" Touch	Yes
MP 377 19" Touch	Yes

Mobile Panels V11.0

HMI devices	SIMATIC S7-200
Mobile Panel 177 6" DP	Yes
Mobile Panel 177 6" PN	Yes
Mobile Panel 277 8"	Yes
Mobile Panel 277 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	Yes
Mobile Panel 277 10"	Yes

Mobile Panels V12.0

HMI devices	SIMATIC S7-200
Mobile Panel 177 6" DP	Yes
Mobile Panel 177 6" PN	Yes
Mobile Panel 277 8"	Yes
Mobile Panel 277 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	Yes
Mobile Panel 277 10"	Yes

Comfort Panels V11.0

HMI devices	SIMATIC S7-200
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes

HMI devices	SIMATIC S7-200
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V12.0

HMI devices	SIMATIC S7-200
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V13.0

HMI devices	SIMATIC S7-200
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes

HMI devices	SIMATIC S7-200
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Runtime V11.0

HMI devices	SIMATIC S7-200
WinCC RT Advanced	Yes

Runtime V12.0

HMI devices	SIMATIC S7-200
WinCC RT Advanced	Yes

Runtime V13.0

HMI devices	SIMATIC S7-200
WinCC RT Advanced	Yes

Permitted data types for SIMATIC S7 200

Permitted data types for connections with SIMATIC S7 200

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
Bool	1 bit
Byte	1 byte
Char	1 byte
Word	2 bytes
Int	2 bytes
DWord	4 bytes

Data type	Length
DInt	4 bytes
Real	4 bytes
StringChar	--
Timer	2 bytes
Array	--

Note

Disconnection with a PPI network

If you are using arrays in the configuration, an array size of approximately 1000 bytes may cause an interruption of the connection.

Use smaller arrays in your configuration.

12.11.12 Communicating with SIMATIC LOGO!

12.11.12.1 Communication with SIMATIC LOGO!

Introduction

This section describes the communication between an HMI device and the SIMATIC LOGO! controller.

You can configure the following communication channels for the SIMATIC LOGO! controller:

- PROFINET
- Ethernet

HMI connection for communication

You configure connections between the HMI device and SIMATIC LOGO! in the "Connections" editor of the HMI device.

Data exchange

Data exchange with the SIMATIC LOGO! control system is possible by means of tags.

Data cannot be exchanged using area pointers.

12.11.12.2 Creating a connection to SIMATIC LOGO!

Introduction

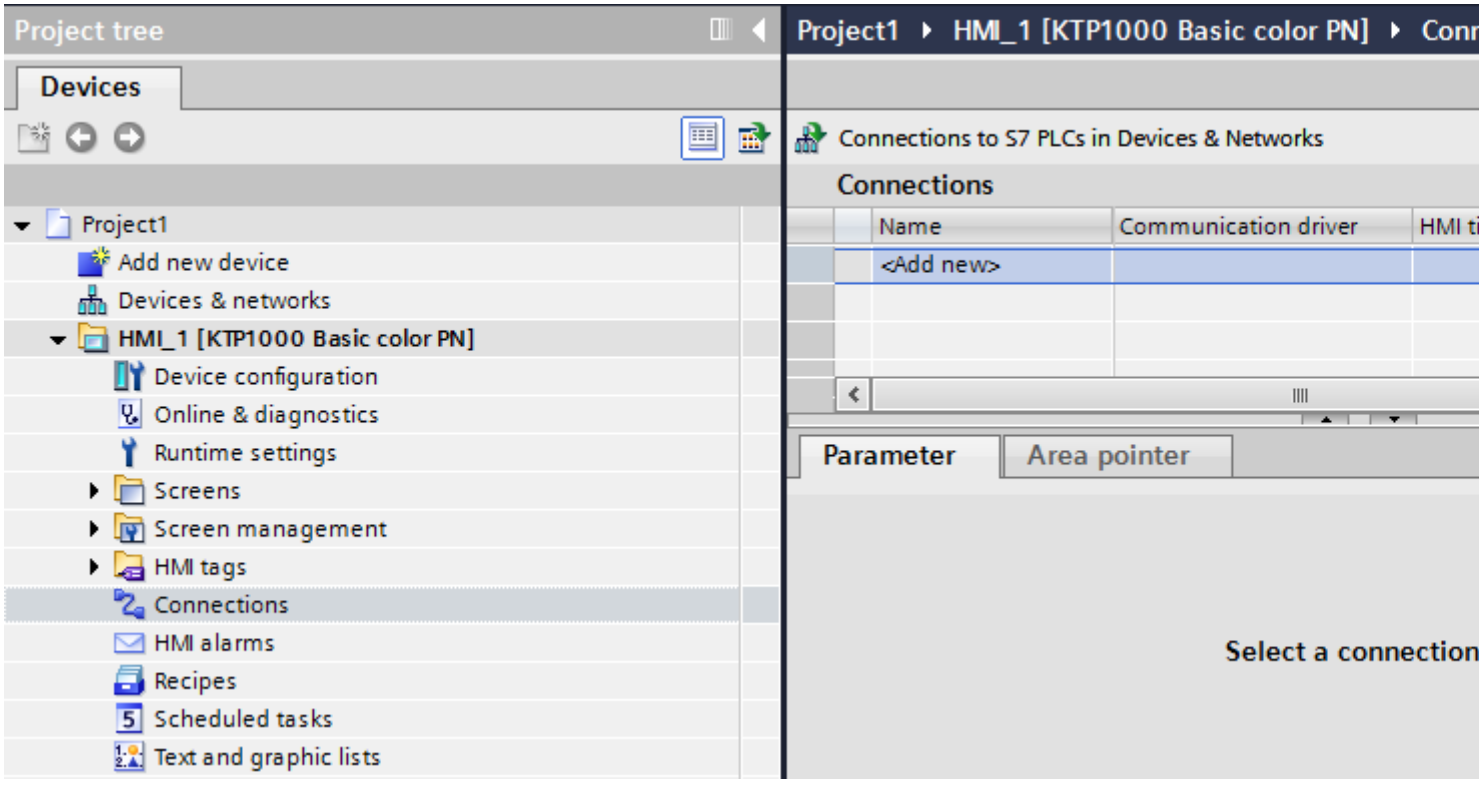
You configure a connection to the SIMATIC LOGO! controller in the "Connections" editor of the HMI device. The interfaces are named differently depending on the HMI device.

Requirements

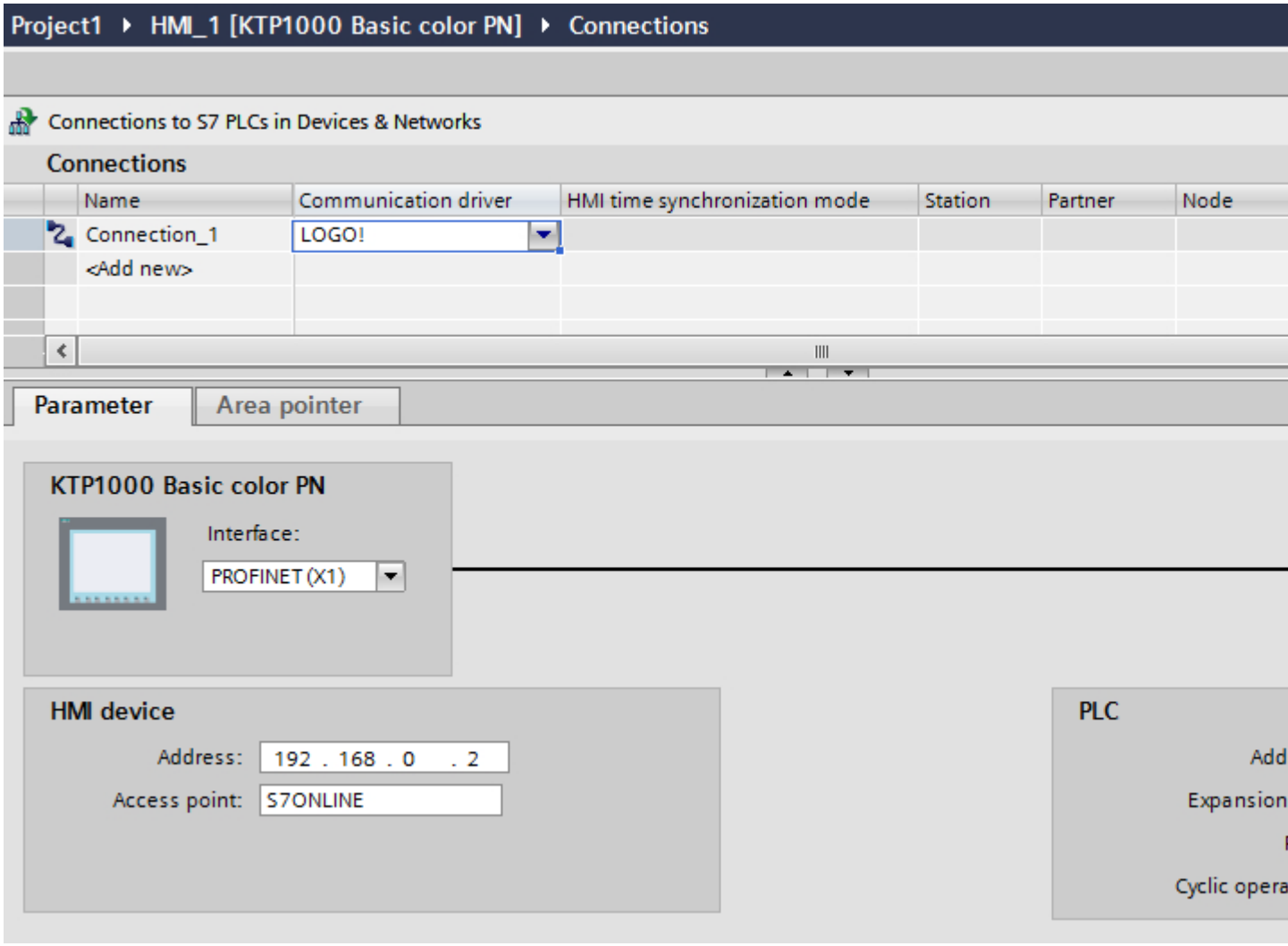
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "SIMATIC LOGO!" driver.
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".



See the chapter "Connection parameters (Page 6505)" for additional details.

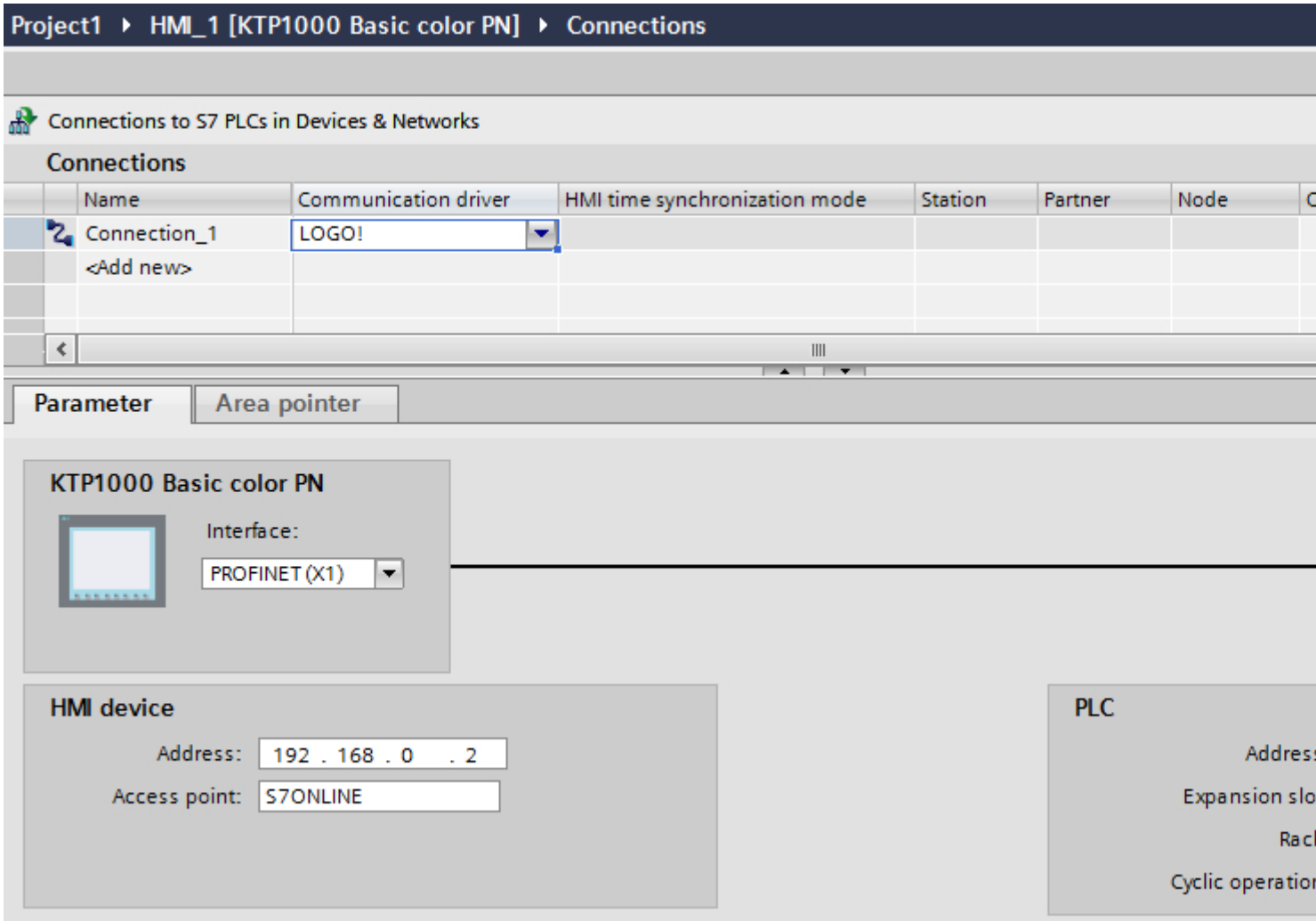
12.11.12.3 Connection parameters

Connection parameters

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"
- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Expansion slot"
  Defines the number of the expansion slot of the CPU to be addressed.

- "Rack"
  Defines the rack number of the CPU to be addressed.
- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance.
  Disable cyclic mode if you are operating several HMI devices in parallel.

Cyclic operation

Handling the "Cyclic operation" selection

If "Cyclic operation" is enabled, the HMI device sends a message frame to the CPU at the beginning of communication indicating that certain tags are required on a recurring basis.

The CPU then always transmits the data at the same cyclic interval. This saves the HMI device from having to output new requests for the data.

If cyclic operation is disabled, the HMI device sends a request whenever information is required.

Additional properties:

- Cyclic operation reduces data transmission load at the HMI device. The PLC resources are used to relieve load on the HMI device.
- The PLC only supports a certain number of cyclic services. The HMI device handles the operation if the PLC cannot provide any further resources for cyclic services.
- The HMI device generates the cycle if the PLC does not support the cyclic mode.
- Screen tags are not integrated in cyclic operation.
- Cyclic mode is only set up at the restart of Runtime.
- The HMI device transfers several jobs to the PLC if cyclic mode is enabled, depending on the PLC.
- The HMI device only transfers one job to the PLC if cyclic mode is disabled.

12.11.12.4 Data exchange

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Auto-Hotspot

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

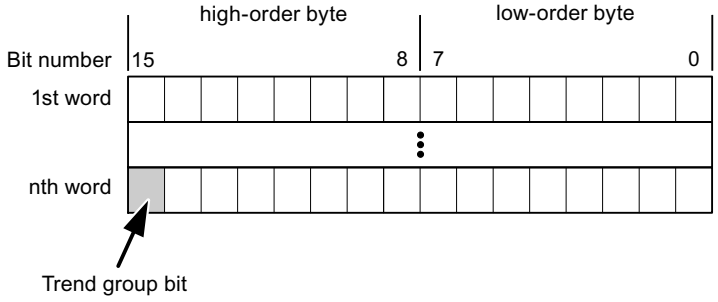
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Permitted data types for trends

For SIMATIC S7

You assign one bit to each trend during configuration. Tags and array tags of the "Word" or "Int" data type are permitted.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Auto-Hotspot

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

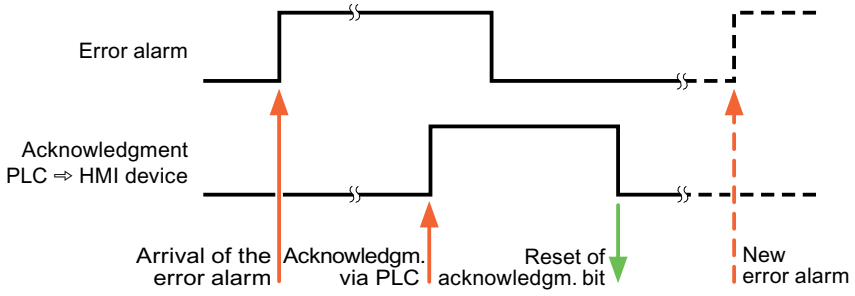
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

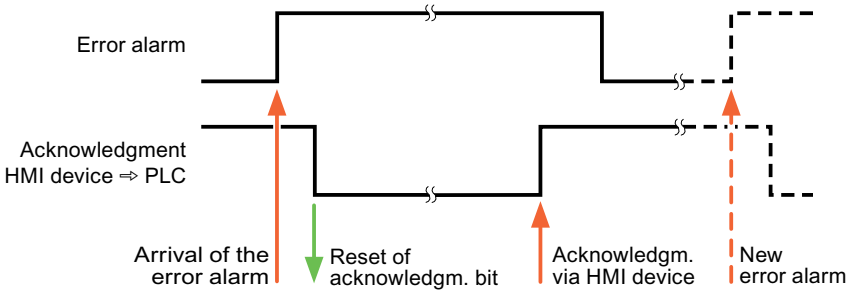
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



12.11.12.5 Performance features of communication

SIMATIC LOGO! device dependency

SIMATIC LOGO! device dependency

If you use devices from an earlier version of the TIA Portal with TIA Portal V13, it may not be possible to configure integrated connections to certain HMI devices.

Basic Panels V11.0

HMI devices	SIMATIC LOGO!
KP300 Basic	No
KP400 Basic	No
KTP400 Basic PN	No
KTP600 Basic DP	No
KTP600 Basic PN	No
KTP1000 Basic DP	No
KTP1000 Basic PN	No
TP1500 Basic PN	No

Basic Panels V12.0

HMI devices	SIMATIC LOGO!
KP300 Basic	Yes
KP400 Basic	Yes
KTP400 Basic PN	Yes
KTP600 Basic DP	Yes
KTP600 Basic PN	Yes
KTP1000 Basic DP	No
KTP1000 Basic PN	No
TP1500 Basic PN	Yes

Basic Panels V13.0

HMI devices	SIMATIC LOGO!
KTP400 Basic PN	Yes
KTP700 Basic PN	Yes
KTP700 Basic DP	Yes
KTP900 Basic PN	Yes
KTP1200 Basic PN	No
KTP1200 Basic DP	No

Panels V11.0

HMI devices	SIMATIC LOGO!
OP 73	No
OP 77A	No

HMI devices	SIMATIC LOGO!
OP 77B	No
TP 177A	No
TP 177A Portrait	No
TP 177B 4"	No
TP 177B 6" mono	No
TP 177B 6"	No
OP 177B 6" mono	No
OP 177B 6"	No
TP 277 6"	No
OP 277 6"	No

Multi Panels V11.0

HMI devices	SIMATIC LOGO!
MP 177 6" Touch	No
MP 277 8" Key	No
MP 277 10" Key	No
MP 277 10" Touch	No
MP 377 12" Key	No
MP 377 12" Touch	No
MP 377 15" Touch	No
MP 377 19" Touch	No

Mobile Panels V11.0

HMI devices	SIMATIC LOGO!
Mobile Panel 177 6" DP	No
Mobile Panel 177 6" PN	No
Mobile Panel 277 8"	No
Mobile Panel 277 8" IWLAN V2	No
Mobile Panel 277F 8" IWLAN V2	No
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	No
Mobile Panel 277 10"	No

Mobile Panels V12.0

HMI devices	SIMATIC LOGO!
Mobile Panel 177 6" DP	Yes
Mobile Panel 177 6" PN	Yes
Mobile Panel 277 8"	Yes
Mobile Panel 277 8" IWLAN V2	Yes

HMI devices	SIMATIC LOGO!
Mobile Panel 277F 8" IWLAN V2	Yes
Mobile Panel 277F 8" IWLAN V2 (RFID tag)	Yes
Mobile Panel 277 10"	Yes

Comfort Panels V11.0

HMI devices	SIMATIC LOGO!
KP400 Comfort	No
KTP400 Comfort	No
KTP400 Comfort Portrait	No
KP700 Comfort	No
TP700 Comfort	No
TP700 Comfort Portrait	No
KP900 Comfort	No
TP900 Comfort	No
TP900 Comfort Portrait	No
KP1200 Comfort	No
TP1200 Comfort	No
TP1200 Comfort Portrait	No
KP1500 Comfort	No
TP1500 Comfort	No
TP1500 Comfort Portrait	No
TP1900 Comfort	No
TP1900 Comfort Portrait	No
TP2200 Comfort	No
TP2200 Comfort Portrait	No

Comfort Panels V12.0

HMI devices	SIMATIC LOGO!
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes

HMI devices	SIMATIC LOGO!
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Comfort Panels V13.0

HMI devices	SIMATIC LOGO!
KP400 Comfort	Yes
KTP400 Comfort	Yes
KTP400 Comfort Portrait	Yes
KP700 Comfort	Yes
TP700 Comfort	Yes
TP700 Comfort Portrait	Yes
KP900 Comfort	Yes
TP900 Comfort	Yes
TP900 Comfort Portrait	Yes
KP1200 Comfort	Yes
TP1200 Comfort	Yes
TP1200 Comfort Portrait	Yes
KP1500 Comfort	Yes
TP1500 Comfort	Yes
TP1500 Comfort Portrait	Yes
TP1900 Comfort	Yes
TP1900 Comfort Portrait	Yes
TP2200 Comfort	Yes
TP2200 Comfort Portrait	Yes

Runtime V11.0

HMI devices	SIMATIC LOGO!
WinCC RT Advanced	No

Runtime V12.0

HMI devices	SIMATIC LOGO!
WinCC RT Advanced	Yes

Runtime V13.0

HMI devices	SIMATIC LOGO!
WinCC RT Advanced	Yes

Valid data types for SIMATIC LOGO!

Valid data types for connections with SIMATIC LOGO!

Data type	Length
Bool	1 bit
Byte	1 byte
Int	2 bytes
DInt	4 bytes
Word	2 bytes
DWord	4 bytes
Array	--

12.11.13 Configuring direct keys

12.11.13.1 Direct keys

Introduction

In addition to their standard use, the F, K and S keys of an HMI device with key operation can be used as direct keys.

With HMI devices with touch operation, you configure the system function "Direct key" to a button

You can configure the following direct keys:

- PROFINET connection: PROFINET IO direct keys
- PROFIBUS connection: PROFIBUS DP direct keys

Operating mode of HMI devices with direct keys

Before accessing the PLC from the HMI device using the direct keys you have to change the operating mode of the HMI device.

12.11.13.2 Changing the operating mode of the HMI device

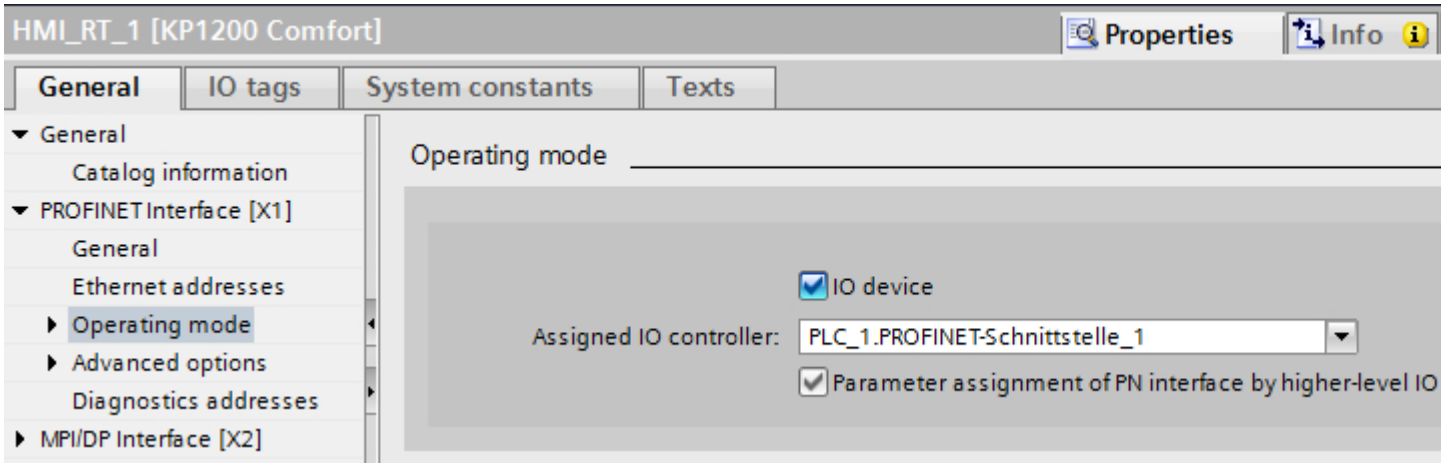
Changing the operating mode for a PROFINET connection

Requirements

The HMI device is networked with a PLC via PROFINET.

Procedure

1. Click the PROFINET interface of the HMI device in the "Devices & Networks" editor.
2. Click "Operating mode" under "Properties > General" in the Inspector window.
3. Select the function "IO device" in the "Operating mode" area.
4. Select the PLC which is networked with the HMI device under "Assigned IO controller".



Changing the operating mode for a PROFIBUS connection

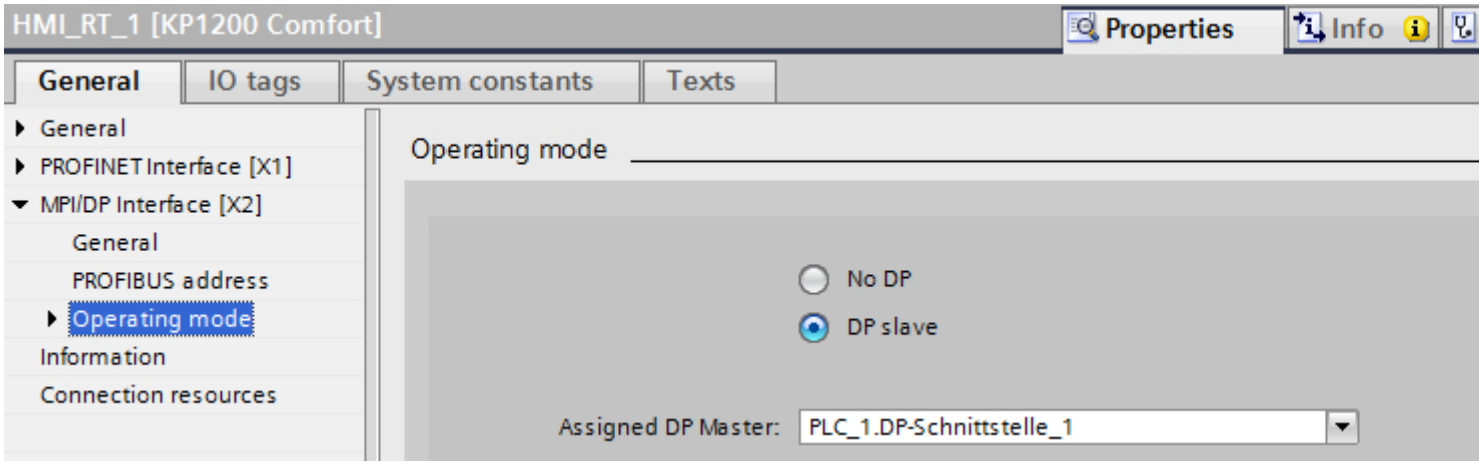
Requirements

The HMI device is networked with a PLC via PROFIBUS.

Procedure

1. Click the PROFIBUS interface of the HMI device in the "Devices & Networks" editor.
2. Click "Operating mode" under "Properties > General" in the Inspector window.

3. Select the function "DP slave" in the "Operating mode" area.
4. Select the PLC which is networked with the HMI device under "Assigned DP master".



12.11.13.3 Configuring direct keys

Configuring direct keys for an HMI device with touch operation

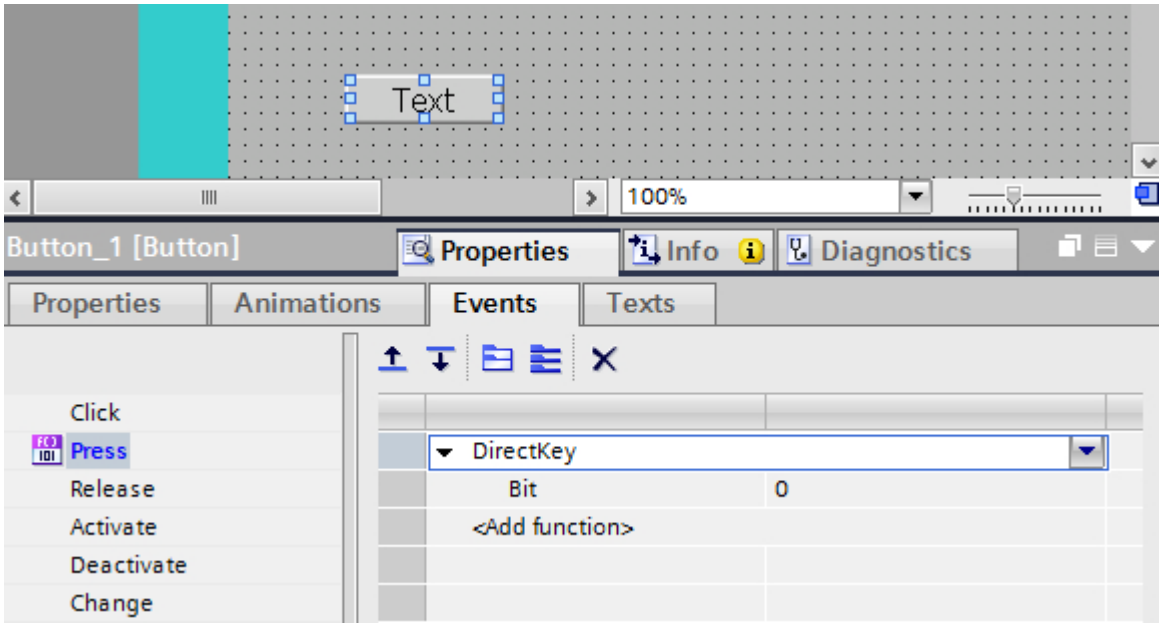
Requirements

- An HMI device with key operation has been created.
- You have created an LED tag.

Procedure

1. Create a new screen.
2. Drag-and-drop a button from the "Tools" task card into the screen.
3. Click the button.
4. In the Inspector window, click "Properties > Events > Press".
5. Click "<Add function>".

6. Select the "Direct key" system function.



7. Under "Bit" enter the correct bit number.
   The correct bit number depends on the HMI device and the input and output assignments on the HMI device.

Assignment of inputs and outputs

The exact assignment of inputs and outputs can be found under:

- PROFINET IO direct keys: Assignment of inputs and outputs (Page 6525)
- PROFIBUS DP direct keys: Assignment of inputs and outputs (Page 6543)

12.11.13.4 PROFINET IO direct keys

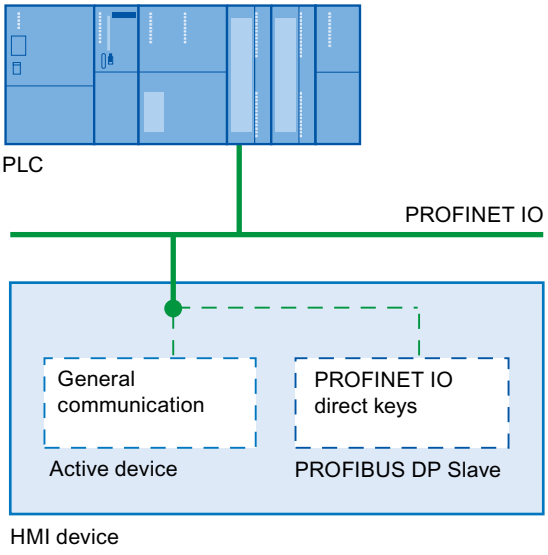
PROFINET IO direct keys

PROFINET IO direct keys

You configure the HMI device created in WinCC as an active communication partner in the automation network.

For the PROFINET IO direct keys, the HMI device can also be configured as a slave in the PROFINET IO network.

The following figure shows the basic configuration based on an automation network with an HMI device and a PLC.



Mode of operation of the PROFINET IO direct keys

The cycle time of the Ethernet bus can be set between 8 ms and 512 ms.

Thus, the reaction time of the PROFINET IO direct keys can also be determined. For a typical configuration with a cycle time of 64 ms, the response time of the PROFINET IO direct keys is < 100 ms.

A response time of < 100 ms to the CPU is usually ensured when using the PROFINET IO direct keys. This time can be exceeded significantly in the following cases:

- Complex functions run in the background, e.g. transmitting recipes, printing reports.
- At the same time, multiple connections can be maintained to the CPU.

HMI devices for the configuration of PROFINET IO direct keys

HMI devices

You can configure PROFINET IO direct keys with the following HMI devices:

HMI device class	HMI device
Panel	OP 177B PN/DP
	OP 277 6"
	TP 177B 4" PN/DP
	TP 177B 6" PN/DP
	TP 277 6"
Mobile Panel	Mobile Panel 177 PN
	Mobile Panel 277 8"
	Mobile Panel 277 10"
	Mobile Panel 277 IWLAN V2
	Mobile Panel 277(F) IWLAN V2
	Mobile Panel 277(F) IWLAN V2 (RFID tags)
Multi Panel	Multi Panel 177 6" Touch
	Multi Panel 277 Key
	Multi Panel 277 Touch
	Multi Panel 377 Key
	Multi Panel 377 Touch
Comfort Panel	KTP400 Comfort
	KP400 Comfort
	KP700 Comfort
	TP700 Comfort
	KP900 Comfort
	TP900 Comfort
	KP1200 Comfort
	TP1200 Comfort
	KP1500 Comfort
	TP1500 Comfort
	TP1900 Comfort
	TP2200 Comfort

Restrictions for PROFINET IO direct keys

Note

Direct keys are still active when the HMI device is in "offline" mode.

Note

If an external application such as Pocket Internet Explorer or Control Panel is started, it will run in the foreground and Runtime is placed in the background. The bit for the function "DirectKeyScreenNumber" is no longer set and the keys or buttons with the configured function "DirectKey" no longer trigger the associated bit in the PLC.

Restrictions

- It is not permitted to operate PROFIBUS IO direct keys and PROFINET DP direct keys simultaneously.
- Establish the following via the "PROFINET IO enabled" option in the control panel of the HMI device:
  - Option disabled = PROFIBUS DP direct keys enabled
  - Option enabled = PROFINET IO direct keys enabled
- If communication is enabled with PROFINET IO, you are not permitted to use the serial interface.
- You can operate direct keys only on local HMI devices. On the Sm@rtClient it is possible to operate the key / button for the direct key. However, No bit is set in the I/O area of the CPU.
- Direct keys which are assigned to a button are only triggered through touch operation. Triggering with a mouse click, for example with a connected USB mouse, is not possible.
- With touch operation, direct keys are triggered independent of any configured password protection.
- The buttons used as direct keys on HMI devices with touch screen functionality may not be modified as follows by means of scripting:
  - Moving
  - Resizing
  - Hiding
  - Disabling for operation
- The LEDs are addressed either via the PROFINET IO direct keys or the HMI Runtime application. Avoid concurrent addressing via the PROFINET IO direct keys and the HMI Runtime application. The LEDs "ACK", "A-Z l", "A-Z r" and "HELP" are reserved for system functions and cannot be configured. It is not advisable to control the "ACK", "A-Z l", "A-Z r" and "HELP" LEDs by means of PROFINET IO direct key functionality.

Inputs and outputs of HMI devices

Assignment of inputs and outputs

Assigning the inputs/outputs

The keys or buttons of the device are assigning bytes in the input area. The LEDs are assigning bytes in the output area. The number of bytes used depends on the HMI device.

The touch panels do not have any permanently assigned keys. They only have user-configurable buttons. Via the "DirectKey system function you can assign a bit to a button in the input area. The counting direction of the bit in the input direction is from right to left. In contrast to the operator panels which have permanently assigned keys, the touch panel buttons can be assigned freely.

Assigning direct keys to screen numbers (only for touch devices)

If a PROFINET IO direct key is using the same bit for different functions in different screens, the SIMATIC PLC must differentiate between the respective functionality by means of the screen number. You can use the "DirectKeyScreenNumber" system function to avoid the delayed updating of the screen number following a screen change.

Using the "DirectKeyScreenNumber"system function you can set within the input area any number of bits in order to identify the screen and simultaneously transfer the direct key bits to the PLC. This ensures unambiguous allocation of a control bit to screen number at all times.

See also

Panel (Page 6525)

Multi Panel (Page 6528)

Mobile Panel (Page 6531)

Comfort Panel (Page 6536)

Panel

OP 177

Direct keys OP 177B

Inputs	Outputs
9 bytes	4 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	K2	K1	F14	F13	F12	F11	F10	F9	n+1
	K10	K9	K8	K7	K6	K5	K4	K3	n+2
	K18	K17	K16	K15	K14	K13	K12	K11	n+3
									n+4
Touch buttons	7	6	5	4	3	2	1	0	n+5
	15	14	13	12	11	10	9	8	n+6
	23	22	21	20	19	18	17	16	n+7
	31	30	29	28	27	26	25	24	n+8

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
K2	K1						
K10	K9	K8	K7	K6	K5	K4	K3
K18	K17	K16	K15	K14	K13	K12	K11

TP 177

Direct keys TP 177B 4"

Inputs	Outputs
5 bytes	–

Direct key assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys					F4	F3	F2	F1	n+0
Touch buttons	7	6	5	4	3	2	1	0	n+1
	15	14	13	12	11	10	9	8	n+2
	23	22	21	20	19	18	17	16	n+3
	31	30	29	28	27	26	25	24	n+4

LED bits

Direct keys TP 177B 6"

Inputs	Outputs
4 bytes	--

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3

LED

No output area

OP 277

Direct keys OP 277 6"

Inputs	Outputs
4 bytes	4 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	K2	K1	F14	F13	F12	F11	F10	F9	n+1
	K10	K9	K8	K7	K6	K5	K4	K3	n+2
	ACK	ALT	CTRL	SHIFT					n+3

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
K2	K1						
K10	K9	K8	K7	K6	K5	K4	K3
ACK	A-Z left	A-Z right	HELP				

TP 277

Direct keys TP 277 6"

Inputs	Outputs
4 bytes	–

	Direct keys assignment									LED
	7	6	5	4	3	2	1	0	Byte	
Touch buttons	7	6	5	4	3	2	1	0	n+0	
	15	14	13	12	11	10	9	8	n+1	No output area
	23	22	21	20	19	18	17	16	n+2	
	31	30	29	28	27	26	25	24	n+3	

Multi Panel

Multi Panel 177

Direct keys Multi Panel 177 6" Touch

Inputs	Outputs
4 bytes	-

	Direct keys assignment									LED
	7	6	5	4	3	2	1	0	Byte	
Touch buttons	7	6	5	4	3	2	1	0	n+0	
	15	14	13	12	11	10	9	8	n+1	No output area
	23	22	21	20	19	18	17	16	n+2	
	31	30	29	28	27	26	25	24	n+3	

Multi Panel 277

Direct keys Multi Panel 277 8" Key

Inputs	Outputs
5 bytes	5 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
	K4	K3	K2	K1			F18	F17	n+2
					K8	K7	K6	K5	n+3
	ACK	ALT	CTRL	SHIFT					n+4

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
						F10	F9
K4	K3	K2	K1				
				K8	K7	K6	K5
ACK	A-Z left	A-Z right	HELP				

Direct keys Multi Panel 277 10" Key

Inputs	Outputs
5 bytes	5 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
	K4	K3	K2	K1	F20	F19	F18	F17	n+2
	K12	K11	K10	K9	K8	K7	K6	K5	n+3
	ACK	ALT	CTRL	SHIFT	K16	K15	K14	K13	n+4

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
				F12	F11	F10	F9
K4	K3	K2	K1				
K12	K11	K10	K9	K8	K7	K6	K5
ACK	A-Z left	A-Z right	HELP	K16	K15	K14	K13

Direct keys Multi Panel 277 8" Touch

Inputs	Outputs
5 bytes	–

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

Direct keys Multi Panel 277 10" Touch

Inputs	Outputs
5 bytes	–

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

Multi Panel 377

Direct keys Multi Panel 377 Key

Inputs	Outputs
5 bytes	5 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Keys	S8	S7	S6	S5	S4	S3	S2	S1	n+0
	S16	S15	S14	S13	S12	S11	S10	S9	n+1
	F8	F7	F6	F5	F4	F3	F2	F1	n+2
	F16	F15	F14	F13	F12	F11	F10	F9	n+3
	ACK	ALT	CTRL	SHIFT	F20	F19	F18	F17	n+4

LED

7	6	5	4	3	2	1	0
S8	S7	S6	S5	S4	S3	S2	S1
S16	S15	S14	S13	S12	S11	S10	S9
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
ACK	A-Z left	A-Z right	INFO	F20	F19	F18	F17

Multi Panel 377 Touch

Inputs	Outputs
5 bytes	–

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

Mobile Panel

Mobile Panel 177

Direct keys Mobile Panel 177 PN

Inputs	Outputs
9 bytes	2 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
			F14	F13	F12	F11	F10	F9	n+1
optional operator controls						T1	S1	S0	n+2
	I7	I6	I5	I4	I3	I2	I1	I0	n+3
	D7	D6	D5	D4	D3	D2	D1	D0	n+4
Touch buttons	7	6	5	4	3	2	1	0	n+5
	15	14	13	12	11	10	9	8	n+6
	23	22	21	20	19	18	17	16	n+7
	31	30	29	28	27	26	25	24	n+8

LED-Bits

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
							T1

S0-S1: Key-operated switch
T1: Illuminated pushbutton 1
I0-I7: Handwheel pulses (forwards)
D0-D7: Handwheel pulses (backwards)

Mobile Panel 277

Direct keys Mobile Panel 277 8"

HMI device	Inputs	Outputs
Mobile Panel 277 8"	10 bytes	4 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
							F18	F17	n+2
optional operator controls				T2		T1	S1	S0	n+3
	I7	I6	I5	I4	I3	I2	I1	I0	n+4
	D7	D6	D5	D4	D3	D2	D1	D0	n+5
Touch buttons	7	6	5	4	3	2	1	0	n+6
	15	14	13	12	11	10	9	8	n+7
	23	22	21	20	19	18	17	16	n+8
	31	30	29	28	27	26	25	24	n+9

LED-Bits

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
						F18	F17
					T2	T1	

S0-S1: Key-operated switch
T1: Illuminated pushbutton 1
T2: Illuminated pushbutton 2
I0-I7: Handwheel pulses (forwards)
D0-D7: Handwheel pulses (backwards)

Direct keys Mobile Panel 277 10"

HMI device	Inputs	Outputs
Mobile Panel 277 10"	5 bytes	--

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

Mobile Panel 277 IWLAN V2

Direct keys Mobile Panel 277 IWLAN V2

The assignment of the inputs and outputs of the direct keys applies to the following HMI devices:

- Mobile Panel 277 IWLAN V2
- Mobile Panel 277F IWLAN V2
- Mobile Panel 277 IWLAN V2 (RFID tags)

Inputs	Outputs
10 bytes	4 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
							F18	F17	n+2
optional operator controls				T2		T1	S1	S0	n+3
	I7	I6	I5	I4	I3	I2	I1	I0	n+4
	D7	D6	D5	D4	D3	D2	D1	D0	n+5
Touch buttons	7	6	5	4	3	2	1	0	n+6
	15	14	13	12	11	10	9	8	n+7
	23	22	21	20	19	18	17	16	n+8
	31	30	29	28	27	26	25	24	n+9

LED-Bits

Byte	7	6	5	4	3	2	1	0
n+0	F8	F7	F6	F5	F4	F3	F2	F1
n+1	F16	F15	F14	F13	F12	F11	F10	F9
n+2							F18	F17
n+3						T2	T1	

S0-S1: Key-operated switch
T1: Illuminated pushbutton 1
T2: Illuminated pushbutton 2
I0-I7: Handwheel pulses (forwards)
D0-D7: Handwheel pulses (backwards)

KTP700 Mobile, KTP900 Mobile

Evaluating operator controls as direct keys

You can configure the operator controls of the HMI device as direct keys. The states of the following operator controls are available directly in the I/O area of the PLC:

- The switching state of the function keys
- The switching state of the key-operated switch
- The switching state of the illuminated pushbuttons

Byte assignment

The following table shows the assignment of the keys (inputs) and LEDs (outputs) to the bytes in the PLC process image. For more information, see your plant documentation.

- KTP700 Mobile and KTP700F Mobile

Direct key bit							
7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
T2	T1	S1	S0				
7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8
23	22	21	20	19	18	17	16
31	30	29	28	27	26	25	24

Byte
n
n + 1
n + 2
n + 3
n + 4
n + 5

LED bit							
7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
T2	T1						

F Bit for function key
S Bit for key-operated switch
T1 Bit for left illuminated pushbutton
T2 Bit for right illuminated pushbutton

The bytes "n+2" to "n+5" contain the direct key bits for the touch buttons.

- KTP900 Mobile and KTP900F Mobile

Direct key bit							
7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
T2	T1	S1	S0			F10	F9
7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8
23	22	21	20	19	18	17	16
31	30	29	28	27	26	25	24

Byte
n
n + 1
n + 2
n + 3
n + 4
n + 5

LED bit							
7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
T2	T1					F10	F9

F Bit for function key
S Bit for key-operated switch
T1 Bit for left illuminated pushbutton
T2 Bit for right illuminated pushbutton

The bytes "n+2" to "n+5" contain the direct key bits for the touch buttons.

Bit coding

The following tables show the bit coding for function keys, key-operated switch and illuminated pushbutton:

- Bit coding of function keys

State	F1 to F8 or F10
Not pressed	0
Pressed	1

- Bit coding of function key LEDs

State	F1 to F8 or F10
LED not illuminated	0
LED is illuminated	1

- Bit coding of key-operated switch

State	S1	S0	Key position
Position 0	0	0	In middle position
Position I	0	1	Turned in clockwise direction up to stop
Position II	1	0	Turned counter-clockwise up to stop

- Bit coding of illuminated pushbuttons

State	K1	K2
Not pressed	0	0
Pressed	1	1

- Bit coding of illuminated pushbutton LEDs

State	K1	K2
Off	0	0
On permanently	1	1

Controlling the LEDs of the function keys by means of system functions

F1 to F8 or F10 to LEDs are integrated in the function keys of the HMI device. The connected controller can control the integrated LEDs. The LED signals the operator that he should use the corresponding function key while the project is running.

The following table shows the possible states of the LEDs and the corresponding entries in bit n+1 and bit n of the LED tags:

Bit n + 1	Bit n	LED status
0	0	Off
0	1	Flashing quickly
1	0	Flashing slowly
1	1	ON, continuous

Comfort Panel

KTP400 Comfort

Direct keys KTP400 Comfort

HMI device	Inputs	Outputs
Touch operation	5 bytes	1 byte

Direct key assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys					F4	F3	F2	F1	n+0
Touch buttons	7	6	5	4	3	2	1	0	n+1
	15	14	13	12	11	10	9	8	n+2
	23	22	21	20	19	18	17	16	n+3
	31	30	29	28	27	26	25	24	n+4

LED bits

3	2	1	0
F4	F3	F2	F1

KP400 Comfort

Direct keys KP400 Comfort

Inputs	Outputs
1 byte	1 byte

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1

KP700 Comfort

Direct keys KP700 Comfort

Inputs	Outputs
3 bytes	3 bytes

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
	F24	F23	F22	F21	F20	F19	F18	F17	n+2

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
F24	F23	F22	F21	F20	F19	F18	F17

TP700 Comfort

Direct keys TP700 Comfort

Inputs	Outputs
4 bytes	--

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3

LED

No output area

KP900 Comfort

Direct keys KP900 Comfort

Inputs	Outputs
4 bytes	4 bytes

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
	F24	F23	F22	F21	F20	F19	F18	F17	n+2
							F26	F25	n+3

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
F24	F23	F22	F21	F20	F19	F18	F17
						F18	F17

TP900 Comfort

Direct keys TP900 Comfort

Inputs	Outputs
5 bytes	--

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

KP1200 Comfort

Direct keys KP1200 Comfort

Inputs	Outputs
5 bytes	5 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
	F24	F23	F22	F21	F20	F19	F18	F17	n+2
	F32	F31	F30	F29	F28	F27	F26	F25	n+3
							F34	F33	n+4

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
F24	F23	F22	F21	F20	F19	F18	F17
F32	F31	F30	F29	F28	F27	F26	F25
						F34	F33

TP1200 Comfort

Direct keys TP1200 Comfort

Inputs	Outputs
5 bytes	--

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

KP1500 Comfort

Inputs	Outputs
5 bytes	5 bytes

Direct key assignment

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
	F24	F23	F22	F21	F20	F19	F18	F17	n+2
	F32	F31	F30	F29	F28	F27	F26	F25	n+3
					F36	F35	F34	F33	n+4

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
F24	F23	F22	F21	F20	F19	F18	F17
F32	F31	F30	F29	F28	F27	F26	F25
				F36	F35	F34	F33

TP1500, TP1900 and TP2200 Comfort

Inputs	Outputs
5 bytes	--

Direct key assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

12.11.13.5 PROFIBUS DP direct keys

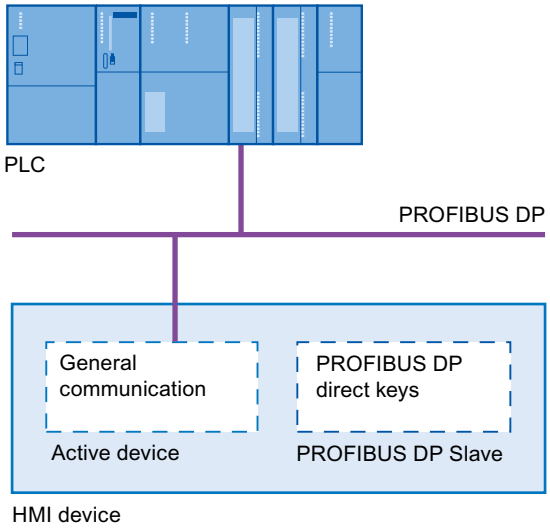
PROFIBUS DP direct keys

PROFIBUS DP direct keys

You configure the HMI device created in WinCC as an active communication partner in the automation network.

For the PROFIBUS DP direct keys, also configure the HMI device as a slave in the PROFIBUS DP network.

The following figure shows the basic configuration based on an automation network with an HMI device and a PLC.



Mode of operation of the PROFIBUS DP direct keys

With PROFIBUS DP direct keys, when the key or button is pressed, a bit is set in the I/O area of the CPU.

The cycle time of the PROFIBUS DP bus is calculated from the total of all configured inputs and outputs.

The number of configured inputs and outputs can influence the reaction time of the PROFIBUS DP direct key. For a typical configuration the response time of the PROFIBUS DP direct key is < 100 ms.

HMI devices for the configuration of PROFIBUS DP direct keys

HMI devices

You can configure PROFIBUS DP direct keys with the following HMI devices:

HMI device class	HMI device
Panel	OP 77B
	OP 177B
	OP 277 6"
	TP 177B 4"
	TP 177B 6"
	TP 277 6"
Mobile Panel	Mobile Panel 177 DP
	Mobile Panel 277 8"
	Mobile Panel 277 10"

HMI device class	HMI device
Multi Panel	Multi Panel 177 6" Touch
	Multi Panel 277 Key
	Multi Panel 277 Touch
	Multi Panel 377 Key
	Multi Panel 377 Touch
Comfort Panel	KTP400 Comfort
	KP400 Comfort
	KP700 Comfort
	TP700 Comfort
	KP900 Comfort
	TP900 Comfort
	KP1200 Comfort
	TP1200 Comfort
	KP1500 Comfort
	TP1500 Comfort
	TP1900 Comfort
	TP2200 Comfort

Restrictions for PROFIBUS DP direct keys

Note

If an external application such as Pocket Internet Explorer or Control Panel is started, it will run in the foreground and Runtime is placed in the background. The bit for the function "DirectKeyScreenNumber" is no longer set and the keys or buttons with the configured function "DirectKey" no longer trigger the associated bit in the PLC.

Restrictions

- It is not possible to operate PROFIBUS DP direct keys and PROFINET IO direct keys simultaneously.
  Establish the following via the "PROFINET IO enabled" option in the control panel of the HMI device:
  - Option disabled = PROFIBUS DP direct keys enabled
  - Option enabled = PROFINET IO direct keys enabled
- You can operate direct keys only on local HMI devices. On the Sm@rtClient it is possible to operate the key / button for the direct key. However, No bit is set in the I/O area of the CPU.

- The buttons used as direct keys on HMI devices with touch screen functionality may not be modified as follows by means of scripting:
  - Moving
  - Resizing
  - Hiding
  - Disabling for operation
- The LEDs are addressed either via the PROFIBUS DP direct keys or the HMI Runtime application. Avoid concurrent addressing via the PROFIBUS DP direct keys and the HMI Runtime application. The LEDs "ACK", "A-Z l", "A-Z r" and "HELP" are reserved for system functions and cannot be configured. We do not recommend addressing the LEDs "ACK", "A-Z l", "A-Z r" and "HELP" with the PROFIBUS DP direct keys.
- The PROFIBUS direct keys on an HMI device are triggered independent of any configured password protection.

Inputs and outputs of HMI devices

Assignment of inputs and outputs

Assigning the inputs/outputs

The keys or buttons of the device are assigning bytes in the input area. The LEDs occupy bytes in the DP output area. The following table shows the number of bytes used in the various HMI devices. The precise assignment is shown in the following screens.

The touch panels do not have any permanently assigned keys. They only have user-configurable buttons. Via the direct keys function you can assign to a button a bit in the input area. The counting direction of the bit in the input direction is from right to left. In contrast to the operator panels which have permanently assigned keys, the touch panel buttons can be assigned freely. For more detailed information, refer to the "WinCC flexible - Configuring WinCC Windows-based Systems" user manual.

Assigning direct keys to screen numbers (only for touch devices)

If a PROFIBUS DP direct key is using the same bit for different functions in different screens, the S7 must differentiate between the respective functionality by means of the screen number. To combat the delayed updating of the screen number in the PLC after a change of screen you use the "DirectKeyScreenNumber" screen function.

Using the "DirectKeyScreenNumber" you can set within the input area any number of bits in order to identify the screen and simultaneously transfer the direct key bits to the PLC. This ensures unambiguous allocation of a control bit to screen number at all times.

See also

Panel (Page 6544)

Multi Panel (Page 6547)

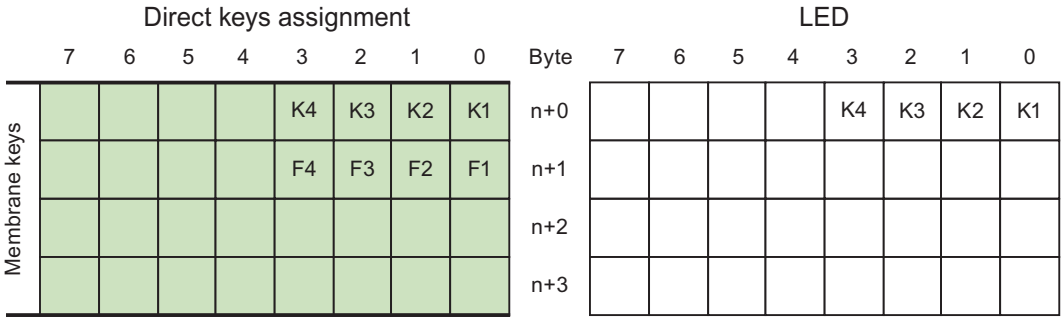
Mobile Panel (Page 6550)
Comfort Panel (Page 6552)

Panel

OP 77

Direct keys OP 77B

Inputs	Outputs
4 bytes	4 bytes



OP 177

Direct keys OP 177B

Inputs	Outputs
9 bytes	4 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	K2	K1	F14	F13	F12	F11	F10	F9	n+1
	K10	K9	K8	K7	K6	K5	K4	K3	n+2
	K18	K17	K16	K15	K14	K13	K12	K11	n+3
									n+4
Touch buttons	7	6	5	4	3	2	1	0	n+5
	15	14	13	12	11	10	9	8	n+6
	23	22	21	20	19	18	17	16	n+7
	31	30	29	28	27	26	25	24	n+8

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
K2	K1						
K10	K9	K8	K7	K6	K5	K4	K3
K18	K17	K16	K15	K14	K13	K12	K11

TP 177

Direct keys TP 177B 4"

Inputs	Outputs
5 bytes	–

Direct key assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys					F4	F3	F2	F1	n+0
Touch buttons	7	6	5	4	3	2	1	0	n+1
	15	14	13	12	11	10	9	8	n+2
	23	22	21	20	19	18	17	16	n+3
	31	30	29	28	27	26	25	24	n+4

LED bits

Direct keys TP 177B 6"

Inputs	Outputs
4 bytes	--

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3

LED

No output area

OP 277

Direct keys OP 277 6"

Inputs	Outputs
4 bytes	4 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	K2	K1	F14	F13	F12	F11	F10	F9	n+1
	K10	K9	K8	K7	K6	K5	K4	K3	n+2
	ACK	ALT	CTRL	SHIFT					n+3

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
K2	K1						
K10	K9	K8	K7	K6	K5	K4	K3
ACK	A-Z left	A-Z right	HELP				

TP 277

Direct keys TP 277 6"

Inputs	Outputs
4 bytes	–

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3

LED

No output area

Multi Panel

Multi Panel 177

Direct keys Multi Panel 177 6" Touch

Inputs	Outputs
4 bytes	-

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3

LED

No output area

Multi Panel 277

Direct keys Multi Panel 277 8" Key

Inputs	Outputs
5 bytes	5 bytes

	Direct keys assignment									LED							
	7	6	5	4	3	2	1	0	Byte	7	6	5	4	3	2	1	0
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0	F8	F7	F6	F5	F4	F3	F2	F1
	F16	F15	F14	F13	F12	F11	F10	F9	n+1							F10	F9
	K4	K3	K2	K1			F18	F17	n+2	K4	K3	K2	K1				
					K8	K7	K6	K5	n+3					K8	K7	K6	K5
	ACK	ALT	CTRL	SHIFT					n+4	ACK	A-Z left	A-Z right	HELP				

Direct keys Multi Panel 277 10" Key

Inputs	Outputs
5 bytes	5 bytes

	Direct keys assignment									LED							
	7	6	5	4	3	2	1	0	Byte	7	6	5	4	3	2	1	0
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0	F8	F7	F6	F5	F4	F3	F2	F1
	F16	F15	F14	F13	F12	F11	F10	F9	n+1					F12	F11	F10	F9
	K4	K3	K2	K1	F20	F19	F18	F17	n+2	K4	K3	K2	K1				
	K12	K11	K10	K9	K8	K7	K6	K5	n+3	K12	K11	K10	K9	K8	K7	K6	K5
	ACK	ALT	CTRL	SHIFT	K16	K15	K14	K13	n+4	ACK	A-Z left	A-Z right	HELP	K16	K15	K14	K13

Direct keys Multi Panel 277 8" Touch

Inputs	Outputs
5 bytes	–

	Direct keys assignment									LED
	7	6	5	4	3	2	1	0	Byte	
Touch buttons	7	6	5	4	3	2	1	0	n+0	
	15	14	13	12	11	10	9	8	n+1	No output area
	23	22	21	20	19	18	17	16	n+2	
	31	30	29	28	27	26	25	24	n+3	
	39	38	37	36	35	34	33	32	n+4	

Direct keys Multi Panel 277 10" Touch

Inputs	Outputs
5 bytes	–

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

Multi Panel 377

Direct keys Multi Panel 377 Key

Inputs	Outputs
5 bytes	5 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Keys	S8	S7	S6	S5	S4	S3	S2	S1	n+0
	S16	S15	S14	S13	S12	S11	S10	S9	n+1
	F8	F7	F6	F5	F4	F3	F2	F1	n+2
	F16	F15	F14	F13	F12	F11	F10	F9	n+3
	ACK	ALT	CTRL	SHIFT	F20	F19	F18	F17	n+4

LED

7	6	5	4	3	2	1	0
S8	S7	S6	S5	S4	S3	S2	S1
S16	S15	S14	S13	S12	S11	S10	S9
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
ACK	A-Z left	A-Z right	INFO	F20	F19	F18	F17

Multi Panel 377 Touch

Inputs	Outputs
5 bytes	–

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

Mobile Panel

Mobile Panel 177

Direct keys Mobile Panel 177 DP

Inputs	Outputs
9 bytes	4 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
			F14	F13	F12	F11	F10	F9	n+1
optional operator controls						T1	S1	S0	n+2
	I7	I6	I5	I4	I3	I2	I1	I0	n+3
	D7	D6	D5	D4	D3	D2	D1	D0	n+4
Touch buttons	7	6	5	4	3	2	1	0	n+5
	15	14	13	12	11	10	9	8	n+6
	23	22	21	20	19	18	17	16	n+7
	31	30	29	28	27	26	25	24	n+8

LED-Bits

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
							T1

S0-S1: Key-operated switch
T1: Illuminated pushbutton 1
I0-I7: Handwheel pulses (forwards)
D0-D7: Handwheel pulses (backwards)

Mobile Panel 277

Direct keys Mobile Panel 277 8"

HMI device	Inputs	Outputs
Mobile Panel 277 8"	10 bytes	4 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
							F18	F17	n+2
optional operator controls				T2		T1	S1	S0	n+3
	I7	I6	I5	I4	I3	I2	I1	I0	n+4
	D7	D6	D5	D4	D3	D2	D1	D0	n+5
Touch buttons	7	6	5	4	3	2	1	0	n+6
	15	14	13	12	11	10	9	8	n+7
	23	22	21	20	19	18	17	16	n+8
	31	30	29	28	27	26	25	24	n+9

LED-Bits

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
						F18	F17
					T2	T1	

S0-S1: Key-operated switch
T1: Illuminated pushbutton 1
T2: Illuminated pushbutton 2
I0-I7: Handwheel pulses (forwards)
D0-D7: Handwheel pulses (backwards)

Direct keys Mobile Panel 277 10"

HMI device	Inputs	Outputs
Mobile Panel 277 10"	5 bytes	--

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

Mobile Panel 277 IWLAN V2

Direct keys Mobile Panel 277 IWLAN V2

The assignment of the inputs and outputs of the direct keys applies to the following HMI devices:

- Mobile Panel 277 IWLAN V2
- Mobile Panel 277F IWLAN V2
- Mobile Panel 277 IWLAN V2 (RFID tags)

Inputs	Outputs
10 bytes	4 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Membrane keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
							F18	F17	n+2
optional operator controls				T2		T1	S1	S0	n+3
	I7	I6	I5	I4	I3	I2	I1	I0	n+4
	D7	D6	D5	D4	D3	D2	D1	D0	n+5
Touch buttons	7	6	5	4	3	2	1	0	n+6
	15	14	13	12	11	10	9	8	n+7
	23	22	21	20	19	18	17	16	n+8
	31	30	29	28	27	26	25	24	n+9

LED-Bits

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
						F18	F17
					T2	T1	

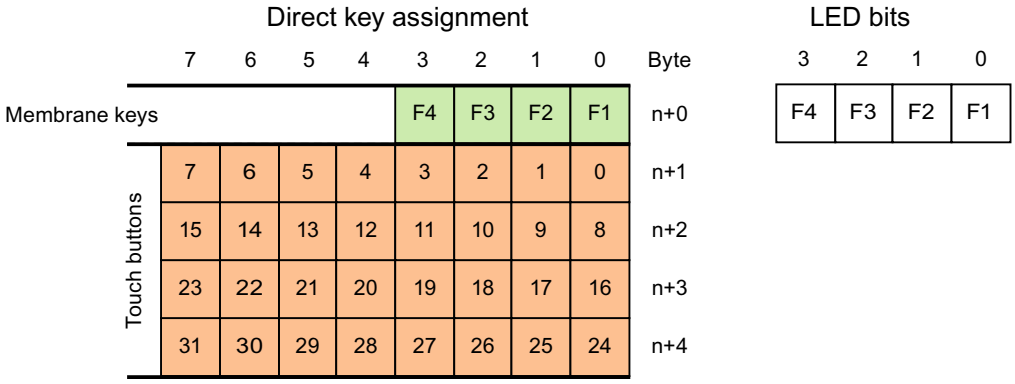
S0-S1: Key-operated switch
T1: Illuminated pushbutton 1
T2: Illuminated pushbutton 2
I0-I7: Handwheel pulses (forwards)
D0-D7: Handwheel pulses (backwards)

Comfort Panel

KTP400 Comfort

Direct keys KTP400 Comfort

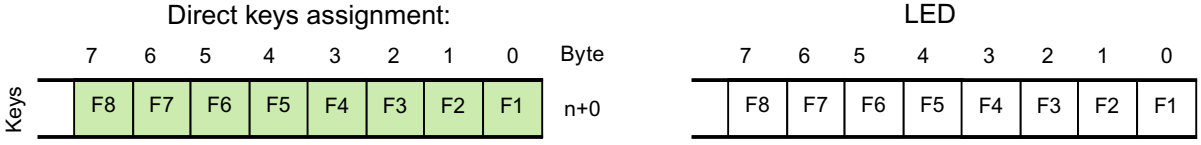
HMI device	Inputs	Outputs
Touch operation	5 bytes	1 byte



KP400 Comfort

Direct keys KP400 Comfort

Inputs	Outputs
1 byte	1 byte



KP700 Comfort

Direct keys KP700 Comfort

Inputs	Outputs
3 bytes	3 bytes

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
	F24	F23	F22	F21	F20	F19	F18	F17	n+2

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
F24	F23	F22	F21	F20	F19	F18	F17

TP700 Comfort

Direct keys TP700 Comfort

Inputs	Outputs
4 bytes	--

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3

LED

No output area

KP900 Comfort

Direct keys KP900 Comfort

Inputs	Outputs
4 bytes	4 bytes

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
	F24	F23	F22	F21	F20	F19	F18	F17	n+2
							F26	F25	n+3

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
F24	F23	F22	F21	F20	F19	F18	F17
						F18	F17

TP900 Comfort

Direct keys TP900 Comfort

Inputs	Outputs
5 bytes	--

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

KP1200 Comfort

Direct keys KP1200 Comfort

Inputs	Outputs
5 bytes	5 bytes

Direct keys assignment

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
	F24	F23	F22	F21	F20	F19	F18	F17	n+2
	F32	F31	F30	F29	F28	F27	F26	F25	n+3
							F34	F33	n+4

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
F24	F23	F22	F21	F20	F19	F18	F17
F32	F31	F30	F29	F28	F27	F26	F25
						F34	F33

TP1200 Comfort

Direct keys TP1200 Comfort

Inputs	Outputs
5 bytes	--

Direct keys assignment:

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

KP1500 Comfort

Inputs	Outputs
5 bytes	5 bytes

Direct key assignment

	7	6	5	4	3	2	1	0	Byte
Keys	F8	F7	F6	F5	F4	F3	F2	F1	n+0
	F16	F15	F14	F13	F12	F11	F10	F9	n+1
	F24	F23	F22	F21	F20	F19	F18	F17	n+2
	F32	F31	F30	F29	F28	F27	F26	F25	n+3
					F36	F35	F34	F33	n+4

LED

7	6	5	4	3	2	1	0
F8	F7	F6	F5	F4	F3	F2	F1
F16	F15	F14	F13	F12	F11	F10	F9
F24	F23	F22	F21	F20	F19	F18	F17
F32	F31	F30	F29	F28	F27	F26	F25
				F36	F35	F34	F33

TP1500, TP1900 and TP2200 Comfort

Inputs	Outputs
5 bytes	--

Direct key assignment

	7	6	5	4	3	2	1	0	Byte
Touch buttons	7	6	5	4	3	2	1	0	n+0
	15	14	13	12	11	10	9	8	n+1
	23	22	21	20	19	18	17	16	n+2
	31	30	29	28	27	26	25	24	n+3
	39	38	37	36	35	34	33	32	n+4

LED

No output area

12.11.14 Communication via SIMATIC HMI HTTP

12.11.14.1 Basic information on SIMATIC HMI HTTP

Introduction

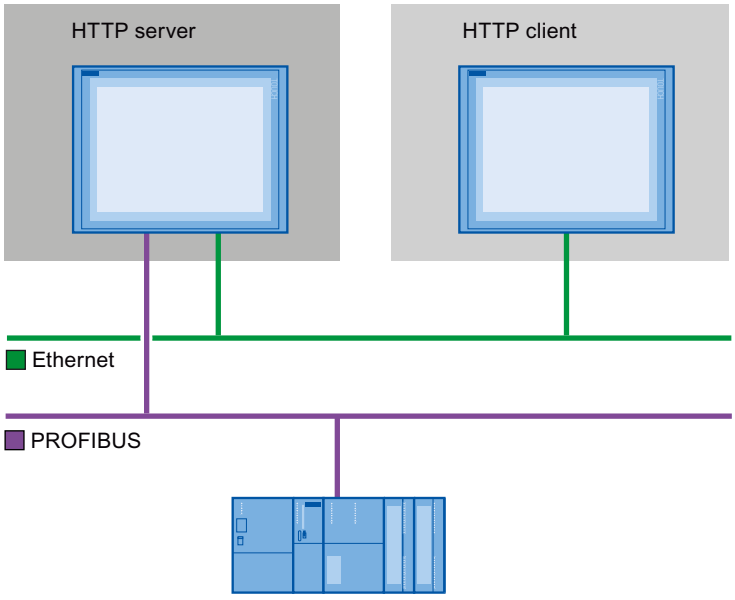
You use the SIMATIC HMI HTTP Protocol for data exchange between HMI devices.

The SIMATIC HMI HTTP Protocol is not suited for exchange of bulk data.

Data exchange occurs according to the Request-Response method. The HTTP client sends its request to the HTTP server, which processes it and returns a response.

A device can be configured as an HTTP client and HTTP server simultaneously.

The client and server establish a connection via the Ethernet interface for data exchange.



HTTP components

The components of the SIMATIC HMI HTTP Protocol are transferred to the HMI device in "Load to device".

HTTP components are:

- HTTP server
- HTTP client

If you use a Standard-PC or SIMATIC IPC, you must install WinCC RT Advanced or WinCC RT Professional first.

SMTP protocol

The SMTP protocol is not a requirement for communication via SIMATIC HMI HTTP Protocol .

HTTP / HTTPS

TheSIMATIC HMI HTTP Protocol offers two standards:

- HTTP
  Is implemented in local networks for a fast, uncoded transfer of uncritical data.
- HTTPS
  Enables a reliable HTTP connection between devices. Initially there is an exchange of secret keys. Then, in a further step, the respective public keys can be securely exchanged in the form of digital certificates. The public keys are used to encode the utility data in order to guarantee a bug-proof communication.

Note

Due to the encryption, the HTTPS protocol has a lower transmission performance than the HTTP protocol.

Caution

End users are responsible for the security of their own networks.

12.11.14.2 Configuring a connection via SIMATIC HMI HTTP

Configuring a connection via SIMATIC HMI HTTP

Communication between HMI devices via SIMATIC HMI HTTP protocol

Several steps are needed to configure communication between HMI devices via the SIMATIC HMI HTTP protocol:

1. Configure HMI device as a server.
2. Configure HMI device as a client.
3. Assign tag parameters in the client.
4. Commission connection.

Access to tags via SIMATIC HMI HTTP protocol

If access to tags via the SIMATIC HMI HTTP protocol is to be enabled, these tags must be defined for the relevant HMI devices and interconnected.

Configuring an HMI device as a server

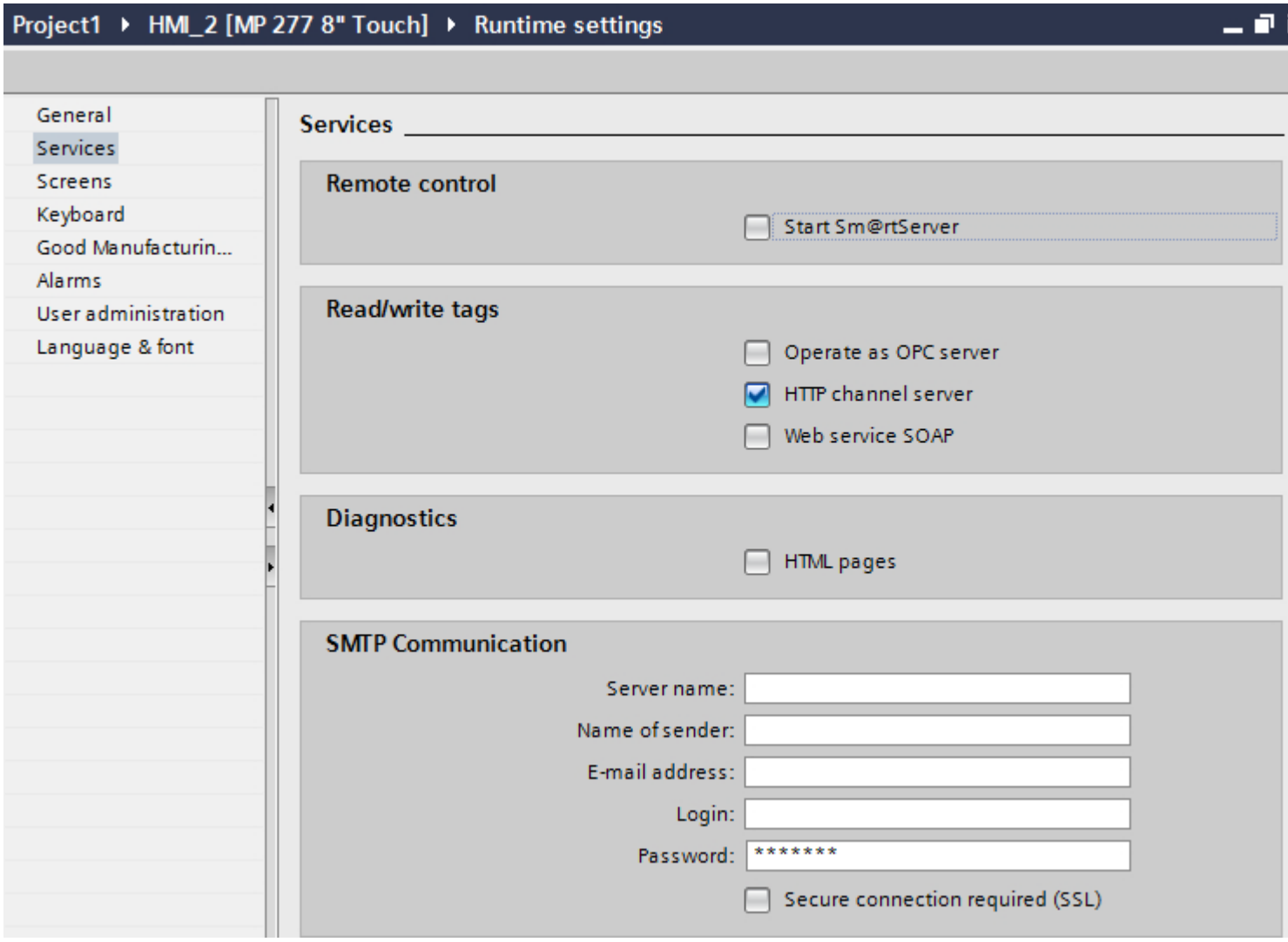
Configuring a server

Requirements

- The HTTP communication channel must be set in the Control Panel of the HMI device.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click "Runtime settings".
   The "Runtime settings" editor opens.
3. Click "Services".
4. Select the "HTTP channel server" option.



If you use the SMTP protocol, you assign the following values under "SMTP settings" according to your project specifications:

- Server name
- Login
- Password

Result

The HMI device has been configured as a server.

Tags in the server

Tags used

The client can use the HTTP protocol for read and write access to tags configured on the server in runtime. This means that it is not necessary to configure additional tags for an HTTP communication.

However, the following aspects must be taken into account to ensure correct data exchange:

- The data type of the server tags must match the data type in the client.
- The tag name configured in the HTTP server must be identical to the name of the address tag of the HTTP client.

Configuring an HMI device as a client

Configuring a client

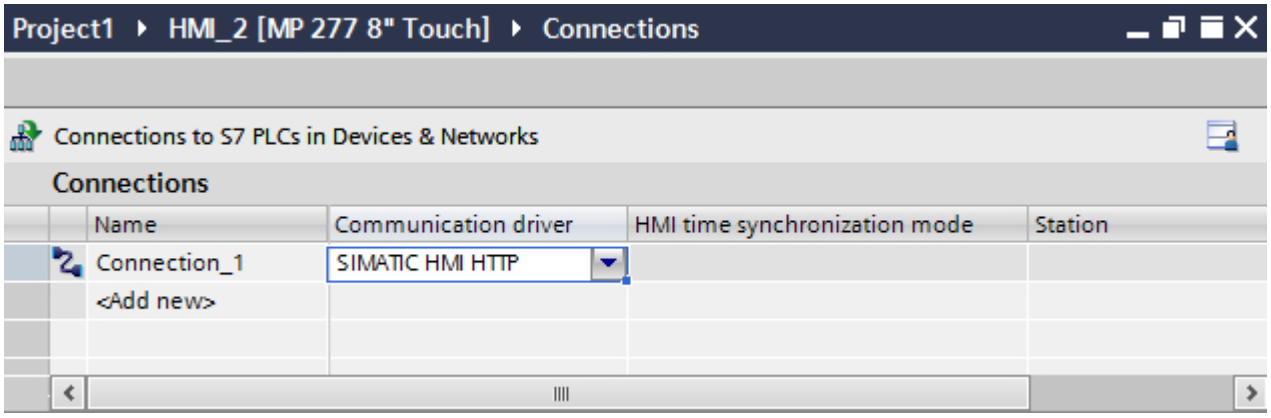
Requirements

- The HTTP communication channel must be set in the Control Panel of the HMI device.
- An HMI device has been configured as a server.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.

4. In the "Communication drivers" column, select the "SIMATIC HMI HTTP Protocol" driver.



5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".
6. Set the parameters in the Property window.
   See "Setting parameters for the client" for more information about the parameters.

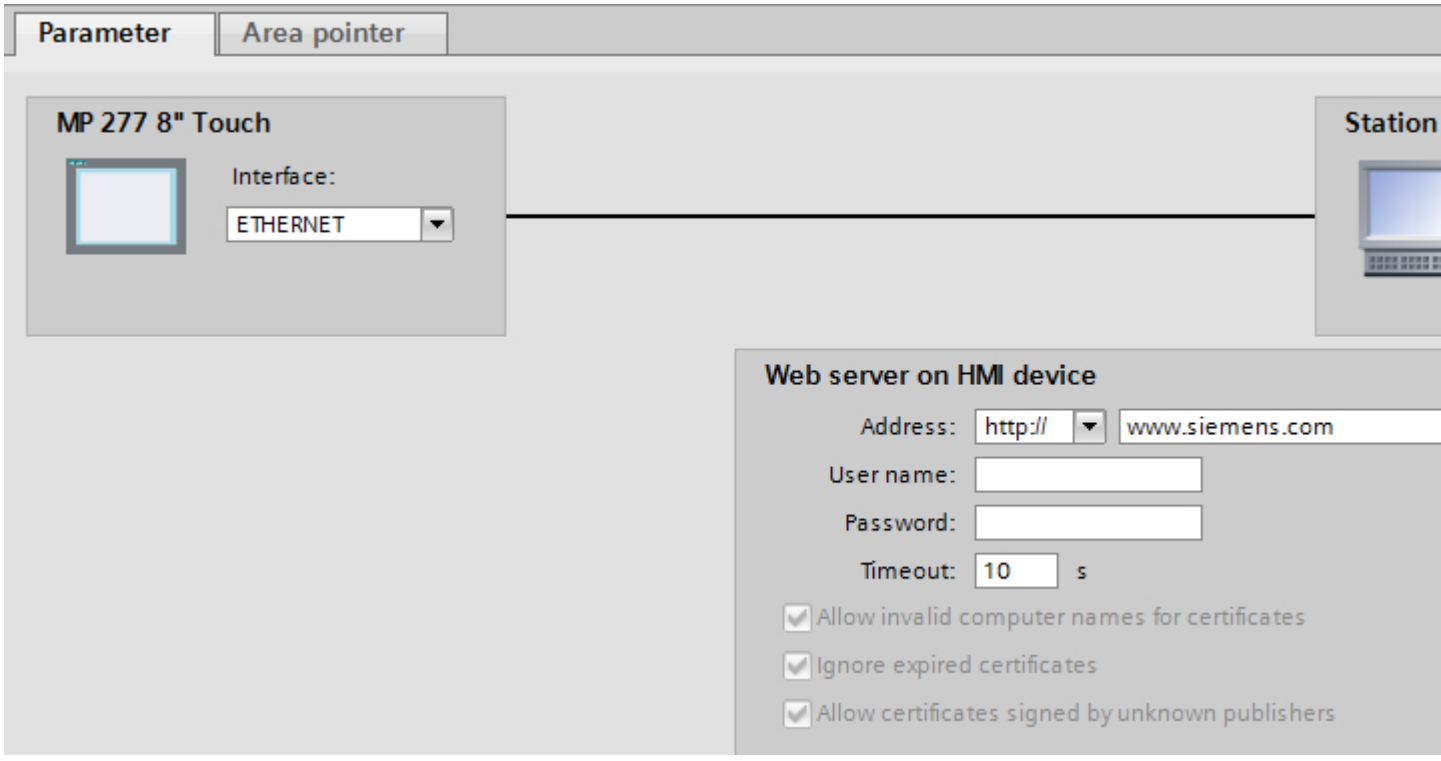
Setting parameters for the client

Requirements

- Connection has been created via SIMATIC HMI HTTP protocol.
- "Connections" editor is open.

Procedure

1. Select "Ethernet" in the Inspector window under "Parameters > Interface".
2. Select protocol http:// or https://.
3. Enter the name of the server or its IP address.



User name and password

If the "Authentication required" check box was selected in the server in the "Control Panel > WinCC Internet Settings > Web Server" dialog, a user name and password must be entered here in the client.

Timeout

Assign a time after which a connection break is recognized.

HTTPS protocol

If the HTTPS protocol is selected, with the following settings you can establish how the HTTPS client should check the properties of the server certificate and how it should react in the event of error:

- "Allow invalid computer names for certificates"
- "Allow expired certificates"
- "Allow certificates signed by unknown publishers"

Reading out an IP address

If the server has already been commissioned, you can read out the IP address on the server as well:

- For Panel
  Click "Start > Programs > Command Prompt" on the server and enter the "ipconfig" command using the screen keyboard. The IP address is displayed after pressing <Ret>.
- For PC / Panel PC
  Click "Start > Run" on the server, enter "Cmd", and press <Ret>: The command interpreter is displayed. Enter the "ipconfig" command. The IP address is displayed after pressing <Ret>.

Assigning tag parameters in the client

Assigning tag parameters in the client

Introduction

In order to be able to access tags on the server, they must be configured in the client as tag addresses.

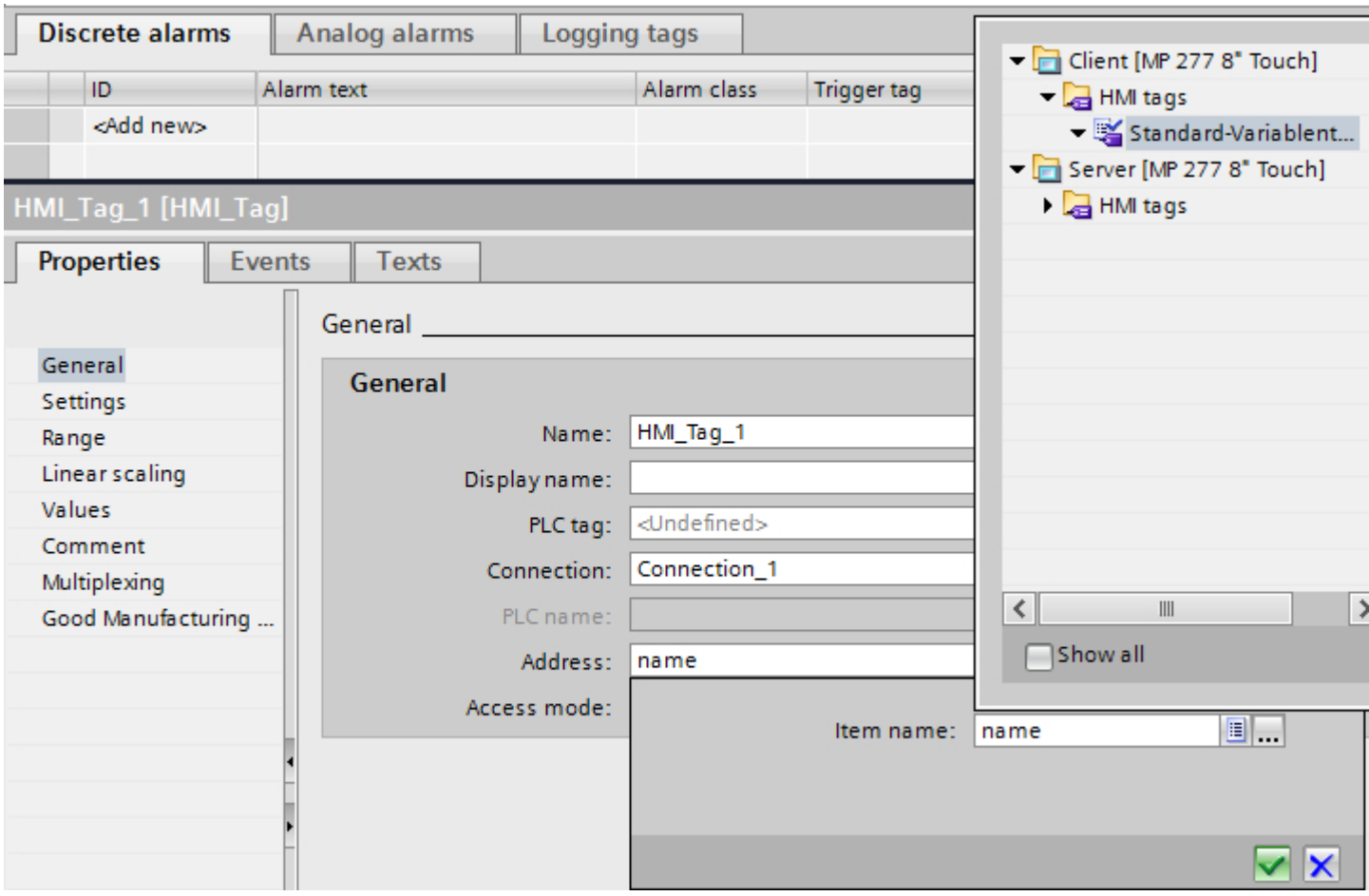
Requirements

- Tags have been created in the "HMI tags" editor of the server.

Procedure

1. Double-click the client under "Devices" in the project tree.
2. Double-click "HMI tags".
3. Create a new tag.
4. Select a data type.

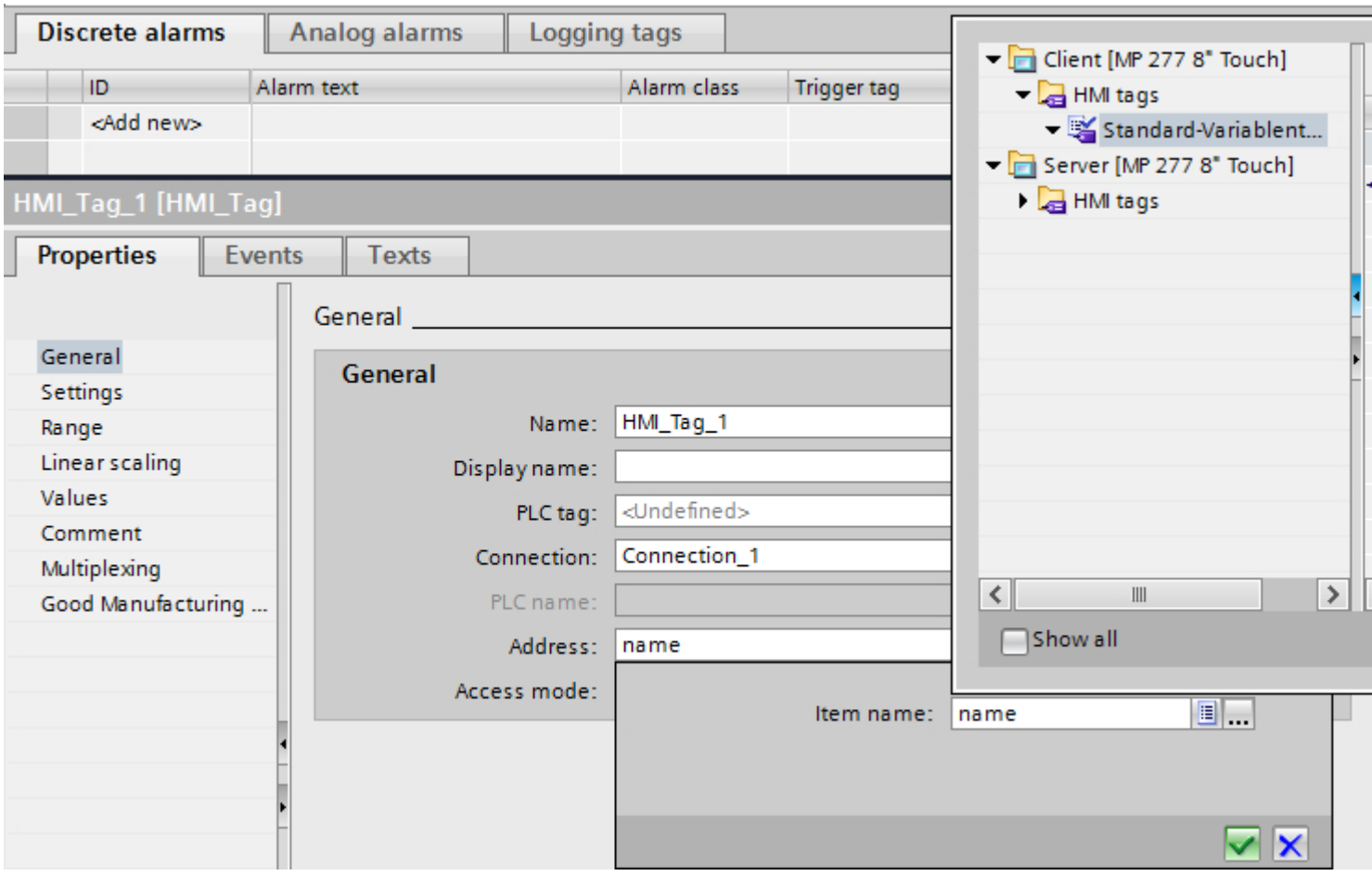
5. Open the selection dialog in the "Addresses" column.



6. Define the parameters in the working area:
   See "Parameters for the client tags" for more information about the parameters.

Parameters for the tags of the client

Parameters for the tags



Data type

- The client does not check the data type. Therefore, pay attention that the data type selected here matches the data type of the tags on the server.
- Array tags are not permitted.

Addressing

If the server and client are not within the same project, note the following:

- Enter the exact name of the tag that is to be communicated with on the server.

Commissioning a connection

Setting the server's Internet Settings

Introduction

To commission an HTTP/HTTPS connection, the WinCC Internet Settings must be set up in the Control Panel of both the server and client, in addition to the configuration in WinCC.

The name and number of tabs contained in the "WinCC Internet Settings" dialog depend on the software installed.

Only the required tabs with the MP 277 HMI device as an example are shown below.

WinCC Internet Settings, Proxy tab

The valid network settings are specified by the network administrator.

Make the following settings:

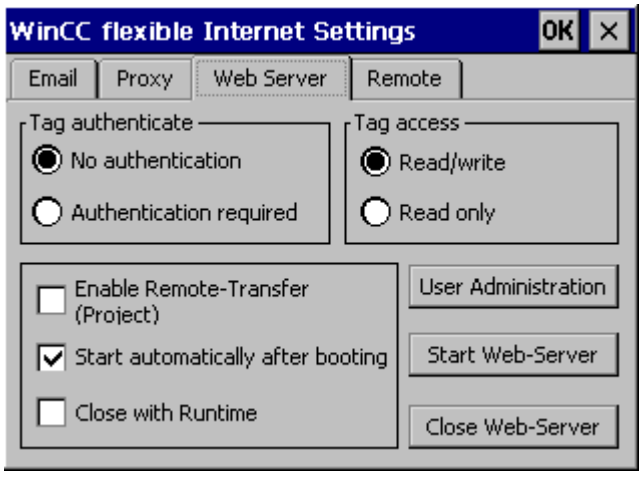
- Internet settings for HMI devices with Windows CE
  In this case, there are several options for calling the Control Panel:
  - In the startup phase, click on the "Control Panel" button in the Windows CE Loader menu.
  - Select "Start > Settings > Control Panel" in the toolbar.
  - During normal operation, pressing the key combination <CTRL + ESC> opens the toolbar. Start the Control Panel by selecting "Start > Settings > Control Panel".
  - Using the RT function "OpenControlPanel"

  The language in the Windows CE based devices is always English.

WinCC Internet Settings, Web Server tab

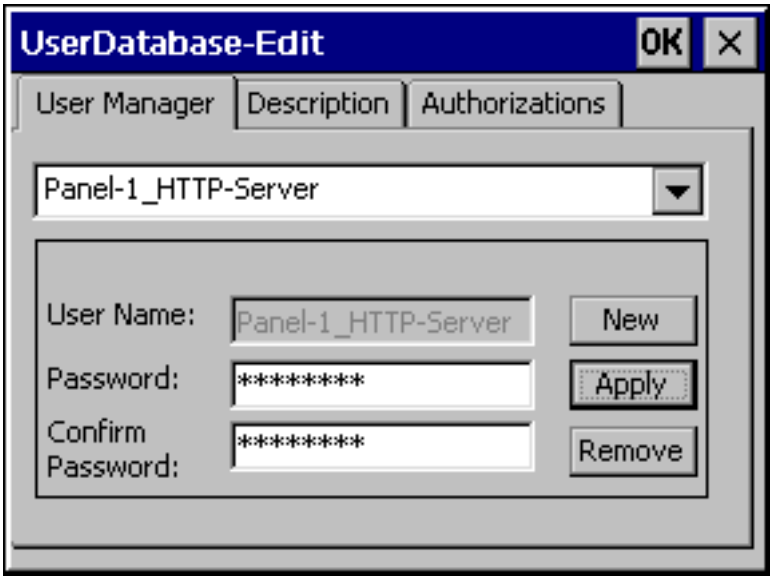
Settings for using the integrated Web Server are made in this tab.

The "Web Server" tab is only available in the WinCC Internet Settings if the Web Server is installed on the HMI device.



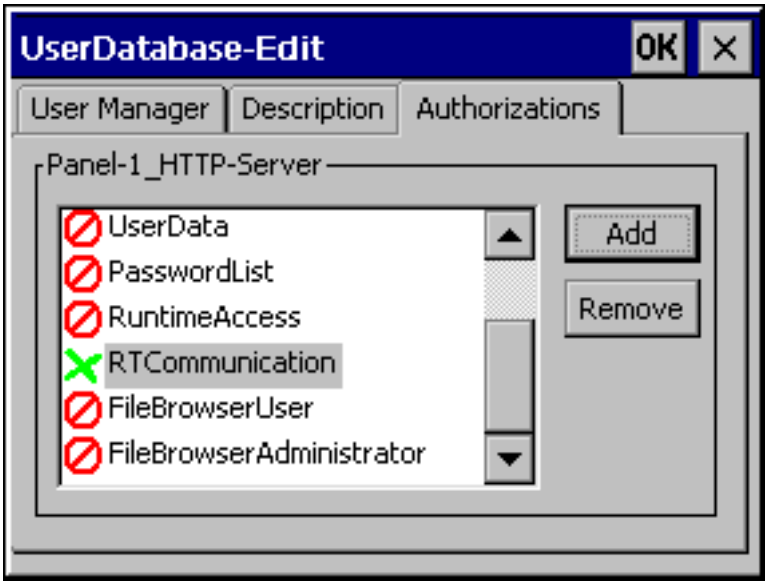
- Tag authenticate
  Controls authorization for tag access:
  - "No authentication": Authorization (user name password) is not required for access.
  - "Authentication required": Authorization (user name password) is required for access. The user name and password for the client are specified during configuration.
- Tag access
  Controls access to tags:
  - "Read/write": Tags can be read and rewritten.
  - "Read only": Tags can only be read.
- Group for defining operational performance
  The following check boxes are not selected in conjunction with tag exchange via the HTTP/ HTTPS protocol:
  - Enable remote transfer (project)
    Select this check box to enable an HTTP transfer from the configuration computer to the HMI device.
  - Start automatically after booting
    Specifies when the HTTP server should be started:
    Enabled: The HTTP server starts immediately after the HMI device is booted, irrespective of the runtime software
    Disabled: The HTTP server starts at the same time as the runtime software.
  - Close with Runtime
    Enabling this check box causes the HTTP server to be closed at the same time as the runtime software.

- User Administration
  This button opens the "UserDatabase-Edit" dialog box.



  In the "User Manager" tab of this dialog box, you can select, delete, or create a new user. The "Description" and "Authorizations" tabs always apply to the user currently selected. (This example dialog box shows a user with the name 'Panel-1_HTTP-Server'.)

- User Manager
  If a user name and password have been entered for the HTTP server under "Connections" in your WinCC project, the same user name and password must be specified on the server.
- Description
  Enables input of a user description.
- Authorizations
  The newly set up user must be assigned the authorizations for "RT Communication on the "Authorizations" tab control.



Establishing an HTTP connection

Introduction

To establish an HTTP connection, you must perform the following actions:

- In the "Connections" editor of WinCC, configure the connection as an "https://" protocol type and define how the HTTPS client should verify the properties of the server certificate and respond to errors.
- Install a valid certificate on the HTTPS client.
  Certificates are necessary for server authentication. Using certificates you can ensure that the server with which the connection is to be developed is actually the server for which it is outputting.

Principle of an HTTPS connection

After runtime start, the HTTPS client establishes a connection to the HTTPS server. The HTTPS server presents its certificate, which the client verifies for authenticity. The session code that can only be read by the HTTPS server is then transmitted. The session code is now available on both sides and enables a symmetrical data encryption.

Note

The certificate contains the current time. The current time can lead to problems if the time zones of the server and client are different. For example, a certificate generated on a server with an Asian time zone only becomes valid on a client with European time zone in the future (8 hours).

Preparation for installing a certificate on the client

With the first HTTPS client access, the HTTPS server generates the certificate itself and then saves it in the "Cert.cer" memory. The file is stored in the following directory:

- On a PC/panel PC (with Windows XP) in the directory "..\\Siemens\Automation\WinCC Runtime Advanced\SystemRoot\SSL"
- On Windows CE-based devices in the directory "Flash\Simatic\SystemRoot\SSL"
  The certificate must be stored on the HTTPS client on a storage medium from which it can be launched with a double click.

Server	Client	Possible file transfer
with Windows XP (PC, Panel PC)	with Windows XP (PC, Panel PC)	• Disk • USB stick • LAN (Ethernet) • Internet Explorer (via TCP/IP, if Service is already running)
With Windows CE (xP 270, xP 277, MP 370, MP 377, xP 177B, Mobile Panel 177 PN, Mobile Panel 277, Comfort Panels)	with Windows XP (PC, Panel PC)	• Memory card • ActiveSync (serial)
with Windows XP (PC, Panel PC)	With Windows CE (xP 270, xP 277, MP 370, MP 377, xP 177B, Mobile Panel 177 PN, Mobile Panel 277, Comfort Panels)	
With Windows CE (xP 270, xP 277, MP 370, MP 377, xP 177B, Mobile Panel 177 PN, Mobile Panel 277, Comfort Panels)	With Windows CE (xP 270, xP 277, MP 370, MP 377, xP 177B, Mobile Panel 177 PN, Mobile Panel 277, Comfort Panels)	• Memory card

Installing a certificate on a client with Windows XP

Insert the storage medium on which you have saved the "Cert.cer" file into the HTTPS client or open the directory in which the file is located. Double click on the file and follow the instructions in the Windows dialog.

Tip: The Internet Explorer provides an easy way to install a certificate. Connect to this device via HTTPS (e.g.: https://<my device>). The browser establishes if a certificate has not yet been imported. In this case, the browser asks if you want to install the certificate. Any errors in the certificate will be displayed.

Installing a certificate on a client with Windows CE

Insert the memory card on which you have saved the converted "Cert.cer" file into the HTTPS client. WinCC provides the "InstallCert.exe" tool for importing certificates with Windows CE.

You can implement the installation as follows:

- In Explorer:
  Double click the "Cert.cer" file to install the certificate.
- At the command prompt:
  Enter "InstallCert /[command parameter] [filename]".
  - command parameters:
    /r parameter must be specified because the certificate used in WinCC Runtime is a root certificate.
    A root certificate is the main certificate and is used to verify the authenticity of all other certificates transferred.
  - filename
    You must specify the certificate file with its complete path (e.g. "\Storage Card\Cert.cer")

A status alarm is output when you completed the installation. Runtime has to be restarted after the installation of a certificate on Windows CE- HMI devices with HTTPS clients. It is necessary to restart Runtime so that an HTTPS connection can be established.

The file "Cert.cer" cannot be opened.

The "Cert.cer" file generated at the HTTPS server cannot be opened on HMI devices based on Windows CE 5.0 by double-clicking the client.

1. Open the Control Panel.
2. Select "Certificates > My Certificates."
3. Click the "Import" button.
   A dialog box opens.
4. Select the "From a File" menu in the file browser and select the "Cert.cer" file.

12.11.14.3 Performance features of communication

Permitted data types

Permitted data types

When configuring tags, the data types listed below can be used.

Data type in the SIMATIC HMI HTTP protocol	Length	Sign	Range of values
Bool	0	No	true (-1) or false (0)
SInt	1 Byte	Yes	-128 to 127
USint	1 Byte	No	0 to 255
Int	2 Byte	Yes	-32768 to 32767
UInt	2 Byte	No	0 to 65535

Data type in the SIMATIC HMI HTTP protocol	Length	Sign	Range of values
DInt	4 Byte	Yes	-2,147,483,648 to 2,147,483,647
UDInt	4 Byte	No	0 to 4,294,967,295
Real	4 Byte	Yes	-3.402823E38 to -1.401298E-45 for negative values and 1.401298E-45 to 3.402823E38 for positive values
LReal	8 Byte	Yes	-1.79769313486231E308 to -4.94065645841247E-324 for negative values and 4.94065645841247E-324 to 1.79769313486232E308 for positive values
String	1 to 255 byte	——	
DateTime	8 Byte	——	1.1.1970 00:00:00 up to 31.12.2037 23:59:59

Note

Data types for third-party PLCs

Data types in third-party PLCs may differ from those in WinCC.

Note the the tag definition of the third-party PLC for the correct assignment.

Note

It is not possible to access array tags from an HTTP client.

Multiplex tags are not supported by the HTTP protocol. If you need a multiplex tag, you must configure it on the HTTP client. You will have to create the tags for the multiplex list on the server and the client and then connect them. You will then add these tags to the multiplex list on the client.

Address multiplex tags are not supported by the HTTP protocol.

12.11.15 Communication via OPC

12.11.15.1 Communication via OPC

Introduction

This section describes the communication between an HMI device and a device via OPC.

In this case, the HMI device is the OPC client and the device is the OPC server.

For more detailed information on OPC and on configuring OPC servers, refer to Section "OPC (Page 7155)".

12.11.15.2 Configuring a connection via OPC

Introduction

You configure a connection to a device via OPC in the "Connections" editor of the HMI device.

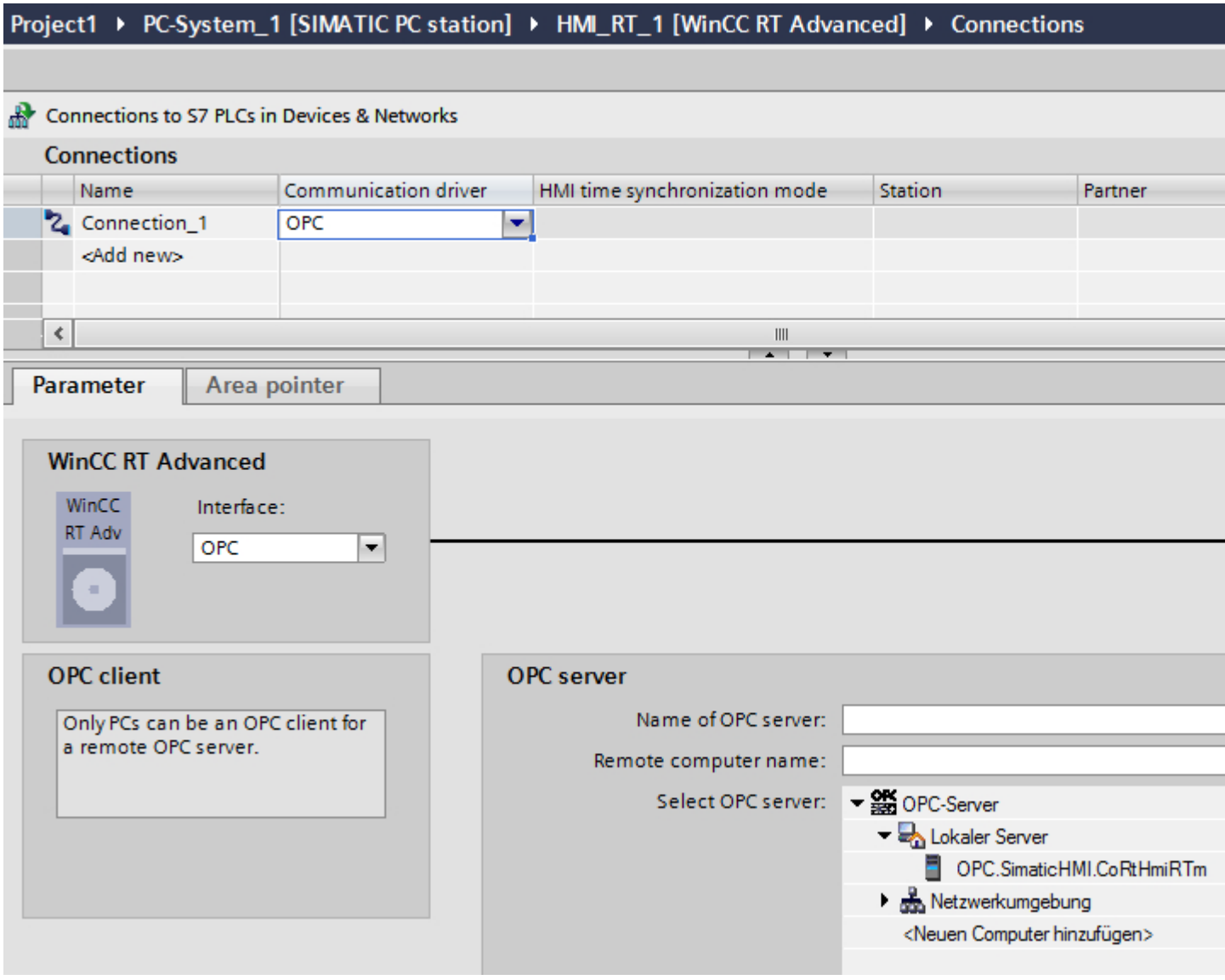
Requirements

- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.
4. In the "Communication drivers" column, select the "OPC" driver.

5. Select the "OPC" interface for the HMI device under "Parameters > Interface".



6. Select an OPC server from the tree view under "Parameters > OPC servers".
   The OPC server is entered in the "Name of OPC server" field.

12.11.15.3 Performance features of communication

Permitted data types for OPC DA

Permitted data types for connections with OPC DA

The table lists the user data types that can be used when configuring tags.

Data type	Length
VT_BOOL	1-bit
VT_I1	1 byte
VT_UI1	1 bytes
VT_I2	2 bytes
VT_UI2	2 bytes
VT_I4	4 bytes
VT_UI4	4 bytes
VT_R4	4 bytes
VT_R8	8 bytes
VT_BSTR	--
VT_DATE	8 bytes
VT_DATE|VT_ARRAY	--
VT_I1|VT_ARRAY	--
VT_UI1|VT_ARRAY	--
VT_I2|VT_ARRAY	--
VT_UI2|VT_ARRAY	--
VT_I4|VT_ARRAY	--
VT_UI4|VT_ARRAY	--
VT_R4|VT_ARRAY	--
VT_R8|VT_ARRAY	--

Permitted data types for OPC XML DA

Permitted data types for connections with OPC DA XML

The table lists the user data types that can be used when configuring tags.

Data type	Length
boolean	1-bit
byte	1 byte
unsignedByte	1 byte

Data type	Length
short	2 bytes
unsignedShort	2 bytes
int	4 bytes
unsignedInt	4 bytes
float	4 bytes
double	8 bytes
string	--
dateTime	--
time	--
date	--

Permitted data types for OPC UA

Permitted data types for connections with OPC UA

The table lists the user data types that can be used when configuring tags.

Data type	Length
Boolean	1-bit
SByte	1 byte
Byte	1 byte
Int16	2 bytes
UInt16	2 bytes
Int32	4 bytes
UInt32	4 bytes
Float	4 bytes
Double	8 bytes
String	--
DateTime	--

12.11.16 Communication via routing

12.11.16.1 Communication via routing

Introduction

If all stations in an automation system are not connected to the same subnet, these stations cannot be accessed directly online.

A connection between partners in different subnets is only possible via routing.

You configure the routing settings with the properties of the interfaces.

Communication from connection partners in various different subnets can be routed via the following links:

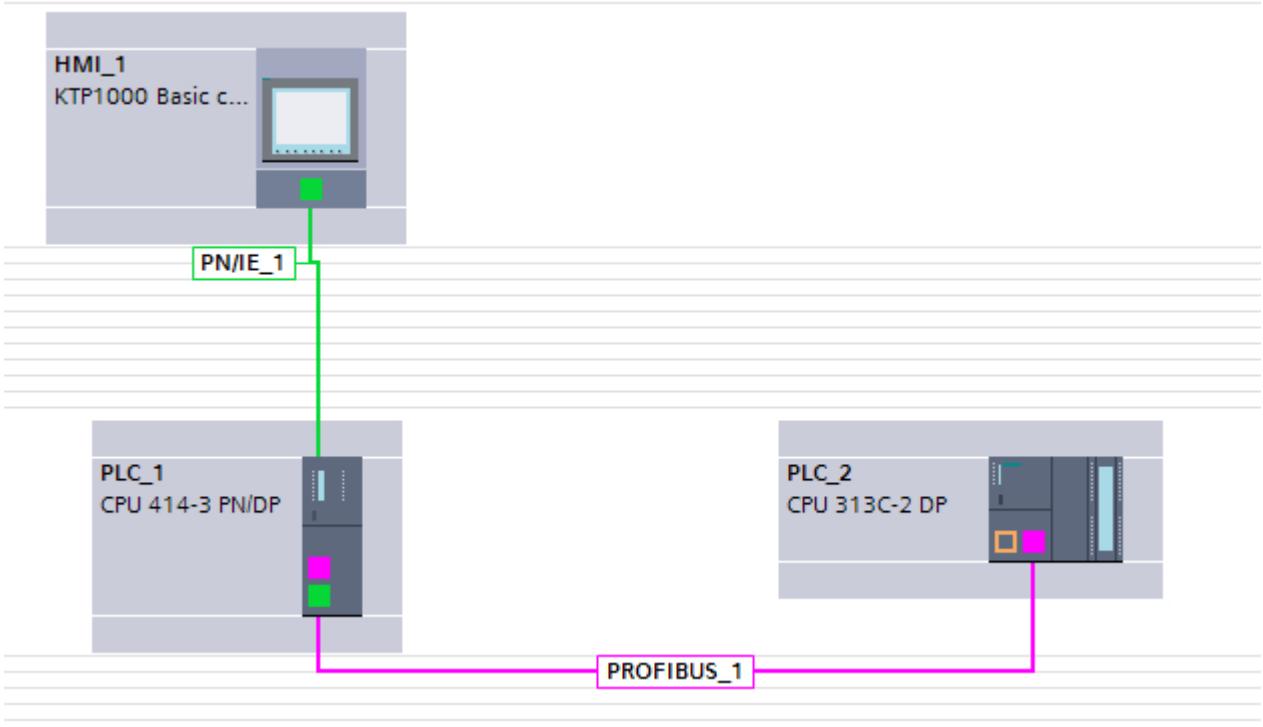
- PROFINET
- PROFIBUS
- MPI

Requirements for routing

To establish a connection to these devices, a router must be interposed. In this case, a SIMATIC station can also act as a router if it has appropriate interfaces with the different subnets.

Modules with communication capability (CPUs or CPs) used to establish gateways between the subnetworks must have routing capability.

A connection between partners in different subnets is only possible with IP routing. The routing settings can be edited in the corresponding interface properties.



Routing path

The routing path is determined in Runtime by the system and cannot be influenced by the user. During configuration, it is not possible to generate information about a faulty connection.

HMI devices as routers

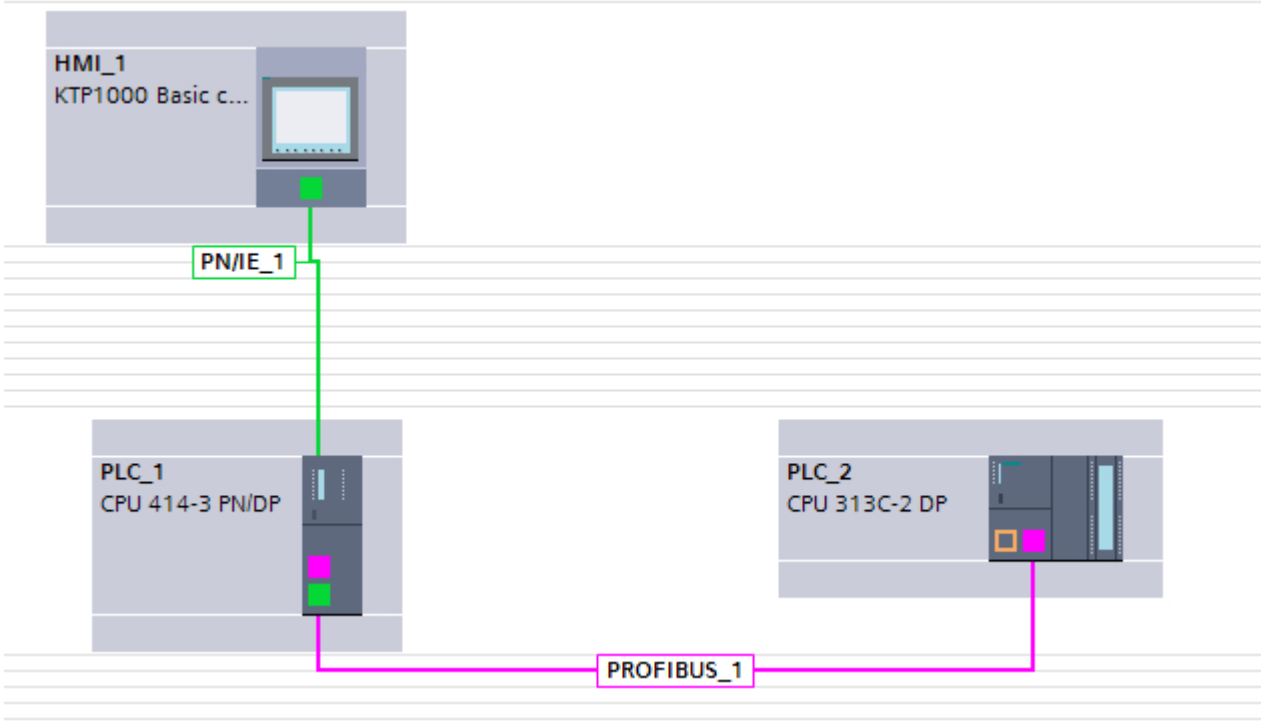
Only SIMATIC PCs or PC stations can be used as a router.

All other HMI devices from the "HMI" area cannot be used as a router.

12.11.16.2 Example for communication via routing

Communication via routing

Illustration of a hardware configuration with a routing connection

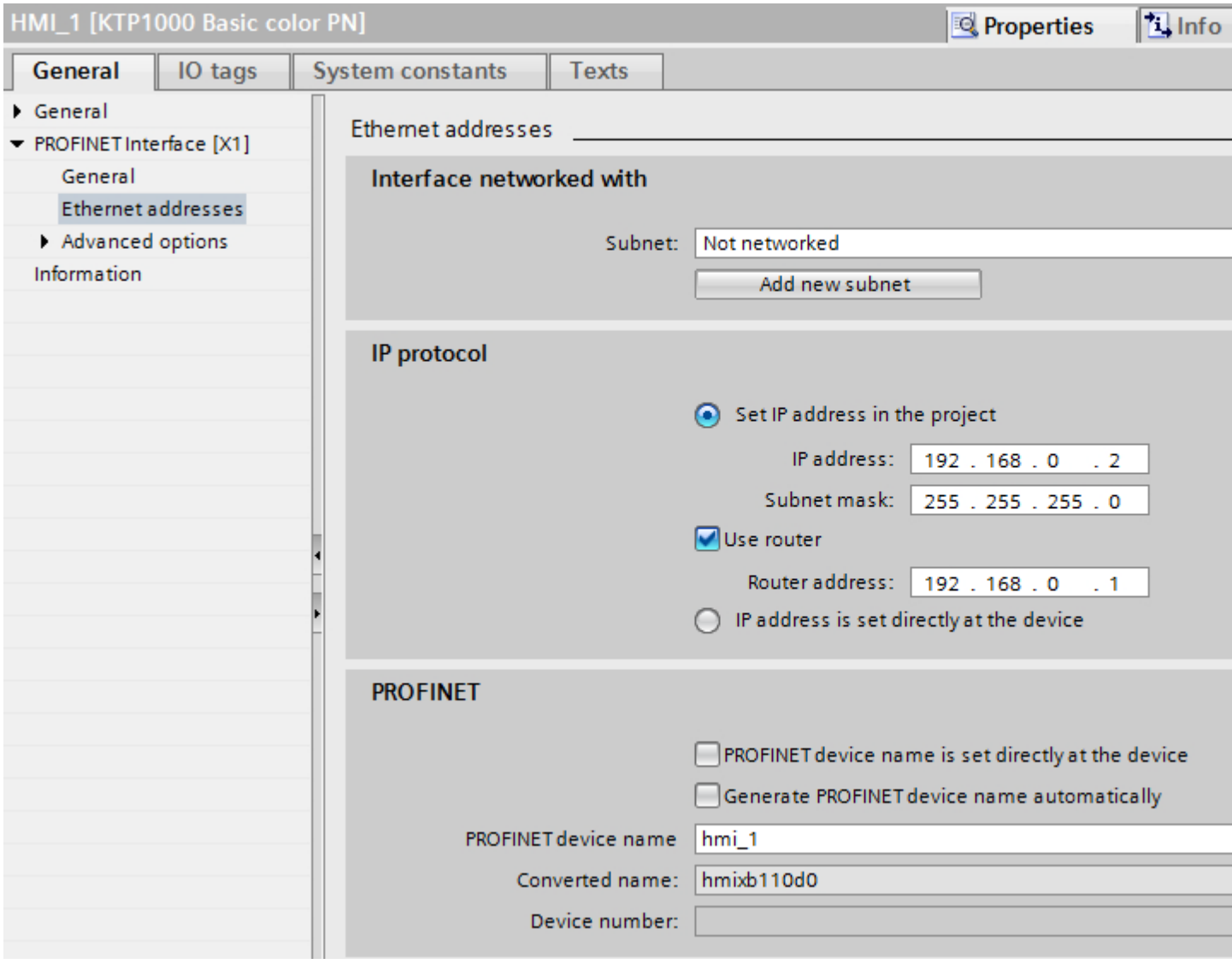


In the figure above, a routing connection has been established between an HMI device and the SIMATIC S7-300 controller.

The SIMATIC S7-400 automation device serves as a router.

A routing connection can be made directly in integrated projects. The communication partners are connected in the "Devices and Networks" editor.

You configure the parameters of the connection and interface in the Inspector window.

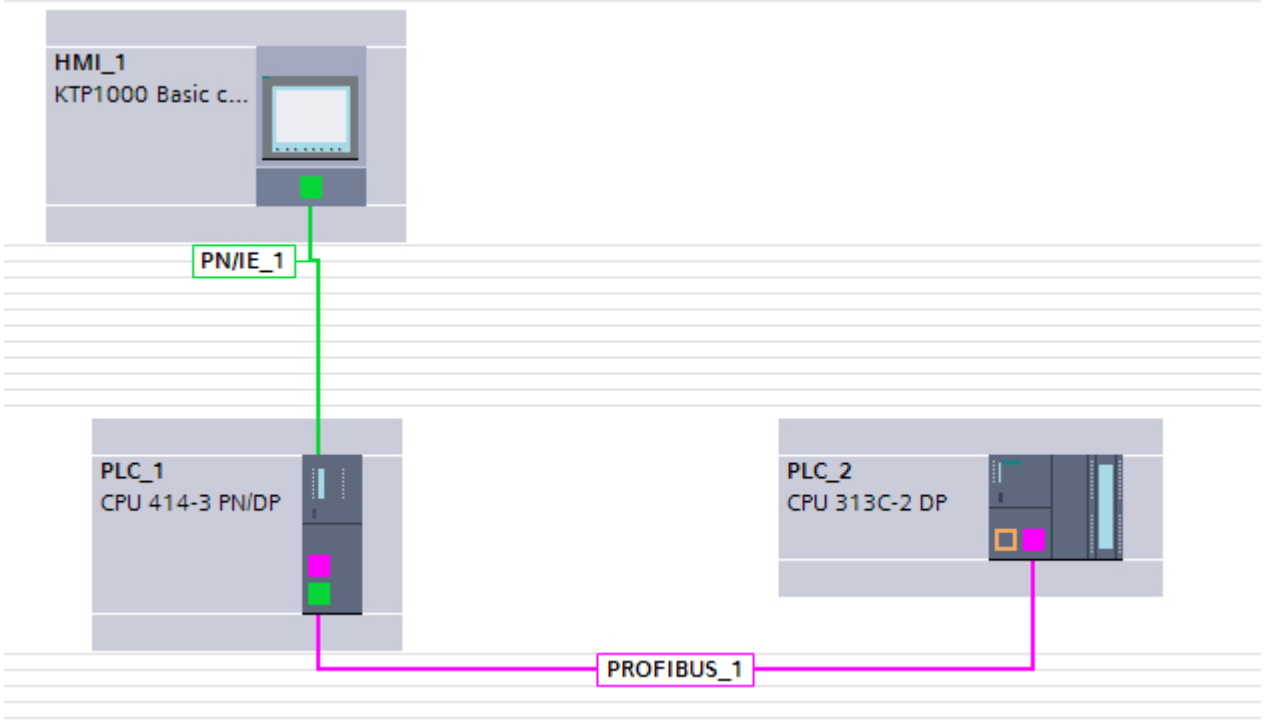

HMI_1 [KTP1000 Basic color PN]
Properties
Info
General
IO tags
System constants
Texts
General
PROFINET Interface [X1]
General
Ethernet addresses
Advanced options
Information
Ethernet addresses
Interface networked with
Subnet:
Not networked
Add new subnet
IP protocol
Set IP address in the project
IP address:
192 . 168 . 0 . 2
Subnet mask:
255 . 255 . 255 . 0
Use router
Router address:
192 . 168 . 0 . 1
IP address is set directly at the device
PROFINET
PROFINET device name is set directly at the device
Generate PROFINET device name automatically
PROFINET device name
hmi_1
Converted name:
hmixb110d0
Device number:

12.11.16.3 Configuring communication via routing

Introduction

The following configuration describes a network for the following communication partners:

- KTP1000 Basic PN
- SIMATIC S7-400 with PROFINET and PROFIBUS interface
- SIMATIC S7-300 with PROFIBUS interface



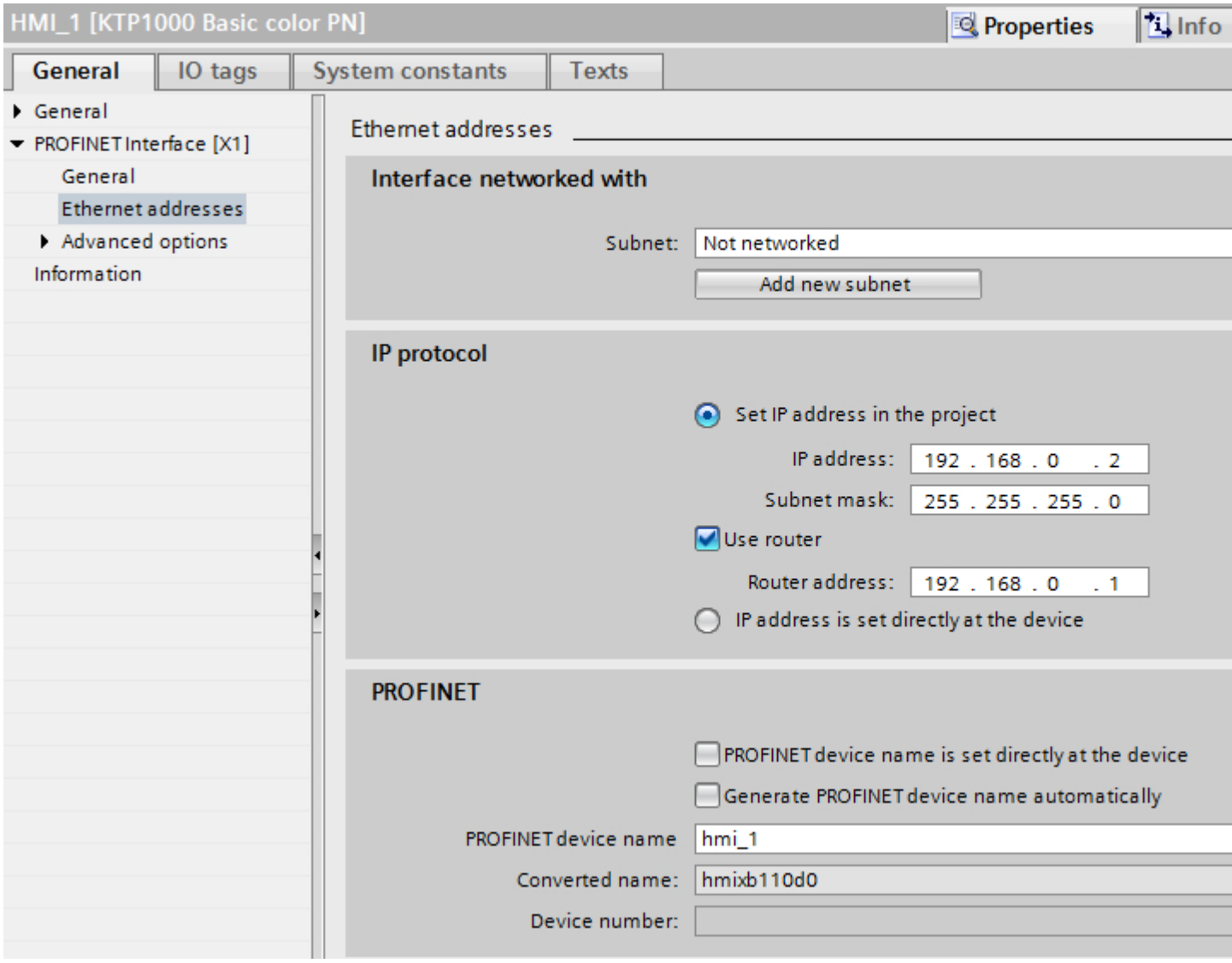
Requirements

- KTP1000 Basic PN is connected via PROFINET to the SIMATIC S7-400.
- SIMATIC S7-400 is connected via PROFIBUS to the SIMATIC S7-300.

Procedure

1. Click on the PROFINET interface of the HMI device.
2. In the Inspector window, click "Properties > General > Ethernet addresses".

3. Click "Use router" in the "IP protocol" area.
4. Enter the IP address of the router in the "Router address" field.



12.11.17 PROFINET IO and IRT

12.11.17.1 PROFINET IO

What is PROFINET IO?

PROFINET IO

PROFINET is an Ethernet-based automation standard of PROFIBUS Nutzerorganisation e.V. (PNO) which defines a manufacturer-neutral communication, automation and engineering model.

Objective

The objective of PROFINET is:

- Integrated communication via field bus and Ethernet
- Open, distributed automation
- Use of open standards

Architecture

The PROFIBUS User Organisation e.V. (PNO) has designated the following aspects for PROFINET architecture:

- Communication between controllers as components within distributed systems.
- Communication between field devices, such as I/O devices and drives.

Implementation by Siemens

The demand for "Communication between controllers as components within distributed systems" is implemented by "Component Based Automation" (CBA). Component Based Automation is used to create a distributed automation solution based on prefabricated components and partial solutions.

The demand for "Communication between field devices" is implemented by Siemens with "PROFINET IO". Just as with PROFIBUS DP, the complete configuration and programming of the components involved is possible using the Totally Integrated Automation Portal.

The following sections deal with the configuration of communication between field devices using PROFINET IO.

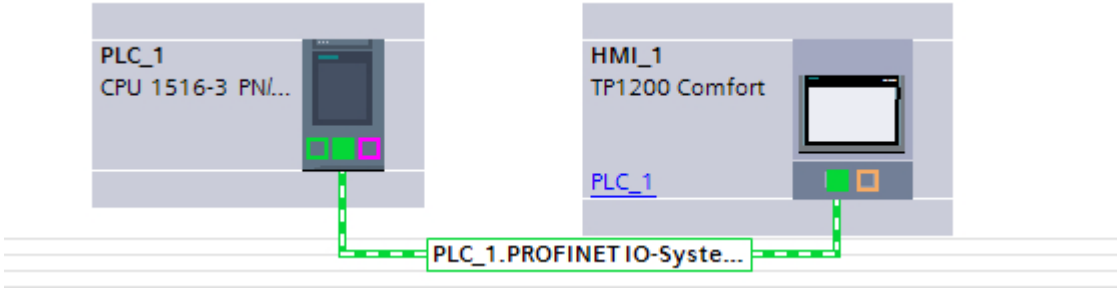
PROFINET IO systems

PROFINET IO system

A PROFINET IO system is comprised of a PROFINET IO controller and its assigned PROFINET IO devices.

- A PROFINET IO controller is a PLC that controls the automation task.
  Example: SIMATIC S7 1500 PLC
- A PROFINET IO device is a device that is controlled by an IO controller.
  Examples: HMI device TP 1200 Comfort, a head module of the distributed IO family ET 200S

As soon as you connect an IO controller to an IO device, a controller-device link is established.



12.11.17.2 IRT communication

Overview of RT classes

RT classes in PROFINET IO

PROFINET IO is a scalable, real-time communication system based on Ethernet technology. The scalable approach is reflected in several real-time classes:

- **RT:** Transmission of data in prioritized, non-isochronous Ethernet frames. The required bandwidth is within the free bandwidth area for TCP/IP communication.
- **IRT:** Isochronous transmission of data with high stability for time-critical applications (for example, motion control). The required bandwidth is from the area of bandwidth reserved for cyclic data.

Device dependency

Depending on the device, not all real-time classes are supported.

You can find further information on device dependency here: Performance characteristics of PROFINET IO and IRT (Page 6592)

Introduction: Isochronous Realtime Ethernet

Introduction: Isochronous Realtime Ethernet

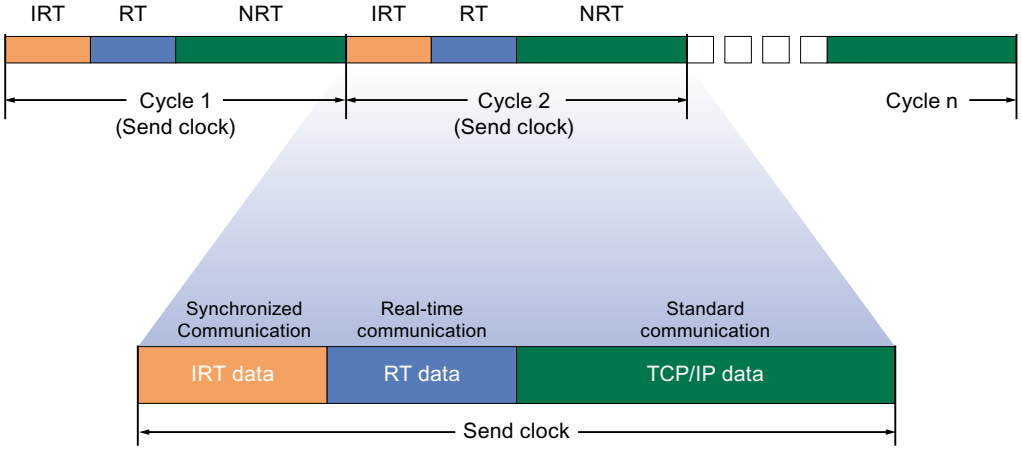
IRT is a transmission method by which PROFINET devices are synchronized with very high accuracy.

A sync master specifies the clock to which sync slaves are synchronized. An IO controller or an IO device can have the role of sync master.

Sync master and sync slaves are always devices in a sync domain. Bandwidth is reserved within the sync domain for IRT communication. Real-time and non-real-time communication (TCP/IP communication) is possible outside of the reserved bandwidth.

Time ranges of communication cycles

The communication cycle is divided into three time ranges, which are represented in the following chart:



- IRT data (synchronized communication)
  This area can be reserved in specific steps, depending on the send clock. Within this time range the IRT data is transmitted.
- RT data (real-time communication)
  In this time range the cyclic RT date is transmitted. RT data has a higher priority than "normal" TCP/IP data. TCP/IP data or Ethernet frames can have a priority between 1 and 7. RT data has a priority of 6.
- TCP/IP data (standard communication)
  Standard communication (TCP/IP, etc.) is transmitted in the remaining interval of the communication cycle.

Applications of IRT

PROFINET with IRT is especially suited for:

- High performance and deterministics for large quantity structures with regard to user data communications (productive data)
- High performance even with many devices in the line topology with regard to user data communications (productive data).
- Parallel transmission of production and TCP/IP data over wire, and securing the forwarding of production data when data volume is high by reserving transmission bandwidth.

Basic procedures for configuring IRT

Basic procedures for configuring IRT

If you want to upgrade equipment with PROFINET IO to IRT, follow these three steps:

1. Configure the PROFINET IO device. The PROFINET device must support IRT.
2. Set which device acts as the sync master and synchronizes the other devices. For this you must configure a sync domain with a sync master and at least one sync slave.
3. Download the configuration to all involved devices.

You can find further information on IRT configuration here: Auto-Hotspot

Creating PROFINET IO configuration for IRT

As a prerequisite for IRT configuration, there must be an existing PROFINET IO configuration, i.e., one or more stations must be configured with an IO controller and IO devices.

You can read which components are suitable for IRT communication in the task card "Information", once you have selected the corresponding components in the hardware catalog.

The requirements for functioning IRT communication are met simply by configuring a PROFINET IO system with components that support IRT.

Sync domain

Sync domain

A sync domain is required for synchronization of PROFINET IO devices. The sync domain assures that all devices belonging to it communicate synchronously.

Prerequisite for IRT communication is a synchronization cycle to distribute a common time base for all PROFINET devices in a sync domain. With this base synchronization, a synchronous transmission cycle of PROFINET devices within a sync domain is achieved. The sync master (usually an IO controller) generates the common synchronization clock and specifies the time basis on which all other sync slaves (usually IO devices) are synchronized.

If the sync master fails, the communication between the IRT devices reverts to RT communication.

Background processes during configuration of an IO system

If you configure an IO controller and network it with an Ethernet subnet, it will automatically be added to the default sync domain of the Ethernet subnet. The default sync domain is always available. The IO controller operates unsynchronized.

If you assign an additional IO device to the IO system of the IO controller, the IO device automatically assigns the IO controller's sync domain.

12.11.17.3 Configuring the HMI device as IO device

Introduction

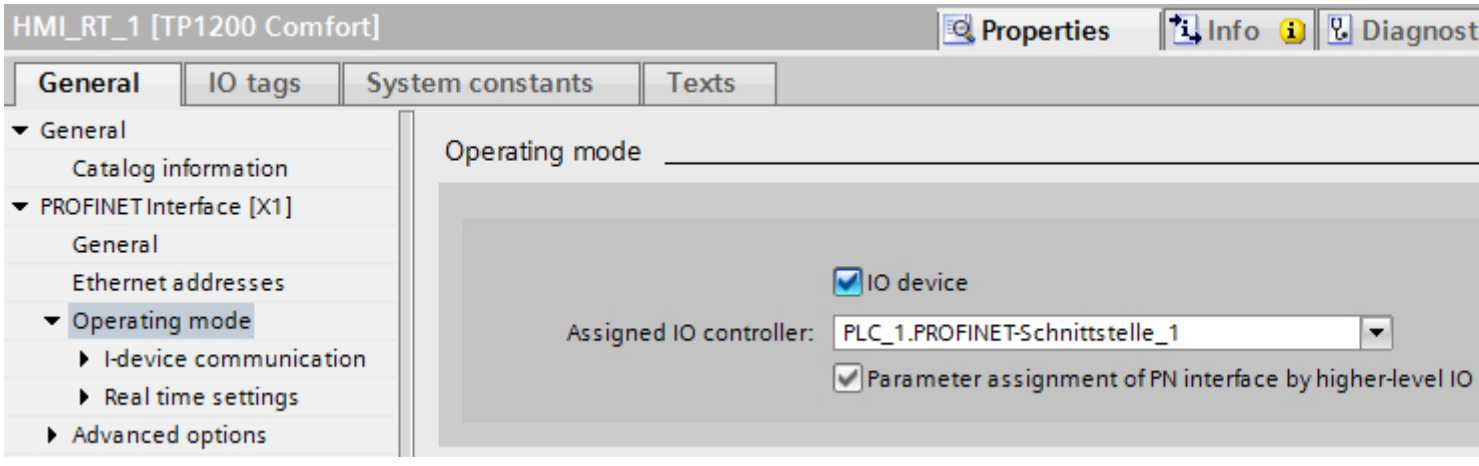
An IO controller and at least one PROFINET IO device are necessary for a PROFINET IO system.

Requirements

- A SIMATIC S7 1200 or SIMATIC S7 1500 PLC has been created.
- An HMI device from the Comfort Panels device family has been created.
- The "Devices & Networks" editor is open.

Procedure

1. You network the PLC with the HMI device:
2. Click the HMI device in the "Devices & Networks" editor.
3. In the inspector window, you select the following:
   "General > PROFINET Interface (X1) > Operating mode"
4. Select the "IO device".



5. You assign the parameters required for your PROFINET IO system
   You can find further information on the parameters for PROFINET IO systems here:
   Parameters for PROFINET IO and IRT (Page 6589)

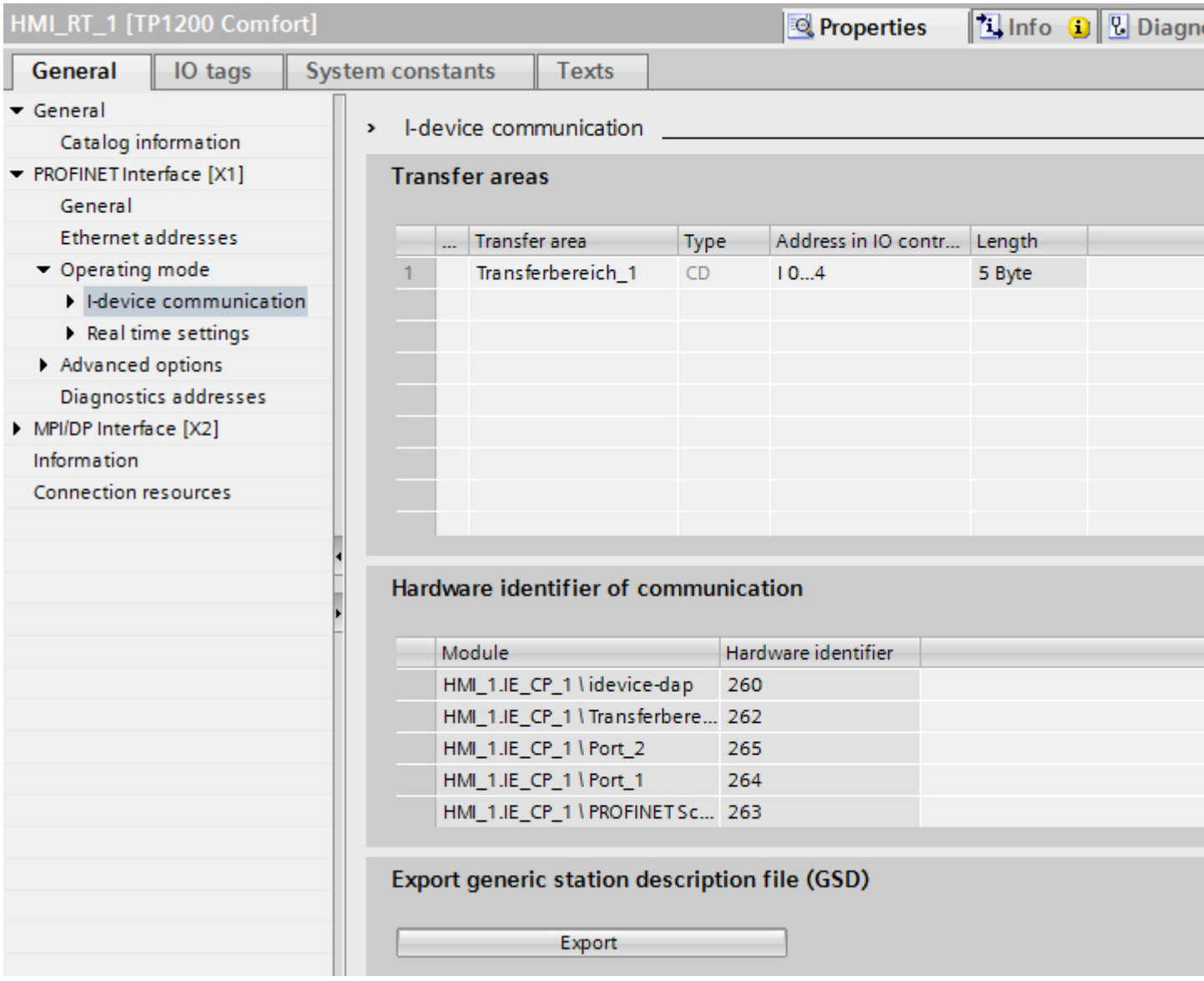
12.11.17.4 Parameters for PROFINET IO and IRT

I-device communication

I-device

The "I-device" functionality (intelligent IO device) of a CPU allows data to be exchanged with an IO controller and therefore to use the CPU, for example, as an intelligent preprocessing unit of subprocesses. The I-device is linked as an IO device to a "higher-level" IO controller.

Parameters for I-device communication



- Transfer areas
  Display transfer areas of the I-device communication
- Diagnostics address of communication
  Display of diagnostic data of the IO device

Configuring an I-device with a GSD file

If you use an I-device in another project, or if the I-device is used in another engineering system, then configure the higher-level IO controller and the I-device as described above.

However, click on the "Export" button after configuring the transfer areas in order to create a GSD file from the I-device. This GSD file represents the configured I-device in other projects.

The "Export" button is found in the "I-device communication" section of the inspector window.

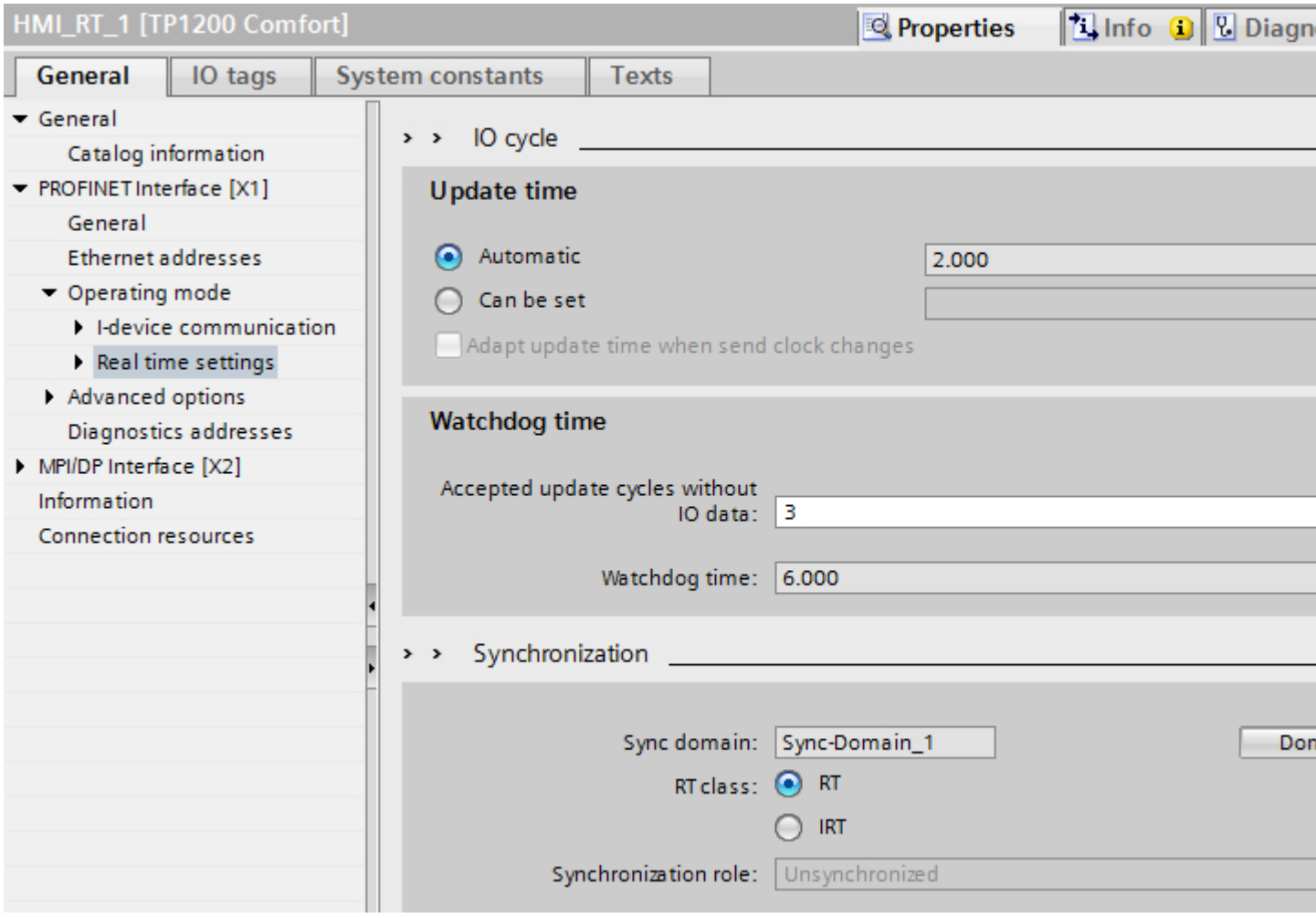
The hardware configuration is compiled and the export dialog opened.

Assign a name for the I-device proxy as well as a description in the fields provided. Click the "Export" button to complete your process.

Finally, import the GSD file, for example, into another project.

Real-time settings

Parameters for real-time settings



- Update time
  - Display of update time used in network.
  - Selection of update times available in network.
- Watchdog time
  - Selection of available update cycles.
  - Display of watchdog time used in network.
- Synchronization
  - Display of sync domain used
  - Selection of RT class.
  - Selection of role in the sync domain.

12.11.17.5 Performance characteristics of PROFINET IO and IRT

Configuration of PROFINET and IRT

Setting up PROFINET with IRT

Keep in mind the following rules for the set up and operation of a PROFINET IO system in IRT operation. They serve to insure an optimal operation of your PROFINET IO system.

- When using IRT, you must configure the topology. This will enable exact calculation of the update time, bandwidth, and additional parameters.
- If you would like to use multiple sync domains, configure a sync domain boundary for the port which is currently connected to the PROFINET device of another sync domain.
- In a sync domain you can only configure one sync master at a time.
- A PROFINET IO system may only be part of a single sync domain.
- If you configure a PROFINET device in a sync domain and want to synchronize with IRT, the PROFINET devices concerned must support IRT communication.
- Wherever possible, use the same PROFINET device as the PROFINET IO controller and sync master.
- If only some of the PROFINET devices in a PROFINET IO system are synchronized, please keep in mind the following: PROFINET devices which are not part of IRT communication are placed outside of the sync domain.

Device dependency

IRT communication

The following HMI devices support IRT communication:

- TP700 Comfort and KP700 Comfort
- TP900 Comfort and KP900 Comfort
- TP1200 Comfort and KP1200 Comfort
- TP1500 Comfort and KP1500 Comfort
- TP1900 Comfort
- TP2200 Comfort

The following connection boxes for Mobile Panels support IRT communication:

- Connection Box Advanced

Communication options

The following configuration options are available for Comfort Panels:

- Comfort Panel with PROFINET Basis functionality without PROFINET IO
- Comfort Panel with PROFINET RT in an RT network networked with an RT controller
- Comfort Panel with PROFINET RT as end point in an IRT network networked with an RT controller
- Comfort Panel with PROFINET IRT in an IRT network networked with an IRT controller

The following configuration options are available for Mobile Panels:

- Mobile Panel with PROFINET Basis functionality without PROFINET IO
- Mobile Panel with PROFINET IO as IO device in a PROFINET network connected to an IO controller
- Mobile Panel with PROFINET basic funtionality or with PROFINET IO as the terminal point in a IRT network (via Connection Box Advanced)

12.11.18 Media redundancy

12.11.18.1 Restrictions with media redundancy

Diagnostic interrupts

If you have activated "Diagnostic interrupts", diagnostic interrupts pertaining to "Media redundancy" are only processed by PROFINET-compatible devices and displayed on the "IO Controller".

If you do not use a PROFINET-compatible device, the "Diagnostic interrupts" selection does not have any function.

Diagnostic interrupts are only detected by the IO controller. There is no separate diagnostic interrupts for HMI devices that are detected and displayed by HMI devices.

12.11.18.2 Media redundancy

Media redundancy options

To increase the network availability of an industrial Ethernet network with optical or electrical line topology, the following options are available:

- Meshing networks
- Parallel switching of transmission modes
- Consolidation of a line topology to a ring topology

Media redundancy in ring topologies

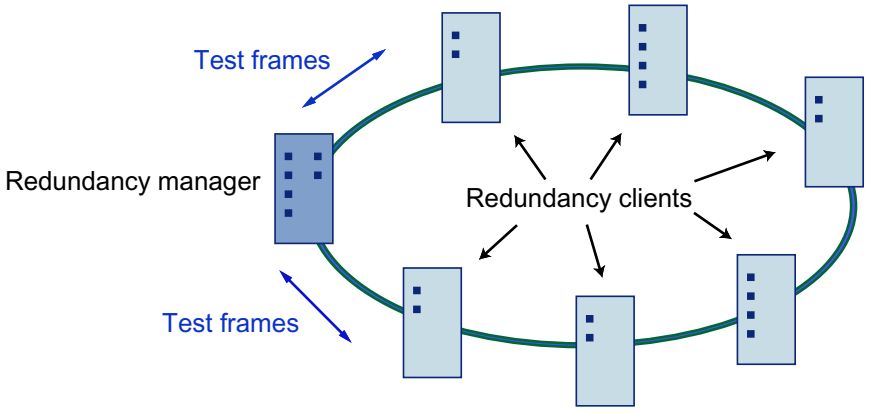
Devices in ring topologies can be the external switches and/or integrated switches of communication modules.

To set up a ring topology with media redundancy you must join both free ends of a line-shaped network topology to one device.

Consolidation of a line topology to a ring takes place over 2 ports (ring ports) in a device in the ring.

This device is the redundancy manager. All other devices in the ring are the redundancy clients.

The following figure shows the ring type interconnection:



The two ring ports in a device are the ports which create the connection to both of its neighboring devices in a ring topology.

The selection and set up of the ring ports takes place during configuration of the respective device.

Download the configuration in the individual devices before physically connecting the ring.

Functions of media redundancy in a ring topology

If a part of the ring is disconnected, then the data paths between the individual devices is automatically reconfigured. After the reconfiguration the devices are once again accessible in the newly created topology.

In the redundancy manager, 2 ring ports will be separated from each other during interruption-free network operation, so that no data frames circulate.

The ring topology is a line when viewed from the standpoint of data transfer. The redundancy manager monitors the ring topology. To do so it sends a test frame not only from ring port 1 but also from ring port 2. The test frame passes through the ring in both directions until it arrives at the other ring port on the redundancy manager.

An interruption in the ring can occur from connection failure between two devices or device failure in the ring.

If the redundancy manager's test telegram does not reach the other ring port because of interruption in the ring, the redundancy manager connects via both ports.

With this alternate route a functioning connection is again created among all remaining devices in a line-shaped topology.

The time between ring interruption and re-establishment of a functional line topology is called the reconfiguration time.

As soon as the interruption is eliminated, the original transmission route is again created, both ring ports in the redundancy manager are separated from each other, and the redundancy clients are notified of the change.

The redundancy clients then use the new route to the other devices.

If the redundancy manager fails, then the ring is changed to a functional line.

12.11.18.3 Media Redundancy Protocol (MRP)

Media Redundancy Protocol (MRP)

The MRP process works in conformity with Media Redundancy Protocol (MRP), which is specified in IEC 61158 Type 10 "PROFINET".

The reconfiguration time following an interruption of the ring amounts to a maximum of 0.2 seconds.

Requirements for trouble-free operation

Requirements for trouble-free operation with media redundancy process MRP are:

- MRP is supported in ring topologies of up to 50 devices. Exceeding the device limit can lead to data traffic failure.
- All devices must be connected to each other via their ring ports.
- "MRP" must be activated on all devices in the ring.
- The connection settings (transfer medium / duplex) must be set to full duplex and for at least 100 Mbit/s for all ring ports. Otherwise data traffic failure can occur.
- In STEP 7 configuration in the properties dialog for all ports in the ring, set the connection in the register "Options" to "Automatic settings".

HMI devices for media redundancy

The following HMI devices support "Media Redundancy":

- KP700 Comfort
- TP700 Comfort
- KP900 Comfort
- TP900 Comfort
- KP1200 Comfort
- TP1200 Comfort
- KP1500 Comfort

- TP1500 Comfort
- TP1900 Comfort
- TP2200 Comfort

Connection boxes for Mobile Panels for media redundancy

The following connection boxes for Mobile Panels support "media redundancy":

- Connection Box Standard
- Connection Box Advanced

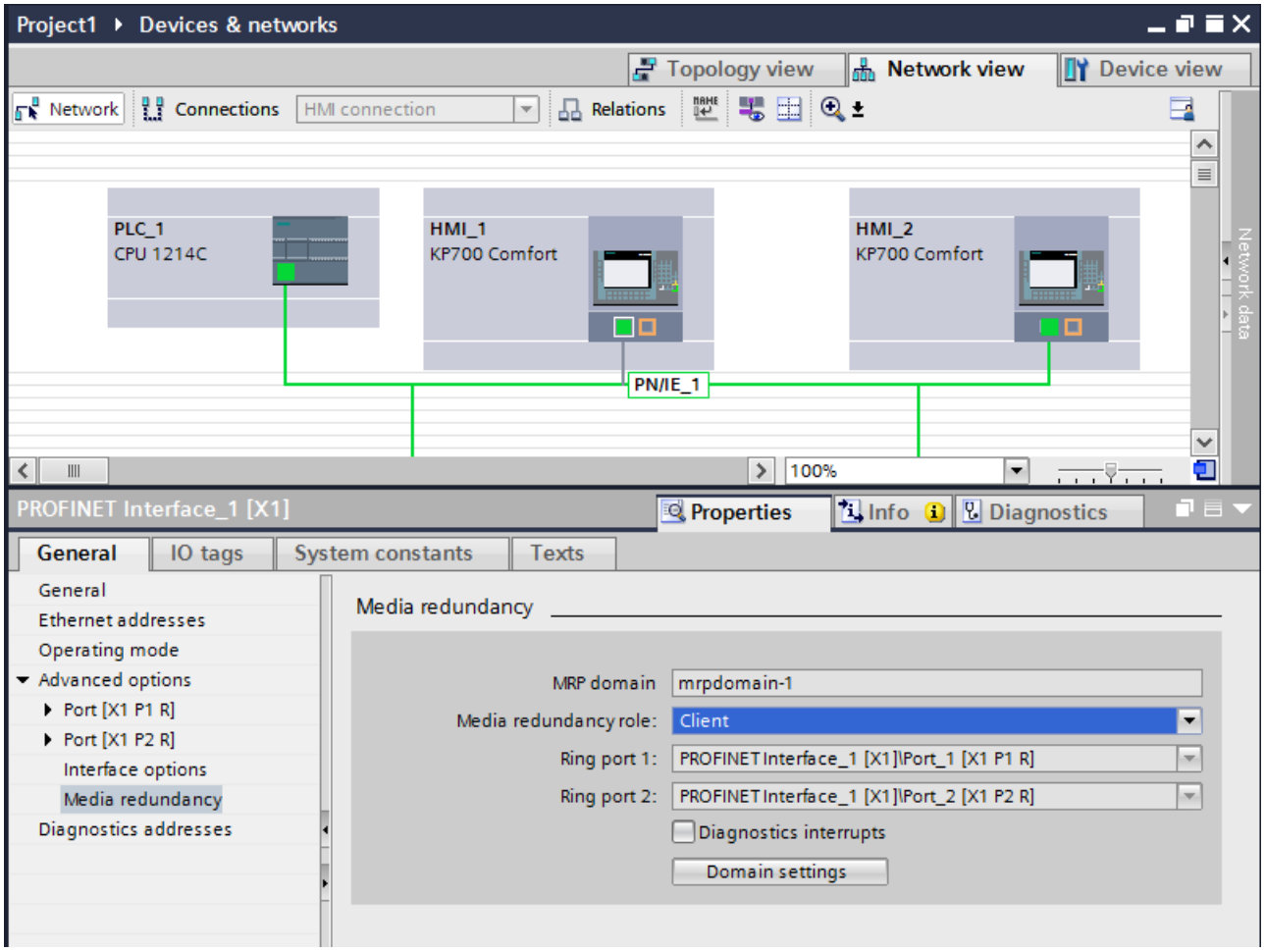
12.11.18.4 Configuring media redundancy for HMI devices

Requirements

- The participating components must support "media redundancy".
- IRT communication is not configured
- One device on the network must be configured as "Redundancy manager".
- A ring topology is set up by means of port interconnections.

Procedure

1. Click on the PROFINET interface of an HMI device.
2. Select "General > Advanced settings > Media redundancy" in the properties.
3. Select "Client" in the "Media redundancy role" area.

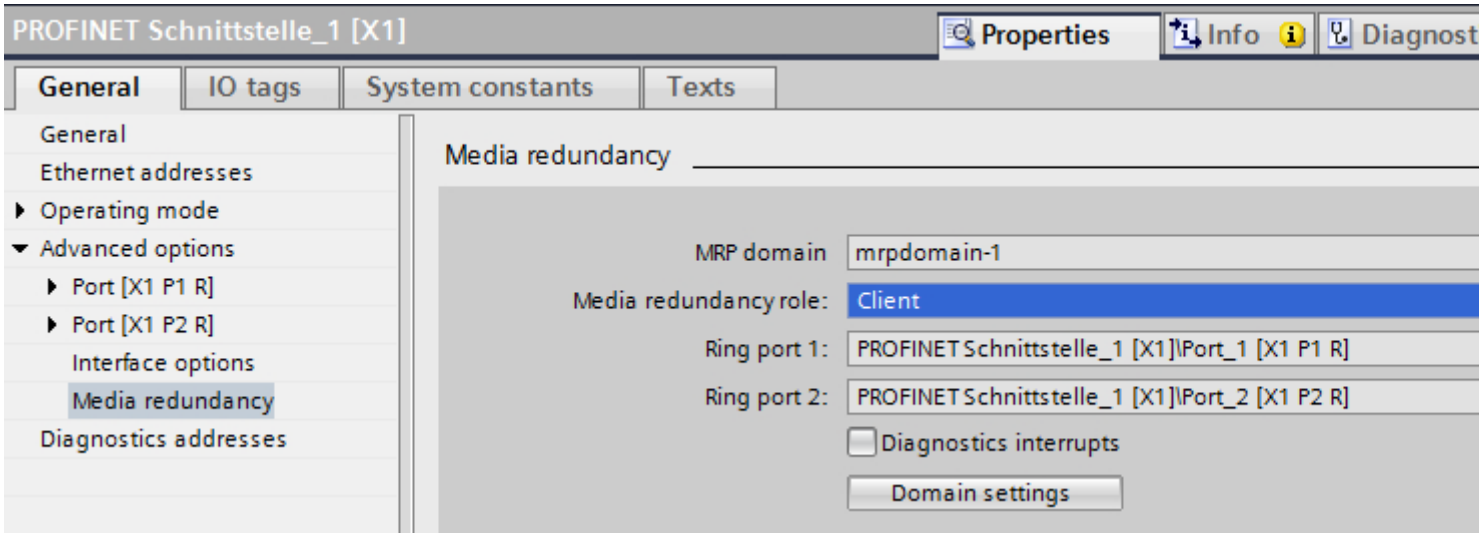


Result

The HMI device is now operating as client on the MRP network.

The default ring ports will automatically be displayed in the subjacent fields.

12.11.18.5 Parameters for media redundancy



MRP domain

This area displays the MRP domain.

You can rename the MRP domain in the "Domain settings" dialog.

Domain settings

Click "Domain settings" to open the "Domain settings" editor.

You also manage the MRP domains in the MRP domain area of the "Domain settings" editor.

Media redundancy role

Assign the role of the HMI device in the "Media redundancy role" area.

You have the following options:

- Client
  The HIM device is operated as "Client" in an MRP domain.
- Not a node of the ring
  The HIM device is not operated within an MRP domain.

Ring port 1 and Ring port 2

Display of available ring ports

Diagnostic interrupts

The following diagnostic interrupts can be generated if "Diagnostic interrupts" is activated:

- Wiring or port error
  Diagnostics interrupts will be generated for the following errors in the ring ports:
  - A neighbor of the ring port does not support MRP
  - A ring port is connected to a non-ring port

12.11.18.6 Managing MRP domains

You can monitor and adjust the media redundancy settings centrally.

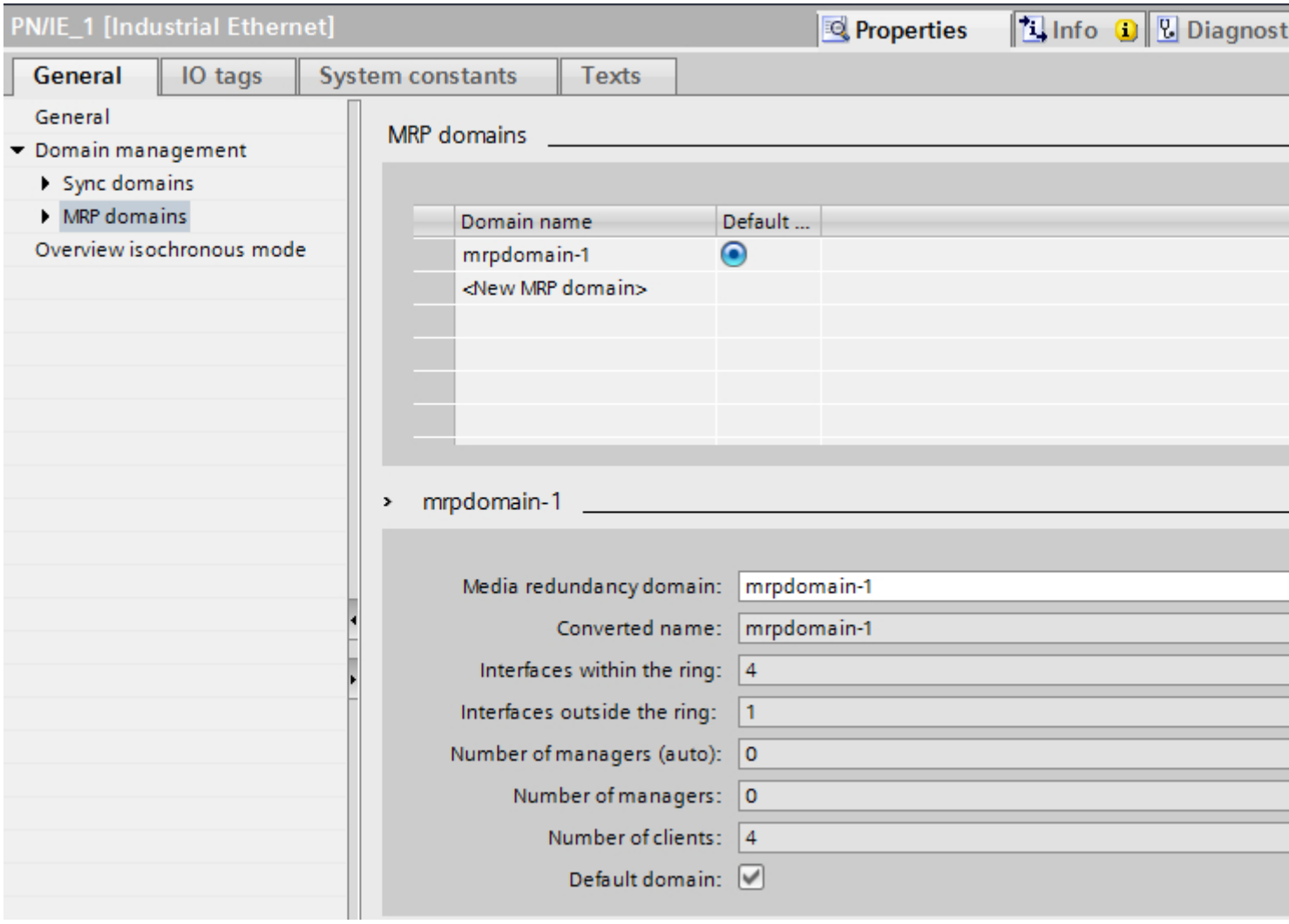
Requirements

The devices are configured and networked.

Procedure

1. Select the PROFINET interface of the HMI device.
2. Select the "Advanced options > Media redundancy" area in the Inspector window.

3. Click the "Domain settings" button.
4. Navigate to the "Domain management > MRP domains" area in the Inspector window.



MRP domains

You can specify the default MRP domain in the "MRP domains" area.

You can rename all domains and check the redundancy roles and ring ports.

For each MRP domain, you receive an overview of how many PROFINET interfaces are interconnected in the ring, how many PROFINET interfaces are located outside the ring, and the number of redundancy managers and redundancy clients.

You can have the devices in the ring, including device name, associated MRP domain, media redundancy role, and utilized ring ports, displayed in groups:

- IO systems
  The list shows you all configured devices of the selected IO system.
- Rings
  The list shows you all ring interconnections of the selected IO system. The identified rings are numbered consecutively. If all devices of this ring belong to one MRP domain, this is also displayed.
- MRP domain
  The list shows you the existing MRP domains of the selected IO system.

12.11.19 Communication with other PLCs

12.11.19.1 Communication with other PLCs

Introduction

Communication with other PLCs is communication with PLCs that are not in the SIMATIC family.

These PLCs have proprietary protocols for data exchange. The protocols are configured as communication drivers in WinCC.

Communication drivers

The following communication drivers are supported in WinCC and are already installed:

- Allen-Bradley
  - Allen-Bradley EtherNet/IP
  - Allen-Bradley DF1
- Mitsubishi
  - Mitsubishi MC TCP/IP
  - Mitsubishi FX
- Modicon Modbus
  - Modicon Modbus TCP/IP
  - Modicon Modbus RTU
- Omron
  - Omron Host Link

Communication drivers in WinCC RT Professional

The following communication drivers are supported for RT Professional:

- Allen-Bradley
  - Allen-Bradley EtherNet/IP
- Mitsubishi
  - Mitsubishi MC TCP/IP
- Modicon Modbus
  - Modicon Modbus TCP

Connections between HMI devices and other PLCs

You configure the connections between HMI devices and other PLCs in the "Connections" editor of the HMI device. These connections are non-integrated connections.

12.11.19.2 Distinctive features when configuring

Distinctive features for data exchange

Distinctive features apply when configuring connections to other PLCs, compared to configuring integrated connections.

Note the following distinctive features when configuring:

- Addressing of tags
- Permitted data types
- Distinctive features when configuring area pointers
- Distinctive features when configuring alarms
- Distinctive features when configuring trends

For more detailed information on distinctive features when configuring, refer to Section "Data exchange" of the respective communication driver.

12.11.19.3 Parallel communication

Parallel communication of communication drivers

The following table shows an overview of which communication drivers you can use simultaneously on one HMI device.

Note

Parallel communication is not approved for Basic Panels.

Parallel communication over Ethernet interfaces

The approved combinations can be operated via the same Ethernet interface. Several Ethernet interfaces are not required.

Parallel communication only concerns the Ethernet-based communication drivers.

	Allen-Bradley Ether-Net/IP	Mitsubishi MC TCP/IP	Modicon Modbus TCPIP	OPC (DA/ XML DA)	OPC UA (DA)	SIMATIC LOGO !	SIMATIC S7 200	SIMATIC S7 300/400	SIMATIC S7 1200	SIMATIC S7 1500	SIMATIC HTTP protocol	Sinumerik NC
Allen-Bradley Ether-Net/IP	--	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Mitsubishi MC TCP/IP	No	--	No	Yes	Yes	No	No	No	No	No	Yes	No
Modicon Modbus TCPIP	No	No	--	Yes	Yes	No	No	No	No	No	Yes	No
OPC (DA/ XML DA)	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OPC UA (DA)	Yes	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SIMATIC LOGO!	Yes	No	No	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes	Yes
SIMATIC S7 200	Yes	No	No	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes
SIMATIC S7 300/400	Yes	No	No	Yes	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes
SIMATIC S7 1200	Yes	No	No	Yes	Yes	Yes	Yes	Yes	--	Yes	Yes	Yes
SIMATIC S7 1500	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	--	Yes	Yes

	Allen-Bradley EtherNet/IP	Mitsubishi MC TCP/IP	Modicon Modbus TCPIP	OPC (DA/ XML DA)	OPC UA (DA)	SIMATIC LOGO!	SIMATIC S7 200	SIMATIC S7 300/400	SIMATIC S7 1200	SIMATIC S7 1500	SIMATIC HTTP protocol	Sinumerik NC
SIMATIC HTTP protocol	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	--	Yes
Sinumerik NC	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	--

Parallel communication over serial interfaces

The following applies for parallel communication over serial interfaces:

- One communication driver per interface.
- One interface per communication driver.

12.11.19.4 Communication drivers

Allen-Bradley

Allen-Bradley communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Allen-Bradley communication drivers.

The following communication drivers are supported:

- Allen-Bradley EtherNet/IP
- Allen-Bradley DF1

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags
  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.
- Area pointers
  Area pointers are used to exchange specific data and are only set up when these data are used.

Allen-Bradley EtherNet/IP

Configuring a connection via Allen-Bradley EtherNet/IP

Introduction

You configure a connection to a PLC with an Allen-Bradley EtherNet/IP communication driver in the "Connections" editor of the HMI device.

The Ethernet interfaces are named differently depending on the HMI device.

Example: PROFINET interface corresponds to the Ethernet interface

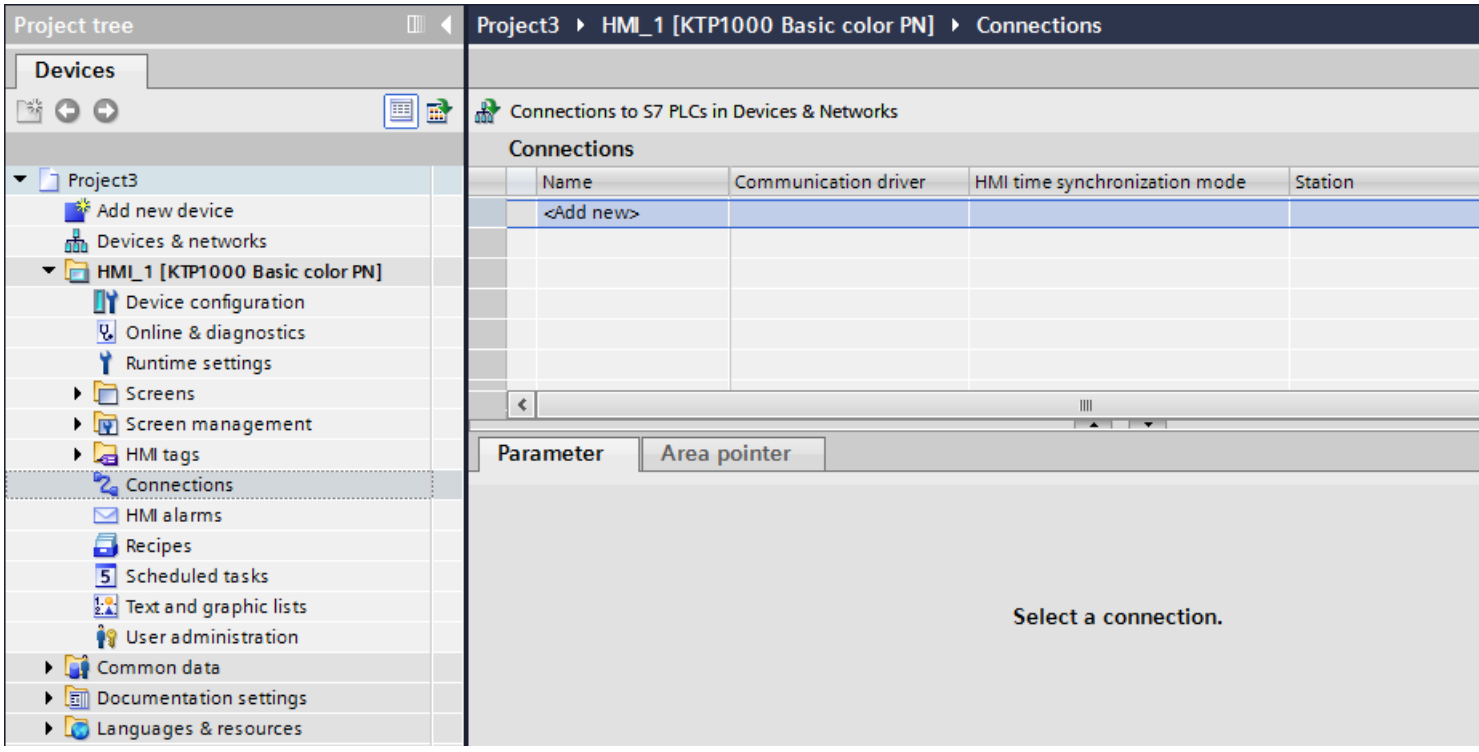
Requirements

- A project is open.
- An HMI device has been created.

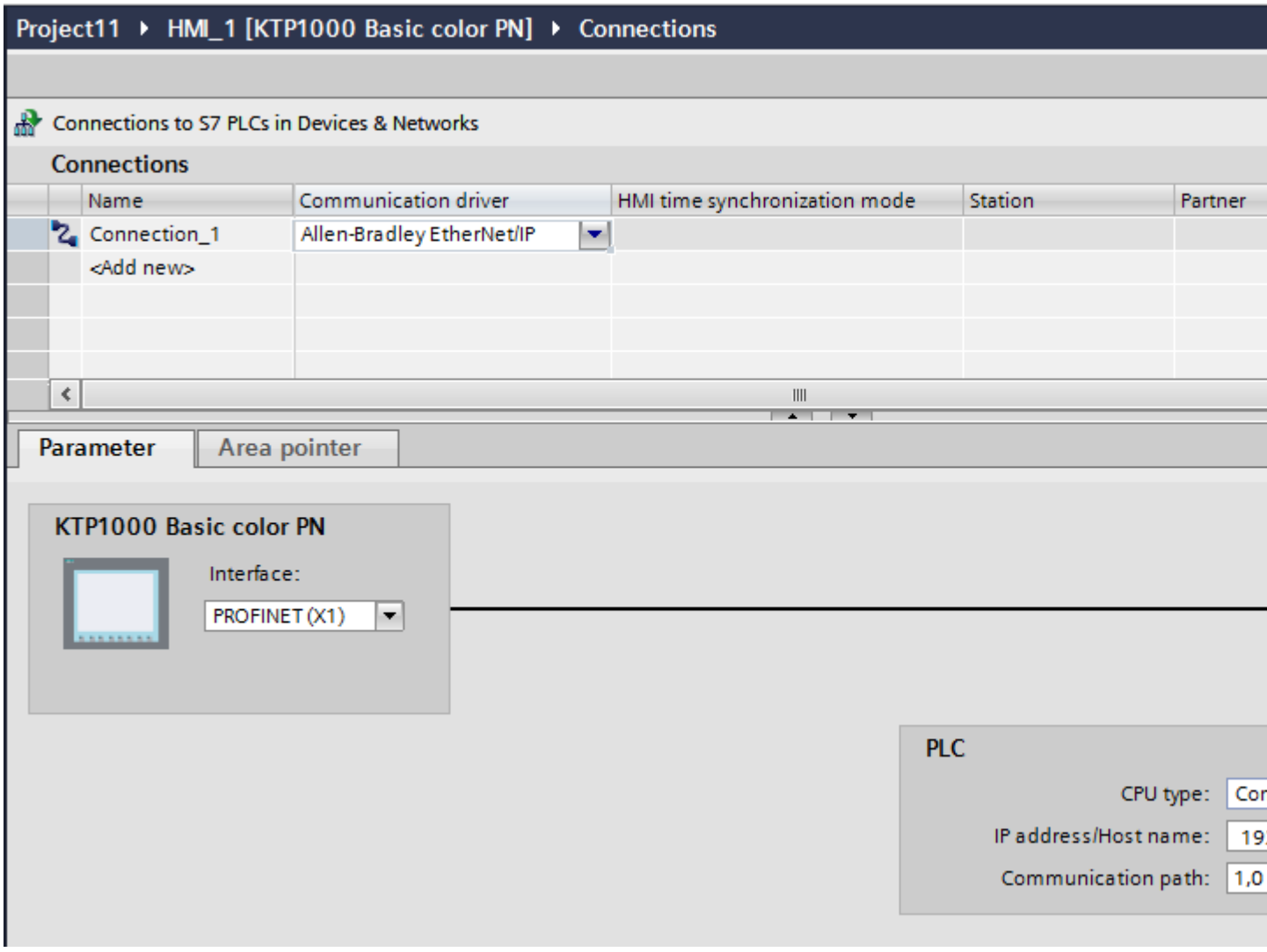
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Allen-Bradley EtherNet/IP" driver.



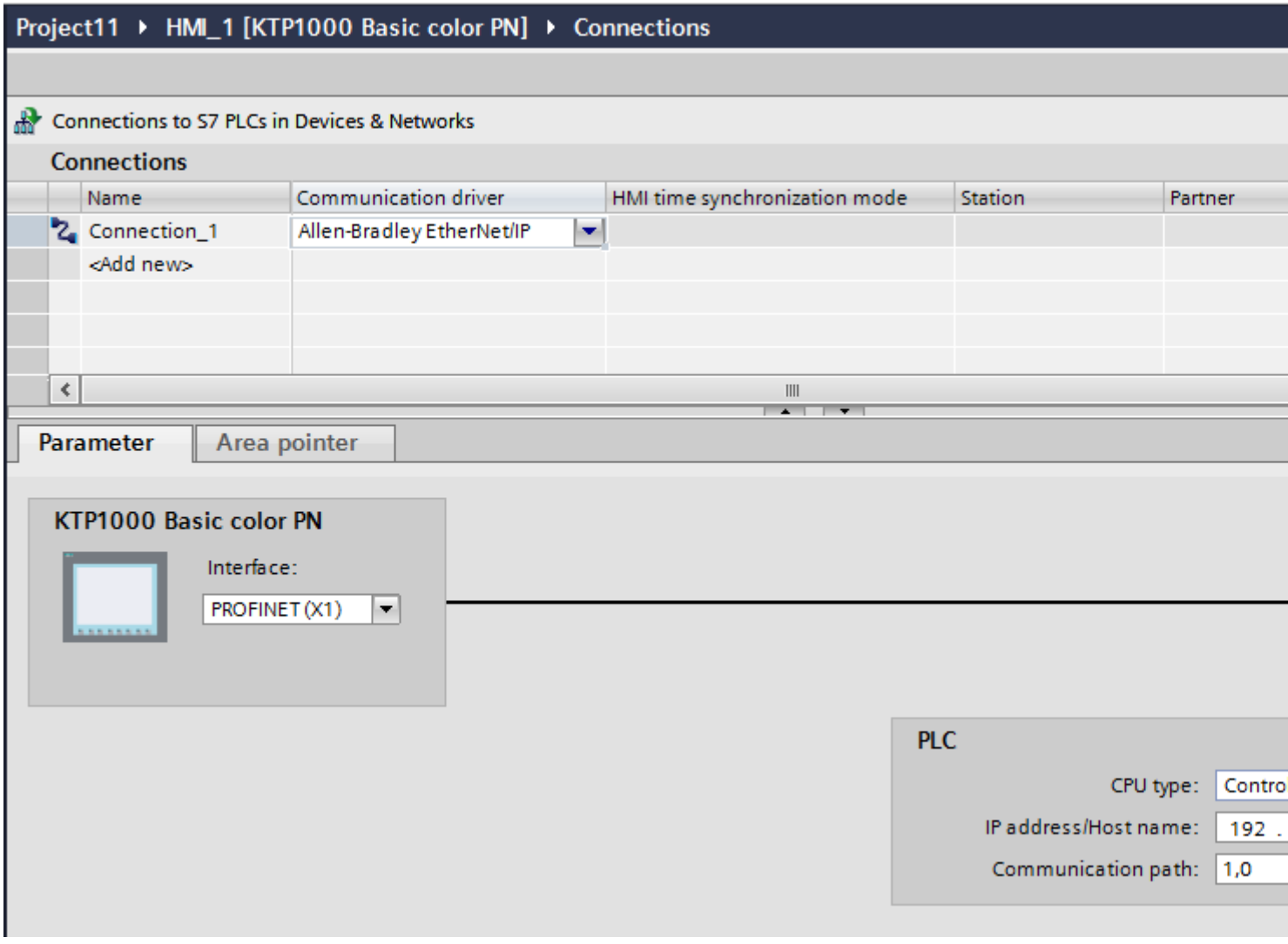
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Allen-Bradley EtherNet/IP)

Parameters to be set

To assign the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "PLC" area is available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select only one interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC. The IP address is transferred to the HMI device upon subsequent loading.

Note

The IP address in the control panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

The IP address already set up in the control panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

To set up the IP address of the HMI device:

1. Click on the HMI device.
2. Open the "Device configuration" editor.
3. Click the Ethernet interface.
4. Assign the IP address in the inspector window under:
   "General > PROFINET interface > Ethernet addresses"

Parameters for the PLC

- CPU type
  For "CPU type", set the CPU type of the PLC used.
- IP address
  Set the IP address or host name of the Ethernet/IP module of the PLC. Only the IP address can be used on a Basic Panel.
- Communication path
  Set the CIP path from the Ethernet module to the PLC. This establishes a logical connection between the Ethernet module and PLC, even if both devices are located in different CIP networks.
  For additional information see: Examples: Communication path

Connecting HMI device to PLC

Connections via Allen-Bradley EtherNet/IP

Connection

The HMI device can be connected to the Allen-Bradley PLC using the following components:

- Existing Ethernet network that also contains the PLCs
- Cross-over Ethernet cable connected directly to the Ethernet interface of the CPU or the communication module

The connection of the HMI device to an Allen-Bradley PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Communication types

Approved communication types with Allen-Bradley EtherNet/IP

The following communication types are system-tested and approved:

- Point-to-point connection to the approved PLCs
- Multipoint connection from a HMI device (Allen-Bradley Ethernet/IP-Client) with up to 4 PLCs with the respectively approved PLCs. CPU types can be mixed.

Connection

Connection with the following PLCs is approved with Allen-Bradley EtherNet/IP:

- CPU type: "ControlLogix, Compact Logix"
  - ControlLogix
    556x(1756-L6x) with Ethernet module 1756-ENBT
  - Guard Logix-System ControlLogix
    556xS(1756-L6xS) with Ethernet module 1756-ENBT
  - CompactLogix
  - 533xE(1769-L3xE) with Ethernet interface onboard
  - 532xE(1769-L2xE) with Ethernet interface onboard
  - 534x (1768-L4x) with Ethernet module 1768-ENBT
- CPU type: "SLC, MicroLogix"
  - MicroLogix 1100 (with Ethernet interface onboard)
  - MicroLogix 1400 (with Ethernet interface onboard)
  - SLC 5/05 (with Ethernet interface onboard)

Performance features of communication

Permitted data types for Allen-Bradley EtherNet/IP

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

CPU type: ControlLogix, CompactLogix

Data type	Length
Bool	1 bit
DInt	4 bytes
Int	2 bytes
Real	4 bytes
SInt	1 byte
String	1 to 82 characters
UDInt	4 bytes
UInt	2 bytes
USInt	1 byte

Permitted data types arrays

Address	Permitted data types
Array	SInt, USInt, Int, UInt, DInt, UDInt, Real
Individual bits from the basic data types of the PLC SInt, USInt, Int, UInt, DInt, UDInt	Bool*

* Any changed value of certain defined bits is written back to the PLC. There is no check to determine whether any other bits have changed. The PLC (or other PLCs) may only read access the value.

CPU type: SLC, MicroLogix

Data type	Operand type	Length
ASCII	A	0 to 80 characters
Bool	N, R, C, T, B, S, I, O	1 bit
DInt	N	4 bytes
Int	N, R, C, T, S	2 bytes
Real	N, F	4 bytes
String	ST	1 to 82 characters
UDInt	N	4 bytes
UInt	N, R, C, T, B, I, O	2 bytes

Permitted data types arrays

Address	Permitted data types
Array	Int, UInt, DInt, UDInt, Real

Distinctive features for connections with Allen-Bradley Ethernet/IP

With the communication driver Allen Bradley Ethernet/IP and the CPU type SLC, MicroLogix, you can only use array tags for discrete alarms and trends.

Note

I/O modules with 8 or 16 ports occupy one data word on the PLC.

I/O modules with 24 or 32 ports occupy two data words.

The HMI device does not output an error message if using non-existent bits.

You should always make sure that I/O modules with 8 or 24 ports only occupy the bits that are actually assigned to a port.

Supported CPU types for Allen-Bradley EtherNet/IP

CPU types

The following CPU types are supported for configuring the Allen-Bradley EtherNet/IP communication driver.

- CompactLogix
  - 1769-L2xE with Ethernet interface onboard
  - 1769-L3xE with Ethernet interface onboard
  - 1768-L4x with Ethernet module 1768-ENBT
- ControlLogix
  - 1756-L6x with Ethernet module 1756-ENBT
- GuardLogix
  - 1756-L61S with Ethernet module 1756-ENBT
  - 1756-L62S with Ethernet module 1756-ENBT
  - 1756-L63S with Ethernet module 1756-ENBT
- MicroLogix
  - MicroLogix 1100 / 1400
- SLC50x
  - SLC5/05

Addressing in the C.Logix CPU type

Addressing

Addressing

A tag is uniquely referenced in WinCC by means of an address in the PLC. The address must correspond with the tag name in the PLC. The tag address is defined by a string with a length of up to 128 characters.

Using characters for addressing

Valid characters for tag addressing:

- Letters (a to z, A to Z)
- Numbers (0 to 9)
- Underscore (_)

The tag address consists of tag name and other character strings used to specify the tag in the PLC.

Tag name properties:

- The tag name may begin but not end with an underscore character.
- Strings with successive underscore and space characters are invalid.
- The address may not exceed a length of 128 characters.

Note

The characters reserved for tag addressing may not be used in program/tag names or at any other address instance.

The reserved characters are listed below:

Reserved character	Function
.	Element delimiter
:	Definition of a program tag
,	Delimiter for addressing multi-dimensional arrays
/	Reserved for bit addressing.
[ ]	Addressing of array elements or arrays

PLC and program tags

The Allen-Bradley EtherNet/IP communication driver supports addressing of PLC tags (global project tags) and/or program tags (global program tags).

A program tag is declared based on the program name in the PLC and actual tag name which are delimited by colon. PLC tags are simply addressed by their name.

Note

Addressing errors

Addressing errors occur when the tag name and data type are inconsistent.

Note that the tag name defined in the address field in WinCC must match the tag name in the PLC. Make sure that the data types of tags in WinCC match the data types in the PLC.

Note

Module-specific tags, e.g. for data on input and output modules, cannot be addressed directly. Instead, use an alias tag in the PLC.

Example: Local:3:O. Data cannot be addressed in WinCC.

If the alias "MyOut" is defined for Local:3:O in the PLC, you can address with WinCC via MyOut.Data.

Addressing syntax

Notation of addresses

The tables below define the notation for the individual addressing options for Allen-Bradley EtherNet/IP.

Table 12-155 Access to arrays, basic data types and structure elements

Data types	Type	Address
Basic data types	PLC tag	Tag name
	Program tag	Programname:tagname
Arrays	PLC tag	Array tag
	Program tag	Program name: array tag
Bits	PLC tag	Tagname/bitnumber
	Program tag	Programname:tagname/bitnumber
Structure elements	PLC tag	Structure tag. Structure element
	Program tag	Program name: structure tag. structure element

Note

Bit addressing with the data types Bool, Real and String is not permitted and will cause an addressing fault.

Description of the syntax

Syntax description:

```
 (Programname:)tagname([x(,y)(,z)]){.tagname([x(,y)(,z)])}(/
bitnumber)
```

- The "( )" defines an optional, single instance of an expression.
- The "{ }" defines an optional expression with multiple single instances.

The address string length may not exceed 128 characters.

Addressing types

Arrays

An array is a data structure that includes a number of data of the same type. WinCC only supports one-dimensional arrays.

In the address column of the tag editor, enter the array name possibly by specifying a start element. The length is defined in the Array Elements input box of the tag editor. If array limits in the PLC are exceeded (due to faulty indexing), addressing errors result.

These arrays must be declared in the PLC as controller or program tags.

Two- or three-dimensional arrays in the PLC can only be addressed in WinCC if these can be mapped area-wise onto one-dimensional arrays .

Note

During all read accesses and all write accesses, all array elements of a tag are always read or written, respectively. The contents of an array tag which is interconnected with a PLC are always transferred whenever there is a change. The HMI device and the PLC cannot concurrently write data to the same array tag for this reason. Instead of writing data only to a single element, the program writes the entire array to the PLC.

Array elements

Elements of one-dimensional, two-dimensional and three-dimensional arrays in the PLC are indexed by setting an index and the corresponding notation in the tag editor. Array addressing starts at element "0", with arrays of all basic types being valid for element addressing. Read/ write operations are only carried out at the addressed element, and not for the entire array.

Bits and bit tags

Bit access is allowed to all basic data types with the exception of Bool, Real and String. Bit addressing is also allowed at array/structure elements. The Bool data type is set in WinCC when bits and bit tags in the basic data types are addressed.

Single-digit bit numbers are addressed with "/x" or "/0x" (x = bit number). Bit numbers are defined by up to two digits.

Note

With the "Bool" data type in the data types SInt, Int and DInt, after changing the specified bit the complete tag is then written in the PLC again. In the meantime, no check is made as to whether other bits in the tag have since changed. Therefore, the PLC may have only read access to the specified tag.

Structures

User-defined data types are created by means of structures. These structures group tags of different data types. Structures may consist of basic types, arrays and of other structures. In WinCC, only structure elements are addressed and not entire structures.

Structure elements

Structure elements are addressed by means of the name of the structure and of the required structure element. This addressing is separated by point. In addition to basic data types, the structure elements may represent arrays or other structures. Only one-dimensional arrays may be used as a structure element.

Note

The nesting depth of structures is only limited by the maximum length of 128 characters for the address.

Address multiplexing

Address multiplexing

Address multiplexing is possible with the CompactLogix, ControlLogix CPU type.

Address multiplexing requires two tags:

- "Tag_1" of data type "String"; contains a logical address such as "HMI:Robot5.Block5" as value.
  The value may change to a second valid address, for example, "HMI:Robot4.Block3".
- "Tag_2" is a tag in which the "Allen-Bradley EtherNet/IP" communication driver is set up as a connection.
  Enter a valid name of an HMI_tag in square brackets as the address.
  - e.g.: "[Tag_1]"
  - The tag must be of the String data type.
  - The square brackets indicate address multiplexing.
  - The address is derived from the actual value in "Tag_1".

  Note

  You can only multiplex entire Allen-Bradley EtherNet/IP addresses. Multiplexing of address elements is not possible. "HMI:Robot[Tag_1].Block5" is an invalid address.

  You can optionally click the arrow right icon in the "Address" column. Replace the "Constant" with the "Multiplex" entry by clicking the arrow on the left edge of the next address dialog box. Now the tag selection list only returns tags of data type "String".

  You can also configure a function triggered by a "change of value" event for multiplexed tags.

Examples for addressing

Example of a table for addressing

The table below defines the basic variants for addressing PLC tags. Other addressing variants are possible by means of combination.

Type	Type	Address
General	PLC tag	Tag name
	Program tag	Program:tagname
Array	Access to an element of a 2-dimensional array	Arraytag[Dim1,Dim2]
	Element of structure array (1-dimensional)	Arraytag[Dim1].structureelement
	Bit in element basic type array (2-dimensional)	Arraytag[Dim1,Dim2]/Bit
Structure	Array in structure	Structuretag.arraytag
	Bit in the element of an array in the substructure	Structuretag.structure2.arraytag [element]/ bit

Note

Program tags are addressed by leading the address with the program name derived from the PLC with colon delimiter.

Example: Programname:arraytag[Dim1,Dim2]

Access to array elements

Type	Address
PLC tag	Arraytag[Dim1]
	Arraytag[Dim1,Dim2]
	Arraytag[Dim1,Dim2,Dim3]
Program tag	Programname:arraytag[Dim1]
	Programname:arraytag[Dim1,Dim2]
	Programname:arraytag[Dim1,Dim2,Dim3]

Examples: Communication path

Example 1:

Connection with a PLC in the same Allen-Bradley rack.

1,0

Number	Meaning
1	Stands for a backplane connection.
0	Stands for a CPU slot number.

Example 2:

Connection with a PLC in remote Allen-Bradley racks. Two Allen-Bradley racks are networked on Ethernet.

1,2,2,190.130.3.101,1,5

Number	Meaning
1	Backplane connection
2	Stands for the CPU slot number of the second Ethernet module.
2	Stands for an Ethernet connection.
190.130.3.101	IP address of a remote AB rack on the network – in particular the third Ethernet module
1	Backplane connection
5	Slot number of the CPU

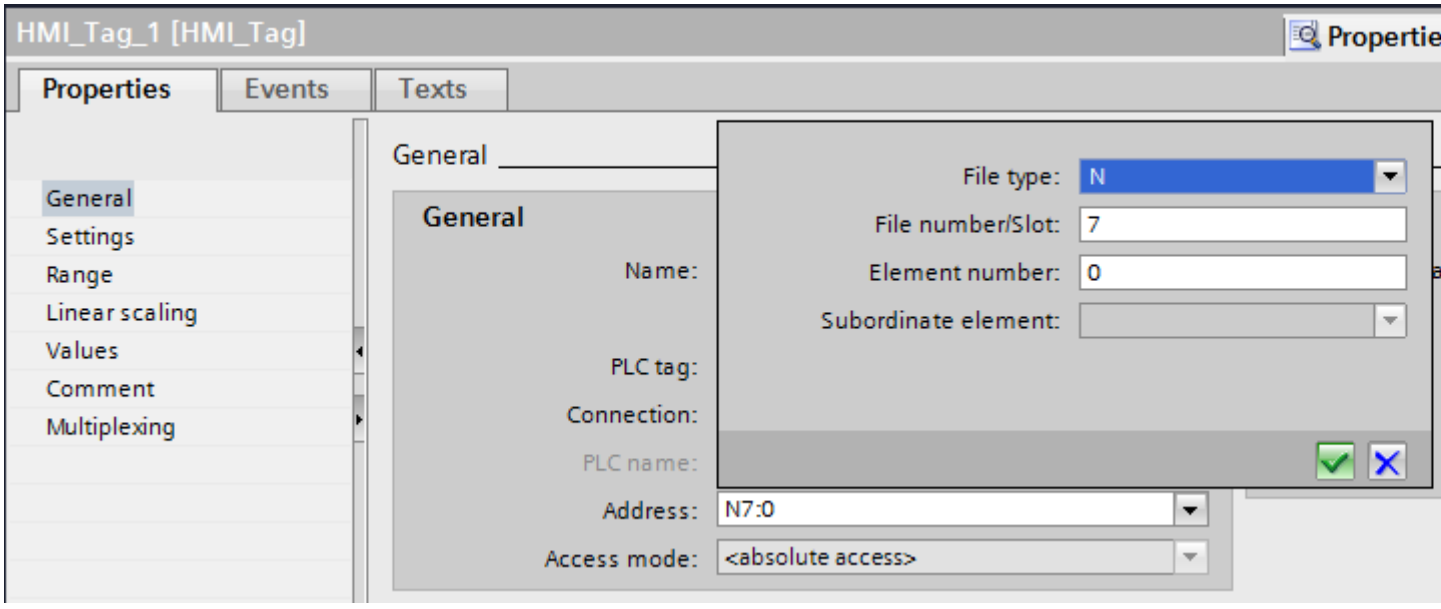
Addressing in the SLC, MicroLogix CPU type

Addressing

The addressing in the SLC, MicroLogix CPU type is entered in the following order:

- Operand type
- File number
- Element number

- Child element
- Bit number



The address then appears in the following format without spaces:

- File type file number : Element number . Child element
- e.g. T10:2.ACC

Operand type

You have the following options under operand type:

- I
- O
- S
- B
- C
- T
- R
- F
- N
- ST
- A

File number

Select the number between two limits under file number:

- Low limit
- High limit

The limit values depend on the selected operand type.

Child element

You can select a child element when you have selected one of the following operand types:

- R
- C
- T

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Note

If running the CompactLogix PLC with firmware earlier than version 18, you will possibly have to restart the HMI device following the transfer of the PLC program.

You could also terminate the connection before transferring the PLC program and set up the connection again after having completed the transfer of the PLC program.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Allen-Bradley DF1

Configuring a connection via Allen-Bradley DF1

Introduction

You configure a connection to a PLC with an Allen-Bradley DF1 communication driver in the "Connections" editor of the HMI device.

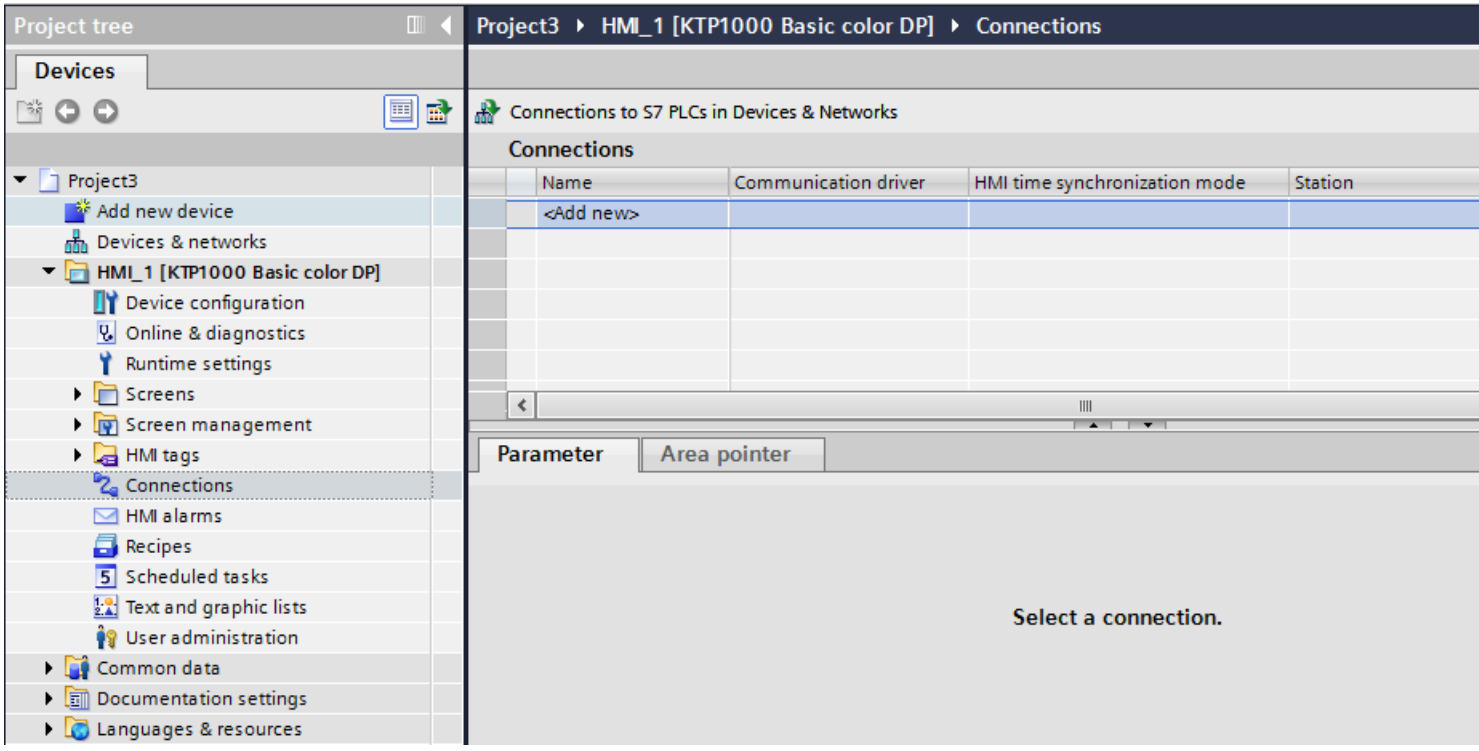
The interfaces are named differently depending on the HMI device.

Requirements

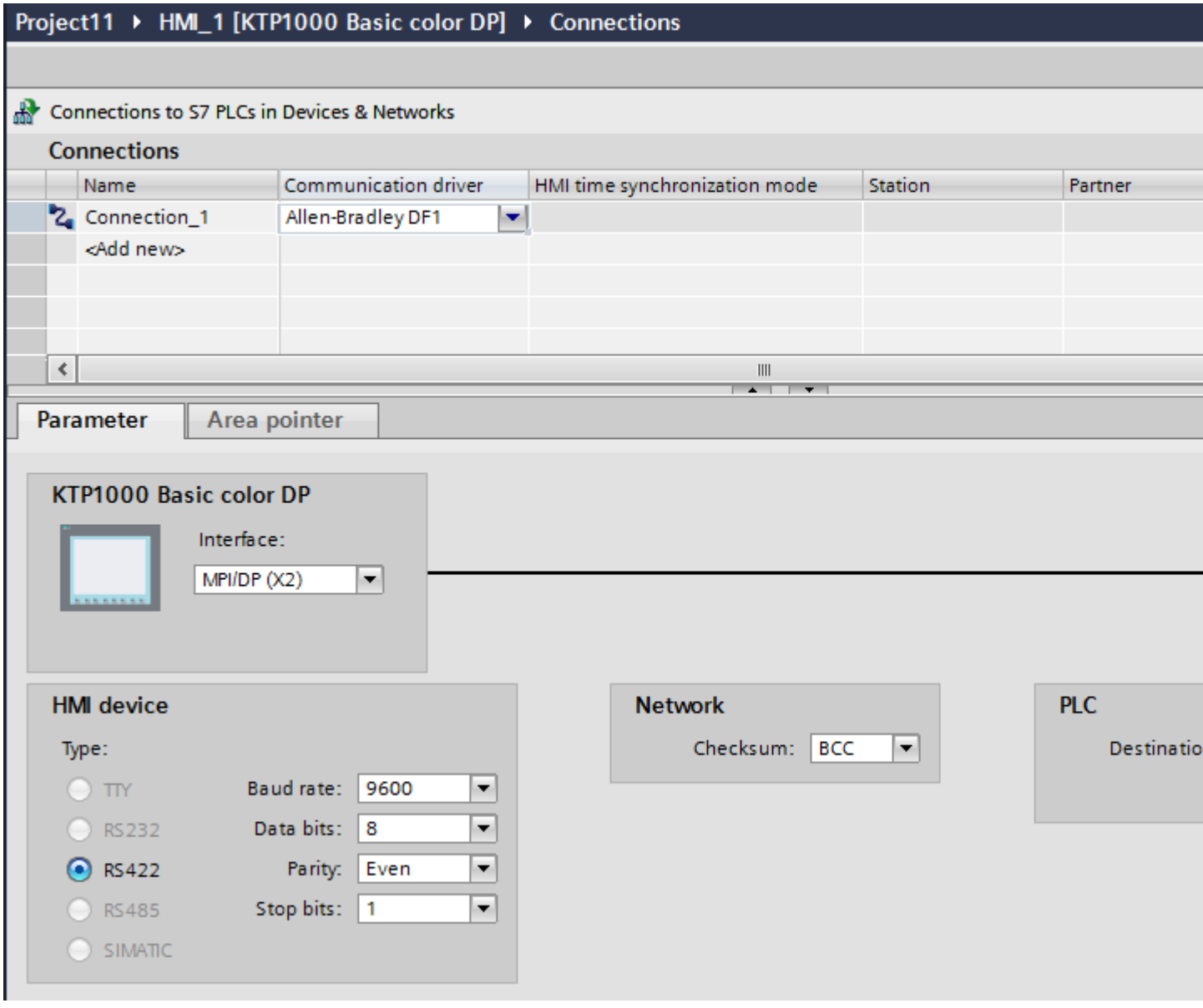
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Allen-Bradley DF1" driver.



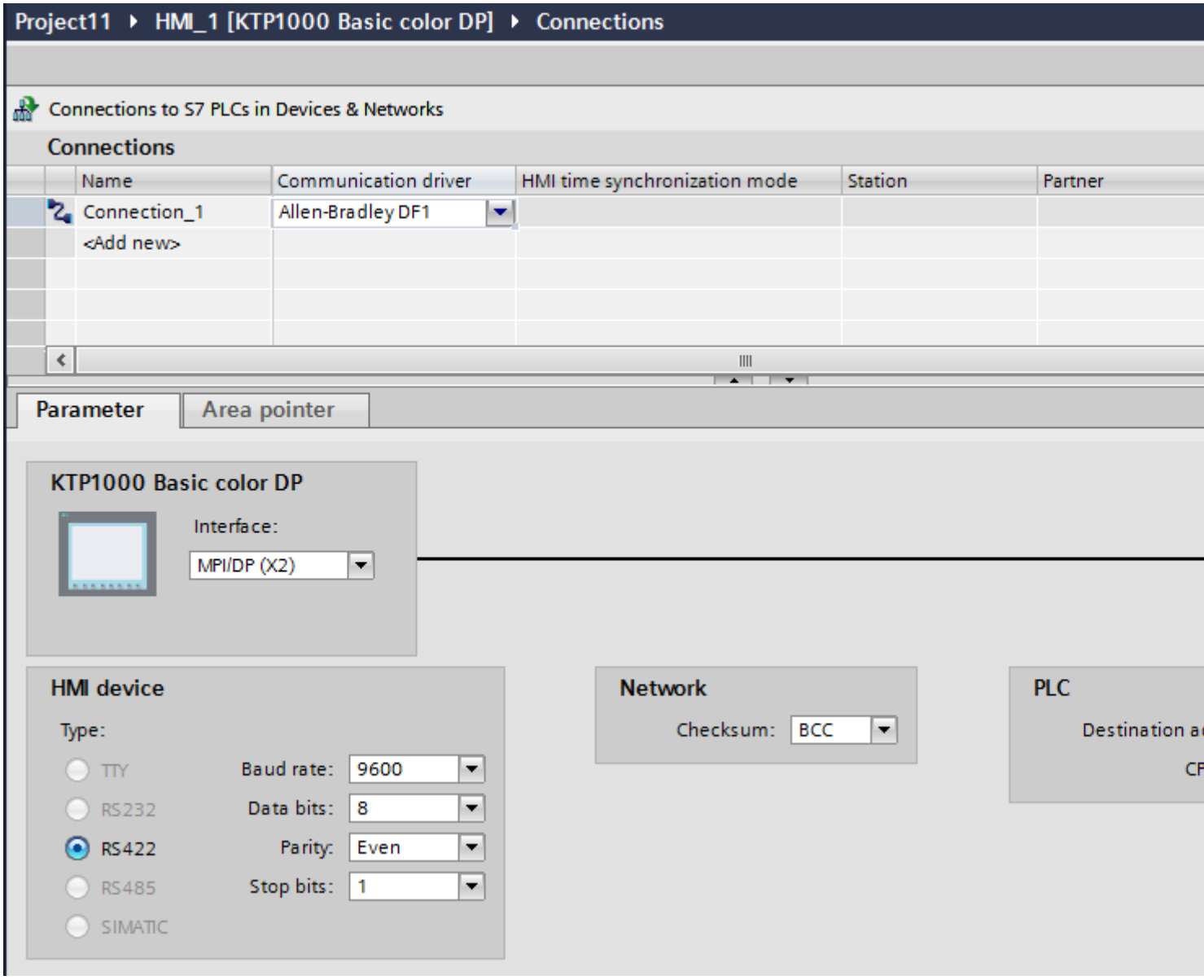
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Allen-Bradley DF1)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

- Interface
  Under "Interface", you select the interface of the HMI device to which the PLC is connected. For additional information, refer to the device manual for the HMI device.
- Type
  Specifies the physical connection used.

  Note

  If you are using the IF1B interface, you must also reconnect the RS-485 receive data and the RTS signal to the rear of the HMI devices via 4 DIP switches.

- Baud rate
  For "Baud rate", select the transmission speed between the HMI device and PLC.
- Data bits
  For "Data bits", you can choose between "7 bits" and "8 bits".
- Parity
  For "Parity", you can choose from "None", "Even", and "Odd".
- Stop bits
  For "Stop bits", you can choose between 1 and 2 bits.

Parameters for the network

- Checksum
  For "Checksum", choose the method for determining the error code: "BCC" or "CRC".

Parameters for the PLC

- Destination address
  For "Destination address", choose the PLC address. If there is a point-to-point DF1 connection, you set the address "0".
- CPU type
  For "CPU type", set the CPU type of the PLC used.

  Note

  Assign the DF1 FULL-DUPLEX driver in the CPU as follows: "NO HANDSHAKING" for "Control Line" and "AUTO-DETECT" for "Embedded Responses".

Connecting HMI device to PLC

Connections via Allen-Bradley DF1

Connection

The connection is established when you have matched the parameters of the PLC and the HMI device. Special blocks for the connection are not required in the PLC.

Note

Rockwell offers a variety of communication adapters for integrating "DF1 devices" for the DH485, DH, and DH+ networks. Of these connections, the direct connection and the connection via KF2 and KF3 module are approved. None of the other connections have been system-tested by SIEMENS AG and are therefore not approved.

Communication partners for Allen-Bradley DF1

Connectable PLCs

The communication drivers listed below support Allen-Bradley PLCs: :

PLC	DF1 (point-to-point) RS-232	DF1 (point-to-point) RS-422	DF1 (multipoint) over KF2 module to DH+ LAN RS-232/RS-422	DF1 (multipoint) over KF3 module to DH485 LAN RS-232
SLC500	–	–	–	X
SLC501	–	–	–	X
SLC502	–	–	–	X
SLC503	X	–	–	X
SLC504	X	–	X	X
SLC505	X	–	–	X
MicroLogix	X	–	–	X
PLC-5 [1)]	X	X	X	–

[1)] Processors released for PLC-5: PLC-5/11, PLC-5/20, PLC-5/30, PLC-5/40, PLC-5/60, and PLC-5/80.

Communication types

PLCs with Allen-Bradley DF1 communication driver

The communication between the HMI device and the following Allen Bradley PLCs is described in this section:

- SLC500
- SLC501
- SLC502
- SLC503
- SLC504
- SLC505
- PLC5
- MicroLogix

In these PLCs the connection is made by the PLC-internal protocols Allen Bradley DF1, Allen Bradley DH485 and Allen Bradley DH+.

The Allen-Bradley DF1 communication driver is used here, the protocol of which is converted into one of the other two PLC-internal protocols in multipoint communication with the communication modules KF2 (Allen Bradley DH+) and KF3(Allen Bradley DH485).

Enabled types of communication with Allen-Bradley DF1

The following communication types are system-tested and enabled:

- HMI (Allen Bradley DF1)
  Point-to-point connection
- HMI (Allen Bradley DF1)
  Via KF2 module to Allen Bradley DH+ (communication with up to 4 PLCs)
- HMI (Allen Bradley DF1)
  Via KF3 module to Allen Bradley DH485 (communication with up to 4 PLCs)

Connectable PLCs

The Allen Bradley DF1 communication driver is available for the following Allen-Bradley PLCs:

PLC	DF1 (point-to-point) RS 232	DF1 (point-to-point) RS 422	DF1 (multipoint) via KF2 module to DH+ LAN RS 232/RS 422	DF1 (multipoint) via KF3 module to DH485 LAN RS 232 [2]
SLC500	–	–	–	X
SLC501	–	–	–	X
SLC502	–	–	–	X
SLC503	X [2]	–	–	X

PLC	DF1 (point-to-point) RS 232	DF1 (point-to-point) RS 422	DF1 (multipoint) via KF2 module to DH+ LAN RS 232/RS 422	DF1 (multipoint) via KF3 module to DH485 LAN RS 232 [2]
SLC504	X [2]	–	X	X
SLC505	X [2]	–	–	X
MicroLogix	X [2]	–	–	X
PLC-5 [1]	X	X	X	–

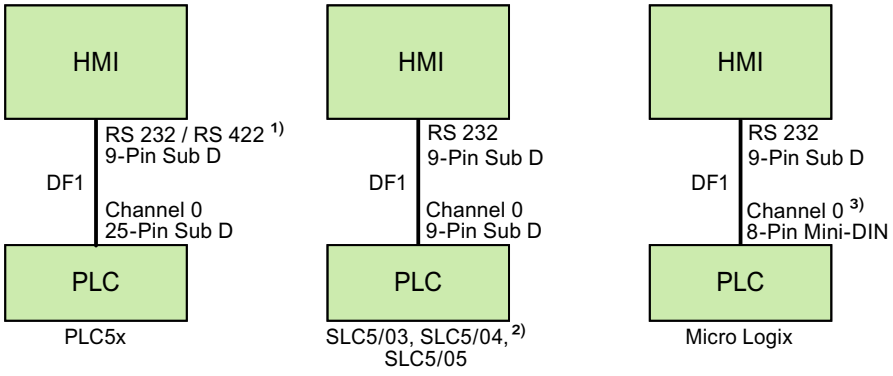
1) Only the following processors are approved for PLC-5: PLC-5/11, PLC-5/20, PLC-5/30, PLC-5/40, PLC-5/60 und PLC-5/80.

2) For HMI devices which only have an RS 422/485 interface and the communication partner is an RS 232 interface, the RS 422/232 converter is tested and approved.
Order number: 6AV6 671-8XE00-0AX0

DF1 protocol with multi-point connection

Point-to-point connection with DF1 protocol

Only point-to-point connections can be established with the DF1 protocol.



1) Only RS 232 is possible for Panel PC and PC.

2) A point-to-point connection to the SLC500, SLC501, and SLC502 PLCs via DF1 is not possible.

3) For MicroLogix ML1500 LRP, Channel 1 (9-pin Sub D) is also possible.

Connecting cable

HMI panel interface used	For connection to PLC5x	For connection to SLC5/03, SLC5/04, SLC5/05	For connection to MicroLogix
RS 232 9-pin	Allen-Bradley cable 1784-CP10	Allen-Bradley cable 1747-CP3	Allen-Bradley cable 1761-CBL-PM02
RS 422 9-pin	Connecting cable 9-pin Sub D RS 422	—	—

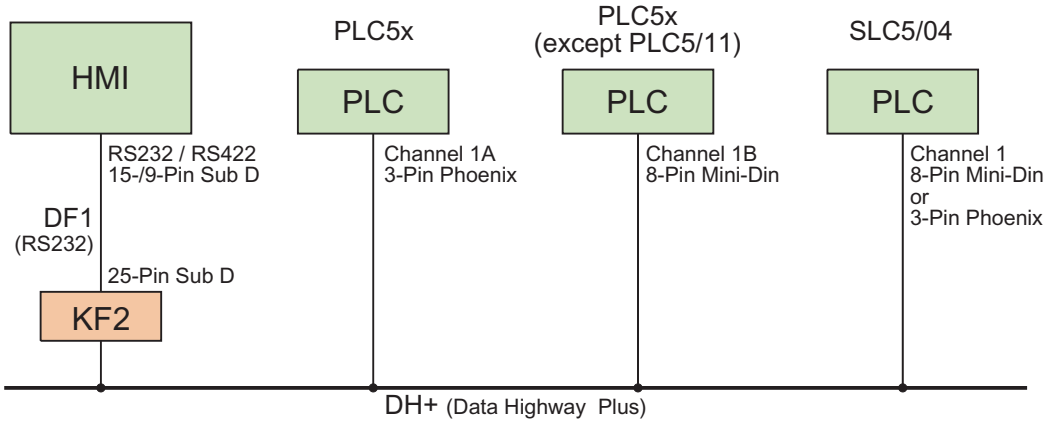
Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Allen-Bradley".

DF1 protocol with multi-point connection via KF2 module

DF1 protocol with multi-point connection via KF2 module to DH+ LAN

The use of a KF2 protocol interface module enables a connection to be made to PLCs in the DH+ LAN (Data Highway Plus Local Area Network).



Connecting cable

HMI panel interface used	For connection to KF2 interface module
RS 232 9-pin	Allen-Bradley cable 1784-CP10 and 25-pin socket/socket adapter
RS 422 9-pin	9-pin Sub D RS 422 connecting cable and 25-pin female/female adapter

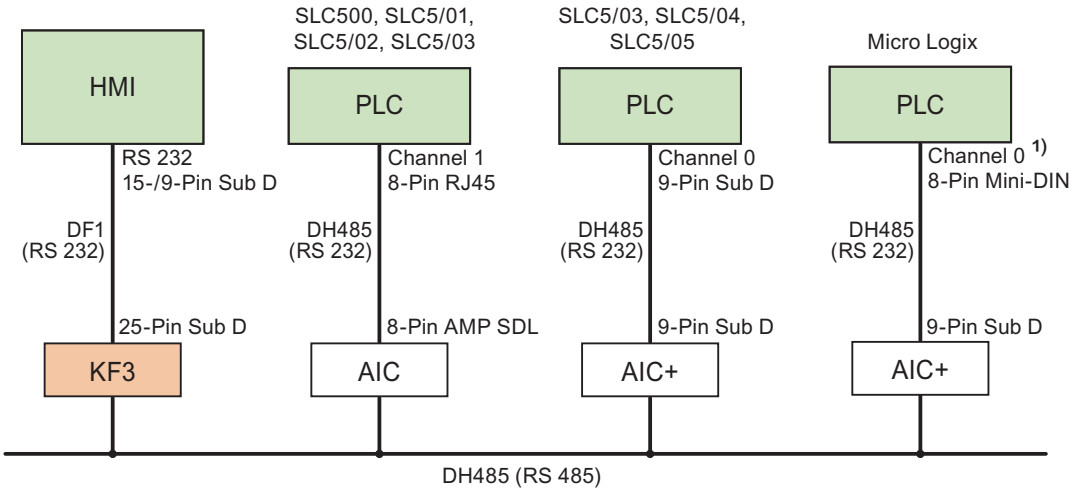
Refer to the Allen-Bradley documentation for the cable connection from the PLCs to the DH+ data bus.

Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Allen-Bradley".

DF1 protocol with multi-point connection via KF3 module

DF1 protocol with multi-point connection via KF3 module to DH485 LAN



1) For MicroLogix ML1500 LRP, Channel 1 (9-pin Sub D) is also possible.

Connecting cable

HMI panel interface used	For connection to KF3 interface module
RS 232 9-pin	Allen-Bradley cable 1784-CP10 and 25-pin socket/socket adapter

Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Allen-Bradley".

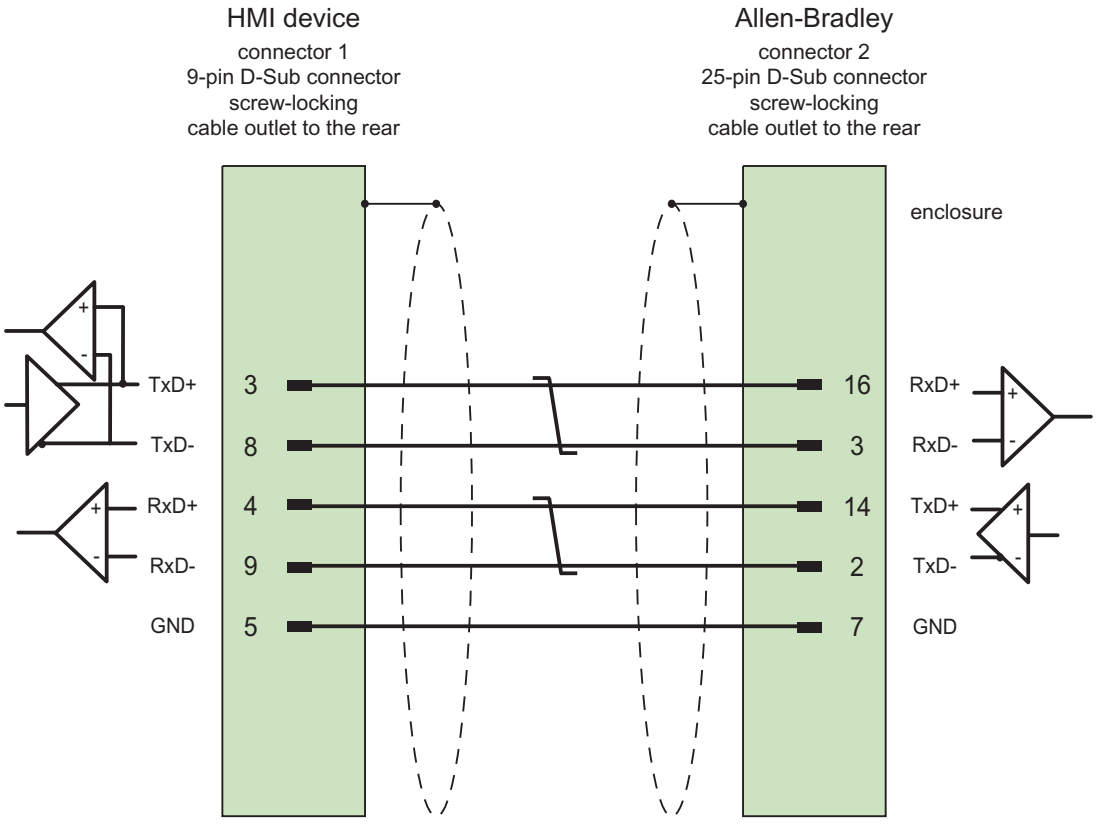
Connecting cables for Allen-Bradley DF1

Connecting cable 9-pin Sub D RS 422 for Allen-Bradley

Connecting cable 9-pin Sub D RS 422

For interconnecting the HMI device (RS 422, 9-pin sub D) - PLC5x, KF2, KF3

You require an additional 25-pin, female / female adapter (gender changer) for interconnections with KF2 and KF3.



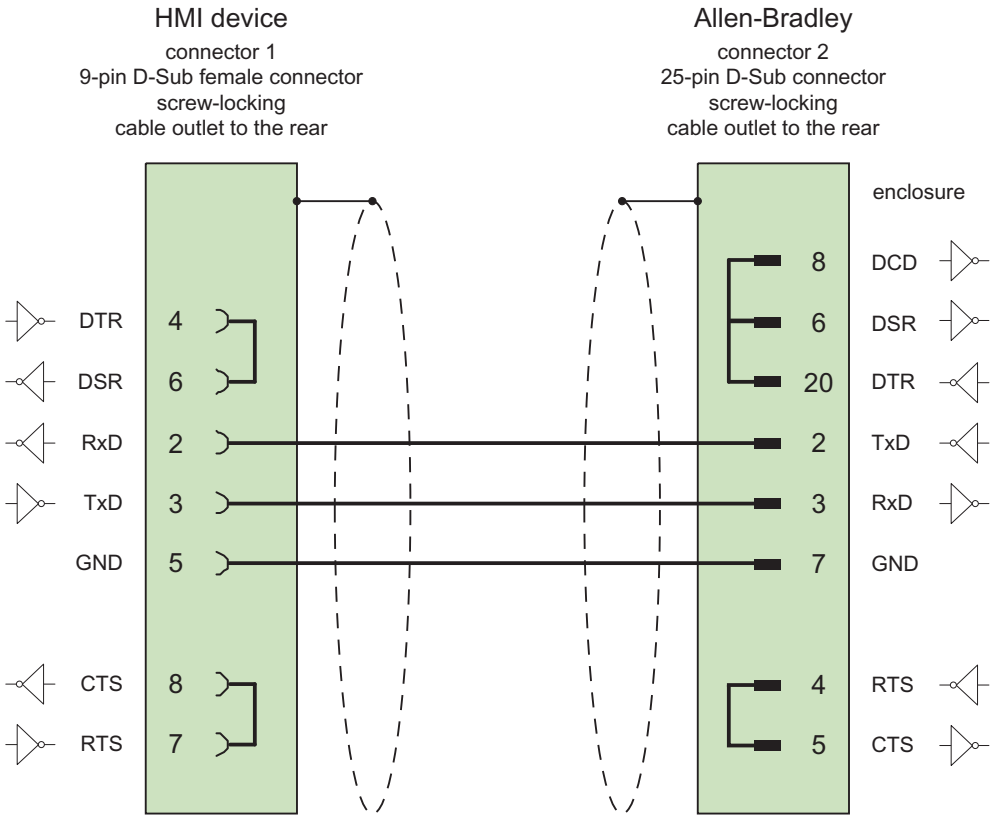
Shield with large-area contact to housing at both ends, interconnected shield contacts
Cable: 3 x 2 x 0.14 mm^2, shielded,
max. length 60 m

Connecting cable 1784-CP10, RS 232, for Allen-Bradley

Allen-Bradley cable 1784-CP10

For interconnecting the HMI device (RS 232, 9-pin sub D) - PLC5x, KF2, KF3

You require an additional 25-pin, female / female adapter (gender changer) for interconnections with KF2 and KF3.



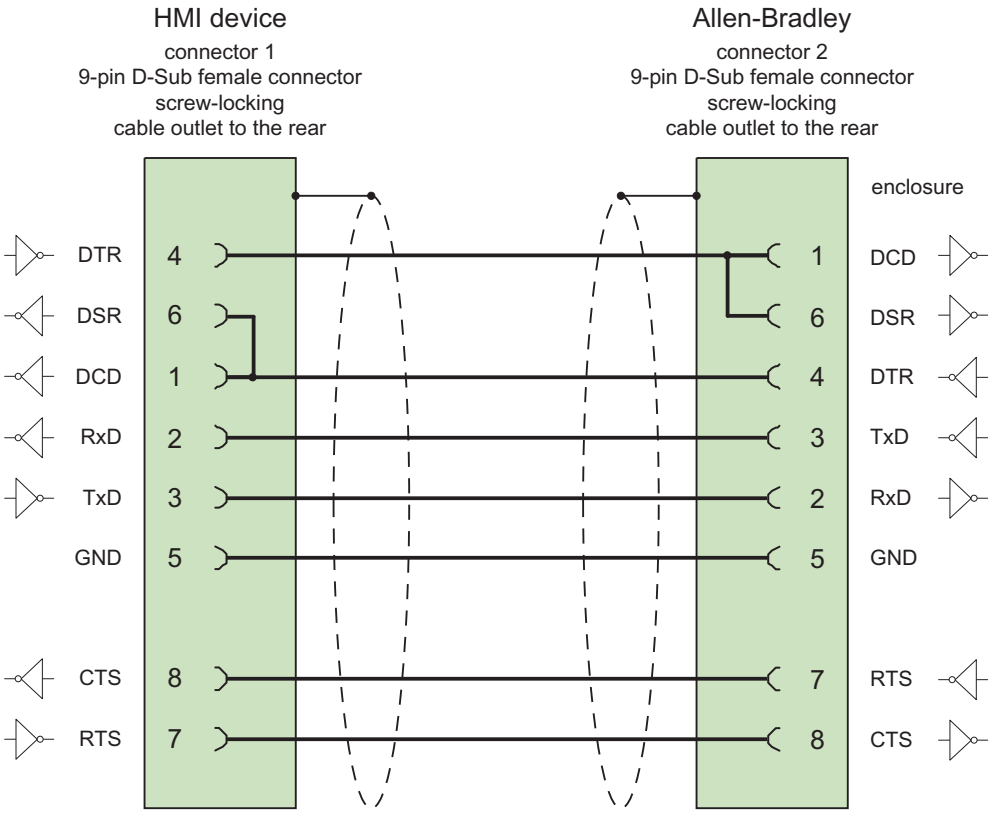
Screen connected with housing over large area on both sides

max. length 15 m

Connecting cable 1747-CP3, RS-232, for Allen-Bradley

Allen-Bradley cable 1747-CP3

For interconnecting the HMI device (RS 232, 9-pin sub D) - SLC503, SLC504, SLC505 (Channel 0), AIC+



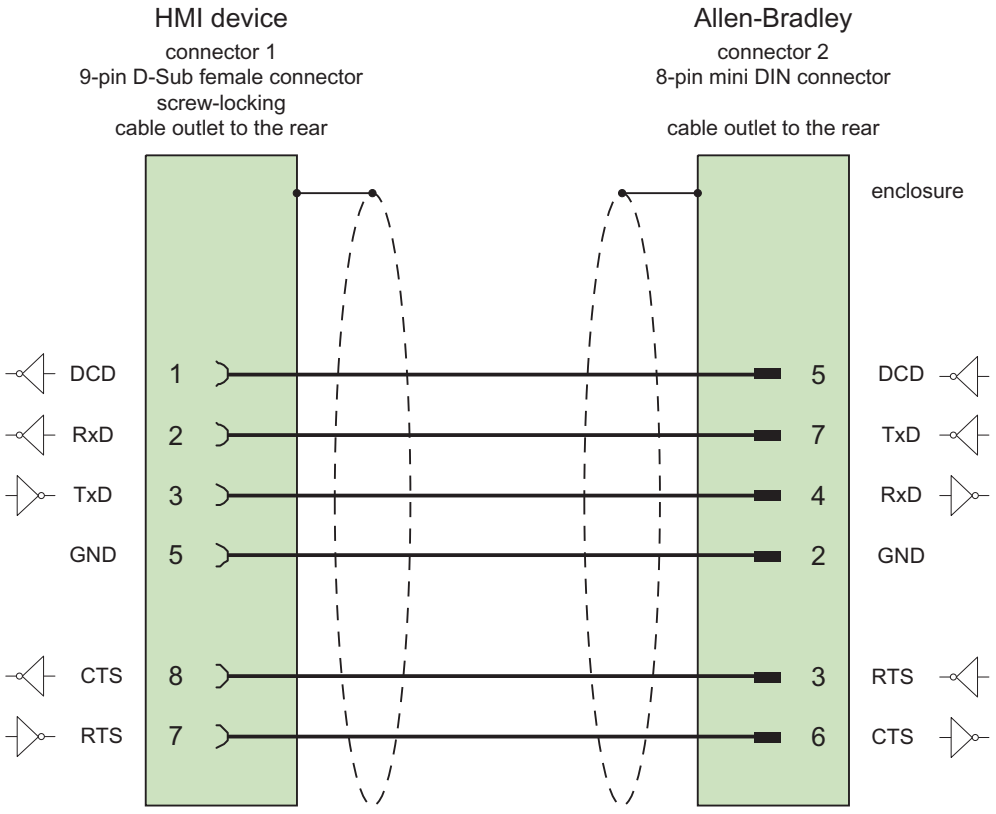
Screen connected with housing over large area on both sides

max. length 3 m

Connecting cable 1761-CBL-PM02, RS-232, for Allen-Bradley

Allen-Bradley cable 1761-CBL-PM02

For interconnecting the HMI device (RS 232, 9-pin sub D) - Micro Logix, AIC+



Screen connected with housing over large area on both sides

max. length 15 m

Performance features of communication

Permitted data types for Allen-Bradley DF1

Permitted data types for Allen-Bradley DF1

The table lists the user data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
ASCII	A [1)]	1 to 80 characters
Bool	N, R, C, T, B, S, I, O	1 bit
Int	N, R, C, T, S	2 bytes
DInt	N	4 bytes
UInt	N, R, C, T, B, I, O	2 bytes
UDInt	N	4 bytes
Real	N, F [1)]	4 bytes

[1] Selectable depending on the selected CPU type.

Abbreviations

In WinCC, formats of the data types are abbreviated as follows:

- UNSIGNED INT = UInt
- UNSIGNED LONG = UDInt
- SIGNED INT = Int
- SIGNED LONG = DInt

Distinctive features for connections with Allen-Bradley DF1

With Allen Bradley DF1, array tags may only be used for discrete alarms and trends.

Note

I/O modules with 8 or 16 ports occupy one data word on the PLC.

I/O modules with 24 or 32 ports occupy two data words.

The HMI device does not output an error message if using non-existent bits.

You should always make sure that I/O modules with 8 or 24 ports only occupy the bits that are actually assigned to a port.

Supported CPU types for Allen-Bradley DF1

CPU types

The following CPU types are supported for configuring the Allen-Bradley DF1 communication driver.

- SLC
  - SLC500
  - SLC501
  - SLC502
  - SLC503
  - SLC504
  - SLC505
- MicroLogix
  - MicroLogix 1x00
  - MicroLogix 1100 / 1400
- PLC 5
  - PLC-5/11
  - PLC-5/20
  - PLC-5/40
  - PLC-5/60
  - PLC-5/80

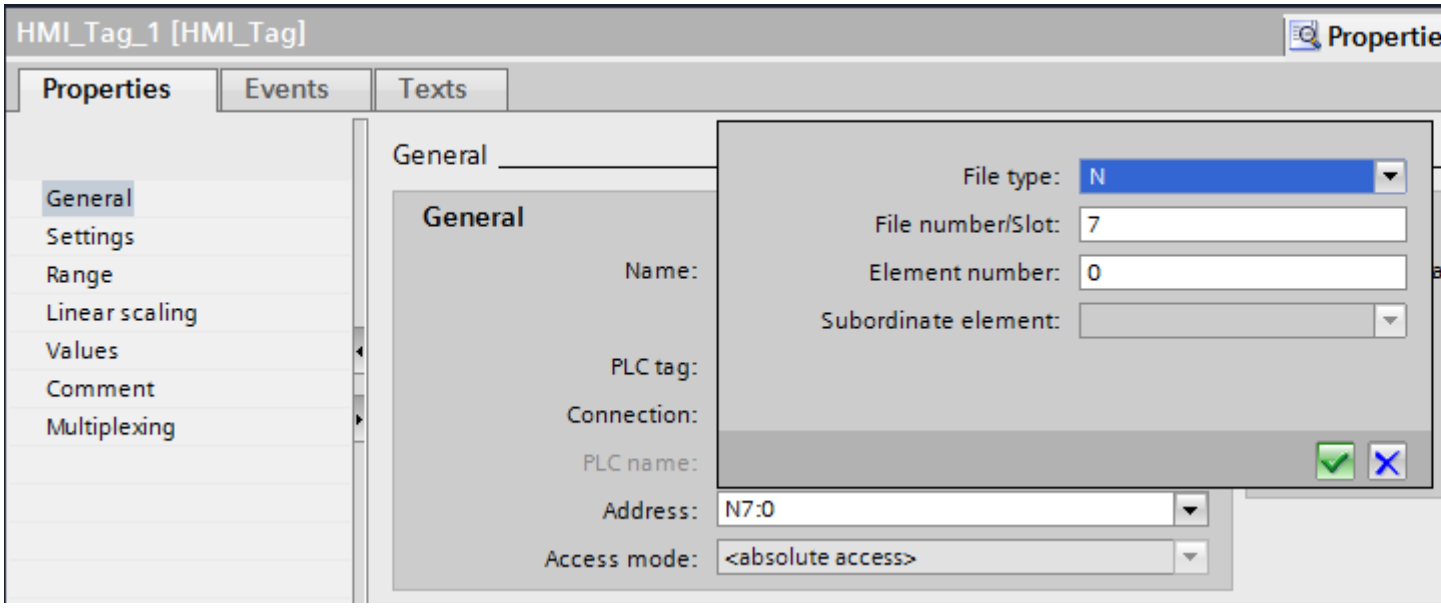
Addressing

Addressing

The addressing is entered in the following order in the Allen-Bradley DF1 communication driver:

- Operand type
- File number
- Element number

- Child element
- Bit number



The address then appears in the following format without spaces:

- File type file number : Element number . Child element
- e.g. T8:2.ACC

Operand type

You have the following options under operand type:

- I
- O
- S
- B
- T
- C
- R
- N
- A
- D only for PLC5 CPU type

File number

Select the number between two limits under file number:

- Low limit
- High limit

The limit values depend on the selected file type.

Child element

You can select a child element when you have selected one of the following data types:

- R
- C
- T

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Allen-Bradley

Area pointers for connections using an Allen-Bradley communication driver

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section "Data exchange using area pointers".

Distinctive features for connections via Allen-Bradley EtherNet/IP

You can configure the following area pointers

Area pointer	Allen-Bradley EtherNet/IP	Allen-Bradley DF1
Screen number	Yes	Yes
Date/time	Yes	Yes
Date/time PLC	Yes	Yes
Coordination	Yes	Yes
Project ID	Yes	Yes
Job mailbox	Yes	Yes
Data record	Yes	Yes

Restrictions Allen-Bradley Ethernet/IP

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
ControlLogix,CompactLogix	Int, UInt	--
SLC, MicroLogix	Int, UInt	N, B

Restrictions Allen-Bradley DF1

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
MicroLogix	--	N, O, I, B
SLC50x	--	N, O, I, B
PLC5	--	N, O, I, B

See also

Data exchange using area pointers (Page 6730)

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Displaying tags with Runtime Advanced and Panels (Page 4270)

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

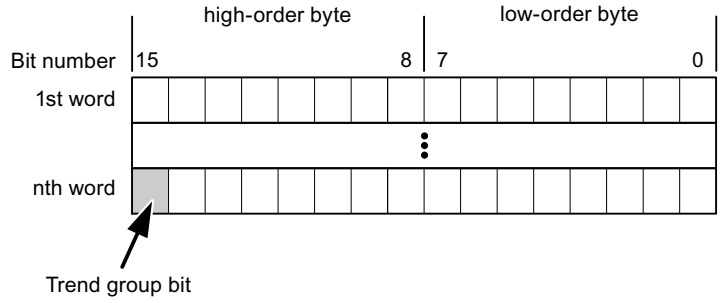
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Restrictions to the trend control

For Allen-Bradley DF1 communication drivers

Tags of the following operand types are permitted:

- "N"
- "O"
- "I"
- "S"
- "B"

Permitted data types:

- "UInt"
- "Int"

Note

The "Array" data type is only available for the trend type "Bit-triggered buffer". The data type "Array" cannot be selected for the "Bit-triggered real time", "Cyclic real time" or "Data log" trend types.

Individual array elements cannot be selected as data type.

You assign a bit to a trend during configuration. This sets a defined bit assignment for all trend areas.

For Allen-Bradley EtherNet/IP communication drivers

Tags of data type "Int" or an array tag of data type "Int" are permitted. You assign a bit to a trend during configuration. This sets a defined bit assignment for all trend areas.

ControlLogix and CompactLogix

The following data types are possible for tags for the ControlLogix and CompactLogix CPU types:

- "UInt"
- "Int"

SLC and MicroLogix

The following operand types are permitted for tags of the SLC and MicroLogix CPU types:

- "N"
- "O"
- "I"
- "S"
- "B"

Permitted data types:

- "UInt"
- "Int"

Note

The "Array" data type is only available for the trend type "Bit-triggered buffer". The data type "Array" cannot be selected for the "Bit-triggered real time", "Cyclic real time" or "Data log" trend types.

Individual array elements cannot be selected as data type.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with Alarms (Page 4293)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Restrictions

Only tags whose "File type" is "N", "O", "I", "S" and "B" are allowed for use as a "trigger tag" for discrete alarms. These tags are only valid for the data types "Int" and "UInt".

Data types

For connections with an Allen-Bradley communication driver, the following data types are supported:

Communication drivers	PLC	Permitted data types	
		Discrete alarms	Analog alarms
Allen-Bradley DF1	SLC500, SLC501, SLC502, SLC503, SLC504, SLC505, PLC5, MicroLogix	Int, UInt	Int, UInt, Long, ULong, Real
Allen-Bradley EtherNet/IP	ControlLogix, Compact-Logix, SLC, Micrologix	Int, UInt	SInt, USInt, Int, UInt, DInt, UDInt, Real

How the bit positions are counted

For connections with an Allen-Bradley communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In Allen-Bradley PLCs	15							8	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

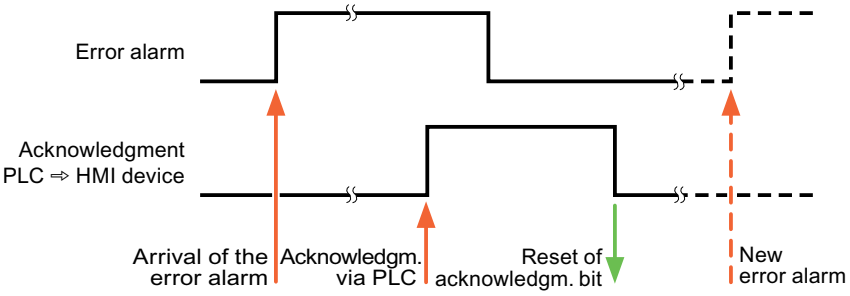
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

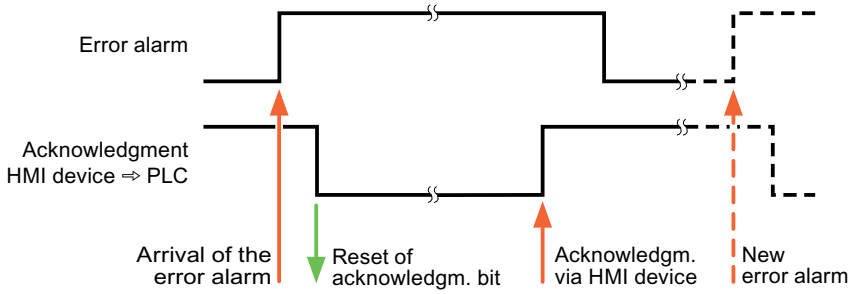
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



LED mapping

Function

Keyboard devices have LEDs in the function keys. These LEDs can be activated from the PLC. Thus, it is possible, for example, to signal to the user which key he should press in a given situation by lighting up an LED.

Note

The LED function cannot be configured for Basic Panels.

Requirements

In order to activate an LED, an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration.

LED assignment

The assignment of the individual LEDs to the bits in the LED tags is specified when the function keys are configured. In this process, the "LED tag" and the assigned "Bit" are specified for each function key in the "General" group of the properties window.

The "Bit" bit number designates the first of two consecutive bits that control the following LED statuses.

Bit n+ 1	Bit n	LED function	
		All Mobile Panels, all Operator Panels, all Multi Panels, all Comfort Panels	Panel PCs
0	0	Off	Off
0	1	Rapid flashing	Flashing
1	0	Slow flashing	Flashing
1	1	On permanently	On permanently

Mitsubishi

Mitsubishi communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Mitsubishi communication drivers.

The following communication drivers are supported:

- Mitsubishi MC TCP/IP
- Mitsubishi FX

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags
  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.
- Area pointers
  Area pointers are used to exchange specific data and are only set up when these data are used.

Mitsubishi MC TCP/IP

Configuring a connection via Mitsubishi MC TCP/IP

Introduction

You configure a connection to a PLC with a Mitsubishi MC TCPI/IP communication driver in the "Connections" editor of the HMI device.

The Ethernet interfaces are named differently depending on the HMI device.

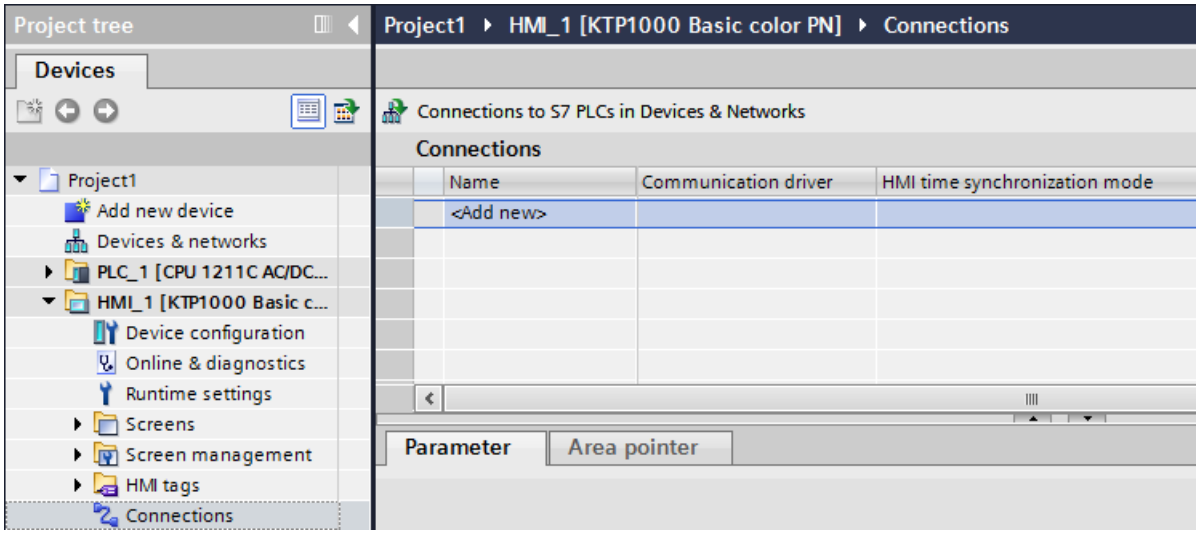
Example: PROFINET interface corresponds to the Ethernet interface

Requirements

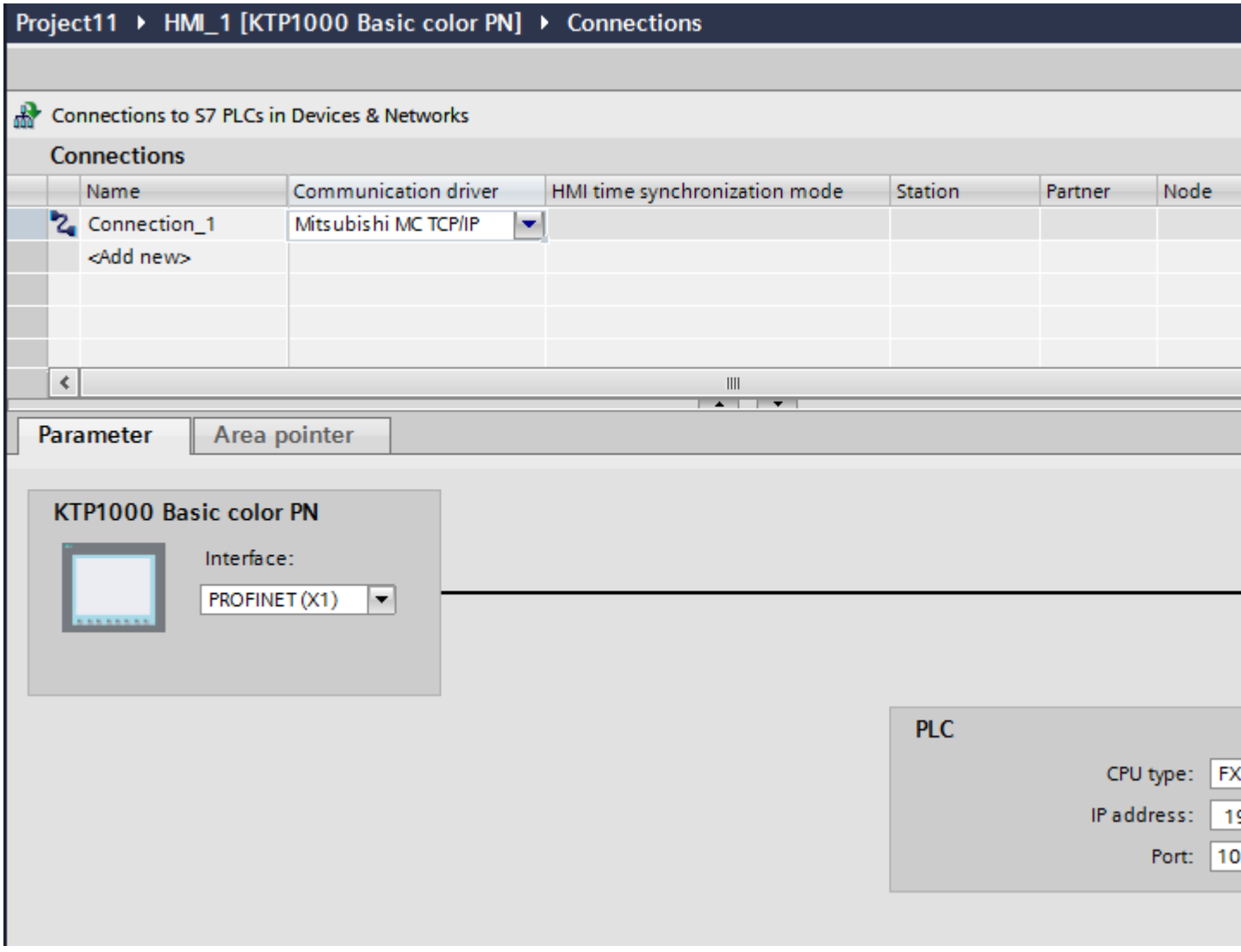
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Mitsubishi MC TCPI/IP" driver.



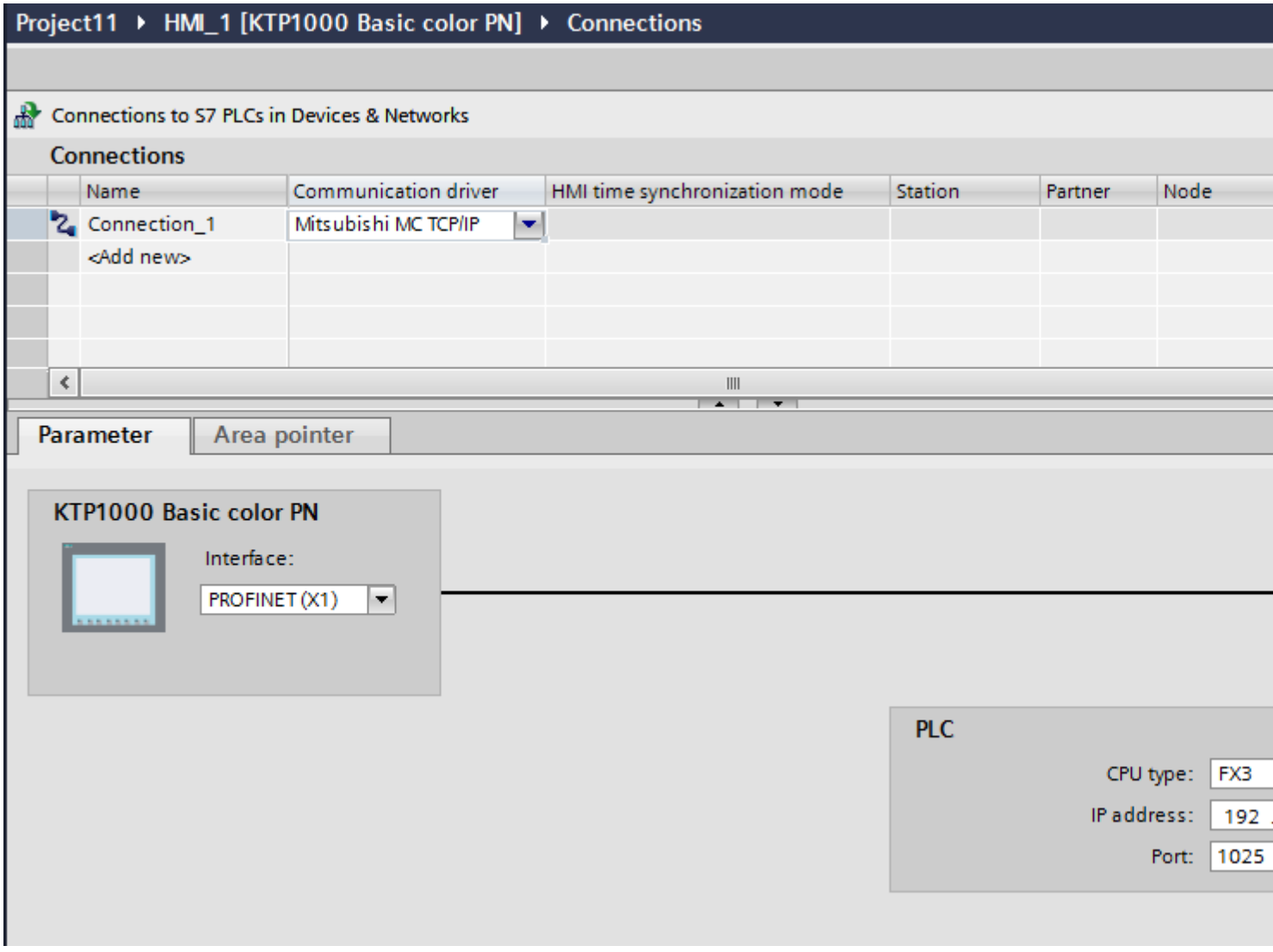
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Mitsubishi MC TCP/IP)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select only one interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC. The IP address is transferred to the HMI device during project transfer.

Note

The IP address in the control panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

The IP address already set up in the control panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

To set up the IP address of the HMI device:

1. Click on the HMI device.
2. Open the "Device configuration" editor.
3. Click the Ethernet interface.
4. Assign the IP address in the inspector window under:
   "General > PROFINET interface > Ethernet addresses"

Parameters for the PLC

- CPU type
  For "CPU type", you set the type of PLC to which the HMI device is connected.
  The following settings are possible:
  –FX3
  –Q
  If you select the FX3 CPU type, the Mitsubishi MC protocol "1E" is used and "3E" for the "Q" CPU type.
  The "Binary code" protocol variant is always used.

  Note

  If the CPU type is changed for a configured connection, tags with the following properties must be revised:

  - Operands that do not exist for the new CPU type, such as "W", "B", "F".
  - Inputs and outputs with different addressing (hexadecimal/octal)
  - Addresses greater than the valid address area of the new CPU type

- IP address
  Set the IP address or host name of the Ethernet/IP module of the PLC. Only the IP address can be used on a Basic Panel.
- Port
  Set the port number of the module of the PLC.

Connecting HMI device to PLC

Connections via Mitsubishi MC TCP/IP

Connection

The HMI device can be connected to the Mitsubishi PLC using the following components:

- Existing Ethernet network that also contains the PLCs
- Cross-over Ethernet cable connected directly to the Ethernet interface of the CPU or the communication module

The connection of the HMI device to a Mitsubishi PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Connect the HMI device to one or several Q-series and/or FX3 PLCs. Connect the HMI device via the following interfaces:

- Communication interface OnBoard
- Approved communication module suitable for the PLC

Note

Timeout response with TCP/IP (Ethernet)

Due to the use of the TCP/IP protocol, the breakdown of a connection is detected at the earliest after approximately one minute. Communication failure cannot be reliably detected if no tags are requested, for example, no output tags in the current screen.

Configure area pointer coordination for each PLC. This setting ensures that a communication failure is recognized after approximately two minutes, even in the aforementioned scenario.

Communication types

Approved communication types

- Only applies for Mitsubishi FX(PG protocol):
  The point-to-point connection from a HMI device to an approved Mitsubishi FX–CPU via Mitsubishi FX is system-tested and approved by Siemens AG.
- Only applies for Mitsubishi MC TCP/IP:
  The following communication types are system-tested and approved:
  - Point-to-point connection to the approved PLCs
  - Multipoint connection from a HMI device with up to 4 PLCs with the respectively approved PLCs. CPU types (FX3 and Q) can be mixed.

Note

The HMI device is a client and the PLC must operate as a server.

Connectable PLCs

Connections can be implemented for the following Mitsubishi PLCs:

PLC	Mitsubishi FX (PG protocol)	Mitsubishi MC TCP/IP
MELSEC FX1n, FX2n	Yes	No
MELSEC FX3U, FX3UC, FX3G with communication module FX3U-ENET	No	Yes
MELSEC System Q • Q-series with the communication module QJ71E71-100 • QnUDEH CPU with Ethernet interface onboard	No	Yes

Parameterization of the communications modules

FX3 PLCs

Procedure

1. Start the FX-Configurator.
2. Select the module.
3. Assign the following settings in the "Operational settings" dialog:
   - Communication data code:
     Binary code
   - Initial timing:
     Always wait for OPEN
   - IP address:
     IP address
   - Send frame setting:
     Ethernet(V2.0)
   - TCP Existence confirmation setting:
     Use the Ping

4. Assign the following settings in the "Open Settings" dialog:
   - Protocol:
     TCP
   - Open system:
     Unpassive
   - Fixed buffer:
     Receive
   - Fixed buffer communication procedure:
     Procedure exist(MC)
   - Pairing open
     Disable
   - Existence confirmation
     No confirm
   - Host station Port No. (DEC)
     Port number

   Note

   The port number chosen in the communication module must match the port number in WinCC. A connection with a port number must be assigned for each connected HMI device.

   You must specify port numbers in decimal values.

5. Confirm the default settings of the other dialog boxes.

The network no. and station no. parameters are not relevant for the connection and can be chosen as required.

Q PLCs

Procedure

1. Click "Edit network parameters".
2. Select the network type:
   - Ethernet
     The network number and the group / station number are not evaluated and can be freely assigned

3. Assign the following settings in the "Operational settings" dialog:
   - Communication data code:
     Binary code
   - Initial timing:
     Always wait for OPEN
   - IP address:
     IP address
   - Send frame setting:
     Ethernet(V2.0)
   - Enable write operations during RUN
4. Assign the following settings in the "Open settings" dialog:
   - Protocol:
     TCP
   - Open system:
     Unpassive
   - Pairing open
     Disable
   - Existence confirmation
     No confirm
   - Host station Port No. (HEX)
     Port-Nummer

   Note

   The port number chosen in the communication module must match the port number in WinCC. A connection with a port number must be assigned for each connected HMI device.

   You must specify port numbers in hexadecimal values.

Internal Ethernet port of the Q0xUDEH CPU

Procedure

1. Assign the following settings in the "Internal Ethernet Port" dialog:
   - IP address:
     IP address
   - Communication data code:
     Binary code
   - Enable online changes
2. Assign the following settings in the "Open settings" dialog:
   - Protocol:
     TCP
   - Open system:
     MC-Protocol
   - Host station Port No. (HEX)
     Port number

   Note

   The port number chosen in the communication module must match the port number in WinCC. A connection with a port number must be assigned for each connected HMI device.

Performance features of communication

Permitted data types for Mitsubishi MC TCPI/IP

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
4-bit block	M, X, Y, B, F	1 byte
8-bit block	M, X, Y B, F	1 byte
12-bit block	M, X, Y B, F	2 bytes
16-bit block	M, X, Y B, F	2 bytes
20-bit block	M, X, Y B, F	4 bytes
24-bit block	M, X, Y B, F	4 bytes
28-bit block	M, X, Y B, F	4 bytes
32-bit block	M, X, Y B, F	4 bytes
Bool	M, D, X, Y B, F	1-bit
DInt	D, W	4 bytes

Data type	Operand type	Length
DWord	D, C, W	4 bytes
Int	D, W	2 bytes
Real 1)	D, W	4 bytes
String 1)	D	1 to 80 characters
Word	D, T, C, W	2 bytes

1) The "String" and "Real" data types are not available for all CPUs.
2) Operand types B, F and W are only available for CPU type "Q".

Note

Note the following for write accesses:

Tags can only be written if "Enable online changes" or "Enable write operations during RUN" was selected when parameterizing the Mitsubishi communication modules.

For data type "Bool" in operand type "D", the entire word is written back to the PLC following a change to the specified bit. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

Note

Array elements in I/O fields cannot be used in communication with a Mitsubishi PLC.

Supported CPU types for Mitsubishi MC TCP/IP

CPU types

The following CPU types are supported for configuring the Mitsubishi MC TCP/IP communication driver.

- FX3 series
  - FX 3G / FX 3G with communication modul FX3U-ENET
  - FX 3U / FX 3U with communication modul FX3U-ENET
  - FX 3UC / FX 3UC with communication modul FX3U-ENET
- Q series
  - Q-Series with QJ71E71-100 communication module
- iQ series / QnUD
  - QnUDEHCPU with built in ethernet module

Addresses for Mitsubishi MC TCP/IP

Address areas for connections via Mitsubishi MC TCP/IP

The address area boundaries differ for the different series; refer to the Mitsubishi Computerlink manuals for this information.

Examples of address area boundaries dependent on the CPU and communication format:

Name	Operand type	Max. address FX3	Max. address Q-Series
Output/Input	Y/X	Octal X/Y 0 - 777	HEX X/Y 0 - 7FF
Bit memory	M	M0 - M3071 and M8000 - M8255	M/L/S 0 - 8191
Data register	D	D0 - 7999 D8000 - D8255	D0 - 8191 D9000 - D9255 becomes SD1000 - SD1255
Counter	C	C0 - 255	C0 - 1023
Timer	T	T0 - 255	T0 - 2047
Link register	W	--	Hex: W0 - FFF
Link flag	B	--	Hex: B0 - FFF
Error flag	F	--	F0 - 2047

Address areas for Mitsubishi MC TCP/IP

FX3

Address areas	Data types														
	Bool	Int	Word	DInt	DWord	Real	String	4-bit block	8-bit block	12-bit block	16-bit block	20-bit block	24-bit block	28-bit block	32-bit block
M	M0 - M9999	--	--	--	--	--	--	M0 - M9996	M0 - M9992	M0 - M9988	M0 - M9984	M0 - M9980	M0 - M9976	M0 - M9972	M0 - M9968
D	D0.0 - D9999.15	D0 - D9999	D0 - D9999	D0 - D9999	D0 - D9999	D0 - D9999	D0 - D9999	--	--	--	--	--	--	--	--
T	--	--	T0 - T999	--	--	--	--	--	--	--	--	--	--	--	--
C	--	C0 - C999	C0 - C999	C0 - C998	C0 - C998	--	--	--	--	--	--	--	--	--	--
X	X0 - X777	--	--	--	--	--	--	X0 - X774	X0 - X770	X0 - X764	X0 - X760	X0 - X754	X0 - X750	X0 - X744	X0 - X740
Y	Y0 - Y777	--	--	--	--	--	--	Y0 - Y774	Y0 - Y770	Y0 - Y764	Y0 - Y760	Y0 - Y754	Y0 - Y750	Y0 - Y744	Y0 - Y740

Q

Address areas	Data types														
	Bool	Int	Word	DInt	DWord	Real	String	4-bit block	8-bit block	12-bit block	16-bit block	20-bit block	24-bit block	28-bit block	32-bit block
M	M0 - M9999	--	--	--	--	--	--	M0 - M9996	M0 - M9992	M0 - M9988	M0 - M9984	M0 - M9980	M0 - M9976	M0 - M9972	M0 - M9968
F	F0 - F9999	--	--	--	--	--	--	F0 - F9996	F0 - F9992	F0 - F9988	F0 - F9984	F0 - F9980	F0 - F9976	F0 - F9972	F0 - F9968
B	B0 - BFFFF	--	--	--	--	--	--	B0 - BFFFC	B0 - BFFF8	B0 - BFFF4	B0 - BFFF0	B0 - BFFEC	B0 - BFFE8	B0 - BFFE4	B0 - BFFE0
D	D0.0 - D65534.15	D0 - D65534	D0 - D65534	D0 - D65533	D0 - D65533	D0 - D65533	D0 - D65534	--	--	--	--	--	--	--	--
T	--	--	T0 - T2047	--	--	--	--	--	--	--	--	--	--	--	--
C		C0 - C2047	C0 - C2047	C0 - C2046	C0 - C2046	--	--	--	--	--	--	--	--	--	--
W	--	W0 - WFFFF	W0 - WFFFF	W0 - WFFFE	W0 - WFFFE	W0 - WFFFE	--	--	--	--	--	--	--	--	--
X	X0 - XFFFF	--	--	--	--	--	--	X0 - XFFFC	X0 - XFFF8	X0 - XFFF4	X0 - XFFF0	X0 - XFFEC	X0 - XFFE8	X0 - XFFE4	X0 - XFFE0
Y	Y0 - YFFFF	--	--	--	--	--	--	Y0 - YFFFC	Y0 - YFFF8	Y0 - YFFF4	Y0 - YFFF0	Y0 - YF-FEC	Y0 - YFFE8	Y0 - YFFE4	Y0 - YFFE0

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Mitsubishi FX

Configuring a connection via Mitsubishi FX

Introduction

You configure a connection to a PLC with a Mitsubishi FX communication driver in the "Connections" editor of the HMI device.

The Mitsubishi FX protocol is also referred to as the Mitsubishi PG protocol.

The interfaces are named differently depending on the HMI device.

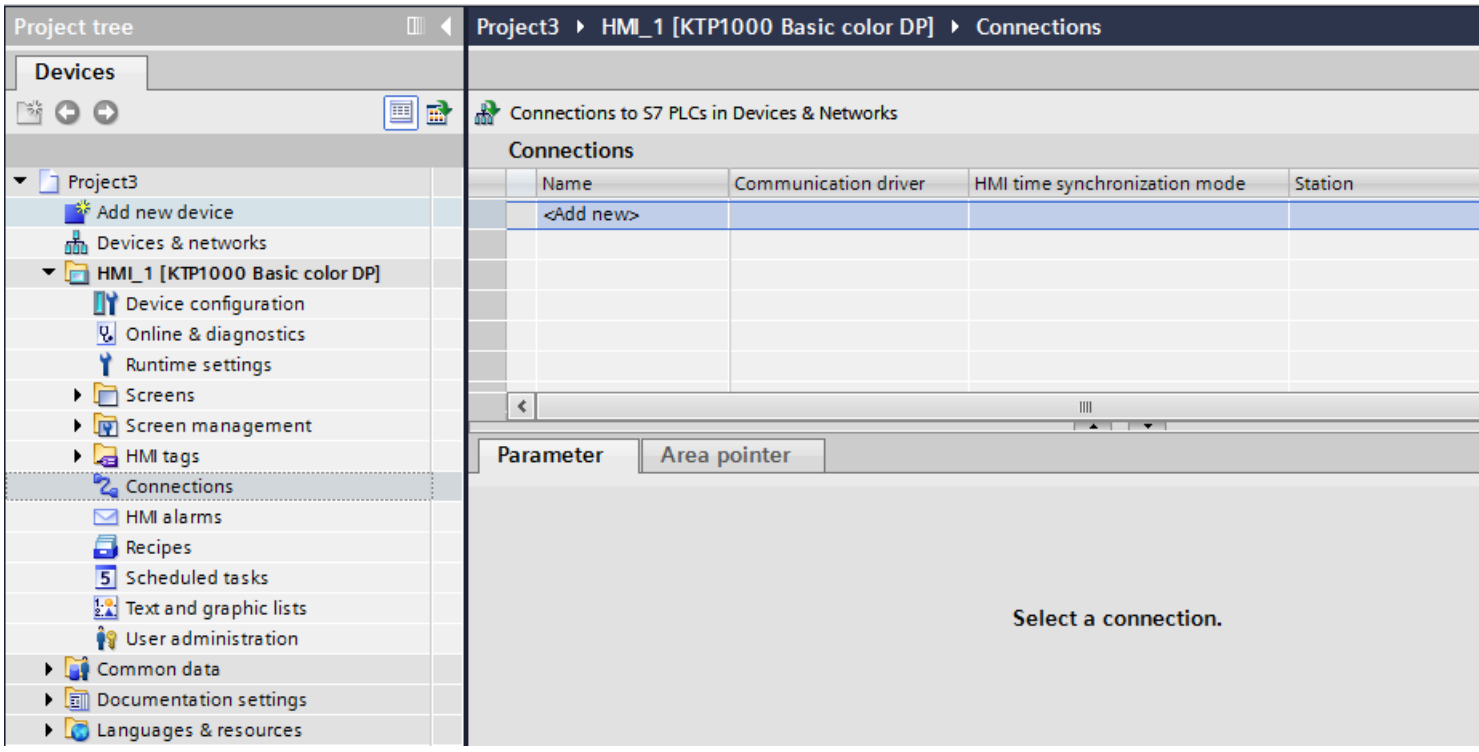
Requirements

- A project is open.
- An HMI device has been created.

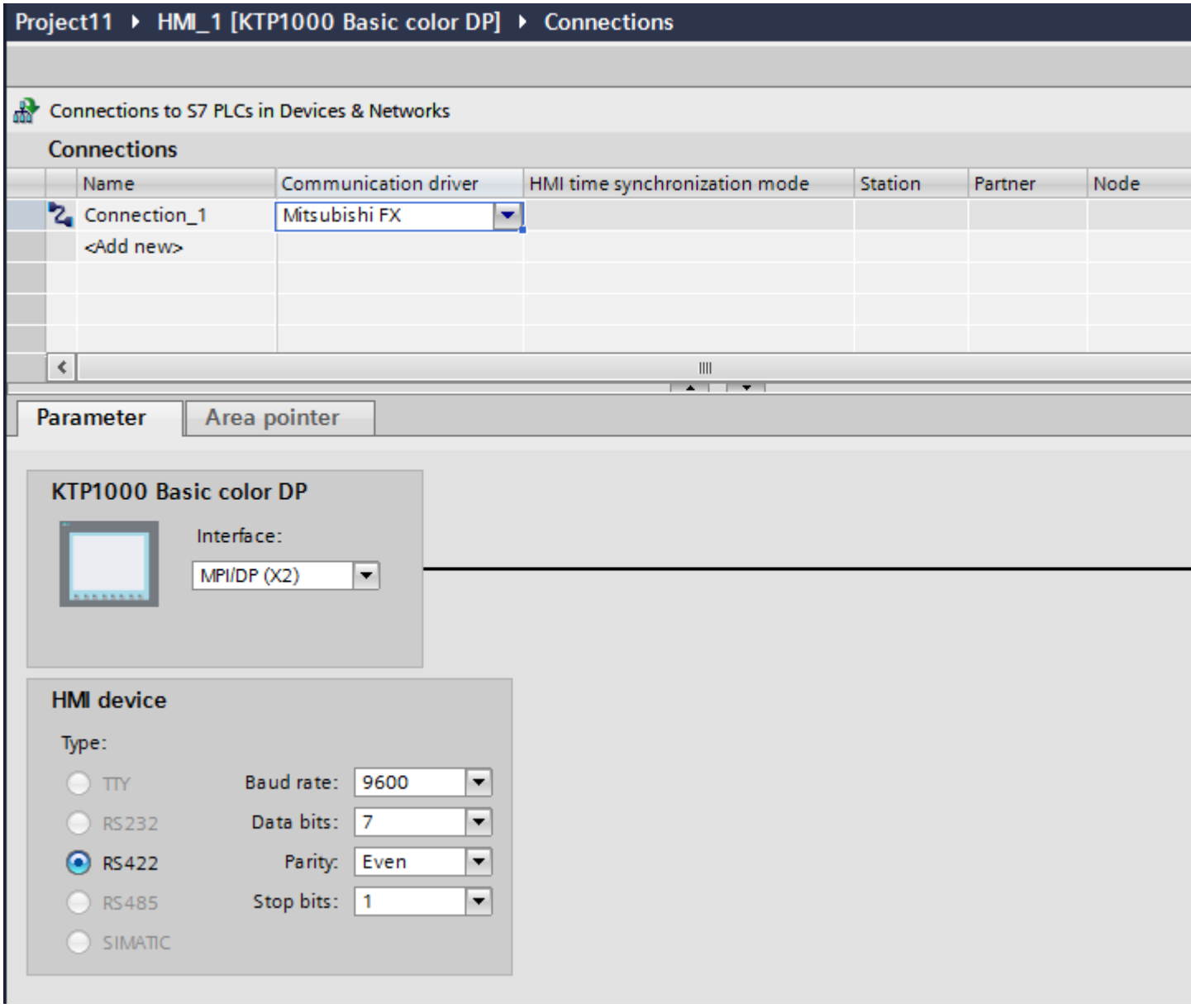
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Mitsubishi FX" driver.



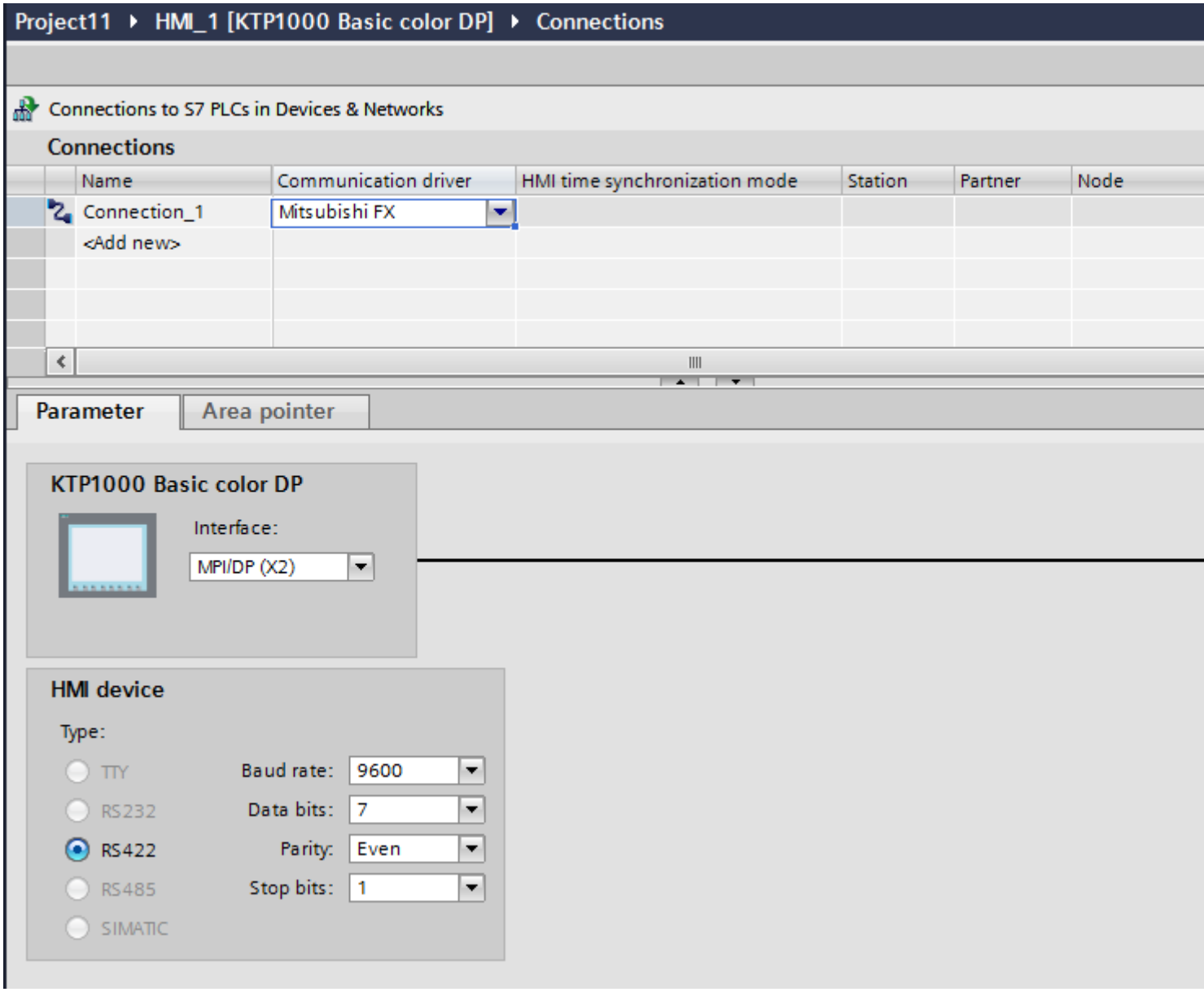
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Mitsubishi FX)

Parameters to be set

To assign the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select an interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

- "Type"
  Specifies the physical connection used.

Note

If you use the IF1B interface, you must switch over the RS422 receive data and the RTS signal additionally by 4 DIL-switches on the back of the HMI device.

Parameters for the PLC

- Baud rate: For "Baud rate", select the transmission speed between the HMI device and PLC. Select the baud rate "9600".
- Data bits: Under "Data bits", select "7 bits".
- Parity: Under "Parity", select "Even".
- Stop bits: Under "Stop bits", select "1 bit".

Connecting HMI device to PLC

Connections via Mitsubishi FX

Connection

Connect the HMI device to the programming interface of the CPU (RS 422) (see documentation of the PLC).

The connection between the HMI device and the Mitsubishi PLC is basically restricted to setting the interface parameters. Special blocks for the connection are not required in the PLC.

Connecting cable

The following connecting cables are available to connect the HMI device to the PLC.

Interface to HMI device or adapter	Mitsubishi Electric PLC via FX protocol
	FX1n, Fx2n, Mini DIN, 8–pin
RS 232, 9-pin	Mitsubishi SC-09 [1]
RS 422, 9-pin	Connecting cable RS422-2P

[1] Since the Mitsubishi PLCs communicate via RS 422 as a standard, the Mitsubishi programming cable SC–09 with integrated RS 422/RS 232 adaptor is necessary for connecting a HMI device via RS 232.

Note

Applies only to RS 232:

Cable length is restricted to 0.32 m.

Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Mitsubishi FX".

Communication types

Approved communication types

- Only applies for Mitsubishi FX(PG protocol):
  The point-to-point connection from a HMI device to an approved Mitsubishi FX–CPU via Mitsubishi FX (PG protocol := protocol for access to the program and memory elements of the FX series PC CPU version V1.21 and after) is system-tested and approved by Siemens AG.
- Only applies for Mitsubishi MC TCP/IP:
  The following communication types are system-tested and approved:
  - Point-to-point connection to the approved PLCs
  - Multipoint connection from a HMI device with up to 4 PLCs with the respectively approved PLCs. CPU types (FX3 and Q) can be mixed.

    Note

    The HMI device is a client and the PLC must operate as a server.

Connectable PLCs

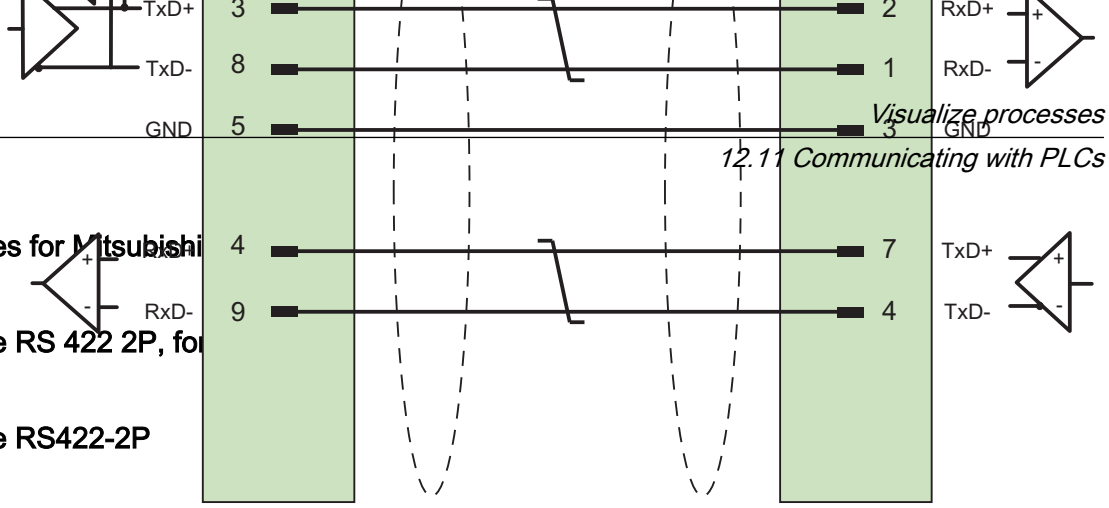
Connections can be implemented for the following Mitsubishi PLCs:

	Mitsubishi FX (PG protocol)	Mitsubishi MC TCP/IP
PLC		
MELSEC FX1n, FX2n	Yes	No
MELSEC FX3U, FX3UC, FX3G with communication module FX3U-ENET	No	Yes
MELSEC System Q • Q-series with the communication module QJ71E71-100 • QnUDEH CPU with Ethernet interface onboard	No	Yes

Connecting cables for Mitsubishi

Connecting cable RS 422 2P, for

Connecting cable RS422-2P



Shield with large-area contact to housing at both ends
Cable: 3 x 2 x 0.14 mm², shielded,
max. length 500 m

Performance features of communication

Permitted data types for Mitsubishi FX

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
4-bit block	M, X, Y	1 byte
8-bit block	M, X, Y	1 byte
12-bit block	M, X, Y	2 bytes
16-bit block	M, X, Y	2 bytes
20-bit block	M, X, Y	4 bytes
24-bit block	M, X, Y	4 bytes
28-bit block	M, X, Y	4 bytes
32-bit block	M, X, Y	4 bytes
Bool	D, M, X, Y	1-bit
DWord	D, C-32 bit	4 bytes
Real	D	4 bytes
String	D	1 to 50 characters
Word	D, T, C-16 bit	2 bytes

Note

Note the following for write accesses:

For data type "Bool" in operand type "D", the entire word is written back to the PLC following a change to the specified bit. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

Note

Array elements in I/O fields cannot be used in communication with a Mitsubishi PLC.

Supported CPU types for Mitsubishi FX

CPU types

The following CPU types are supported for configuring the Mitsubishi FX communication driver.

- FX1 series
  - FX1n
- FX2 series
  - FX2n

Configurable address areas for Mitsubishi FX

Permitted address areas

The following address areas can be configured for the "Mitsubishi FX" communication driver

X	Input	Octal 0- 377
Y	Output	Octal 0- 377
M	Bit memory	0...9999
D	Data register	0...999
C	Counter	0...199
T	Timer	0...25

Address areas for Mitsubishi FX

FX1n and FX2n

Address areas	Data types												
	Bool	Word	DWord	Real	String	4-bit block	8-bit block	12-bit block	16-bit block	20-bit block	24-bit block	28-bit block	32-bit block
M	M0 - M9999					M0 - M9996	M0 - M9992	M0 - M9988	M0 - M9984	M0 - M9980	M0 - M9976	M0 - M9972	M0 - M9968
D	D0.0 - D999.15	D0 - D999	D0 - D998	D0 - D998	D0 - D998								
T		T0 - T255											

Address areas	Data types												
	Bool	Word	DWord	Real	String	4-bit block	8-bit block	12-bit block	16-bit block	20-bit block	24-bit block	28-bit block	32-bit block
C-16-Bit		C-16-Bit 0 - C-16-Bit 199											
C-32-Bit			C-32-Bit 200 - C-32-Bit 255										
X	X0 - X255					X0 - X252	X0 - X248	X0 - X244	X0 - X240	X0 - X236	X0 - X232	X0 - X228	X0 - X224
Y	Y0 - X255					Y0 - Y252	Y0 - Y248	Y0 - Y244	Y0 - Y240	Y0 - Y236	Y0 - Y232	Y0 - Y228	Y0 - Y224

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Mitsubishi

Area pointers for connections via Mitsubishi communication drivers

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section "Data exchange using area pointers".

Special considerations for connections via Mitsubishi communication drivers

You can configure the following area pointers

Area pointers	Mitsubishi MC TCP/IP	Mitsubishi FX
Screen number	Yes	Yes
Date/time	Yes	Yes
Date/time PLC	Yes	Yes
Coordination	Yes	Yes
Project ID	Yes	Yes
Job mailbox	Yes	Yes
Data record	Yes	Yes

Restrictions for Mitsubishi FX and MC TCP/IP

The following restrictions apply for configuring area pointers.

CPU type	Data types	Operand type
FX/FX3	Int, Word	D
Q	Int, Word	D

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Auto-Hotspot

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

See also

Displaying tags with Runtime Advanced and Panels (Page 4270)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

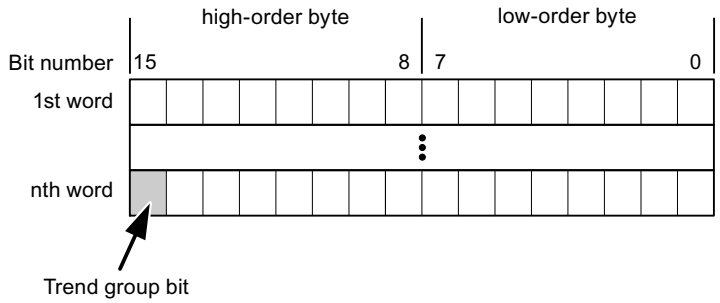
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Restrictions to the trend control

Data types for Mitsubishi MC TCPI/IP

For trend control you can create tags and array tags of the operand type "D" with the data type "Word" or "Int".

Data types for Mitsubishi FX

For trend control you can create tags and array tags of the operand type "D" with the data type "Word".

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with Alarms (Page 4293)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a Mitsubishi communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
FX1n, FX2n, FX3 series, Q-Series, iQ-Series	Word, Int [1)]	4-bit block, 8-bit block, 12-bit block, 16-bit block, 20-bit block, 24-bit block, 28-bit block, 32-bit block, Word, DWord, Int [1)], DInt [1)], Real,
[1)] Not for Mitsubishi FX communication driver		

How the bit positions are counted

For connections with a Mitsubishi communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In Mitsubishi PLCs	15							8	7							0
In WinCC you configure:	15							8	7							0

Restrictions on alarms

- Mitsubishi MC TCP/IP
  Only tags of operand type "D" and data types "Word" and "Int" are permitted as trigger tags for discrete alarms. You can use array tags (operand type: "D"; data types: "ARRAY [x..y] of Word" or "ARRAY [x..y] of Int") for discrete alarms.
- Mitsubishi FX
  Only tags of operand type "D" and data type "Word" are permitted as trigger tags for discrete alarms. You can use array tags (operand type "D"; data type "ARRAY [x..y] of Word") for discrete alarms."

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

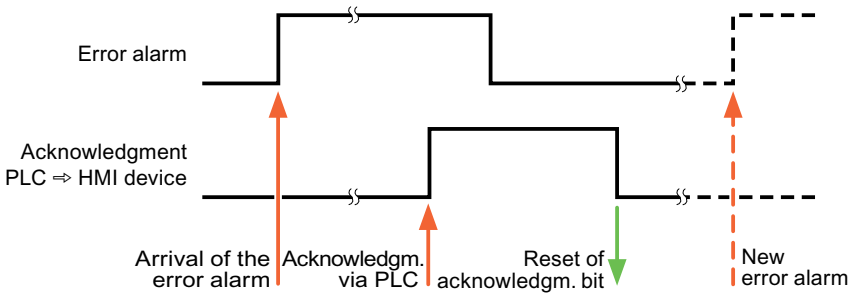
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

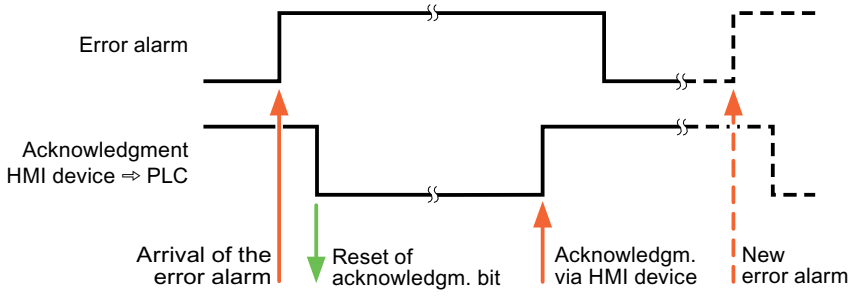
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



LED mapping

Function

Keyboard devices have LEDs in the function keys. These LEDs can be activated from the PLC. Thus, it is possible, for example, to signal to the user which key he should press in a given situation by lighting up an LED.

Note

The LED function cannot be configured for Basic Panels.

Requirements

In order to activate an LED, an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration.

LED assignment

The assignment of the individual LEDs to the bits in the LED tags is specified when the function keys are configured. In this process, the "LED tag" and the assigned "Bit" are specified for each function key in the "General" group of the properties window.

The "Bit" bit number designates the first of two consecutive bits that control the following LED statuses.

Bit n+ 1	Bit n	LED function	
		All Mobile Panels, all Operator Panels, all Multi Panels, all Comfort Panels	Panel PCs
0	0	Off	Off
0	1	Rapid flashing	Flashing
1	0	Slow flashing	Flashing
1	1	On permanently	On permanently

Modicon Modbus

Modicon Modbus communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Modicon Modbus communication drivers.

The following communication drivers are supported:

- Modicon Modbus TCP/IP
- Modicon Modbus RTU

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags
  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.
- Area pointers
  Area pointers are used to exchange specific data and are only set up when these data are used.

Modicon Modbus TCP/IP

Configuring a connection via Modicon Modbus TCP/IP

Introduction

You configure a connection to one of the PLCs with Modicon Modbus TCP/IP communication driver in the "Connections" editor of the HMI device.

The Ethernet interfaces are named differently depending on the HMI device.

Example: PROFINET interface corresponds to the Ethernet interface

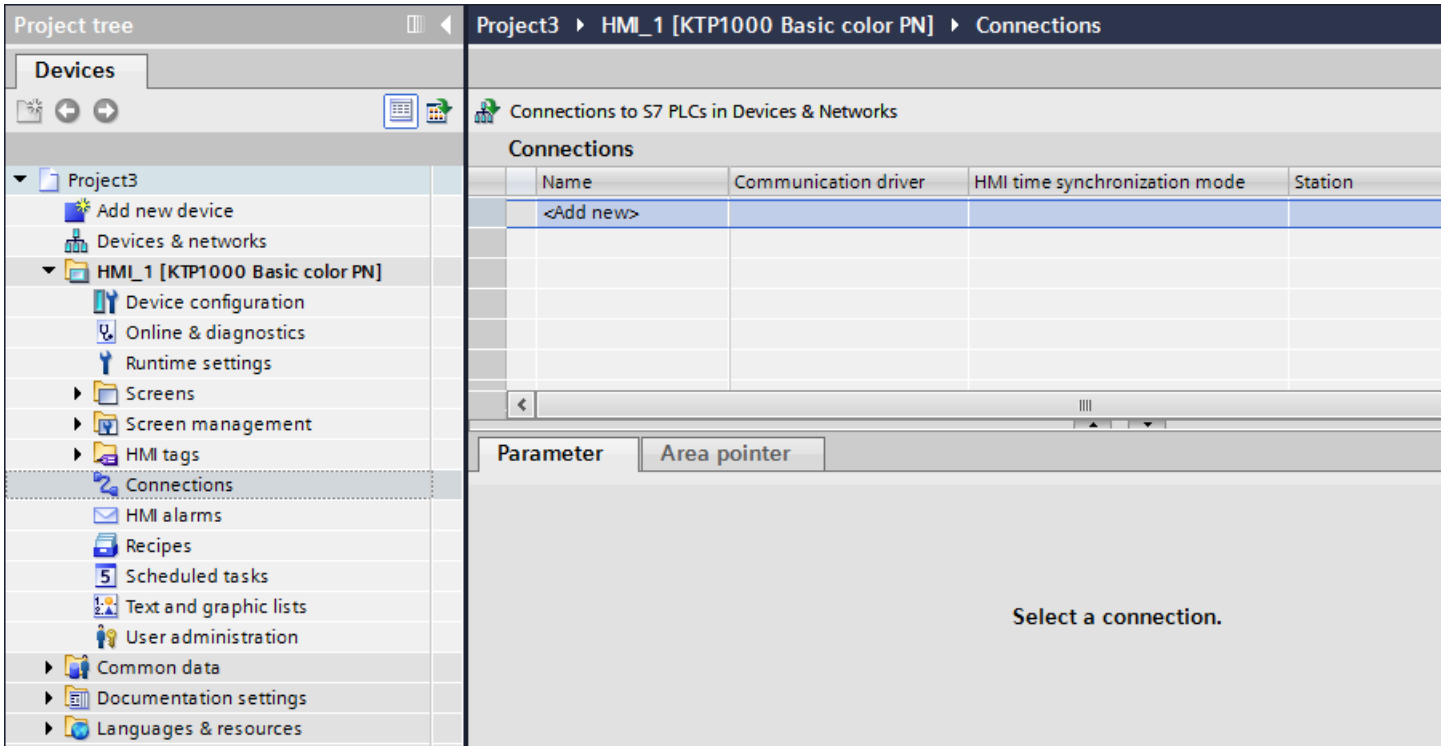
Requirements

- A project is open.
- An HMI device has been created.

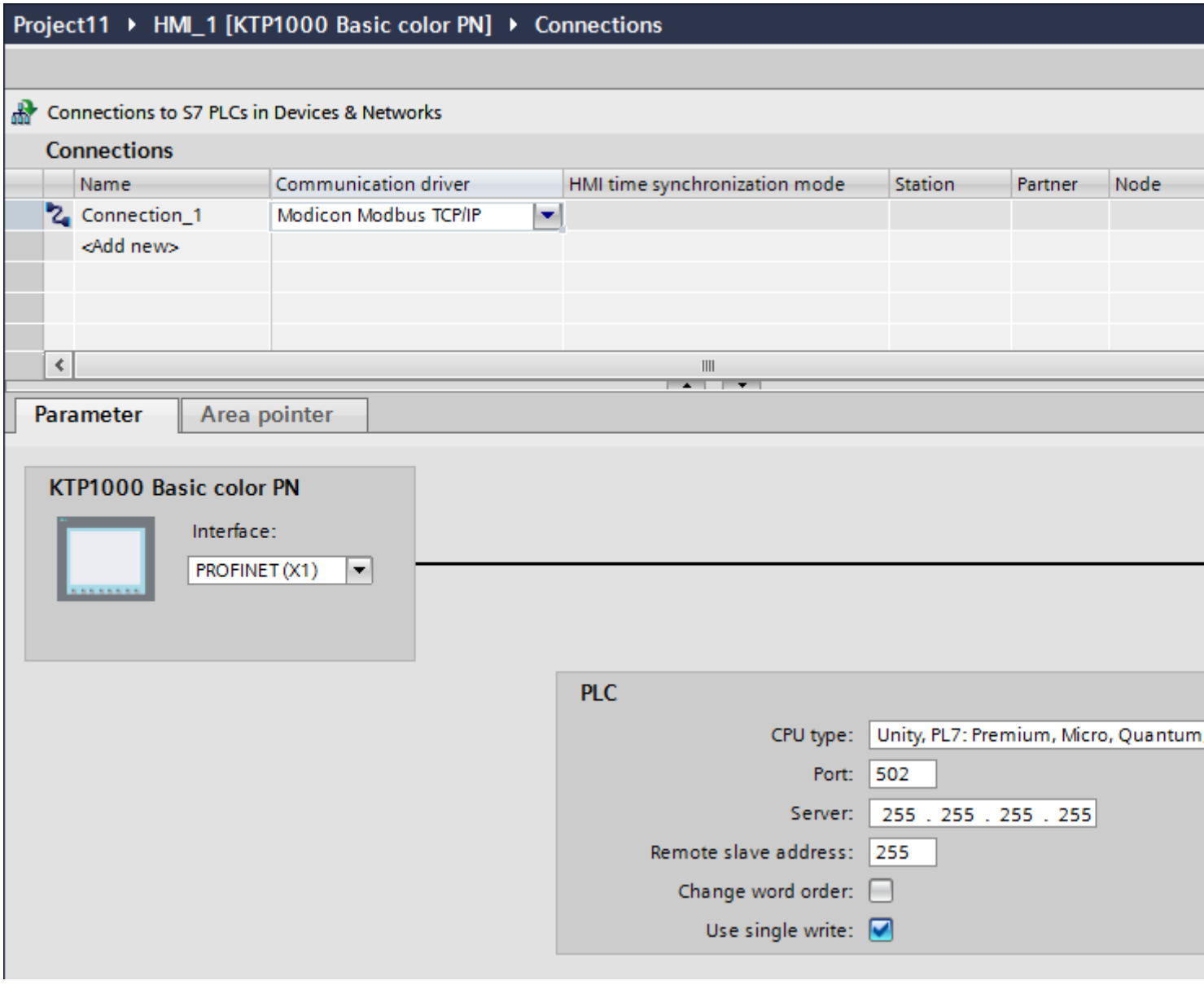
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. Select the "Modicon Modbus TCP" driver in the "Communication driver" column.



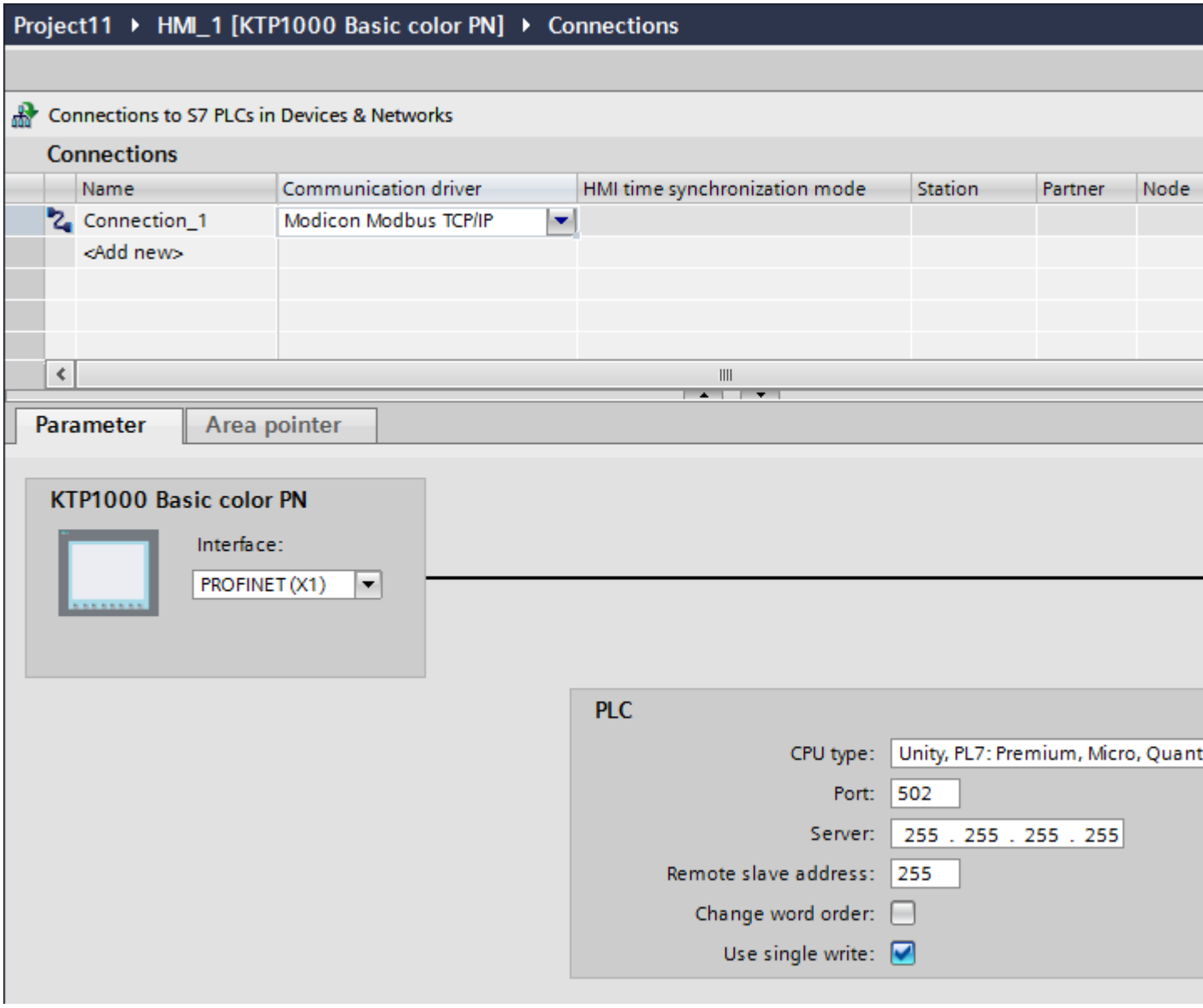
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Modicon Modbus TCP/IP)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select only one interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC. The IP address is transferred to the HMI device during project transfer.

Note

The IP address in the control panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

The IP address already set up in the control panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

To set up the IP address of the HMI device:

1. Click the HMI device.
2. Open the "Device configuration" editor.
3. Click the Ethernet interface.
4. Assign the IP address in the inspector window under:
   "General > PROFINET interface > Ethernet addresses"

Parameters for the PLC

- "CPU type"
  For "CPU type", you set the Modicon PLC to which the HMI device is connected.
- "Port"
  For "Port", you set the port that is used for the TCP/IP connection. The port used by the Modicon PLCs is 502.
- "Server"
  You set the IP address or host name of the PLC under "Server". Only the IP address can be used on a Basic Panel.
- "Distributed slave address"
  For "Distributed slave address", set the slave address of the remote PLC, but only if a bridge is used.
  If a bridge is not used, the default value "255" (or "0") must be retained.

- "Change word order"
  The "Change word order " parameter only affects the word order of the 32-bit values display. The setting pertains to the data types Double, Double+/-, and Float. The byte order cannot be changed.
  - "Change word order" not activated
    The most significant byte is sent first.
    For double words, the least significant word is sent before the most significant word.
    This setting has been system-tested for all approved PLCs.
  - "Change word order" activated
    The most significant byte is sent first.
    For double words, the most significant word is sent before the least significant word.

    Note

    This setting must be used for the SIEMENS SENTRON PAC3200 and PAC4200 multi-function meters and can be used for PLCs of other manufacturers.

- "Use single write"
  If you deselect this function, only function codes 15H and 16H are used for writing into the PLC.
  If this function remains selected, the function codes 05H, 06H, 15H, and 16H are used.

Connecting HMI device to PLC

Connections via Modicon Modbus TCP/IP

Connection

The HMI device can be connected to the Modicon Modbus PLC using the following components:

- Existing Ethernet network that also contains the PLCs
- Cross-over Ethernet cable connected directly to the Ethernet interface of the CPU or the communication module

The connection of the HMI device to a Modicon Modbus PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Note

Timeout response with TCP/IP (Ethernet)

Due to the use of the TCP/IP protocol, the breakdown of a connection is detected at the earliest after approximately one minute. Communication failure cannot be reliably detected if no tags are requested, for example, no output tags in the current screen.

Configure area pointer coordination for each PLC. This setting ensures that a communication failure is recognized after approximately two minutes, even in the aforementioned scenario.

Communication types

Approved communication types

The following communication types are system-tested and approved:

- Point-to-point coupling:
- Multiple point coupling of a HMI device (Modbus TCP/IP Client) with up to 4 PLCs, each with different couplings. CPU types can be mixed.
  The following couplings are possible:
  - Coupling the Ethernet CPU interface of the TSX Unity Quantum.
  - Coupling via the communication modules for Ethernet 140 NOE 771 01 for the TSX Quantum and TSX Unity Quantum series
  - Coupling via the Ethernet interface of the 171 CCC 980 30 CPU adapter of the Momentum series
  - Coupling the Ethernet CPU interface of the TSX Unity Premium.
  - Coupling via the Ethernet TCP/IP connect module TSX ETY 110 for the TSX Premium and TSX Unity Premium series
  - Coupling via the Ethernet TCP/IP connect module TSX ETY 410 for the Micro series
  - Coupling via the Ethernet TCP/IP Modbus Plus Bridge 174 CEV 200 40 to the Modbus Plus interface of the Compact, the TSX Quantum and the TSX Unity Quantum series

Via the TCP/IP Modbus Plus Bridge, 174 CEV 200 40, the PLCs can be accessed at their Remote Slave Address via the Ethernet interface of this bridge.

Note

Integration of the HMI device in a Modbus network via a bridge is not possible. The HMI device is the Modbus master.

Restrictions

The coupling of the HMI device to PLCs of other manufacturers who offer a Modbus TCP/IP interface is not system-tested and thus, not enabled.

However, if another PLC is to be used, observe the following instructions:

- Use the following CPU types, because these operate without address offset and in the usual bit count manner.
  - Unity, PL7: Premium, Micro, Quantum, M340
- The following function codes are used for the respective data areas:

Reading function codes		Address range	
01	ReadCoilStatus	0x / %M	DIGITAL_OUT
02	ReadInputStatus	1x / %I	DIGITAL_IN
03	ReadHoldingRegisters	4x / %MW	USERDATA

Reading function codes		Address range	
04	ReadInputRegisters	3x / %IW	ANALOG_IN
20 (14Hex)	ReadGeneralReference	6x / –	EXTENDEDMEMORY (not for all CPUs)

Writing function codes		Address range	
06 [1]	PresetSingleRegister	4x / %MW	USERDATA Single
16 (10Hex)	PresetMultipleRegisters	4x / %MW	USERDATA Multiple
05 [1]	ForceSingleCoil	0x / %M	DIGITAL_OUT with BIT
15 (0FHex)	ForceMultipleCoils	0x / %M	DIGITAL_OUT with 16 BIT GROUP
21 (15Hex)	WriteGeneralReference	6x / –	EXTENDEDMEMORY (not for all CPUs)

[1] Select use with "Use single write".

Connectable PLCs

Connections can be implemented for the following Modicon Modbus PLCs:

Modicon Modbus PLC	Supported protocol	
Modicon Modbus RTU [2]	Modicon Modbus TCP/IP	
TSX Compact	x	x [1]
TSX Quantum	x	x
Momentum	x	x
Premium	-	x
Micro	-	x
M340 20x0 (without 2010)	-	x

[1] Only via Ethernet TCP/IP Modbus Plus Bridge

[2] Communication via RS 232 is tested and enabled for the PLC. In the HMIs that only have a RS 422/485 interface, the RS 422/232 converter with the order number 6AV6 671-8XE00-0AX0 was tested and enabled.

Performance features of communication

Permissible data types for Modicon Modbus TCP/IP

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Note

If you change the Modicon Modbus RTU communication driver to Modicon Modbus TCP/IP, the string in the "String" data type may be different.

Permitted data types for CPU type "Unity, PLC: Premium, Micro, Quantum M340"

Data type	Operand type	Length
+/- Double	%MW	4 bytes
+/- Int	%MW, %IW	2 bytes
16-bit group	%MW, %I	2 bytes
ASCII	%MW	0 to 80 characters
Bit	%MW, %IW, %M, %I	1-bit
Double	%MW	4 bytes
Float	%MW	4 bytes
Int	%MW, %IW	2 bytes

Note

The ranges "%I" and "%IW" are not supported for the following CPU types:

- Premium
- Micro
- M340

Permitted data types for CPU type "Concept, ProWORX: Compact, Quantum, Momentum"

Data type	Operand type	Length
+/- Double	4x, 6x	4 bytes
+/- Int	3x, 4x, 6x	2 bytes
16-bit group	0x, 1x	2 bytes
ASCII	4x, 6x	0 to 80 characters
Bit	0x, 1x, 3x, 4x, 6x	1-bit
Double	4x, 6x	4 bytes

Data type	Operand type	Length
Float	4x, 6x	4 bytes
Int	3x, 4x, 6x	2 bytes

Bit counting method

The usual bit counting method "16 LSB - 1 MSB" in the following CPU types is only used in the "HMI tags" editor with the selected "Bit" data type:

- Concept, ProWORX: Compact, Quantum, Momentum

The following bit location assignment applies:

	Left byte								Right byte							
Counting with tags	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Format for "Signed"

The placeholder "+/-" stands for the data types "Signed Int" and "Signed Double".

Supported CPU types for Modicon Modbus TCP/IP

CPU types

The following CPU types are supported for configuring the Modicon Modbus TCP/IP communication driver.

- Compact
- Momentum
- Quantum
  - Concept Quantum
  - Unity Quantum
- Micro
- Premium
- Modicon M340
  - 20x0 (except 2010)

Address areas for Modicon Modbus TCP/IP

Unity, PI7

Address areas	Data types							
	Bool	16 Bit Group	Int	+/- Int	DInt	+/- DInt	Float	ASCII
%1	%I0 - %I65535	%I65535 %I0 - %I65520	--	--	--	--	--	--
%M	%M0 - %M65535	%M65535 %M0 - %M65520	--	--	--	--	--	--
%IW	%IW0.0 - %IW65535.15	--	%IW0 - %IW65535	%IW0 - %IW65535	--	--	--	--
%MW	%MW0.0 - %MW65535.15	--	%MW0 - %MW65535	%MW0 - %MW65535	%MW0 - %MW65534	%MW0 - %MW65534	%MW0 - %MW65534	%MW0 - %MW65535

Concept, ProWORX

Address areas	Data types							
	Bool	16 Bit Group	Int	+/- Int	DInt	+/- DInt	Float	ASCII
0x	0x1 - 0x65535	0x1 - 0x65520	--	--	--	--	--	--
1x	1x100001 - 1x165535	1x100001 - 1x165520	--	--	--	--	--	--
3x	3x300001.1 - 3x365535.16	--	3x300001 - 3x365535	3x300001 - 3x365535	--	--	--	--
4x	4x400001.1 - 4x465535.16	--	4x400001 - 4x465535	4x400001 - 4x465535	4x400001 - 4x465534	4x400001 - 4x465534	4x400001 - 4x465534	4x400001 - 4x465535
6x	6x60000.1:1 - 6x69999.16:10	--	6x60000:1 - 6x69999:10	6x60000:1 - 6x69999:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69999:10

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Modicon Modbus RTU

Configuring a connection via Modicon Modbus RTU

Introduction

You configure a connection to a PLC with a Modicon Modbus RTU communication driver in the "Connections" editor of the HMI device.

The interfaces are named differently depending on the HMI device.

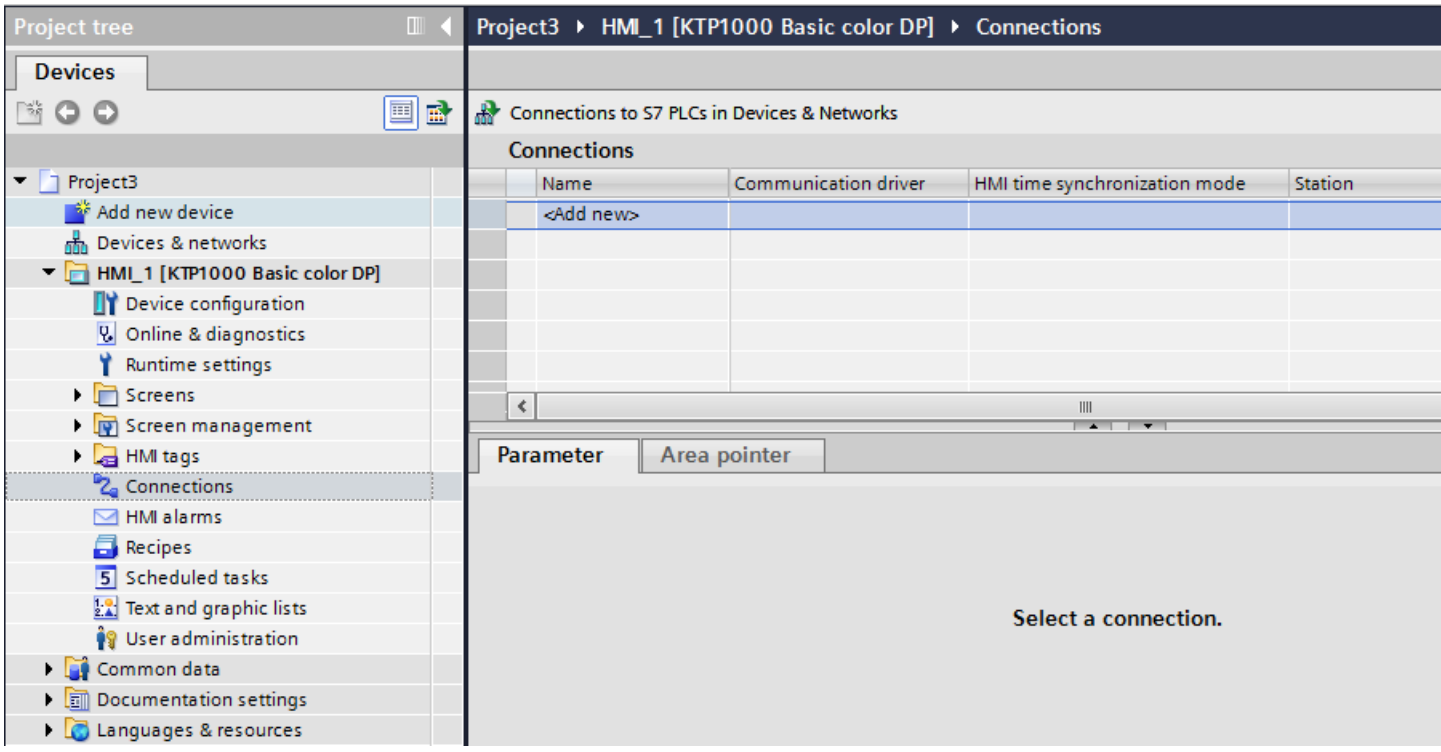
Requirements

- A project is open.
- An HMI device has been created.

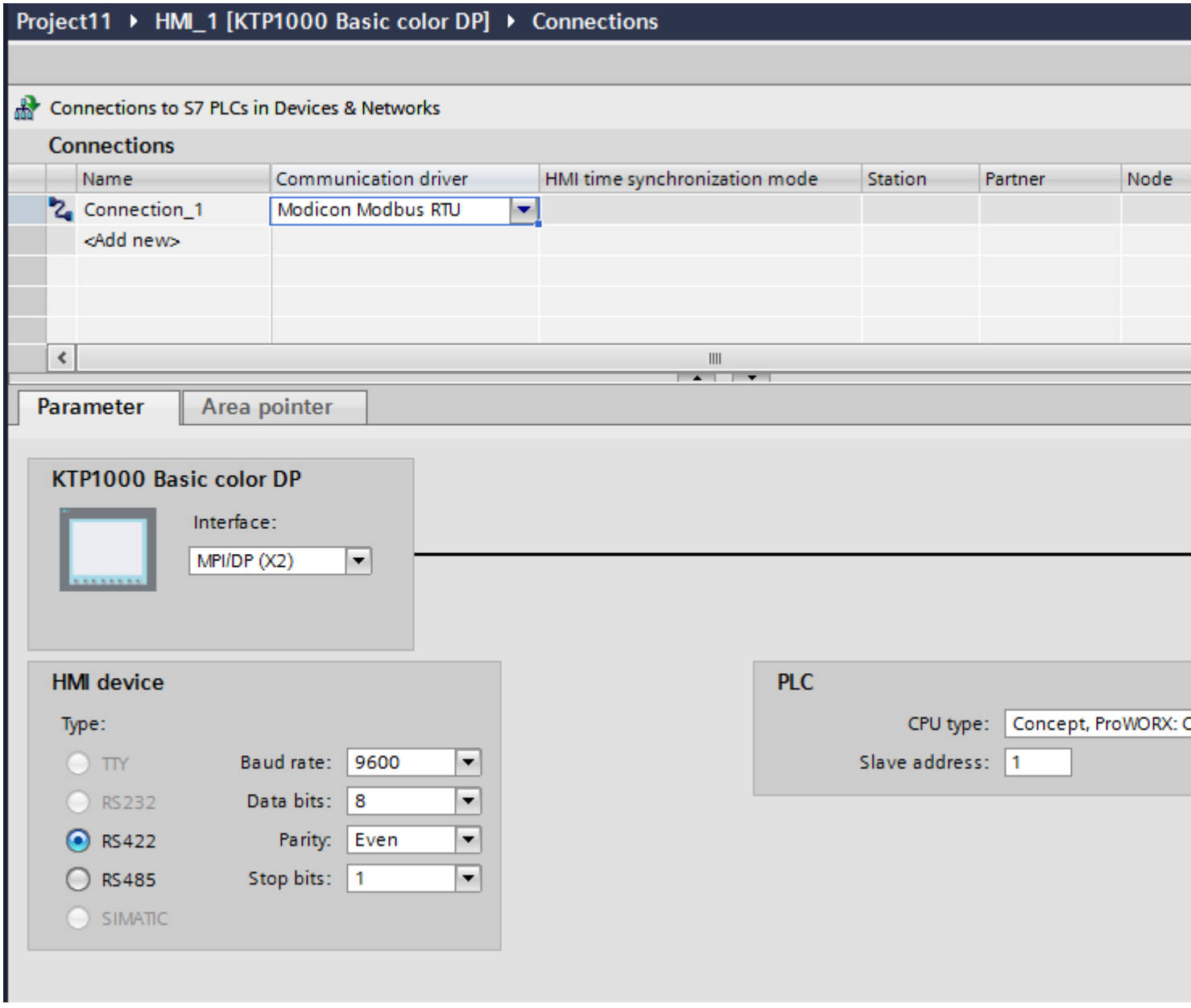
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Modicon Modbus RTU" driver.



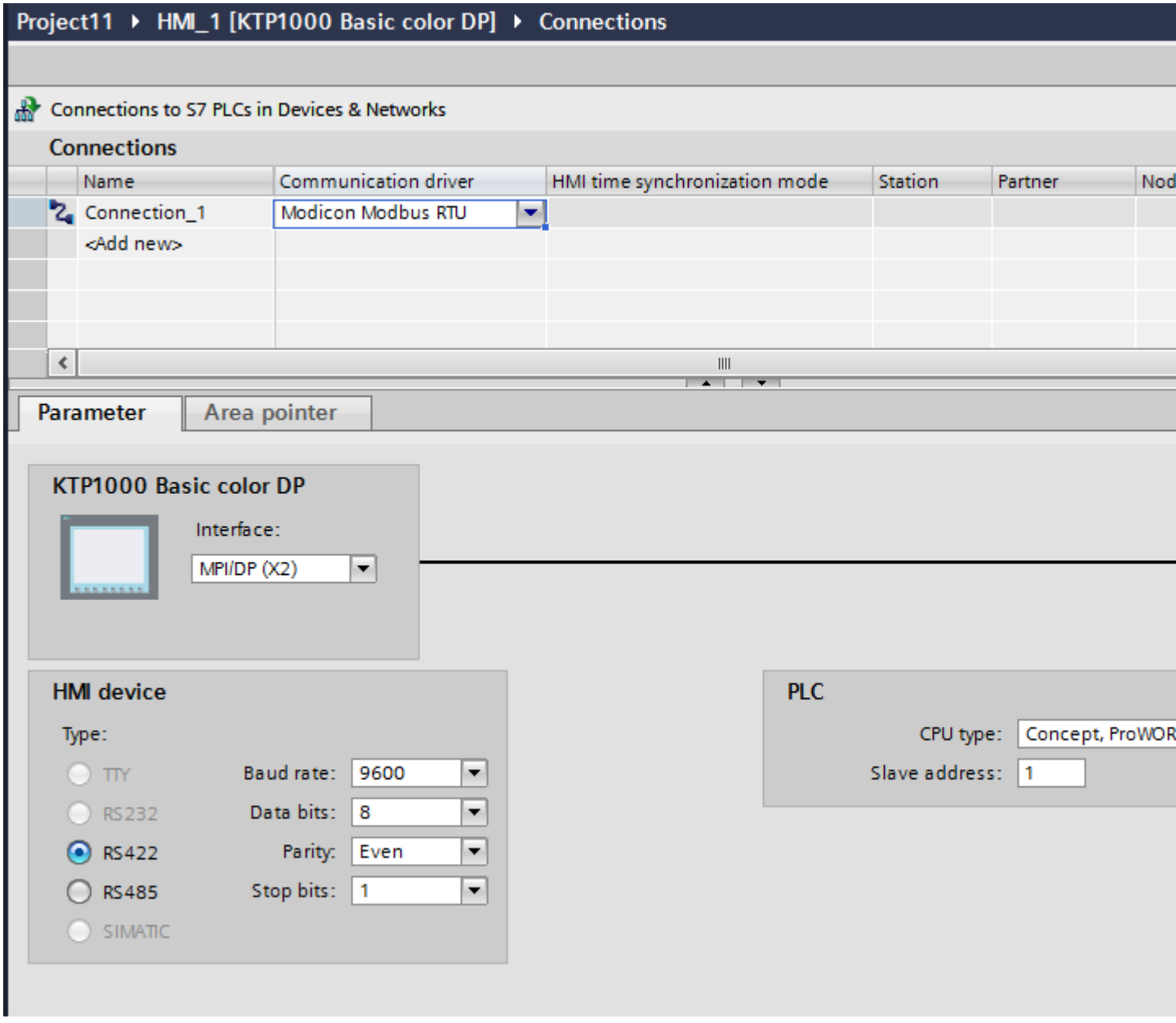
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Modicon Modbus RTU)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select an interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

- Type
  Only RS 232 is system-tested.
  No warranty is given for RS 485.

  Note

  RS 422 is only approved in combination with the RS 422-RS 232 converter.

  Order number: 6AV6 671-8XE00-0AX0

  Note

  If you use the IF1B interface, you must switch over the RS422 receive data additionally by 4 DIL-switches on the back of the HMI device.

- Baud rate
  For "Baud rate", select the transmission speed between the HMI device and Modicon PLC.
  A baud rate of 19200 or 9600 can be selected for the communication.
  A baud rate of 4800 can be selected for certain HMI devices.
- Data bits
  For "Data bits", only the value "8" can be selected.
- Parity
  For "Parity", you can choose from "None", "Even", and "Odd".
- Stop bits
  For "Stop bits", you can choose between 1 and 2 bits.

Parameters for the PLC

- CPU type
  For "CPU type", you set the Modicon PLC to which the HMI device is connected.
  You can select the following CPUs:
  – Concept, ProWORX: Compact, Quantum
- Slave address
  Under "Slave address" you set which slave address the CPU has.

Connecting HMI device to PLC

Connections via Modicon Modbus RTU

Connection

Connect the HMI device to the Modicon Modbus RTU interface of the Modicon Modbus RTU slave.

The connection of the HMI device to Modicon is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Connection cable

The following connecting cables are available to connect the HMI device to Modicon Modbus.

Interface to the HMI device	Modicon PLC		
	directly via Modbus interface (RS232) 9-pin Sub D male connector	Via MB Bridge (RS 232)	directly via Modbus interface (RS232) 8-pin RJ45 connector
RS 232, 9-pin	PP1	PP1	PP2

The cable pin assignments can be found in Section "Connecting cables for Modicon Modbus RTU".

Communication types

Approved communication types

The following communication types are system-tested and approved:

- Ppoint-to-point connection only via the RS-232 interface.
- Multipoint connection from a HMI device (Modbus-Master) with up to 4 PLCs: The HMI device must be connected with a Modbus Plus Bridge or a Compact, Momentum CPU or TSX Quantum CPU which is configured as a Modbus Plus Bridge.
- You connect the other PLCs via the Modbus Plus connection on the first PLC. The PLCs can be reached under their address via the bridge functionality of the first PLC.

  Note

  It is not possible to integrate the HMI device into a Modbus network because the HMI device is Modbus-Master.

- Integration of the HMI device into a Modbus Plus network via the "bridge mode" of the Compact, Momentum or Quantum (logical point-to-point communication of the HMI device with a Compact, Momentum or Quantum).

Restrictions

The connection of the HMI device to PLCs of other manufacturers which offer a Modicon Modbus interface is not system-tested and therefore not approved.

If you use another PLC nevertheless, observe the following information:

- These drivers only work for tags with the bit counting method typical for Modicon PLCs from left (bit1 = most significant bit) to right (bit16 = least significant bit in data type INT).
- The address offset displayed in the configuring is subtracted at protocol level in the message frame. E.g. in Holding Register 4x the offset "40001". The configured address "40006" therefore becomes address "5" in the message frame. The address (e.g. "5") transferred in the message frame is transformed to the PLC-specific address range in the different Non-Modicon PLCs.
- A reply message frame without "ExceptionCode" is expected within 500 ms.
- The following function codes are used for the respective data areas:

Reading function codes		Address range	
01	ReadCoilStatus	0x	DIGITAL_OUT
02	ReadInputStatus	1x	DIGITAL_IN
03	ReadHoldingRegisters	4x	USERDATA
04	ReadInputRegisters	3x	ANALOG_IN
20 (14Hex)	ReadGeneralReference	6x	EXTENDEDMEMORY (not for all CPUs)

Writing function codes		Address range	
06	PresetSingleRegister	4x	USERDATA Single
16 (10Hex)	PresetMultipleRegisters	4x	USERDATA Multiple
05	ForceSingleCoil	0x	DIGITAL_OUT with data type Bit
15 (0FHex)	ForceMultipleCoils	0x	DIGITAL_OUT with data type 16 bit group
21 (15Hex)	WriteGeneralReference	6x	EXTENDEDMEMORY (not for all CPUs)

Connectable PLCs

Connections can be implemented for the following Modicon Modbus PLCs:

Modicon Modbus PLC	Supported protocol	
Modicon Modbus RTU [2]	Modicon Modbus TCP/IP	
TSX Compact	x	x [1]
TSX Quantum	x	x
Momentum	x	x
Premium	-	x
Micro	-	x
M340 20x0 (without 2010)	-	x

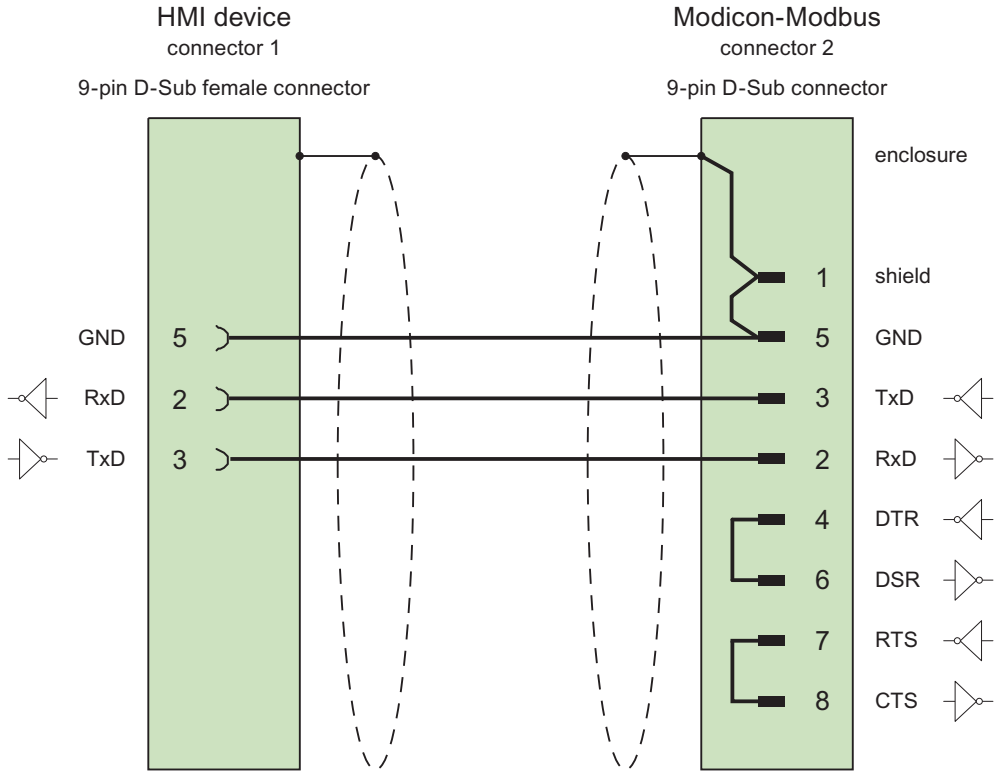
1) Only via Ethernet TCP/IP-Modbus Plus Bridge

2) Communication via RS 232 is tested and enabled for the PLC. In the HMI devices which only have an RS 422/485 interface, the RS 422/232 converter with the order number 6AV6 671-8XE00-0AX0 was tested and approved.

Connecting cables for Modicon Modbus RTU

Connecting cable PP1, RS-232, for Modicon

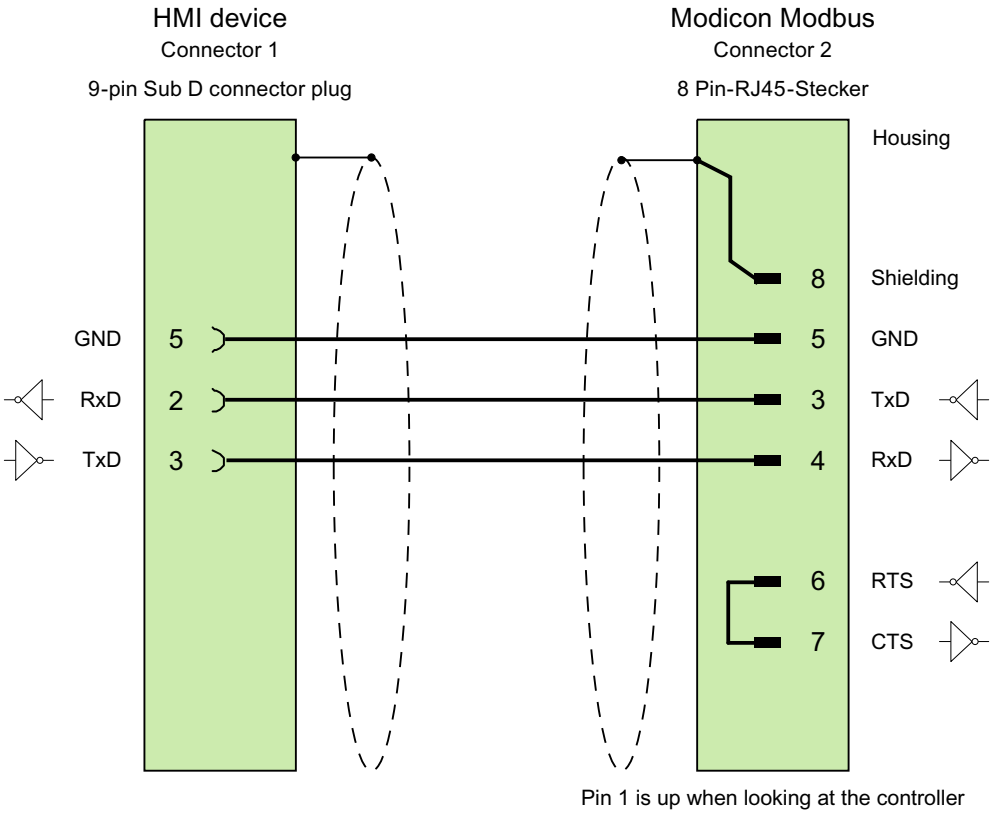
Point-to-point cable 1: PLC > PC ...



Cables: 3 x 0.14 mm^2, shielded,
max. length 15 m

Connecting cable PP2, RS-232, for Modicon

Point-to-point cable 2: PLC (TSX Compact) > PC...



Cables: 3 x 0.14 mm^2, shielded,
max. length 15 m

Performance features of communication

Permitted data types for Modicon Modbus RTU

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
+/- Double	4x, 6x	4 bytes
+/- Int	3x, 4x, 6x	2 bytes
16-bit group	0x, 1x	2 bytes
ASCII	4x, 6x	0 to 80 characters
Bit [1)]	0x, 1x, 3x, 4x, 6x	1-bit
Double	4x, 6x	4 bytes
Float	4x, 6x	4 bytes
Int	3x, 4x, 6x	2 bytes

1) Note the following for write accesses:

For data type "Bit" with the operand types "4x" and "6x", the entire word is written back to the PLC following a change to the specified bit. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

The usual bit counting method (16 LSB - 1 MSB) in the following CPU types is only used in the "HMI tags" editor with the selected "Bit" data type:

- Concept ProWORX: Compact, Quantum

The following bit location assignment applies:

	Left byte								Right byte							
Counting with tags	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Format for "Signed"

The placeholder "+/-" stands for the data types "Signed Int" and "Signed Double".

Supported CPU types for Modicon Modbus RTU

CPU types

The following CPU types are supported in the configuration of the Modicon Modbus RTU communication driver.

- Compact
- Momentum
- Quantum

Address areas for Modicon Modbus RTU

Concept, ProWORX

Address areas	Data types							
	Bool	16 Bit Group	Int	+/- Int	DInt	+/- DInt	Float	ASCII
0x	0x1 - 0x65535	0x1 - 0x65520	--	--	--	--	--	--
1x	1x100001 - 1x165535	1x100001 - 1x165520	--	--	--	--	--	--
3x	3x300001.1 - 3x365535.16	--	3x300001 - 3x365535	3x300001 - 3x365535	--	--	--	--
4x	4x400001.1 - 4x465535.16	--	4x400001 - 4x465535	4x400001 - 4x465535	4x400001 - 4x465534	4x400001 - 4x465534	4x400001 - 4x465534	4x400001 - 4x465535
6x	6x60000.1:1 - 6x69999.16:10	--	6x60000:1 - 6x69999:10	6x60000:1 - 6x69999:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69999:10

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Modicon Modbus

Area pointers for connections via Modicon Modbus communication drivers

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section "Data exchange using area pointers".

Special considerations for connections via Modicon communication drivers

You can configure the following area pointers

Area pointers	Modicon Modbus TCP/IP	Modicon Modbus RTU
Screen number	Yes	Yes
Date/time	Yes	Yes
Date/time PLC	Yes	Yes
Coordination	Yes	Yes
Project ID	Yes	Yes
Job mailbox	Yes	Yes
Data record	Yes	Yes

Restrictions Modicon Modbus TCP/IP

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
Concept, ProWORX: Compact, Quantum, Momentum	+/- Int, Int	4x, 6x
Unity, PL7: Premium, Micro, Quantum, M340	+/- Int, Int	%MW

Modicon Modbus RTU restrictions

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
Concept, ProWORX: Compact, Quantum, Momentum	+/- Int, Int	4x, 6x

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Displaying tags with Runtime Advanced and Panels (Page 4270)

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

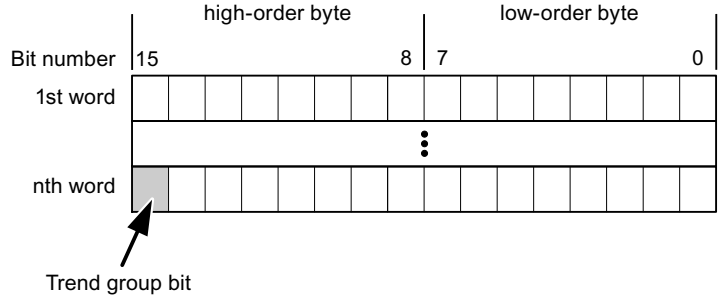
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Permitted data types for trend control

Modicon Modbus TCP/IP

Tags of the following operand types are permitted for CPU type "Concept, ProWORX: Compact, Quantum, Momentum".

- "4x"
- "6x"

Tags of the following operand types are permitted for CPU type "Unity, PL7: Premium, Micro, Quantum, M340".

- "%MW"

The tags for trend control must have data type "Int" or "+/- Int" or be an array tag with data type "Int" or "+/-Int".

You assign a bit to a trend during configuration. This sets a defined bit assignment for all areas.

Modicon Modbus RTU

Tags of the following operand types are permitted for CPU type "Concept, ProWORX: Compact, Quantum".

- "4x"
- "6x"

The tags for trend control must have data type "Int" or "+/- Int" or be an array tag with data type "Int" or "+/-Int".

You assign a bit to a trend during configuration. This sets a defined bit assignment for all areas.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with Alarms (Page 4293)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a Modicon Modbus communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
All Modicon series	Int, +/-Int	16 Bit Group, Int, +/-Int, Double, +/-Double, Float

Arrays and array tags cannot be used for discrete alarms.

How the bit positions are counted

For connections with a Modicon Modbus communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In WinCC you configure:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

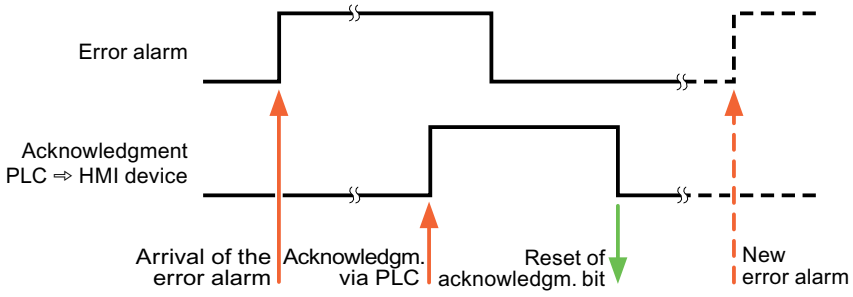
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment

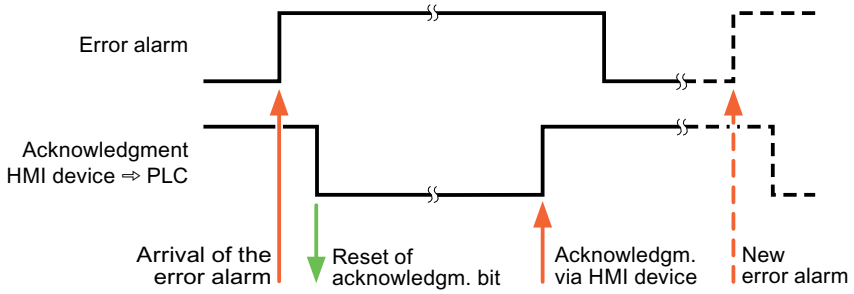
tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



LED mapping

Function

Keyboard devices have LEDs in the function keys. These LEDs can be activated from the PLC. Thus, it is possible, for example, to signal to the user which key he should press in a given situation by lighting up an LED.

Note

The LED function cannot be configured for Basic Panels.

Requirements

In order to activate an LED, an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration.

LED assignment

The assignment of the individual LEDs to the bits in the LED tags is specified when the function keys are configured. In this process, the "LED tag" and the assigned "Bit" are specified for each function key in the "General" group of the properties window.

The "Bit" bit number designates the first of two consecutive bits that control the following LED statuses.

Bit n+ 1	Bit n	LED function	
		All Mobile Panels, all Operator Panels, all Multi Panels, all Comfort Panels	Panel PCs
0	0	Off	Off
0	1	Rapid flashing	Flashing
1	0	Slow flashing	Flashing
1	1	On permanently	On permanently

Omron

Omron communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Omron communication drivers.

The following communication drivers are supported:

- Omron Host Link

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags
  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.
- Area pointers
  Area pointers are used to exchange specific data and are only set up when these data are used.

Omron Host Link

Configuring a connection via Omron Host Link

Introduction

You configure a connection to a PLC with an Omron Host Link communication driver in the "Connections" editor of the HMI device.

Note

Connection with Omron Host Link

A connection will not automatically be established when runtime is started if you have configured a connection via Omron.

A tag which is in the valid PLC memory area must be configured in the runtime start screen.

The connection will otherwise only be established once a corresponding screen has been selected.

This tag will be accessed when runtime is started and a connection will then be established.

The interfaces are named differently depending on the HMI device.

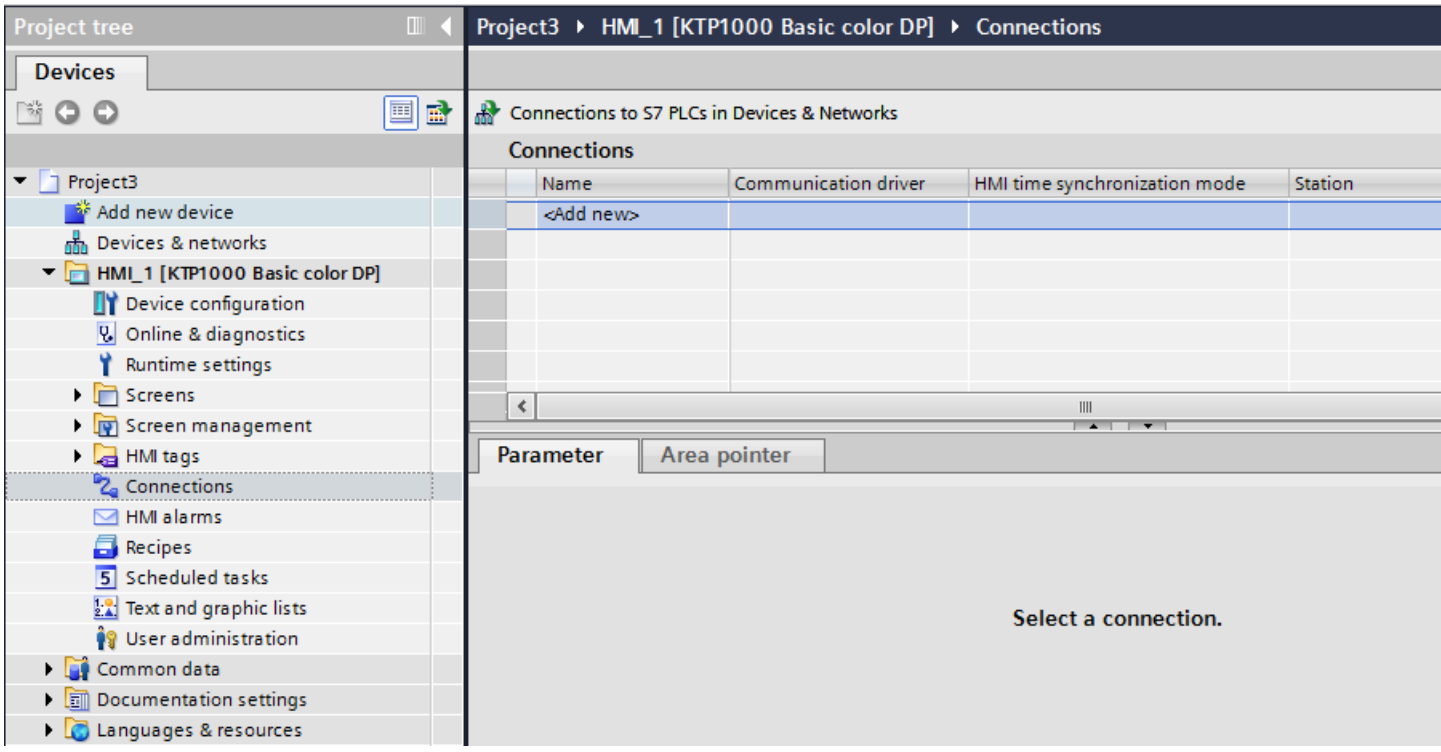
Requirements

- A project is open.
- An HMI device has been created.

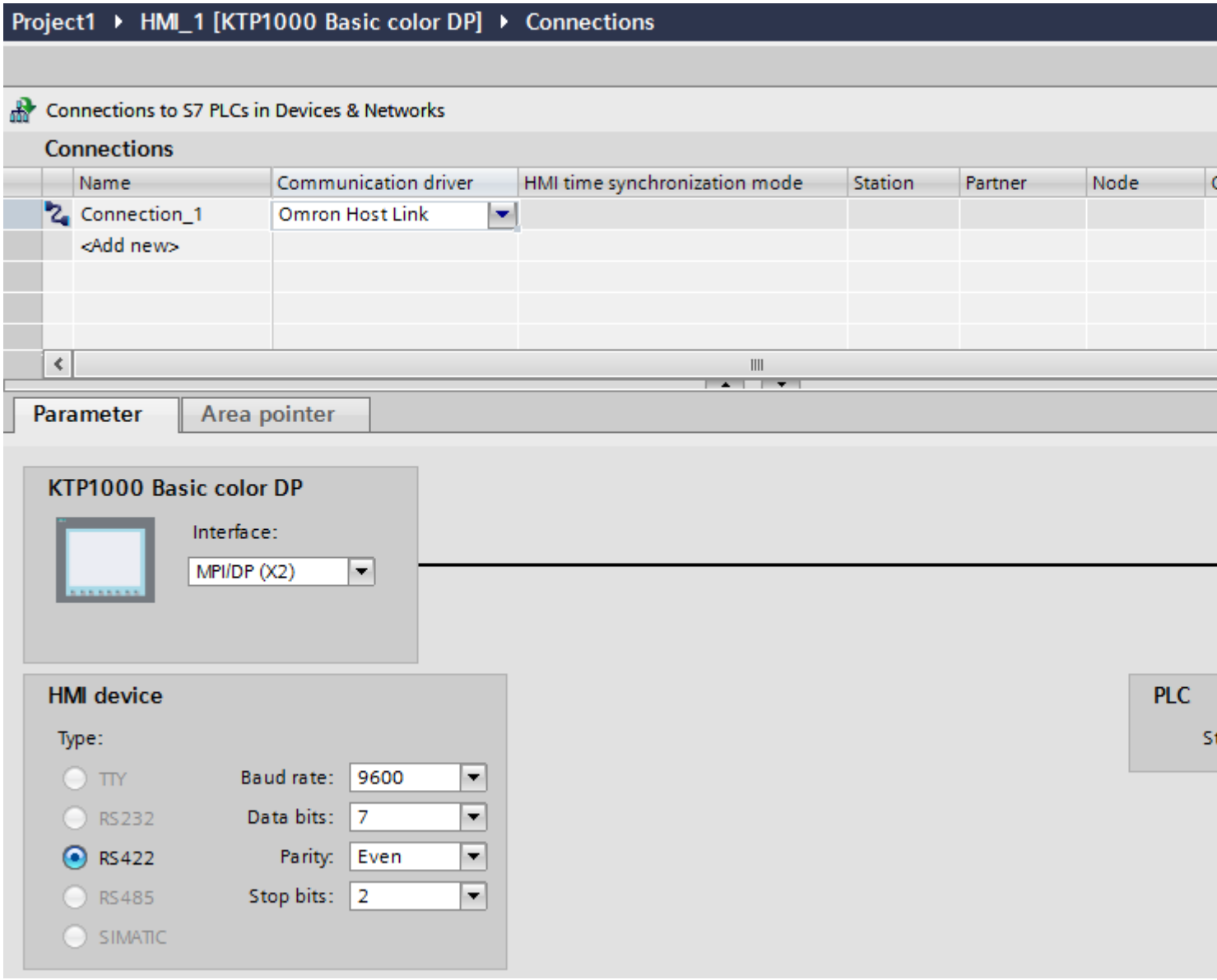
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. Select the "Omron Host Link" driver in the "Communication driver" column.



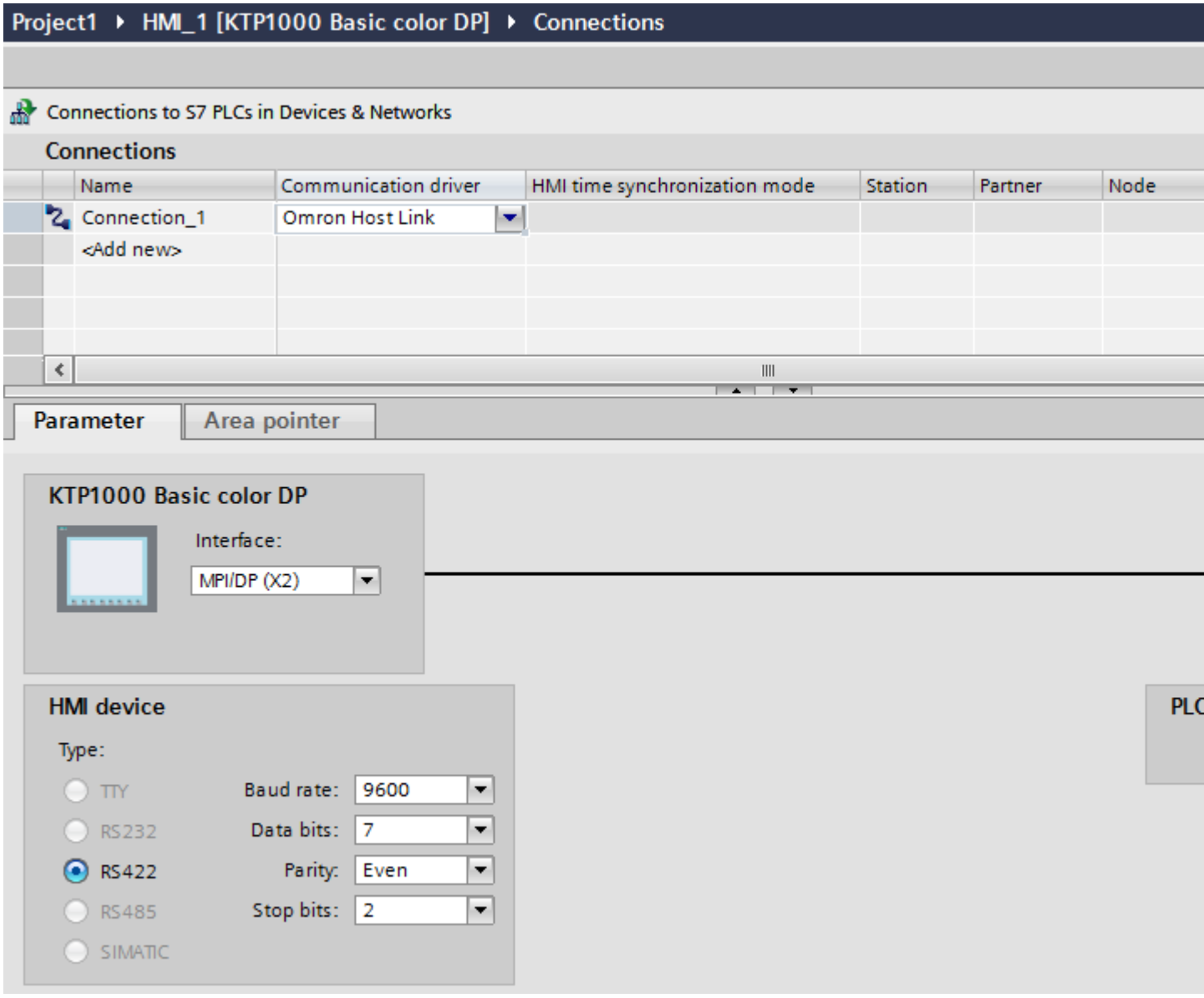
5. Select all necessary connection parameters for the interface in the inspector window under "Parameters".

Parameters for the connection (Omron Hostlink)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select an interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

- Type
  Specifies the physical connection used.
- Baud rate
  For "Baud rate", you set the transmission speed of the HMI device to OMRON. A baud rate of 19200 or 9600 can be selected for the communication.
- Data bits
  For "Data bits", you can choose between "7 bits" and "8 bits".
- Parity
  For "Parity", you can choose from "None", "Even", and "Odd".
- Stop bits
  For "Stop bits", you can choose between 1 and 2 bits.

Parameters for the PLC

- Station address
  For "Station address", set the station number of the connected PLC.

Connecting HMI device to PLC

Connections via Omron Host Link

Connection

The connection of the HMI device to an OMRON PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Connection cable

The following connecting cables are available to connect the HMI device to an Omron PLC.

Interface to the HMI device	Omron PLC			
	RS232, 9-pin	RS232 I/O port	RS422, 9-pin	RS422, terminals/ pins
RS232, 9-pin	PP1	Programming cable (standard cable of Omron)	—	—
RS232 via converter	—	—	—	Multi-point cable 1
RS422, 9-pin	—	—	PP2	Multi-point cable 2

Refer to the relevant device manual to determine which HMI device interface is to be used.

Communication types

Approved communication types

The connection from a HMI device to an OMRON-CPU with the Omron Host Link protocol via RS232 and via RS 422 is system-tested and approved by Siemens AG.

This concerns the following CPU types:

- CP1x (CP1L, CP1H, CP1E)
- CJ1x (CJ1M, CJ1H, CJ1G)
- CJ2H
- CS1x (CS1G, CS1H, CS1D)
- CPM2C

Note

Only the following CPU types have been tested and released for Basic Panels, TP 177A and OP 77A:

- CP1x (CP1L, CP1H, CP1E)
- CJ1x (CJ1M, CJ1H, CJ1G)

Multipoint connection

A multipoint connection to the up to 4 approved OMRON PLCs in a RS422-four-wire connection can be implemented with communication modules on the PLCs and is system-tested and approved by Siemens AG.

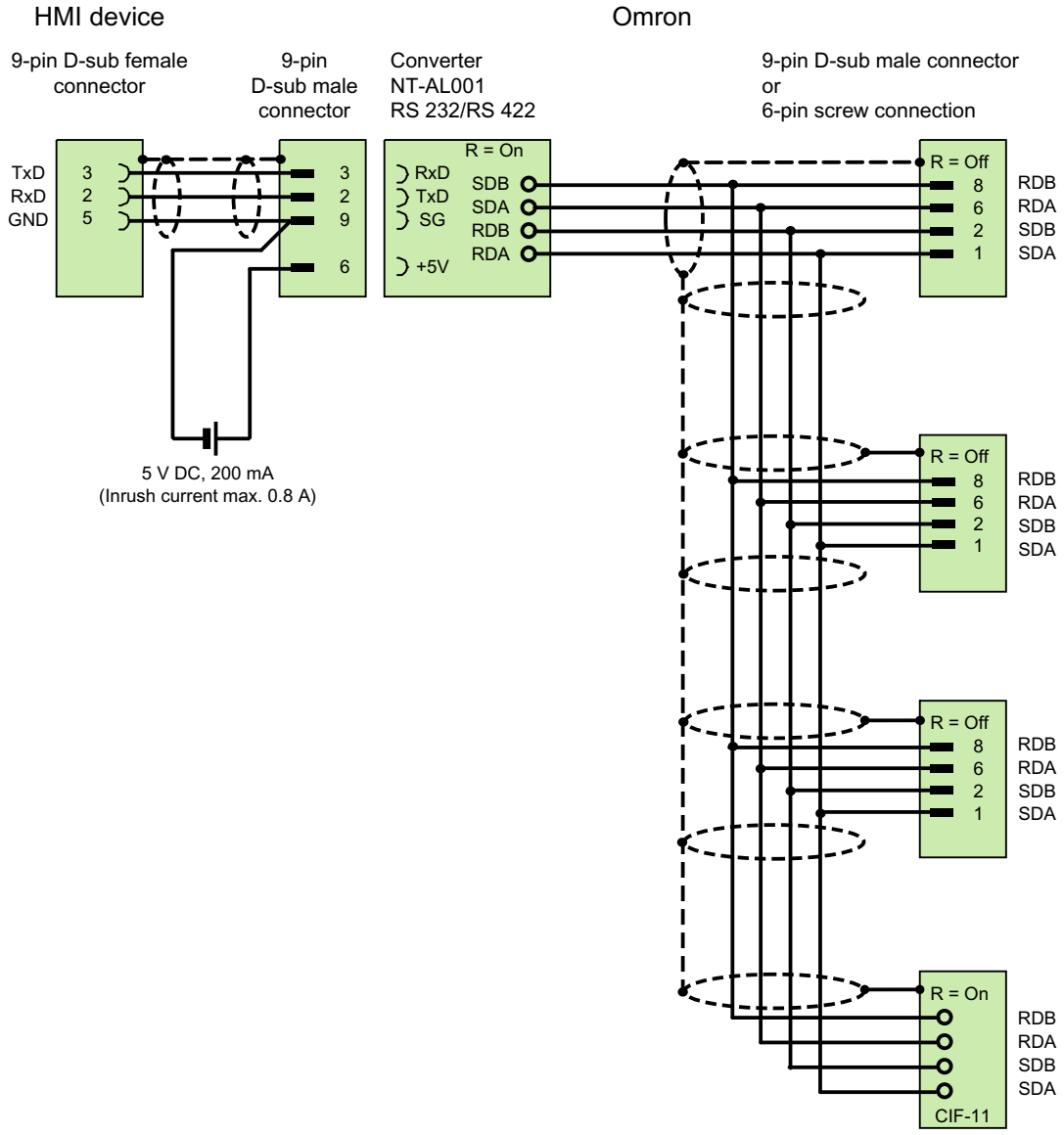
Note

The HMI device can only be operated as a master. Exactly one master is possible in the RS422-four-wire-Multidrop connection.

Connecting cables for Omron Host Link

Connecting cable MP1, RS-232, over converter, for Omron

Multipoint cable 1: MP/TP/PC > PLC

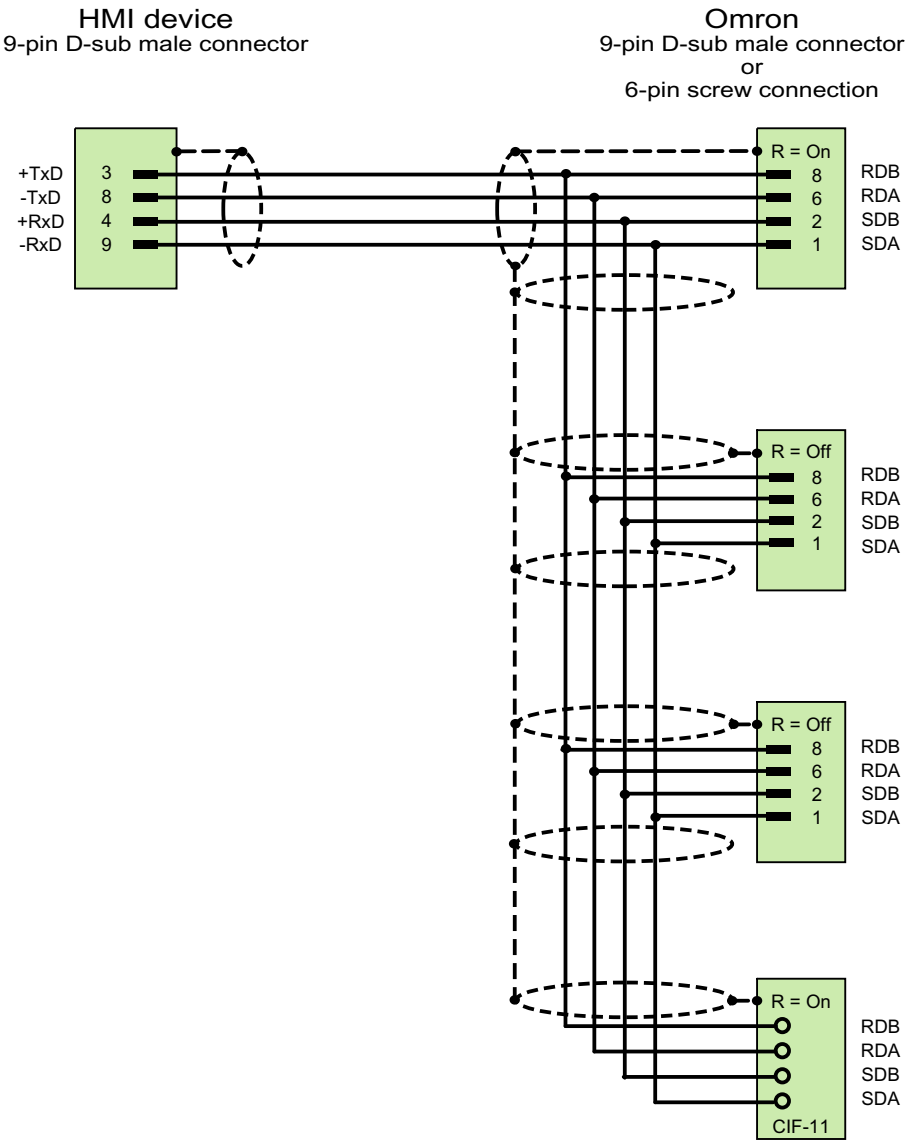


[1] Inrush current max. 0.8 A

shielded, max. length 500 m

Connecting cable MP2, RS-422, for Omron

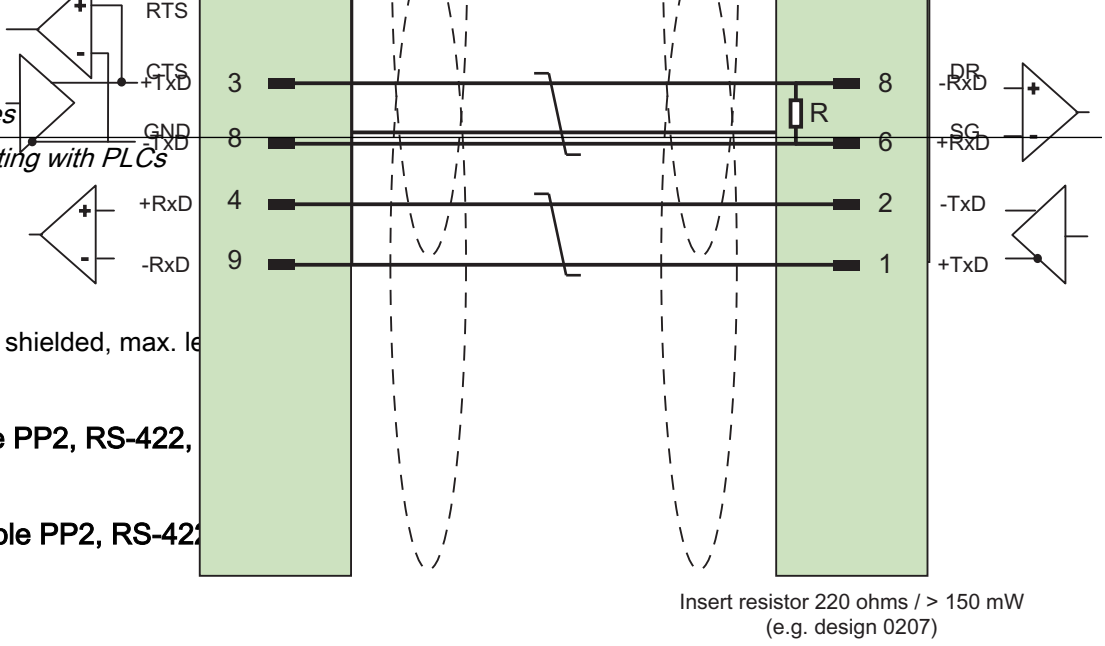
Multipoint cable 2: RS422, MP/TP/PC > SPS_



shielded, max. length 500 m

Connecting cable PP1, RS-232, for Omron

Point-to-point cable PP1, PC/TP/OP - PLC



Insert resistor 220 ohms / > 150 mW
(e.g. design 0207)

shielded, max. le

Connecting cable PP2, RS-422,

Point-to-point cable PP2, RS-42

shielded, max. length 500 m

Performance features of communication

Permissible data types for Omron Host Link

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
Bool	I/O, HR, AR, LR, DM, T/C bit, CPU status	1-bit
Byte	CPU type	1 byte
DInt	HR, AR, LR, DM	4 bytes
Int	I/O, HR, AR, LR, DM, T/C Val	2 bytes
Real	HR, DM	4 bytes
String	HR, AR, LR, DM	0 to 80 characters
UDInt	HR, AR, LR, DM	4 bytes
UInt	I/O, HR, AR, LR, DM, T/C Val	2 bytes

Note

Read and write operations of all data areas in the OMRON PLC can only be reliably carried out in "STOP" or "MONITOR" mode.

"I/O" refers either to the IR/SR area or the CIO area depending on the PLC series. The operand types "LR", "HR" and "AR" are not available in all PLC series.

Note

Note the following for write accesses:

For the "Bool" data type with the operand types "I/O", "HR", "AR", "LR" and "DM", the entire word is written back into the PLC when the specified bit is changed. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

Operand type old PLC	Operand type CS and CJ PLC
CPU Status	CPU Status
I/O	CIO
HR	H Range 0-511
AR	A
LR	n/a 1)
DM	D
T/C	T/C
CPU type	CPU type

1) You do not get an error message when you read or write the LR area in the following PLCs
- CS
- CJ
- CP

Supported CPU types for Omron Host Link

CPU types

The following CPU types are supported in the configuration of the Omron Host Link communication driver.

- CP1
  - CP1L
  - CP1H
  - CP1E
- CJ1
  - CJ1M
  - CJ1H
  - CJ1G

- CJ2
  - CJ2H
- CS1
  - CS1G
  - CS1H
  - CS1D
- CPM
  - CPM2C

Addressing in Omron Host Link

Addressing of PLCs in Omron Host Link

In PLCs of the series CS, CP and CJ, the timers 0-4095 are addressed with T/C 0-2047.

The counters 0-4095 must be addressed with an offset of 2048 (T/C 2048-4095 correspond to the counters 0-2047). Counters and timers with addresses > 2047 cannot be addressed via Host Link.

Counters and timers with addresses > 2047 cannot be addressed via Host Link.

Example:

If you want to address counter C20, you must address T/C 20+2048 = T/C 2068.

Address areas for Omron Hostlink

Omron

Address areas	Data types							
	Bool	Byte	UInt	Int	UDInt	DInt	Real	String
I/O	I/O 0.0 - I/O 9999.15	--	I/O 0 - I/O 9999	I/O 0 - I/O 9999	--	--	--	--
HR	HR 0.0 - HR 9999.15		HR 0 - HR 9999	HR 0 - HR 9999	HR 0 - HR 9998	HR 0 - HR 9998	HR 0 - HR 9999	HR 0 - HR 9999
AR	AR 0.0 - AR 9999.15		AR 0 - AR 9999	AR 0 - AR 9999	AR 0 - AR 9998	AR 0 - AR 9998		AR 0 - AR 9999
LR	LR 0.0 - LR 9999.15		LR 0 - LR 9999	LR 0 - LR 9999	LR 0 - LR 9998	LR 0 - LR 9998		LR 0 - LR 9999
DM	DM 0.0 - DM 9999.15		DM 0 - DM 9999	DM 0 - DM 9999	DM 0 - DM 9998	DM 0 - DM 9998	DM 0 - DM 9999	DM 0 - DM 9999
T/C Bit	T/C Bit 0 - T/C Bit 4095							

Address areas	Data types							
	Bool	Byte	UInt	Int	UDInt	DInt	Real	String
T/C Val			T/C Val 0 - T/C Val 4095	T/C Val 0 - T/C Val 4095				
CPU Status	RUN, MONITOR							
CPU type		CPU type						

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.

- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Omron

Area pointers in connections via Omron communication drivers

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section:

"Data exchange using area pointers".

Special features of connections via Omron Host Link

Area pointers can only be created in the following "File types": "DM", "I/O", "HR", "AR", and "LR".

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Displaying tags with Runtime Advanced and Panels (Page 4270)

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

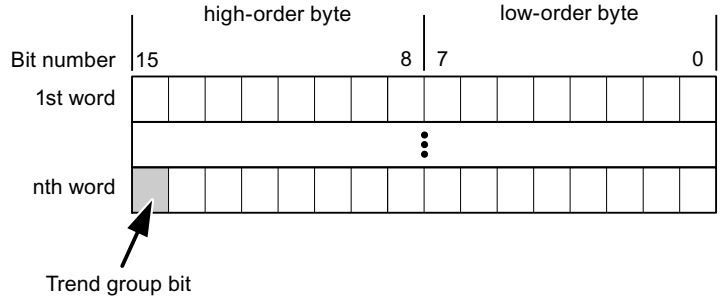
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Restrictions to the trend control

For Omron Host Link communication drivers

Tags of "Operand type" "DM", "I/O", "HR", "AR" or "LR" are permitted.

They must be data type "UInt", "Int" or an array tag of data type "UInt", "Int". You assign a bit to a trend during configuration. This sets a defined bit assignment for all areas.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with Alarms (Page 4293)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with an Omron communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
CP1, CJ1, CJ2, CS1, CPM	Uint, int	UInt, Int, UDInt, DInt

How the bit positions are counted

For connections with an Omron communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In Omron PLCs	15							8	7							0
In WinCC you configure:	15							8	7							0

Only tags for the "DM", "I/O", "HR", "AR", and "LR" file types are allowed for use as a trigger tag for discrete alarms.

Configuring discrete alarms

Use arrays for discrete alarms and append each individual alarm to one bit of the array tags themselves and not to the individual subelements.

Only tags for the "DM", "I/O", "HR", "AR", LR" areas and the "Int" and "UInt" file types are permitted for discrete alarms and arrays.

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

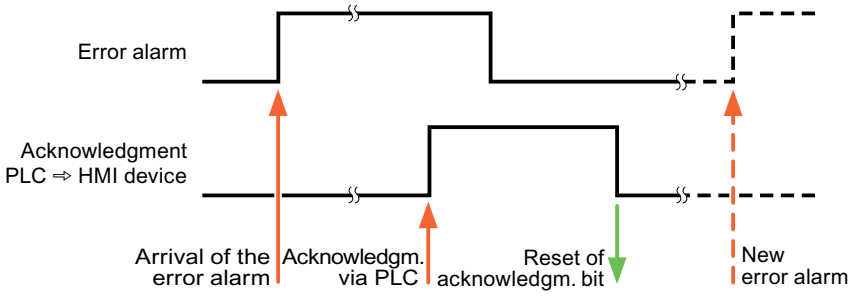
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

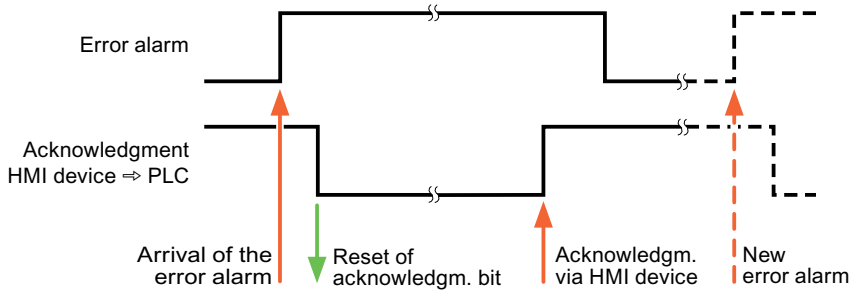
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



LED mapping

Function

Keyboard devices have LEDs in the function keys. These LEDs can be activated from the PLC. Thus, it is possible, for example, to signal to the user which key he should press in a given situation by lighting up an LED.

Note

The LED function cannot be configured for Basic Panels.

Requirements

In order to activate an LED, an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration.

LED assignment

The assignment of the individual LEDs to the bits in the LED tags is specified when the function keys are configured. In this process, the "LED tag" and the assigned "Bit" are specified for each function key in the "General" group of the properties window.

The "Bit" bit number designates the first of two consecutive bits that control the following LED statuses.

Bit n+ 1	Bit n	LED function	
		All Mobile Panels, all Operator Panels, all Multi Panels, all Comfort Panels	Panel PCs
0	0	Off	Off
0	1	Rapid flashing	Flashing
1	0	Slow flashing	Flashing
1	1	On permanently	On permanently

Communication Service Packages

CSP_1

Parallel communication

Parallel communication of communication drivers

The following table shows an overview of which communication drivers you can use simultaneously on one HMI device.

Note

Parallel communication is not approved for Basic Panels.

Parallel communication over Ethernet interfaces

The approved combinations can be operated via the same Ethernet interface. Several Ethernet interfaces are not required.

Parallel communication only concerns the Ethernet-based communication drivers.

	Allen-Bradley Ether-Net/IP	Mitsubishi MC TCP/IP	Modicon Modbus TCPIP	OPC (DA/ XML DA)	OPC UA (DA)	SIMATIC LOGO!	SIMATIC S7 200	SIMATIC S7 300/400	SIMATIC S7 1200	SIMATIC S7 1500	SIMATIC HTTP protocol	Sinumerik NC
Allen-Bradley Ether-Net/IP	--	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Mitsubishi MC TCP/IP	No	--	No	Yes	Yes	No	No	No	No	No	Yes	No
Modicon Modbus TCPIP	No	No	--	Yes	Yes	No	No	No	No	No	Yes	No
OPC (DA/ XML DA)	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OPC UA (DA)	Yes	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SIMATIC LOGO!	Yes	No	No	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes	Yes
SIMATIC S7 200	Yes	No	No	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes	Yes
SIMATIC S7 300/400	Yes	No	No	Yes	Yes	Yes	Yes	--	Yes	Yes	Yes	Yes
SIMATIC S7 1200	Yes	No	No	Yes	Yes	Yes	Yes	Yes	--	Yes	Yes	Yes
SIMATIC S7 1500	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	--	Yes	Yes
SIMATIC HTTP protocol	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	--	Yes
Sinumerik NC	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	--

Parallel communication over serial interfaces

The following applies for parallel communication over serial interfaces:

- One communication driver per interface.
- One interface per communication driver.

CSP_2

CSP_3

CSP_4

CSP_5

12.11.19.5 Data exchange using area pointers

General information on area pointers

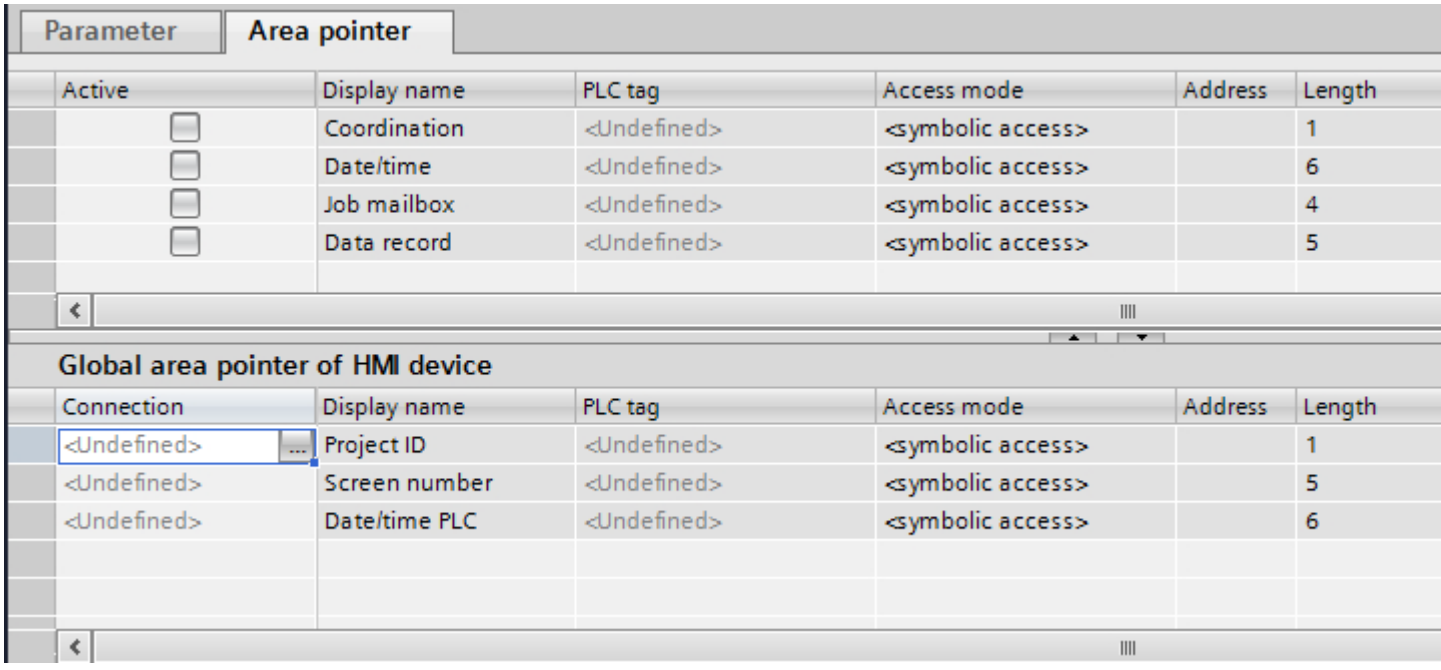
Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use the area pointer, you enable it in "Connections ▸ Area pointers". You then assign the area pointer parameters.



- Active
  Enables the area pointer.
- Pointer name
  Name of the area pointer specified by WinCC.
- PLC tag
  Here you select the PLC tag or the tag array that you have configured as the data area for the area pointer.
- Address
  No address is entered into this field because of the symbolic access.
- Length
  WinCC specifies the length of the area pointer.
- Acquisition cycle
  You specify the acquisition cycle in this field for area pointers that are read by the HMI device. Note that a very short acquisition time may have a negative impact on HMI device performance.
- Comment
  Enter a comment, for example, to describe the purpose of the area pointer.

Accessing data areas

Accessing data areas

The following table shows how HMI devices and PLCs access individual data areas for read (R) or write (W) operations.

Data area	Required for	HMI device	PLC
Screen number	Evaluation by the PLC in order to determine the active screen.	W	R
Data record	Transfer of data records with synchronization	R/W	R/W
Date/time	Transfer of the date and time from the HMI device to the PLC	W	R
Date/time PLC	Transfer of the date and time from the PLC to the HMI device	R	W
Coordination	Requesting the HMI device status in the PLC program	W	R
Project ID	Runtime checks for consistency between the WinCC project ID and the project in the PLC	R	W
Job mailbox	Triggering of HMI device functions by the PLC program	R/W	R/W

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Use

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1. word	Current screen type															
2. word	Current screen number															
3. word	Reserved															
4th word	Current field number															
5. word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

The date/time data area has the following structure:

Data word	Left byte								Right byte								
	15							8	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the Date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute if your process can handle it.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The "Date/time PLC" data area has the following structure:

Data word	Left byte			Right byte		
	15	. . . . .	8	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functionality:

- Detecting the startup of the HMI device in the control program
- detection in the control program of the current HMI device operating mode
- detection in the control program of the HMI devices ready to communicate state

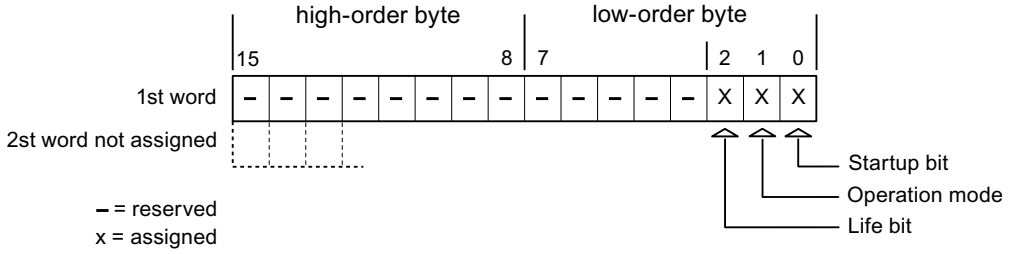
The "Coordination" area pointer has a length of one word.

Use

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program may not make changes to the coordination area for this reason.

Assignment of bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The state of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not the connection to the HMI device is still up by querying this bit in the control program.

"Project ID" area pointer

Function

You can check whether the HMI device is connected to the correct PLC at the start of Runtime. This check is important when operating with several HMI devices.

The HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of configuration data with the control program. If discrepancy is detected, a system event is displayed on the HMI device and Runtime is stopped.

Application

To use this area pointer, set up the following during the configuration:

- Define the version of configuration. Possible values between 1 and 255.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:
  You enter the data address in the editor "Communication > Connections".

Connection failure

A connection failure to a device on which the "Project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following prerequisites:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

"PLC job" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Left byte	Right byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

Note

Please note that not all HMI devices support job mailboxes.

No.	Function	
14	**Setting the time (BCD coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	**Setting the date (BCD coded)**	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	**User logon**	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-

No.	Function	
14	**Setting the time (BCD coded)**	
24	**User logoff**	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	**Transfer date/time to PLC**	
	(in the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	**Transfer date/time to PLC**	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs in order to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	**Update tags**	
	Causes the HMI device to read the current value of the tags•from the PLC whose update ID matches the value transferred in parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	**Screen selection** [1]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	**Read data record from PLC**	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	**Write data record to PLC**	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1] OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.

"Data record" area pointer

"Data mailbox" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization over the data record

Data records are always transferred directly. That is, the tag values are read from an address or written to an address configured for this tag directly, without redirecting the values by means of interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system event.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example, when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then process, edit, or save these values, for example.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data record free
2	0000 0010	Transferring.
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer started by the operator in the recipe display

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data record and sets the data record number to 0.	Abort with system event.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data record.	Abort with system event.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a PLC job

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxes No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data record.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the function from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data record.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data record.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data record.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible

- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system alarms
- Triggered by function
  Output of system alarms
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data record.

12.11.20 Special features of WinAC MP

12.11.20.1 WinAC MP basics

WinAC MP basics

WinAC MP is a software package for HMI devices. WinAC MP integrates a software PLC compatible with SIMATIC S7 in WinCC Runtime.

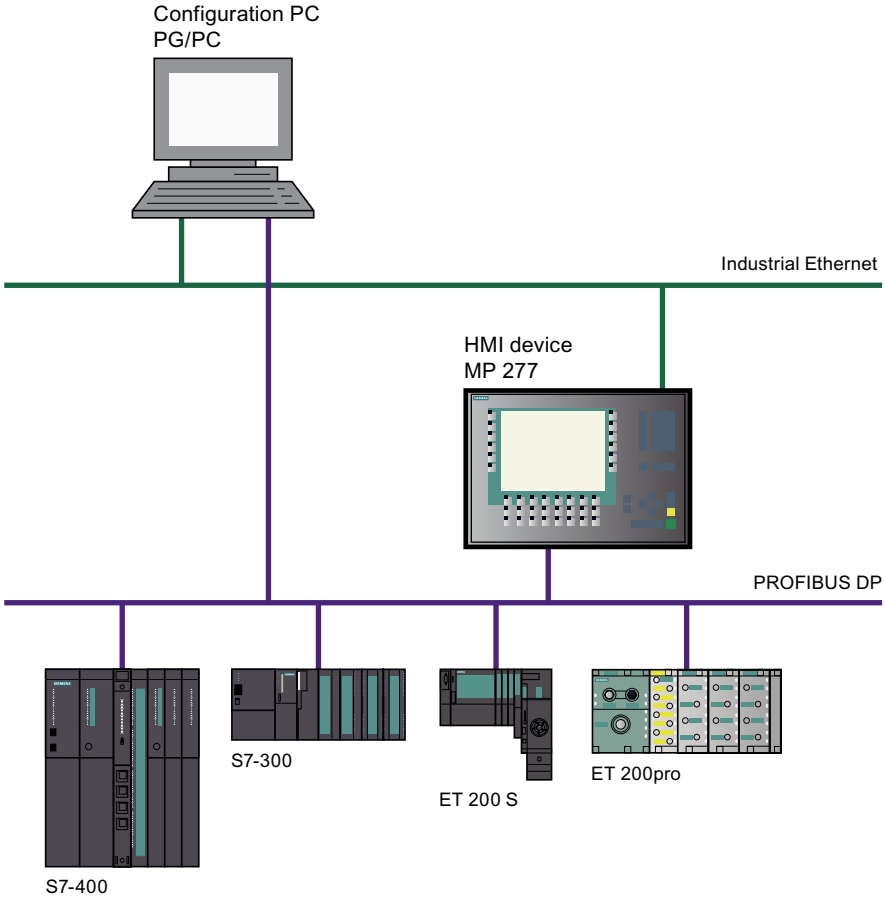
Using WinAC MP you can execute STEP 7 programs on the HMI device.

The common use of WinAC MP and WinCC provides process control and visualization on one HMI device.

The communications driver for WinAC MP supports the following protocols for controlling the process components:

- Industrial Ethernet
- PROFIBUS DP

The following screen shows the principal connection of the process components on the PROFIBUS DP example.



If you wish to control your process using an HMI device and WinAC MP, then you must install WinAC MP on your configuration PC. Then transfer WinAC MP and the project to your HMI device.

Note

Notes about instructions

The following describes how to install WinAC MP and transfer WinAC MP to an HMI device.

To transfer a STEP 7 project with a WinAC MP controller to an HMI device, you require experience with STEP 7.

12.11.20.2 Communication options with WinAC MP

Definition of "routing"

If there are stations in an automation system that are not connected to the same bus, these stations cannot be accessed directly online. To establish a connection to these devices, a router must be included between them.

An MP x77 HMI device with WinAC MP can function as the router. As a router, it connects the Ethernet and PROFIBUS networks.

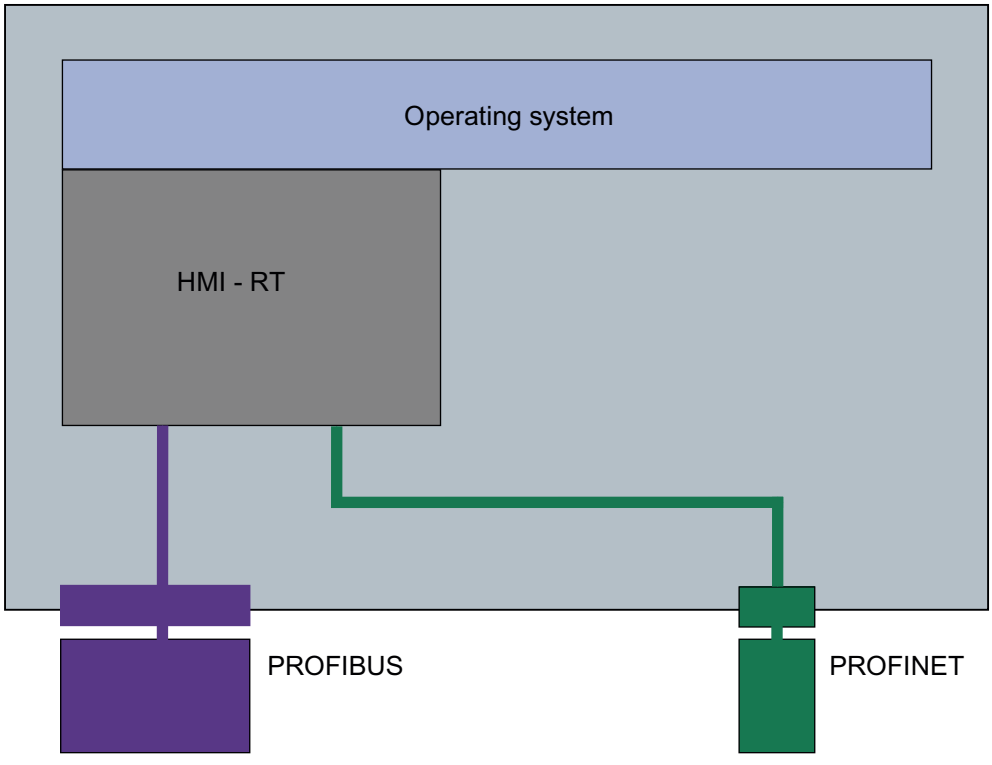
You can use routing, for example, to do the following:

- Download STEP 7 user programs
- Download a hardware configuration
- Execute testing and diagnostics functions

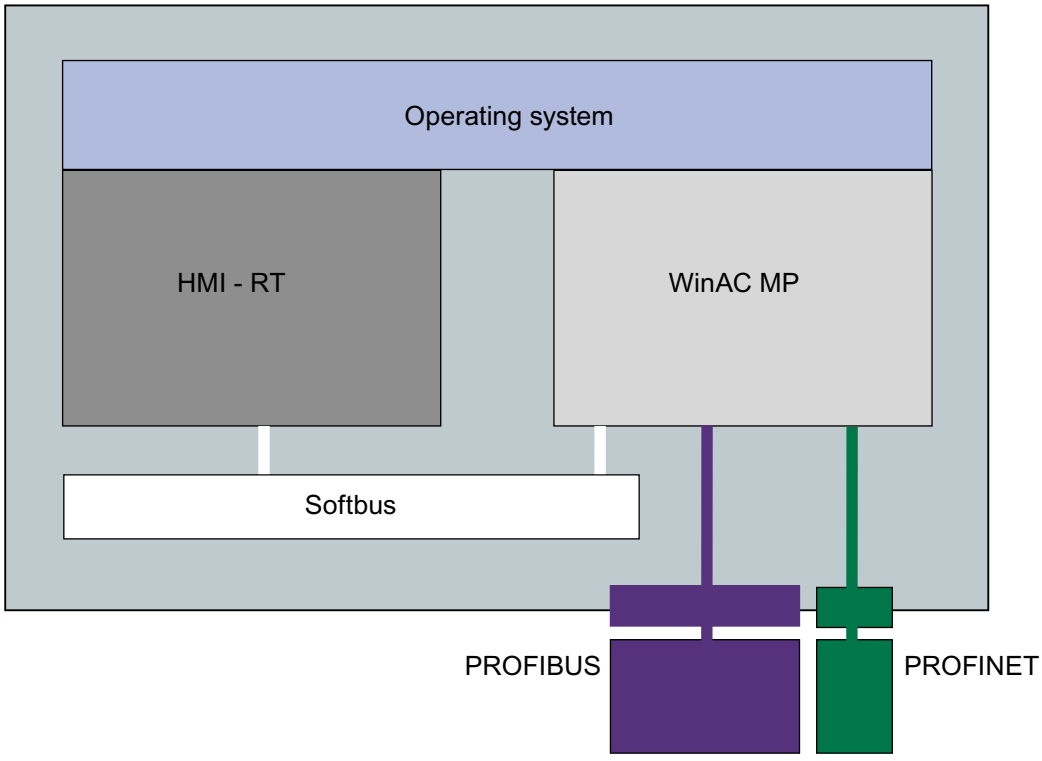
Definition of "Softbus"

Virtual bus that allows data exchange between WinCC Runtime and WinAC MP. This bus is also installed when you install WinCC . After installing WinAC MP, S7 connections from WinCC Runtime to external stations on PROFIBUS or Ethernet are routed over Softbus.

Prior to installation of WinAC MP (without Softbus)



After installation of WinAC MP (with Softbus)



Routing from Ethernet to PROFIBUS DP with WinAC MP

With STEP 7 on Industrial Ethernet, you can access all nodes on PROFIBUS DP from the HMI device. With WinCC, you can access the HMI device but not the nodes connected over PROFIBUS DP.

If the configuration PC is not connected directly to PROFIBUS DP, the nodes on PROFIBUS can nevertheless be reached because the MP x77 HMI device serves as a router.

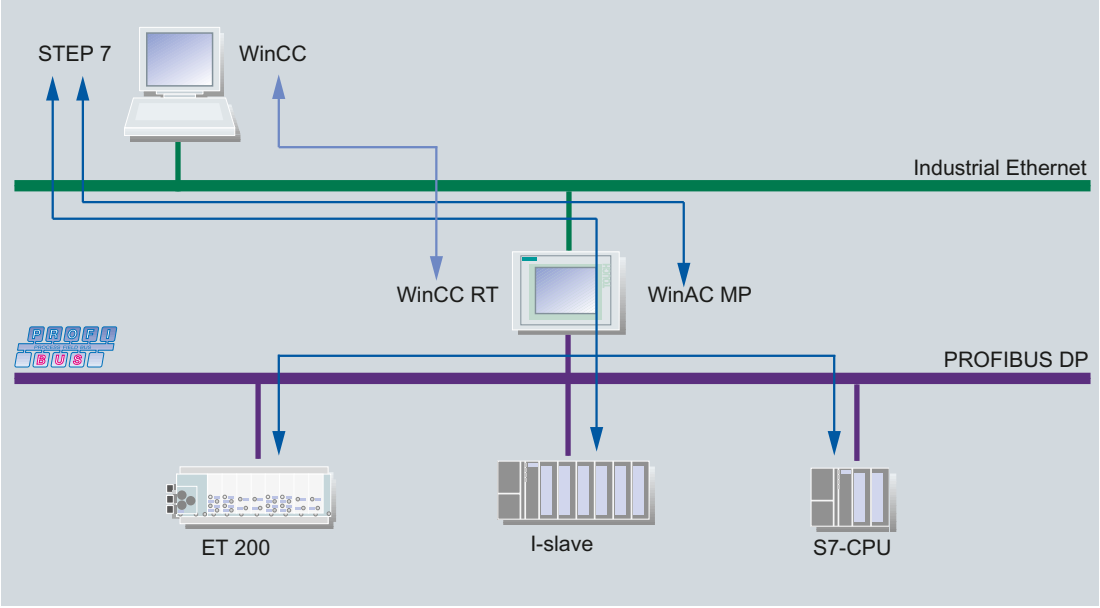


Figure 12-6 Routing from Ethernet to PROFIBUS DP with WinAC MP

Routing from PROFIBUS DP to Ethernet with WinAC MP

With STEP 7 on PROFIBUS DP, you can access all nodes on Industrial Ethernet from the HMI device. With WinCC on PROFIBUS DP, you have access to the HMI device and to OPs connected over PROFIBUS DP.

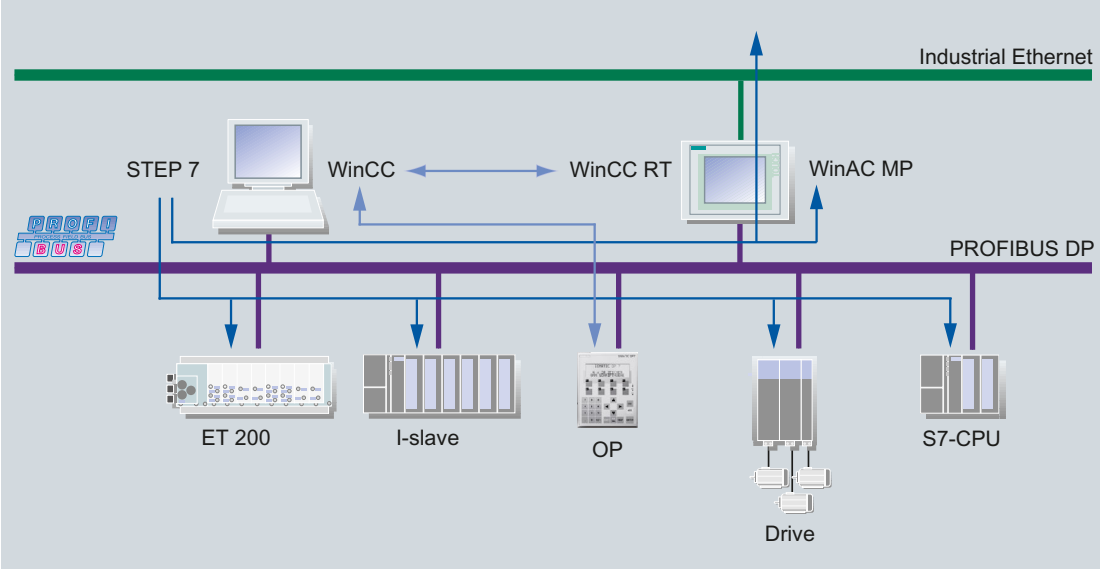


Figure 12-7 Routing with WinAC MP over PROFIBUS

Visualization over WinAC MP

Visualization over the HMI device between Industrial Ethernet and PROFIBUS DP is possible with WinCC.

It is not necessary to program the communication links. A PC serves as the visualization platform.

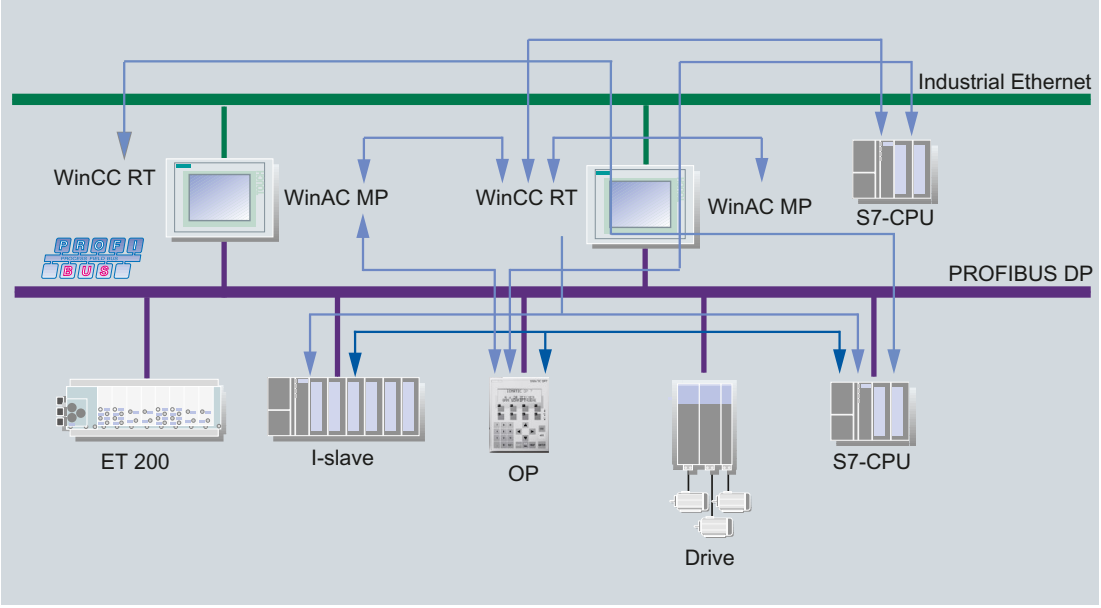


Figure 12-8 Visualization over WinAC MP

CPU-CPU communication over WinAC MP

CPU-CPU communication is possible with the HMI device.

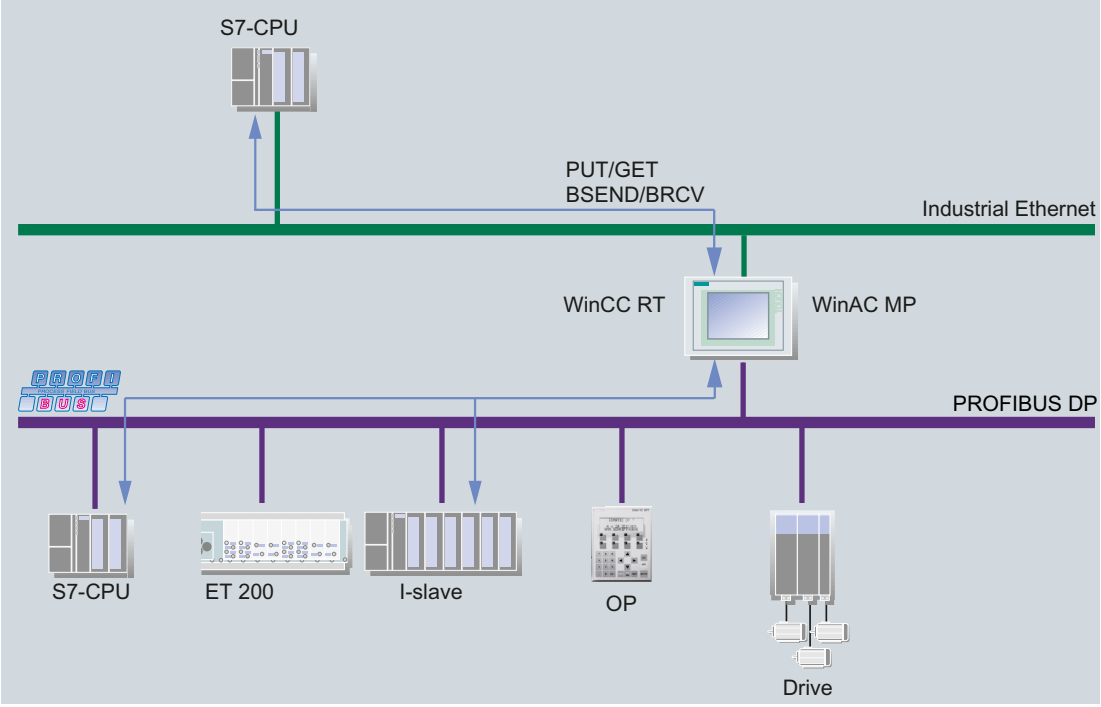


Figure 12-9 CPU-CPU communication over WinAC MP

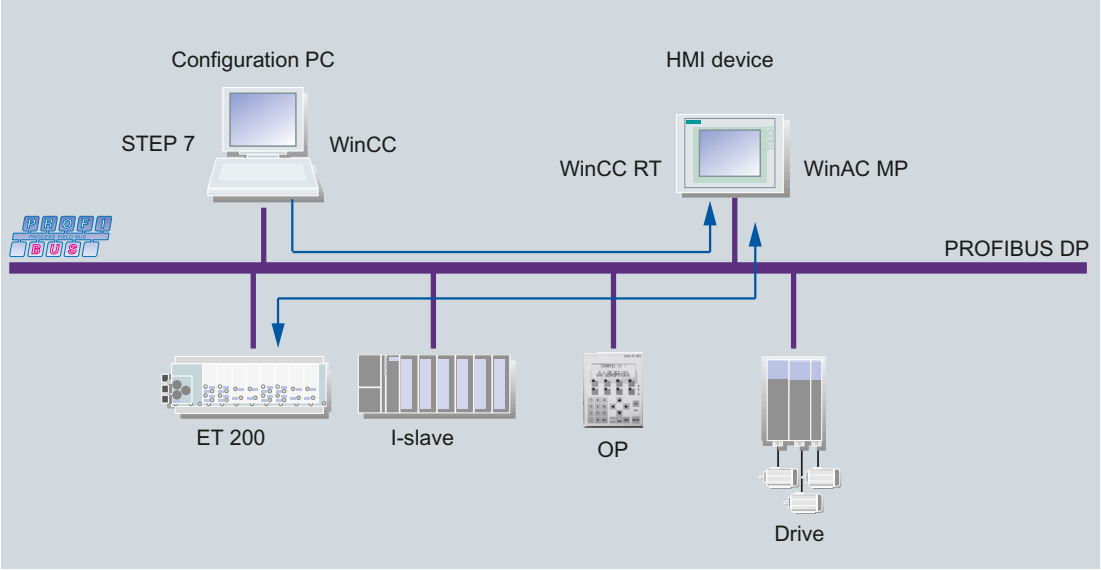
12.11.20.3 Standard procedure for communication with WinAC MP

Overview

The section below shows how to configure WinAC MP in a SIMATIC HMI station using a configuration PC on which STEP 7 and WinCC are installed.

The configuration PC and WinAC MP in a SIMATIC HMI station are interconnected via PROFIBUS DP. Configuration via Ethernet is also possible. A description of the procedure can be found in the "WinAC MP" manual.

You can integrate DP slaves for WinAC MP in the STEP 7 project.



Procedure

The basic steps are as follows:

1. Configure WinAC MP in STEP 7.
2. Create a connection between WinAC MP and WinCC Runtime.
3. In STEP 7, download the configuration to WinAC MP.
4. Configure (visualization) HMI objects with WinCC .
5. Download the (visualization) configuration (for example, visualization with histogram) from WinCC to the HMI device.

12.11.20.4 Configuring the WinAC MP communications driver

Requirements for the use of WinAC MP

Requirements

In order to use WinAC MP in the process control, you must perform the installation and development steps on your configuration PC. Transfer the necessary components to the HMI device.

Basic procedure

The following steps are described in more detail in the following:

1. Check that your system meets the system requirements:
   - STEP 7 and the authorization are installed on the configuration PC.
   - WinCC and the authorization are installed on the configuration PC.
     WinCC Runtime and ProSave were installed along with WinCC.
2. When WinCC is installed, the WinAC MP option is also automatically installed. Authorization of the WinAC MP is carried out via separate licenses.
   The following components are installed alongside WinAC MP:
   - WinAC MP Runtime files
   - WinAC MP system library
3. Develop a STEP 7 project for controlling your process with WinAC MP on the HMI device.
4. Develop a WinCC project for operating and visualizing your controller.
5. Connect the process components via the PROFIBUS network and configure the hardware.
6. Transfer the WinAC MP Runtime files from the configuration PC to the HMI device.
   Use ProSave for this.
7. Transfer the WinAC MP authorization from its storage location to the HMI device.
   Use the Automation License Manager for this.
8. Transfer the STEP 7 project to the HMI device.
9. Transfer WinCC Runtime and the WinCC project to the HMI device.
   To do this, use the "Transfer" function.

Changing the device type

Introduction

There are different ways to create a SIMATIC HMI-station with an HMI device and a configured WinAC MP controller.

Procedure 1

Create an HMI device in a SIMATIC HMI-station in the device and network view. Then add the corresponding variant of WinAC MP.

The advantage of this procedure is that you can choose the device type (MP 177, MP 277 or MP 377) and the design of the device type (monitor size and touch or key) directly on selection in the hardware catalog.

Procedure 2

Insert the desired variant of WinAC MP directly in the device and network view.

This procedure creates, by default, the largest touch design of the selected HMI device in the device and network view.

Changing the device type

Select a device. You can change the device type using the shortcut menu command "Change device". When the device type MP277 is changed to MP377 and MP 177 or vice versa with a configured WinAC MP controller in a SIMATIC HMI station, the WinAC MP controller is automatically moved to the storage location of the non slotted modules. It can therefore not be used in the new HMI device. If the the device type is changed, the WinAC MP-configuration must be created again.

Connecting the configuration PC and HMI device

Procedure for creating a connection via Ethernet or PROFIBUS DP

Basic procedure

WinAC MP supports the communication between the configuration PC and the HMI device via the following connections:

- Industrial Ethernet
- PROFIBUS DP
- USB

To achieve successful communication, follow these steps:

1. Connect the configuration PC to the HMI device.
2. Make the parameter settings for data transfer on the control panel.
3. Make the parameter settings for data transfer on the configuration PC.
4. Set the communications parameters in ProSave.
   With ProSave you can load the WinAC MP Runtime files in the HMI device. Therefore, you must set the communication parameters both in ProSave as well as on the HMI device.
5. Set the communications parameters in WinCC.
   Use WinCC to download your project to the HMI device using the "Download" function.

Establishing a connection over Ethernet

Connect the configuration PC to the HMI device (Ethernet)

In order to communicate via an Ethernet network you can either use a direct connection or a networked connection.

To connect a configuration PC to an HMI device via Ethernet, follow these steps:

1. Connect the interfaces of the configuration PC and the HMI device with an Ethernet cable.

Note

Use a crossover cable to create a direct connection between the configuration PC and the HMI device.

To connect the configuration PC and the HMI device to an Ethernet (LAN) use a 1 to 1 cable with RJ45 connector.

The interface to be used on the HMI device can be established from the operating instructions of your HMI device.

Configuring the transmission settings in the Control Panel (Ethernet)

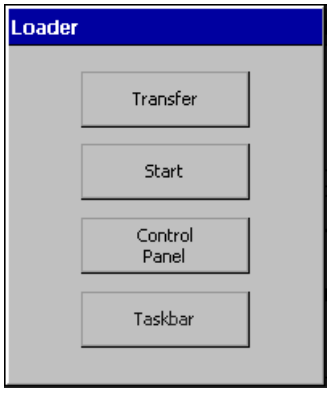
Requirements

The configuration PC and HMI device are connected to each other over an Ethernet cable.

Setting up an IP address on the control panel

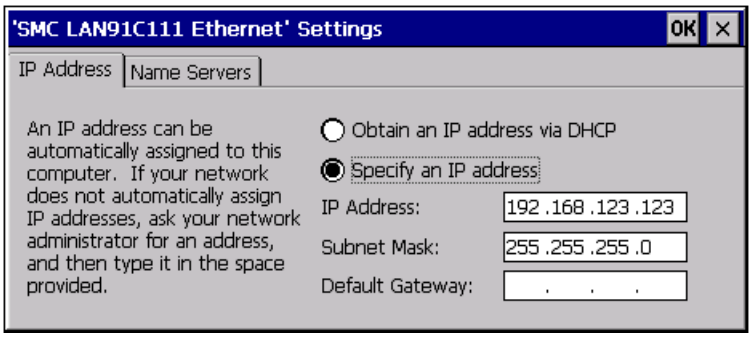
The set up the IP address on the control panel, follow these steps:

1. Switch on the HMI device.
   Before the HMI device opens an existing project, the "Loader" selection field appears.
2. Select "Control Panel".



3. Double click on the "Network and Dial-up Connection" symbol.
4. Double-click on the LAN connection.
5. Select the "IP Adress" tab.

6. Select "Specify an IP address".

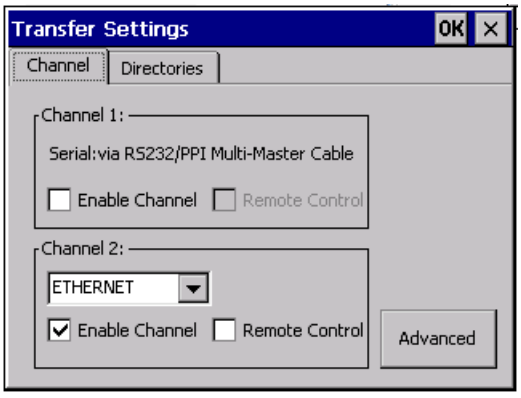


7. Enter the IP address and the subnet mask that you were given by your network administrator.
8. Confirm your entries with "OK".

Entering transfer settings

To make the data transfer settings, follow these steps:

1. In the control panel, double click on the "Transfer" symbol.
2. Select the "Channel" tab.
3. Select the "Ethernet" option in the "Channel 2" area.
4. Enable "Enable Channel" and confirm your entries.



5. Click "OK" and end the dialog "Transfer Settings".

The transmission settings are configured on the configuration PC

Requirement

- The configuration PC and HMI device are connected to each other over an Ethernet cable
- The data transfer settings have been made on the control panel

Procedure

To make the data transfer settings on the configuration PC, follow these steps:

1. Open the network settings of your PC.
2. Select the "Internet Protocol (TCP/IP)".
3. Enable the "General" tab.
4. Select the "Use the following IP address" check box in the protocol properties.
5. Enter an "IP address".
6. Enter a "subnetwork mask".
7. Confirm your entries.

Note

The IP addresses of the configuration PC and the HMI device must not be the same.

The subnet mask must be the same for the configuration PC and the HMI device.

Set the communication parameters (Ethernet)

Requirement

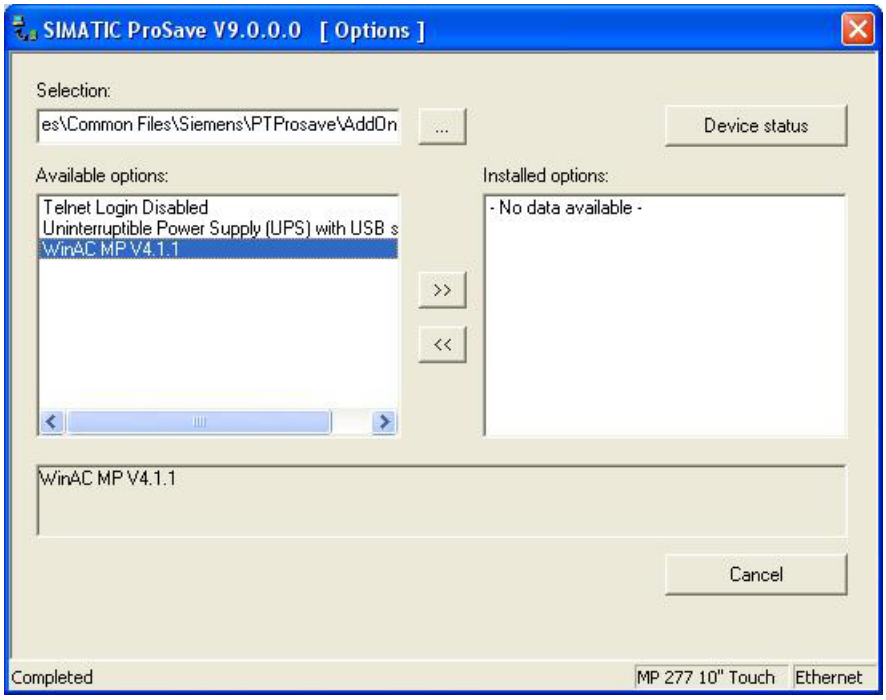
- The configuration PC and HMI device are connected to each other over an Ethernet cable
- The data transfer settings have been made on the control panel
- The data transfer settings have been made on the configuration PC

Procedure

To make the communications parameter settings in ProSave, follow these steps:

1. Make the transfer settings first.
2. To do this, select the desired component in the project tree.
3. Select the "Download to device > Software" command in the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog will automatically open. In this case, set all the necessary parameters for the connection and click "Load".
   - If you have already defined an online connection, the project data will be compiled immediately and the "Load preview" dialog will open. This dialog displays messages and proposes actions necessary for uploading.
4. Select the "Options" command from the "Online > HMI device maintenance" menu to start ProSave .

5. Select an element under "Available options".
6. Click ">>".



Setting the communication parameters in WinCC (Ethernet)

Requirement

- The configuration PC and HMI device are connected to each other over an Ethernet cable.
- The data transfer settings have been made on the control panel.
- The data transfer settings have been made on the configuration PC.

Procedure

To make the connection settings in WinCC, follow these steps:

1. Open the transfer settings.
2. Select the desired HMI device in the project tree.
3. Select the "Download to device > Software" command in the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog will automatically open. In this case, set all the necessary parameters for the connection and click "Load".
   - If you have already defined an online connection, the project data will be compiled immediately and the "Load preview" dialog will open. This dialog displays messages and proposes actions necessary for uploading.

4. Select the "Ethernet" mode.
5. Specify the IP address or the computer name of the HMI device.
6. Switch the HMI device to "transfer" mode.
7. Transfer the WinCC project to the HMI device.

Warning

Death, serious bodily harm and / or damage to property from unexpected process or machine behavior.

If you use DHCP it is not guaranteed that every time the same IP address will be issued if a node is switched on.

Nodes could lose the connection in the Industrial Ethernet or be connected with incorrect nodes. Always enter a static IP address for the HMI device. Contact your network administrator for address allocations.

Establishing a connection via PROFIBUS DP

Connect the configuration PC to the HMI device (PROFIBUS DP)

1. Connect the interfaces from the configuration PC and the HMI device with an PROFIBUS DP cable.

Note

To communicate via an PROFIBUS DP network you must create a direct connection.

Check if your configuration PC has a suitable interface for connection to the PROFIBUS DP.

If this interface is missing, you must perform the following steps:

- Build a card into the configuration PC, e.g. CP 5611 for desktop PCs.
- Install the respective drivers.

Configuring the transmission settings in the Control Panel (PROFIBUS DP)

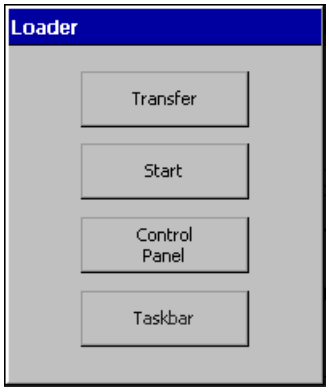
Requirement

- The configuration PC and HMI device are linked with each other via a PROFIBUS DP cable.

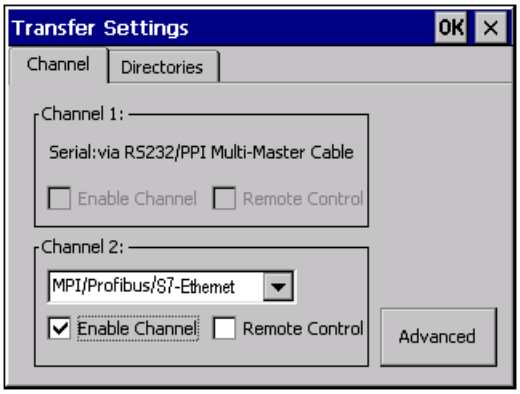
Procedure

Perform the following steps on the HMI device:

1. Switch on the HMI device.
   Before the HMI device opens an existing project, the "Loader" selection field appears.
2. Select "Control Panel".



3. Double click on "Transfer".
4. Select the "MPI/PROFIBUS" option from the "Channel 2" selection list under the "Channel" tab.
5. Select "Enable Channel".



6. Confirm with "OK".
7. "Transfer Settings" End the dialog.

Setting the communication parameters in ProSave (PROFIBUS DP)

Requirement

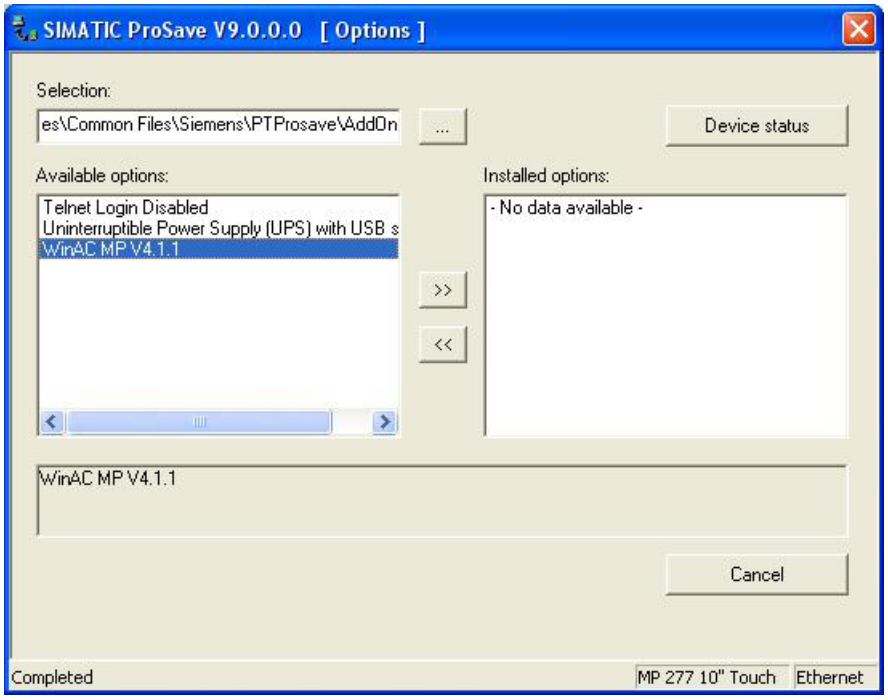
The configuration PC and HMI device are linked with each other via a PROFIBUS DP cable.

The data transfer settings have been made on the control panel.

Procedure

To make the communications parameter settings in ProSave, follow these steps:

1. Make the transfer settings first.
2. To do this, select the desired component in the project tree.
3. Select the "Download to device > Software" command in the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog will automatically open. In this case, set all the necessary parameters for the connection and click "Load".
   - If you have already defined an online connection, the project data will be compiled immediately and the "Load preview" dialog will open. This dialog displays messages and proposes actions necessary for uploading.
4. Select the "Options" command from the "Online > HMI device maintenance" menu to start ProSave .
5. Select an element under "Available options".
6. Click ">>".



Set the communication parameters in WinCC (PROFIBUS DP)

Requirement

- The configuration PC and HMI device are linked with each other via a PROFIBUS DP cable.
- The data transfer settings have been made on the control panel.

Procedure

To make the connection settings in WinCC, follow these steps:

1. Open the transfer settings.
2. Select the desired HMI device in the project tree.
3. Select the "Download to device > Software" command in the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog will automatically open. In this case, set all the necessary parameters for the connection and click "Load".
   - If you have already defined an online connection, the project data will be compiled immediately and the "Load preview" dialog will open. This dialog displays messages and proposes actions necessary for uploading.
4. Select the "MPI / PROFIBUS DP" mode.
5. Specify the PROFIBUS address that is assigned to the HMI device.
   The PROFIBUS address is the station address and corresponds to the address configured for the WinAC MP controller.
6. Switch the HMI device to "transfer" mode.
7. Transfer the WinCC project to the HMI device.

Note

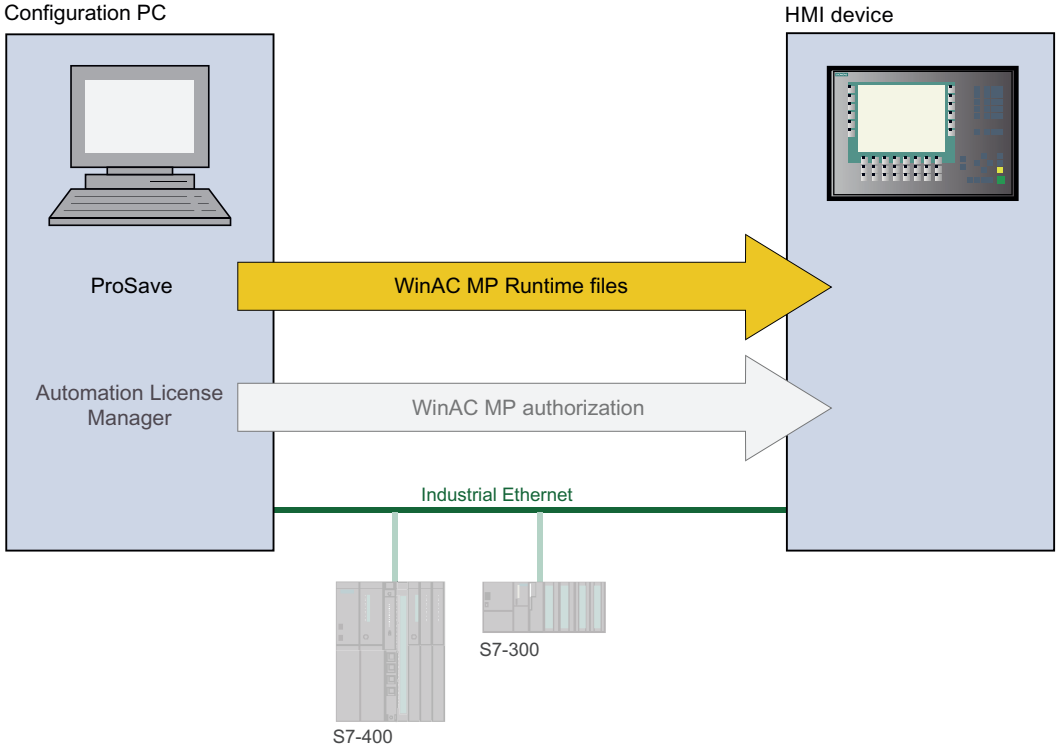
For PROFIBUS DP transmission the WinAC MP control must be started on the HMI device.

12.11.20.5 Transferring WinAC MP to the HMI device

Procedure for transferring WinAC MP

Introduction

If you want to execute WinAC MP on the HMI device you must transfer the WinAC MP Runtime files from the configuration PC to the HMI device. For the transfer there must be a data connection between the configuration PC and the HMI device. The following screen shows the arrangement using example Industrial Ethernet.



Basic procedure

To transfer WinAC MP Runtime files to an HMI device, follow these steps:

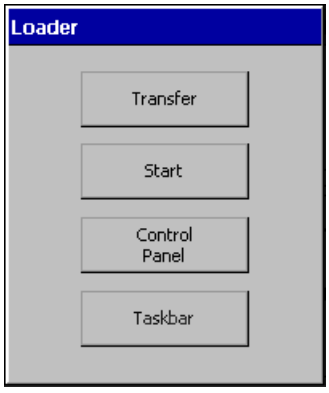
1. Setting up the HMI device.
2. Configure the ProSave on the configuration PC.
3. Start transfer.

Switch the HMI device to transfer mode

Procedure

For the transfer of data you must switch the HMI device to "Transfer":

1. Switch on the HMI device.
   Before the HMI device opens an existing project, the "Loader" selection field appears.
2. Click on the "Transfer" button.
   The dialog window displays the message "Connecting to host ...".



Configuring ProSave on the configuration PC

Requirement

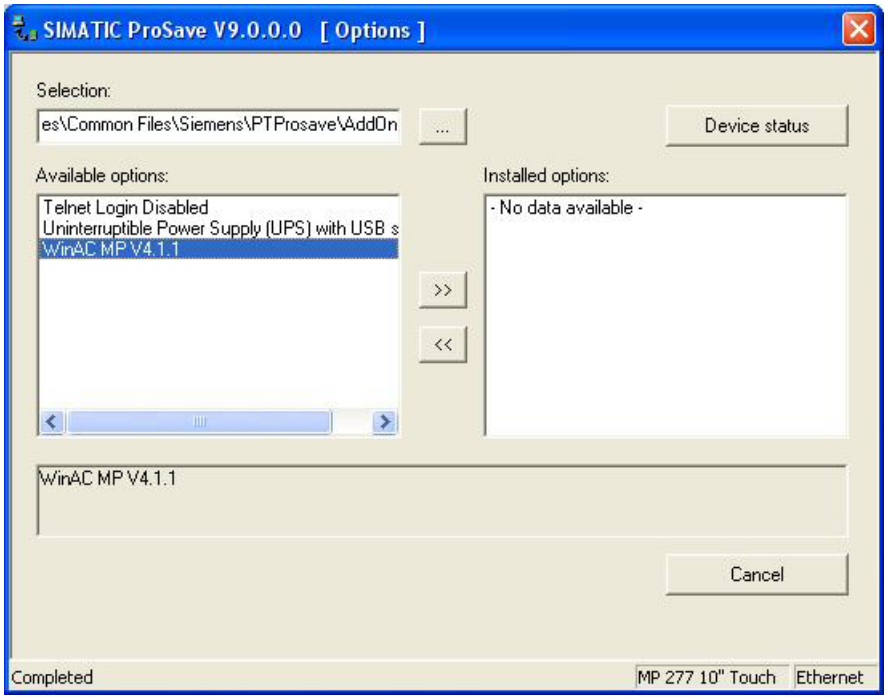
The HMI device was switched to transfer mode.

Procedure

To set up a configuration PC to transfer WinAC MP, proceed as follows:

1. Start ProSave.
2. Select the "General" tab.
3. Select the device type of your HMI device.
4. Select the "Ethernet" connection type.

5. Specify the IP address or the computer name of the HMI device.
6. Start the transfer of WinAC MP.



Start transfer

Requirement

- The HMI device was switched to transfer mode.
- ProSave is set up on the configuration PC.

Procedure

To start the transfer of WinAC MP files, follow these steps:

1. Open the "Options" tab in transfer mode and click the "Device status" button. If no error message is shown then the communication connection is valid.

   Note

   If there is a communication error, then you must check the following circumstances:

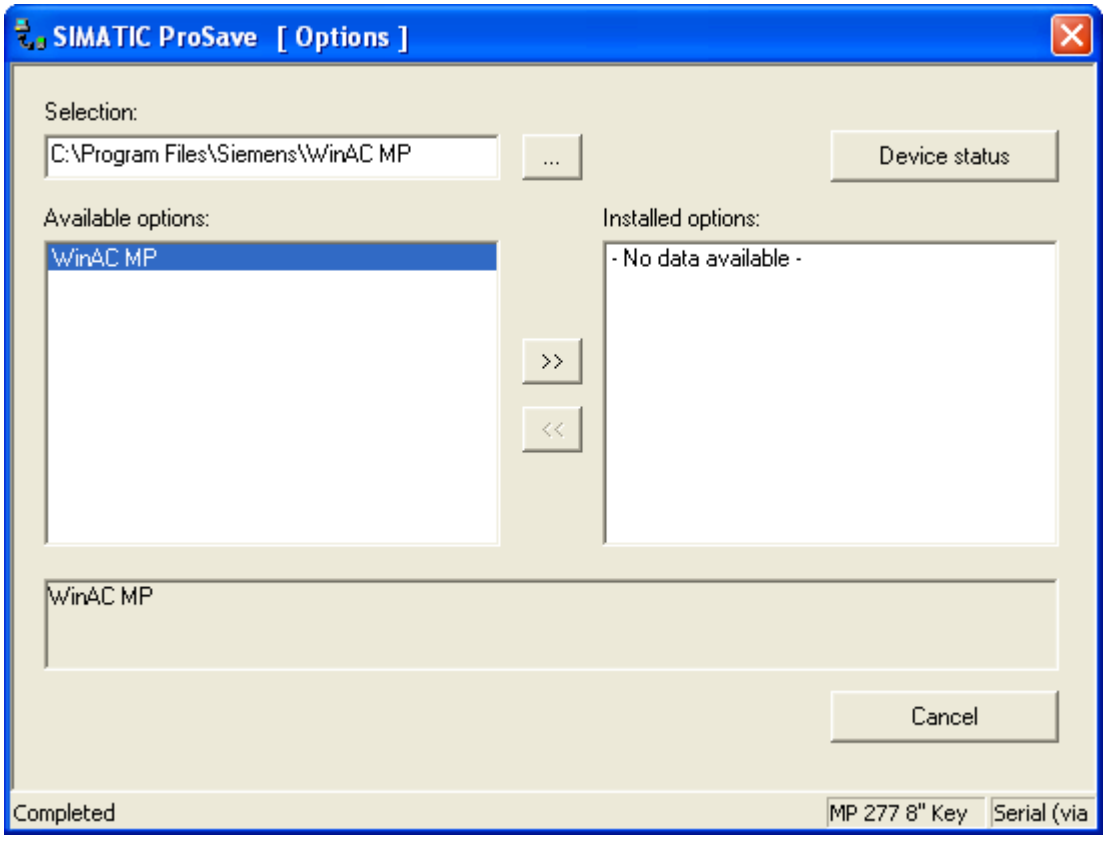
   • The HMI device must be in Transfer mode.

   • Check the cable connections for a correct and firm connection.

   • The settings in ProSave and on the HMI device must be correctly defined for the connection.

2. In the "Options" tab, select "WinAC MP" below "Available options".

3. To transfer WinAC MP to the HMI device, click the >> button.



   ProSave then starts to load the software of the WinAC MP control to the HMI device. While the files are being transferred the configuration PC shows the progress of the transfer. The following messages are displayed on the HMI device in the "Transfer" dialog box:

   – Progress of the file receipt on the HMI device.
   – Unzip data in the Flash file system.
   – Save data in the Flash file system.

   At the end of the transfer, a message confirms a successful transfer.

4. When transfer is complete, you will be asked to restart the device. Restart the device.

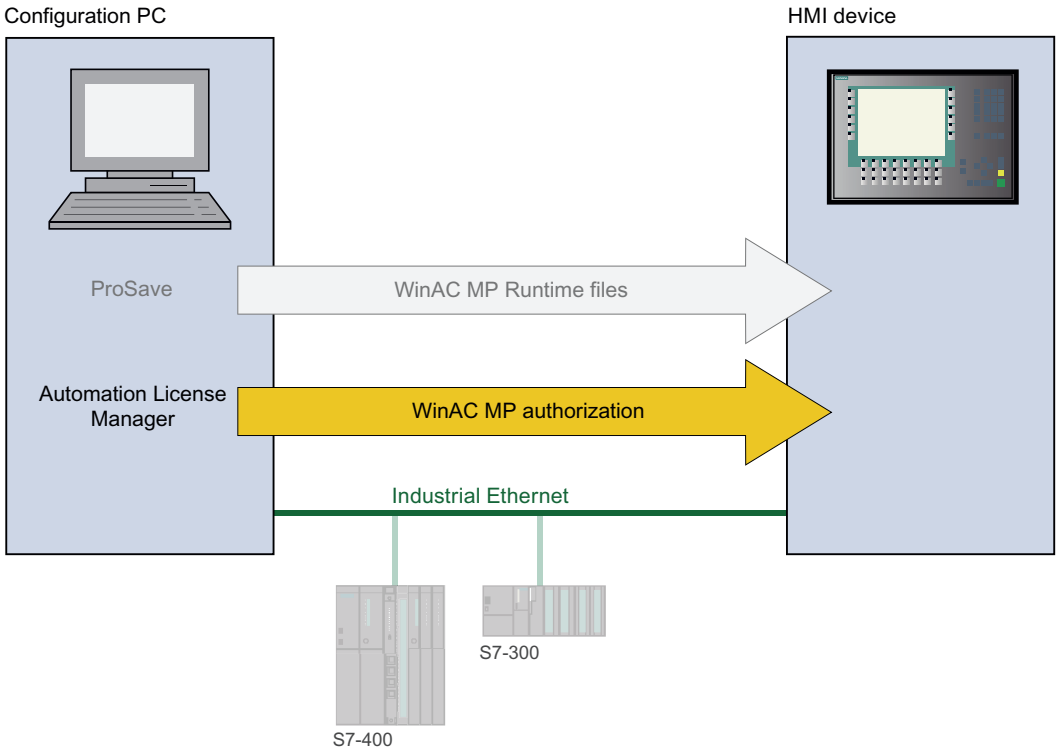
Result

The WinAC MP control is now installed on the HMI device. Now, transfer the authorization for WinAC MP to the HMI device.

12.11.20.6 Transferring authorization to the HMI device

Introduction

You need a license key to operate WinAC MP. The license key is transferred with the Automation License Manager to the HMI device.



Requirements

The storage location of the license key can be called from the configuration PC.

Procedure

To transfer the WinAC MP authorization to an HMI device, follow these steps:

1. Select the "Authorize/License" command from the "Online > Device maintenance" menu to start WinAC MP authorization.
   Automation License Manager is launched.
2. Select the "Connect HMI device" menu command.
3. Select the device type of the HMI device.
4. Select the type of connection and set the parameters for the connection.
   The connection to the HMI device is established. The connected HMI device is displayed.
5. Select the source drive.

6. Drag one or more license keys to the HMI device.
   The license keys are then transferred to the HMI device.
7. Close the Automation License Manager after a successful transfer.

12.11.20.7 WinAC MP system library

Screens from the system library

The screens from the system library are optimized for a 6' display with a resolution of 320 x 240. If you use a display of a different size, you can adapt the screens for your display in WinCC.

Standard screens and tags

The WinAC MP-system library offers you the standard screens "WinAC_MP_Home" and "WinAC_MP_Tuning" and the tags "WinAC_MP_Tags". To use the standard screens, move them to the "Screens" folder in the project tree.

In addition to the standard screens, the tags must be copied separately. To do this, drag "WinAC_MP_Tags" from the WinAC MP-system library to the "HMI_Tags" folder in the project tree.

12.12 Using global functions

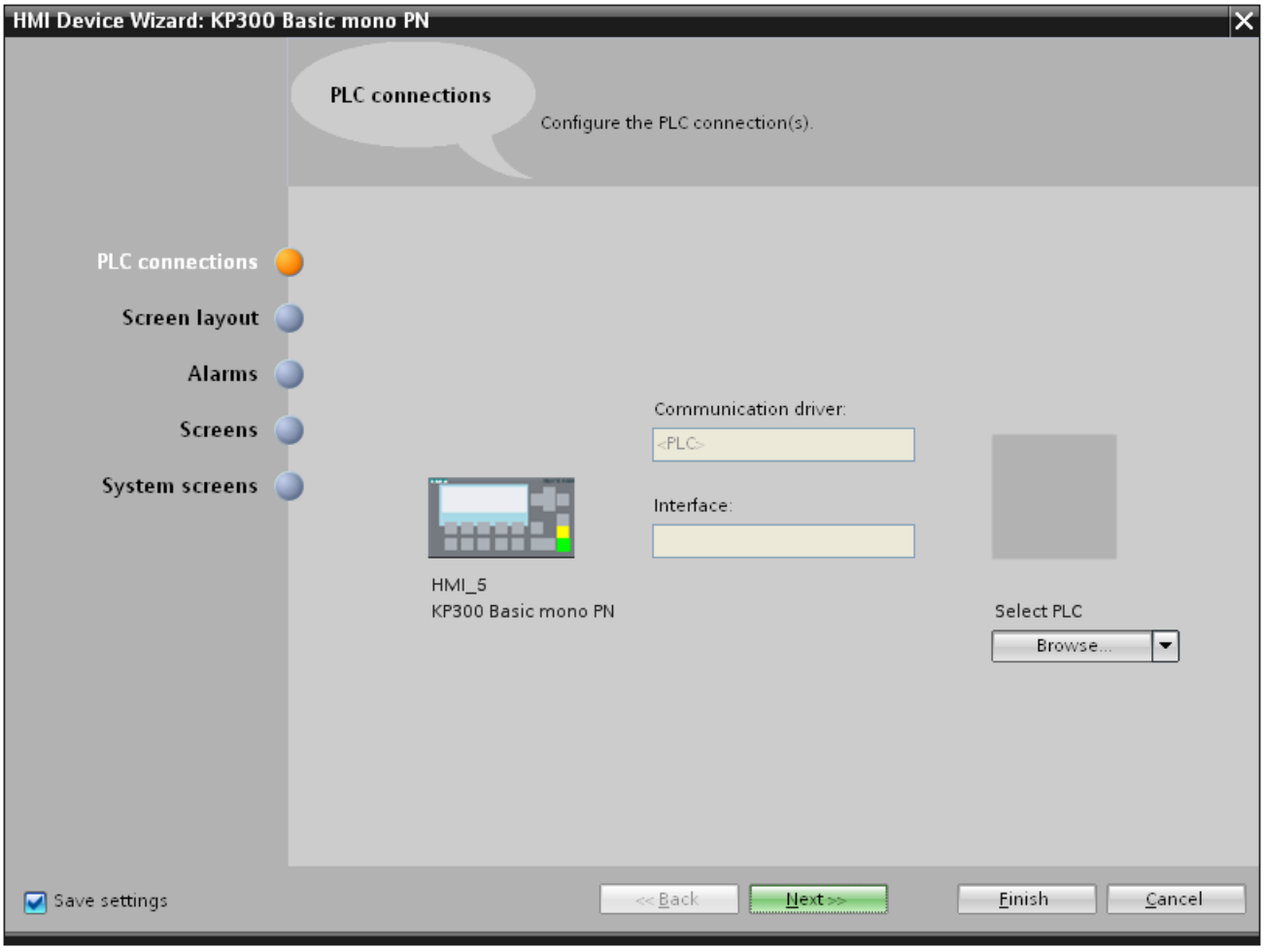
12.12.1 HMI device wizard basics

Introduction

The HMI device wizard will automatically start when you create a new HMI device in your project.

HMI device wizard

The HMI device wizard will guide you through each dialog step by step and help you set up a device. You use the HMI device wizard to specify the basic settings for your HMI device, such as screen layout and the connection to your PLC.



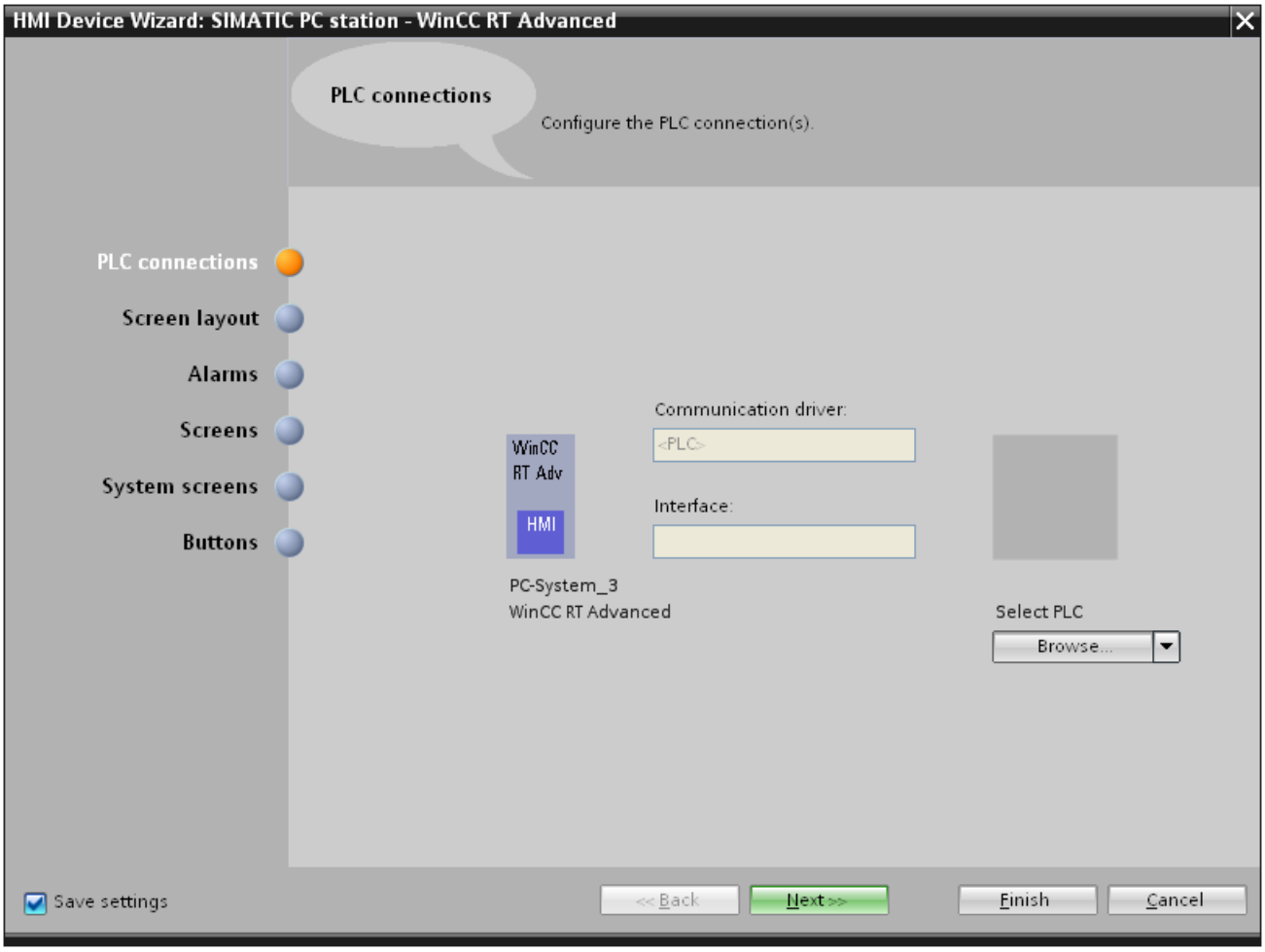
See also

HMI device wizard basics (Page 6770)

12.12.2 HMI device wizard basics

Introduction

The HMI device wizard will guide you through each dialog step by step and help you set up a device. You use the HMI device wizard to specify the basic settings for your HMI device, such as screen layout and the connection to your PLC.



Opening the HMI device wizard

1. Add a new device to your project, for example, RT Advanced.
2. Click on the device in the project tree.
3. Select "Start the HMI device wizard" in the shortcut menu. The "HMI device wizard" opens.

Note

If you make changes after adding the device, such as adding a new screen, the HMI device wizard will not open again.

Note

When you create a device with a color display using the HMI device wizard, the graphics of the navigation buttons may be displayed in black and white. This error only occurs, however, if the new device is created with the same name as a device with a monochrome display which has been deleted in the meantime.

You can avoid this error by always deleting the associated graphics in the Graphics collection whenever you delete a device from the project.

See also

HMI device wizard basics (Page 6766)

12.12.3 Working with libraries

12.12.3.1 Basics on libraries

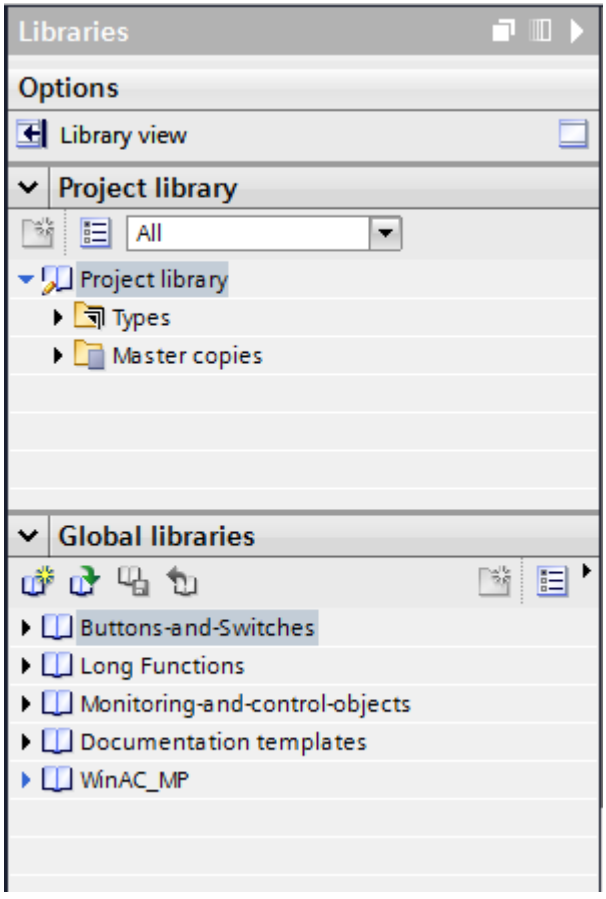
Introduction

Store all objects you need frequently in the libraries. An object that is stored in the library only has to be configured once. It can then be used repeatedly as often as required. Library objects extend the number of available screen objects and increase the effectiveness during configuration through the multiple usage of ready-to-use objects.

Your WinCC software package is supplied with comprehensive libraries that contain, for example, "Motor" or "Valve" objects. You may, however, define your own library objects.

You manage libraries in the "Libraries" task card or library view. The following libraries are available:

- Project library
- Global libraries



Note

There is a symbol library in the "Tools" task card in the "Graphics" palette.

Project library

There is one library for each project. Objects of the project library are stored alongside with the project data and are available only for the project in which the library was created. If the project is moved to another PC, any project library created in it is also moved.

To use the library object of the project library in other objects, move or copy the object into a global library.

Global libraries

A global library is saved independently of the project data in its own file with the extension *.al12.

A project can access several global libraries. A global library may be used concurrently in several projects.

When a library object is changed by a project, this library will be changed in all projects in which these libraries are open.

Library objects

A library can contain all WinCC objects. Examples:

- Complete HMI device
- Screens
- Display and control objects including tags and functions
- Graphics
- Tags
- Alarms
- Text and graphics lists
- Faceplates
- User data types

See also

Displaying library objects (Page 6786)

Storing an object in a library (Page 6789)

Inserting a library object (Page 6790)

Creating a global library (Page 6781)

Saving a global library (Page 6782)

Opening a global library (Page 6783)

Master copies and types (Page 6775)

Screen basics (Page 3889)

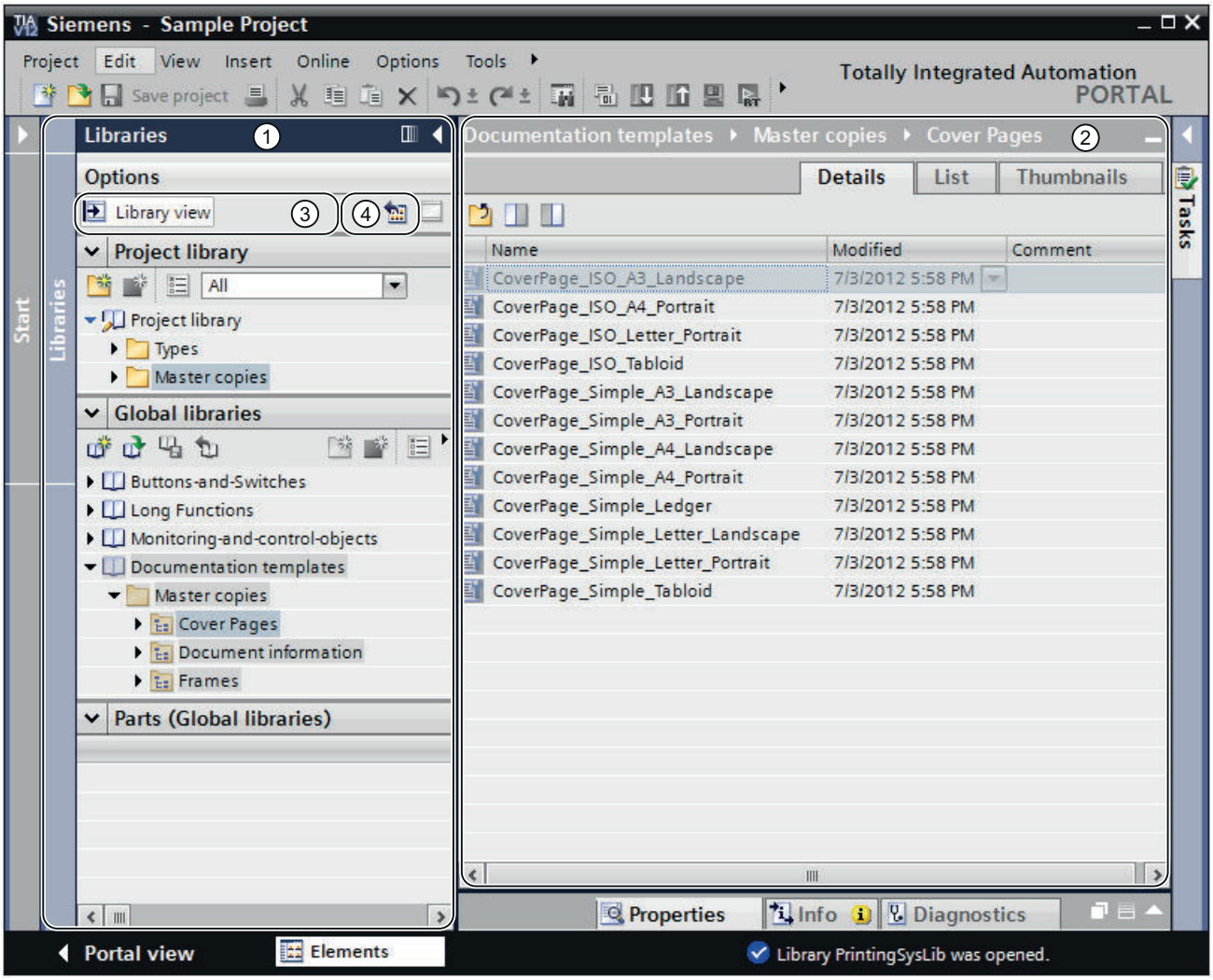
12.12.3.2 Overview of the library view

Function of the library view

The library view combines the functionality of the "Libraries" task card and the overview window. In the library view, the elements of a library are displayed in various views. In the details view, for example, you see additional properties of the individual elements. You can also edit and version the types in the library view.

Layout of the library view

The following figure shows the components of the library view:



1	Library tree
2	Library overview
3	"Library view" button
4	"Open or close library overview" button

Library tree

The library tree is similar to the "Libraries" task card, apart from a few minor differences. In contrast to the task card, there is no "Elements" palette, because the elements are displayed in the library overview. In addition, you can close the library view in the library tree, or open and close the library overview.

Library overview

The library overview corresponds to the overview window and displays the elements of the currently selected object in the library tree. You can display the elements in three different views. You can also perform the following actions in the library overview, for example:

- Copying elements
- Moving elements
- Versioning types
- Editing faceplates and HMI user data types
- Editing type instances

12.12.3.3 Master copies and types

Introduction

Both the "Project library" and the "Global library" contain the two folders "Master copies" and "Types". You can create or use the library objects either as a master copy or a type.

Master copies

Use master copies to create independent copies of the library object.

Types

Create instances of objects of the "Types" folder and use these in your project. The instances are bound to their respective type.

Administration of the library objects

You can copy and move library objects to another library. You can only copy master copies to the "Master copies" folder or any sub-folder of "Master copies". You can also only insert types in the "Types" folder or any sub-folder of "Types".

See also

Basics on libraries (Page 6769)

State of type versions (Page 6791)

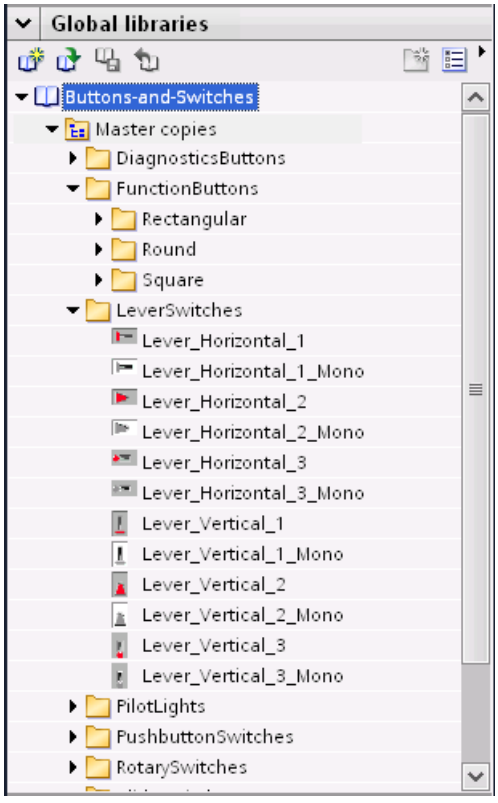
12.12.3.4 Libraries in WinCC

Introduction

The WinCC software package includes extensive libraries. Sorted by topic into folders, they contain preassembled graphic objects which you can use in screens for operation and monitoring of your plant.

Global library "Buttons and Switches"

The libraries "Buttons and Switches" offer a wide selection of buttons and switches.



The folders divide switches and buttons into categories. The "DiagnosticsButtons" folder contains the object "System diagnostics indicator", for example. You use the "System diagnostics indicator" object for system diagnostics in your plant.

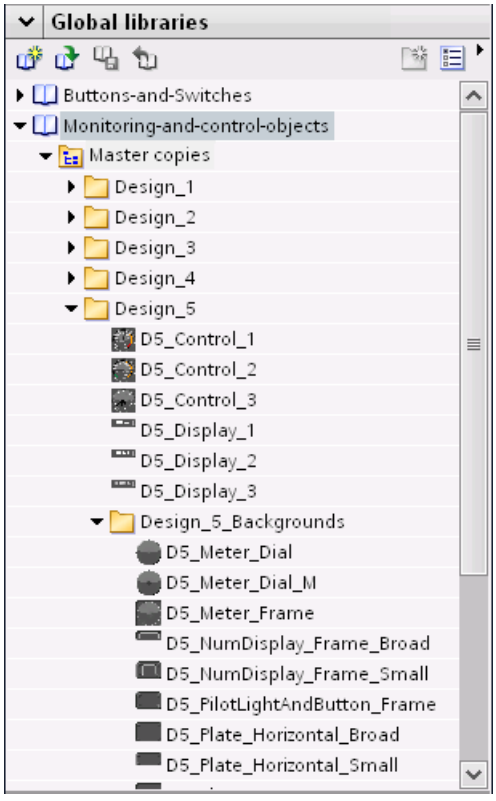
Note

You can only use the objects in the "DiagnosticsButtons" folder on Comfort Panels.

You cannot use objects which have "Switch" in the object name or in the associated folder name in Runtime Professional.

Global library "Monitoring and Control objects"

The "Monitoring and Control objects" library offers more complex display and operating objects in several designs and corresponding control lights, buttons and switches.



In addition, graphics views for the designs are stored in the "Design_Backgrounds" folder; they can be used as object backgrounds for the custom extension of the scope of the library.

Note

You cannot use objects which have "Switch" in the object name in Runtime Professional. The same applies for the object "D5_Display_3" with the date/time field it contains.

12.12.3.5 Managing libraries

Overview of the library management

Function of the library management

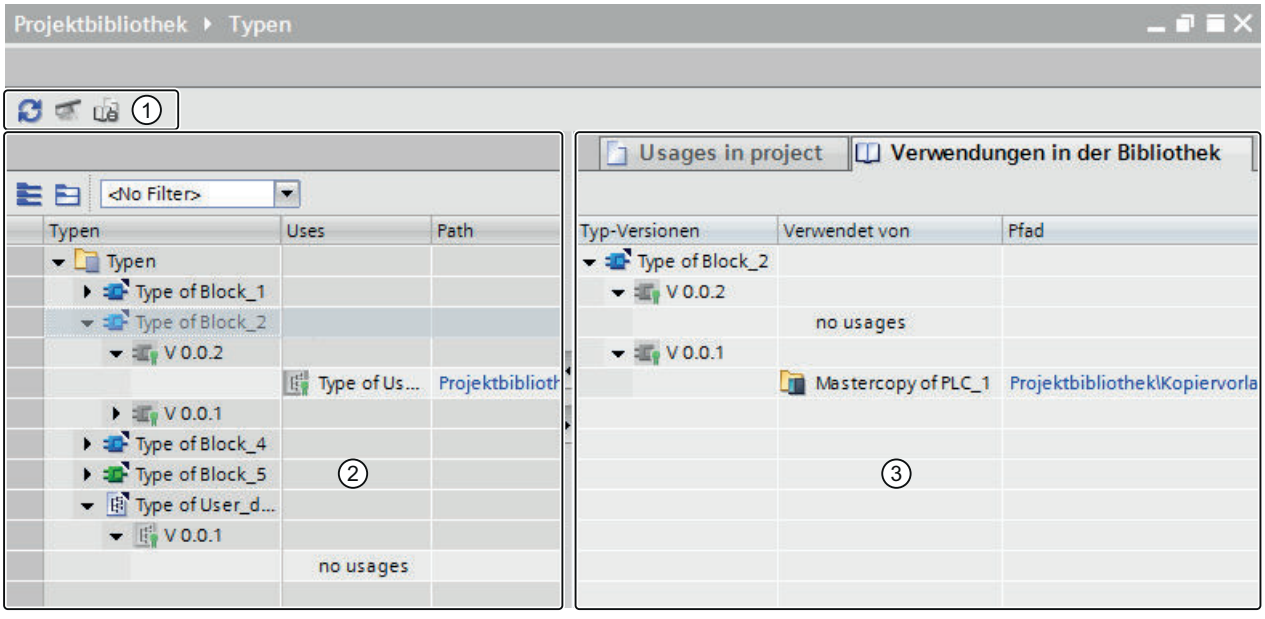
Master copies and types with dependencies to other library elements are subject to some functional restrictions. They can, for example, not be deleted as long as dependencies still exist. This prevents other library elements from becoming useless. The library management is used to identify the dependencies and to create an overview of the work progress.

The library management offers the following functions:

- Display of the correlations of types and master copies
  If a type is referenced in other types or master copies, the correlations are displayed in the library management. You will also be able to see which library elements reference a type or a master copy.
- Display of points of use of types in the project
- Display all types that include a version with the "In testing" or "In progress" status

Layout of the library management

The following figure shows the components of the library management:



① Toolbar of the library management
② "Types" area
③ "Uses" area

Toolbar of the library management

You can perform the following tasks in the toolbar of the library management:

- Update view
  If the project was changed, you can update the view of the library management.
- Clean up library
  You can clean up the project library and global libraries. By cleaning up a library, you delete all types and type versions that are not linked to an instance in the project.
- Harmonize project
  By harmonizing a project, you adapt the names and the path structures of type uses in the project to the corresponding names and path structures of the types within a library.

"Types" area

The "Types" area displays the contents of the folder that you have selected in the library view. You can open or close all types by using the buttons in the toolbar of the "Types" area. You can also use the "Filter" drop-down list to filter the view and display all types with the "In testing" or "In progress" status only. For each type, the types that it references are displayed.

"Uses" area

The "Uses" area gives you an overview of the points of use of the selected types and master copies. The "Uses" area is divided into two tabs:

- "Uses in the project" tab
  The "Uses in the project" tab is used to show the instances of type versions and their respective point of use in the project. When you select an instance, you can show the cross references of the instance in the project in the Inspector window.
- "Uses in the library" tab
  The "Uses in the library" tab is used to show all points within the library at which a type or a master copy is used.

Opening library management

Procedure

To open the library management, follow these steps:

1. Open the library view.
2. Select a type or any folder that contains types.
3. Select the "Library management" command from the shortcut menu.

Result

The library management opens and the types are displayed with their versions.

Filtering types in the library management

Introduction

A filter function in the library management enables you to limit the displayed types. The following filters are available:

- Display all types that include a version with "in progress" status
- Display all types that have no instances in the project
- Display all types that have more than one version
- Display all released types

Requirement

At least one type has been created.

Filtering for all types with "in progress" status

1. Select the "Types" folder in the project library.
2. Select the "Library Management" command in the shortcut menu of the "Types" folder. The library management opens.
3. Select "Pending changes" from the "Filter" drop-down list. The "Types" area only displays types that have the "in progress" status.

Filtering for all types that have no instances in the project

1. Select the "Types" folder in the project library.
2. Select the "Library Management" command in the shortcut menu of the "Types" folder. The library management opens.
3. Select "No instances in project" from the "Filter" drop-down list. The "Types" area only displays types that have the "in progress" status.

Filtering for types with multiple versions

1. Select the "Types" folder in the project library.
2. Select the "Library Management" command in the shortcut menu of the "Types" folder. The library management opens.
3. Select "Multiple versions" from the "Filter" drop-down list.
4. The "Types" area only displays types that have more than one version.

Filtering for all released types

1. Select the "Types" folder in the project library.
2. Select the "Library Management" command in the shortcut menu of the "Types" folder.
   The library management opens.
3. Select "Released types" from the "Filter" drop-down list.
   The "Types" area only displays types that have released versions.

See also

Opening library management (Page 6777)

Creating a faceplate type (Page 4062)

Creating a global library

Introduction

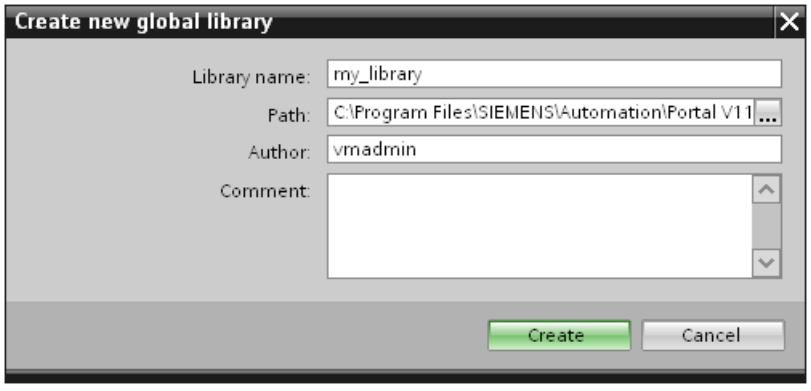
In the libraries you store the configured objects that you want to use several times in your configuration. To use objects in several projects, create a global library.

Requirement

- A project is open.
- The "Libraries" task card is displayed or the "Library view" is open.

Procedure

1. Click the icon under "Global library".
   The "Create new global library" dialog opens.



2. Enter a name.
3. Select the path where the new library is to be stored.
4. Click "Create".

Result

The new library is shown in the "Global libraries" palette. The global library contains the "Types", "Master copies" and "Common data" folders. Under the "Common data" you can find reports for the global library.

A folder with the name of the global library is created in the file system at the storage location of the global library. This actual library file is given the file name extension ".al12".

See also

Basics on libraries (Page 6769)

Saving a global library

Introduction

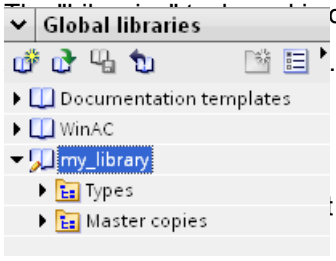
A global library is stored in a separate file on your hard disk drive. The file contains the objects of the global library including the referenced objects. E.g. the reference of a tag which was configured on an I/O field is also saved in the library.

WinCC prompts you to save the global libraries when you close WinCC or your project without saving. You also can store the global library during configuration, without storing the entire project.

Requirement

- You have opened a project which contains at least one library.
- displayed or the library view is open.
- .

Procedure

1. 

you want to save.
2. Click the icon in the "Global library" palette.

You can alternatively select the "Save Library" command in the shortcut menu.

If you want to store the global library in a different folder, select "Save as" in the shortcut menu. Select the path in which you want to store the new library and enter a file name.

Result

The global libraries are saved under their current file name or a file name you have specified.

See also

Basics on libraries (Page 6769)

Opening a global library

Introduction

In WinCC, the global libraries are stored in separate files. You can use a global library in every project.

Requirement

- You have saved a global library.
- A project is open.
- The "Libraries" task card is displayed or the library view is open.

Procedure

1. Click the icon in the "Global library" palette.
   The "Open global library" dialog box is displayed.
2. Select the path in which the library is stored.
3. Click "Open".

Note

To have the access to a global library from multiple projects, open the global library read-only. As soon as a global library is not opened read-only, access from other projects is blocked.

Result

WinCC displays the opened global library in the "Global library" palette.

See also

Basics on libraries (Page 6769)

Displaying logs of global libraries

Logs listing all changes made to the global library are created when global libraries are updated. The logs are stored together with the global library and are always available once you have opened the global library.

Procedure

To open the logs of a global library, follow these steps:

1. Open the global library in the "Libraries" task card or in the library view.
2. Open "Common data > Logs" in the lower-level folder.
3. Double-click the required log.
   The log opens in the work area.

Updating a project with the contents of a library

Introduction

After you have edited several types in the project library, update all instances in the project to the most recent version of the types from the project library.

Requirement

The "Libraries" task card or the library view is open.

Procedure

1. Select the project library.
2. Select "Update > Project" from the shortcut menu. A dialog opens.
3. Select either the entire project or individual devices for the update.
4. Select the "Delete all unused versions of affected types" check box to delete all older versions of the updated types from the project library.
5. Click "OK" to confirm.

Result

All instances of the types are updated in the project to the most recent version of the selected types in the project library.

You can find a log of the update process in the project tree under "Common data".

Updating a library with the contents of another library

The following options are available for updating libraries:

- Updating a global library with types from another global library
- Updating the project library with types from a global library

Each of the following elements can be selected as source for the update:

- An entire library
- Individual folders within a library
- Individual types

Requirement

To update a global library, open the library with write rights.

Requirement

The "Libraries" task card or the library view is open.

Procedure

To update a library with the contents of a different library, follow these steps:

1. Select the entire library or a folder within the library or individual types.
2. Right-click the source and select the "Update > Library" command from the shortcut menu. The "Update library" dialog box opens.
3. Select the type of library you want to update:
   - Select "Update the project library" to update the project library with types from a global library.
   - Select "Update a global library" if you want to update a global library.
4. Optional: Select the global library you want to update from the drop-down list.
5. Select the "Delete all unused versions of affected types" check box to delete all older versions of the updated types from the project library.
6. Click "OK" to confirm.

Result

- Types not yet available in the target library are supplemented there with all their versions. More recent versions are added to the types that already exist in the target library.
  If a more recent version of a type already exists in the target library, the latest version is nevertheless copied and automatically assigned a newer version number.
- A log listing all performed changes to the target library is created for the update process.
  If you have updated the project library, you can find the log in the project tree under "Common data > Logs".
  If you have updated a global library, you can find the log in the "Common data > Logs" folder in the level below the global library.

12.12.3.6 Managing objects in a library

Displaying library objects

Introduction

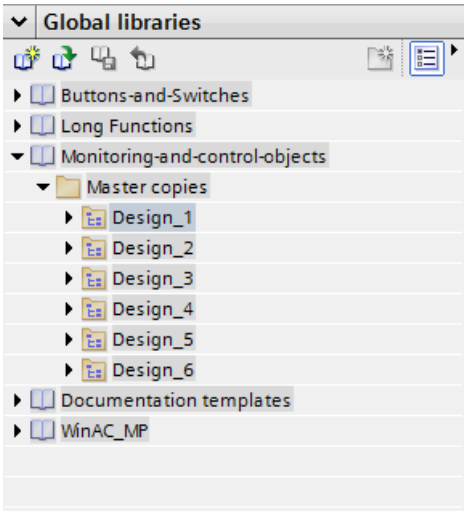
The libraries are displayed as file folders in the corresponding palette. The elements contained in the library are displayed in the file folder and in the "Elements" palette.

Requirement

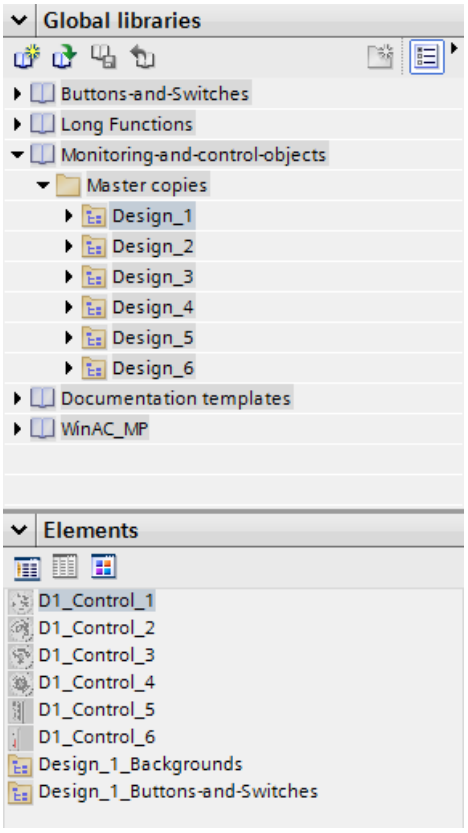
- At least one library object has been created in a library.
- The "Libraries" task card is opened.

Procedure

1. Select the library in the corresponding palette whose library objects you want to display.



2. Click .
The contained library objects are displayed in the "Elements" palette.



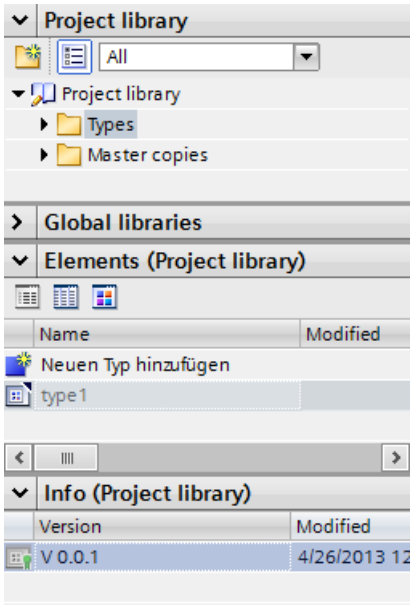
3. Click one of the following icons:

Icon	Description
	Element view in detailed mode
	Element view in list mode
	Element view in overview mode with icons

When several objects are assigned to the library with a multiple selection, only one of the objects is shown in the "Elements" palette. The individual components of this element are displayed in the "Parts" palette.

Show parts of the library objects

1. Select the library in the corresponding palette from which you want to view the components of an element.
2. Click .
3. The contained library objects are displayed in the "Elements" palette.
4. Select the element.
   The "Parts" palette shows the objects of which the element consists.



Result

The library objects are displayed in accordance with the configuration. The components of the faceplates are displayed.

See also

Basics on libraries (Page 6769)

Storing an object in a library

Introduction

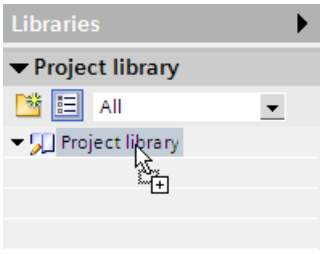
You can store all of WinCC objects, such as screens, tags, graphic objects or alarms in your libraries. You can use drag-and-drop to move the corresponding object from the work area, project window or detail view to the library. In a library you have divided into categories, you can directly add objects to a specific category.

Requirement

- The "Screens" editor is open.
- A screen object has been created in the work area of the screen.
- The created libraries are displayed.

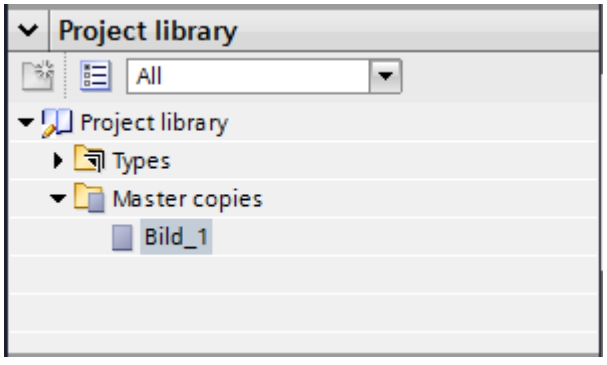
Procedure

1. Select the object in the work area of the "Screens" editor.
2. Drag-and-drop the object from the work area to the desired library.
   The mouse pointer is transformed into a crosshair with an appended object icon.



Result

The object is saved to the library for further use in multiple instances of your configuration.



See also

Basics on libraries (Page 6769)

Inserting a library object

Introduction

The system always assigns the inserted library object a name which consists of the name of the object type and of a consecutive number.

If the inserted object already exists, you have the option of replacing the object or of saving it under a new name.

You cannot insert library objects that are not supported by the HMI device.

Note

If you insert a screen with interconnected template from the library, the template will also be inserted. Any existing matching template is not used.

Requirement

- The "Libraries" task card is opened.
- The editor in which you want to insert the library object is open.

Procedure

1. Select a library object from the library.
2. Drag-and-drop the library object to the position in the work area where you want to insert the object.
   The library object is inserted.

Result

If the object was contained in the "Copy templates" folder, you have inserted an independent copy of the library object in the editor.

If the object was contained in the "Types" folder, you have inserted an instance of the library object in the editor.

See also

Basics on libraries (Page 6769)

12.12.3.7 Using types and their versions

State of type versions

Introduction

Depending on the point of use, the version of a type has different statuses.

Released version

The "released version" status is available for all types, regardless of the point of use.

If you want to edit a released version, you must first create a new test version or an "in progress" version.

Released type versions of scripts and screens can be opened and viewed at their instance.

"In progress" version

The "in progress" status is possible for the following types:

- Faceplates
- User data
- Styles
- Style sheets

When you create a new type or a new version of a released type, the type is assigned the status "in progress".

Types with the "in progress" status can be edited in the library view without the need for a reference to an instance in the project. Upon release, the compatibility of the type is tested by a consistency check.

"In testing" version

Only versions of scripts and HMI screens have the "in testing" status.

When you create a new version of a type, the type is assigned the status "in testing".

A version with "in testing" status is linked to an instance in the project. You can set only one version to "in testing" for each type at a given time.

A version in testing may only be linked to a single instance in the project. Therefore, it is not possible to copy an instance to the clipboard, to duplicate it or to create an additional type from the instance as long as it has "in testing" status.

See also

Generating a script as a type (Page 6792)

Generating a screen as a type (Page 6792)

Creating a faceplate type (Page 4062)

Create a new version of a type (Page 6794)

Master copies and types (Page 6773)

Generating a script as a type

Requirement

- A project is open.
- An HMI device has been created and opened.
- The project tree is open.
- The "Libraries" task card is opened.

Procedure

1. Open the "Scripts" editor in the project tree.
2. Create a new script.
3. Select the script in the project tree.
4. Drag-and-drop the script into a library in the "Libraries" task card. A dialog opens.
5. Enter a name.
6. Enter a comment.

Result

You have created a type version of a script in the library. The created type is stored as a released version in the library. An instance of the type is used in the project.

To change the script create a new version of the script.

See also

State of type versions (Page 6789)

Generating a screen as a type

Requirement

A project is open.

An HMI device has been created and opened.

The project tree is open.

The "Libraries" task card is opened.

Procedure

1. Open the "Screens" editor in the project tree.
2. Create a new screen.
3. Select the screen in the project tree.
4. Drag-and-drop the screen into a library in the "Libraries" task card. A dialog opens.
5. Enter a name.
6. Enter a comment.

Result

You have created a type of a screen in the library.

The created type is stored as a released version in the library. An instance of the type is used in the project.

To change the screen create a new version of the screen.

See also

State of type versions (Page 6789)

Generating a style as a type

Introduction

To define a new style, add a new type in the project library.

Adding a new style

1. Open the "Libraries" task card.
2. Select the "Add new type" command under "Types" in the shortcut menu of the project library. A dialog opens.
3. Select "HMI style".

Result

The new style is created and displayed under its selected name in the project library. The HMI style type is assigned the status "in progress" and the version 0.0.1.

Creating a style sheet as a type

Introduction

To define a new style sheet, add a new type in the project library.

Adding a new style sheet

1. Open the "Libraries" task card.
2. Select the "Add new type" command under "Types" in the shortcut menu of the project library. A dialog opens.
3. Select "HMI style sheet".
4. Assign a meaningful name to the style sheet.
5. Select the style sheet category from the list.

Result

The new style sheet is created and displayed under its selected name in the project library.

The HMI style sheet type is assigned the status "in progress" and the version 0.0.1.

Create a new version of a type

Principle

If you create a new version of a type, the point of use of the type determines the status of the newly created version.

Requirement

The "Libraries" task card is opened.

A type has been created and released.

Procedure

1. Select the released type.
2. Select "Edit type" in the shortcut menu.

Result for types of faceplates, user data, and styles

A new version of the type is created.

The version has the status "in progress". The library view opens.

Result for types of scripts and screens

A dialog opens.

After you have selected the settings in the dialog, the version is set to the status "in testing". The instance used in the project is set to the status "in testing". The library view opens.

See also

State of type versions (Page 6789)

12.12.4 Importing and exporting project data

12.12.4.1 Importing and exporting project data

Introduction

WinCC gives you the option of exchanging project data between the projects or copying them to external applications.

Exporting and importing between projects

You can export the following project data from a project and import them into another project.

- Recipe data records
- Alarms
- Tags
- Text lists
- Project texts

Exporting and importing reduces the workload. Instead of creating new data records, you use data already created in previous projects.

Editing the export file

The following file formats are available for export and import depending on the editor:

- *.xlsx for alarms, tags, project texts and text lists
- *.csv for recipe data records

You can edit the import file in Excel, for example.

XLSX file format

XLSX format is a file format for Excel tables based on the Open XML format. XLSX files are optimized for Microsoft Excel 2007.

You can sort the columns as required in the XLSX file.

CSV file format

CSV stands for Comma Separated Value. In this format, the columns of the table that contain the names and the value of the entry are separated by semicolons. Each table row terminates with a line break. You can also open the CSV file for editing in Excel.

Importing project data

When project data is imported, the objects in the project are created.

The syntax of the import file is checked during import. The accuracy of the values imported and dependencies between the imported values are not checked.

Any errors found in the imported data are reported when the project is compiled.

Copying in Excel format

In all spreadsheet editors you can copy the content in Excel format into the buffer of your PC. Then you insert the project data in Excel format directly into any application outside the TIA Portal. To this purpose you use the corresponding command in the shortcut menu of the work area:

- If you select the command in the shortcut menu of the line header, the complete line is copied into the buffer.
- If you select the command in the shortcut menu of a cell, only the cell content is copied into the buffer.
- If you select several lines and select the command, all marked data are always copied into the buffer.

This data exchange is only possible as an export.

12.12.4.2 Import and export of recipes

Exporting recipes

Introduction

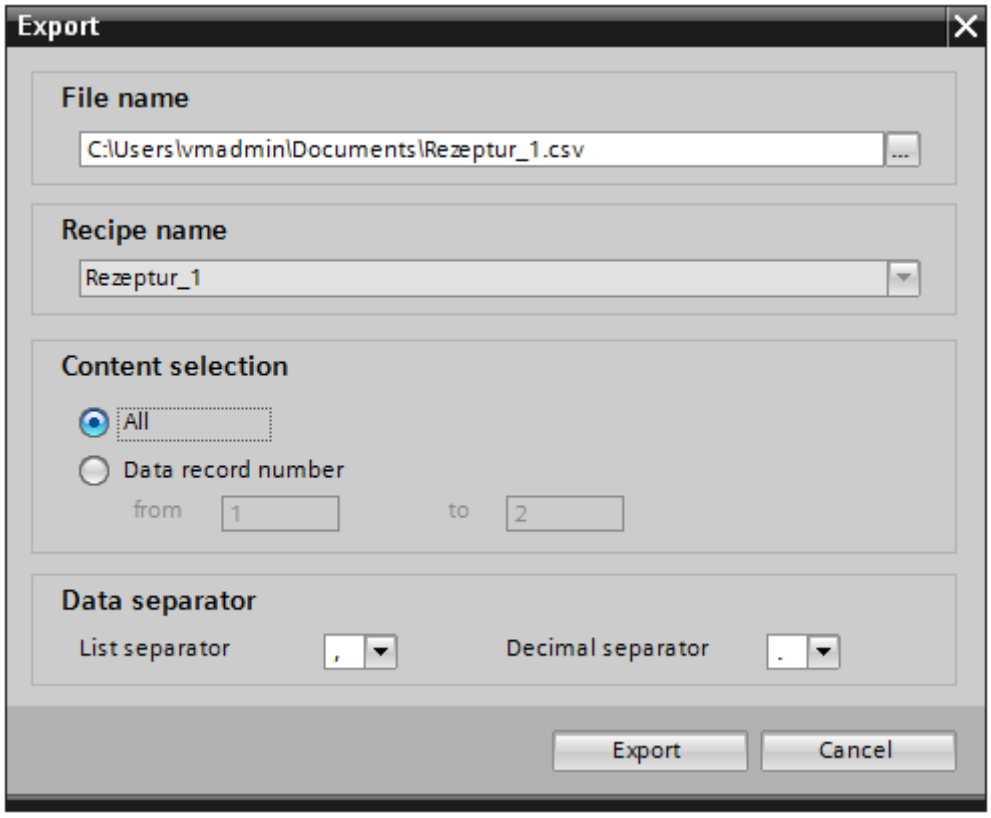
WinCC features an export function for exporting data records from recipes.

Requirements

- The WinCC project for the export is open.
- Recipes have been created in a project.
- The "Recipes" editor is open.

Exporting recipes

1. In the "Recipes" editor, select the recipe with the data records you want to export.
2. Click .
   The "Export" dialog box opens.



   The selected recipe is shown under "Recipe selection".
3. Under "Content selection", specify if all or only selected data records are to be exported.
4. Under "File selection", specify the file in which the recipe data is to be stored.
5. Specify the list separator and decimal separator under "Data separation".
6. Click "Export."
   The export will start.

Result

The exported data has been written to a CSV file. The CSV file will be stored in the specified directory.

Importing recipes

Introduction

Recipes are identified by their name. The recipe name must therefore be unique. Open the import file in a simple text editor to check that it has the correct data structure.

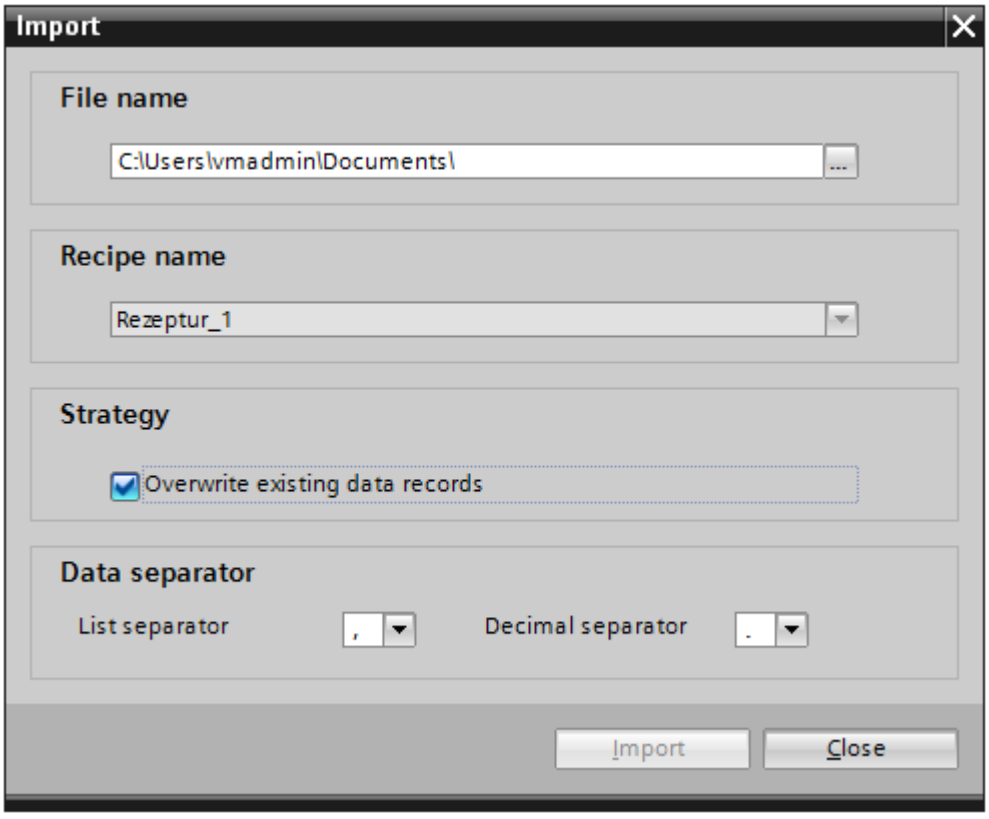
Specify whether or not existing data records should be overwritten by records with the same name during the import.

Requirements

- A CSV file containing at least one recipe has been created.
- The WinCC project for the import is open.
- The "Recipes" editor is open with at least one recipe.

Importing a recipe

1. In the "Recipes" editor, select the recipe with the data records you want to import.
2. Click .
   The "Import" dialog box opens.



The selected recipe is shown under "Recipe selection".

3. Select the file you want to import under "File selection".
4. Under "Strategy", specify if existing data records should be overwritten by records of the same name.
5. Under "Data separation", select the list separator and the decimal separator to use in the CSV file.
6. Click "Import".
   The import will start.

Result

The data records are created in the selected recipe. Depending on the setting for "Strategy", existing data records are overwritten by records with the same name from the CSV file.

Existing data records with the same name will also be imported from the CSV file if you deactivate the "Overwrite existing data records" option.

Format of recipe data

Introduction

This section describes the required format of the file for the import of recipes. The file containing the data of the recipes must be available in "*.csv" format. :

Structure of recipe data

The structure of the import file is fixed. The following example shows the structure of a recipe containing two recipe elements, each with two data records:

```
List separator=<List separator>Decimal symbol=<Decimal
separator><List separator><Line break>
<Name of the recipe><List separator><List separator><Line break>
LANGID_<ID of the language><List separator>
<Display name, recipe element 1><List separator>
<Display name, recipe element 2><Line break>
<Number recipe><List separator>
<Recipe data record number 1><List separator>
<Recipe data record number 2><Line break>
<Tag recipe element 1><List separator>
<Recipe data record 1 value 1><List separator>
<Recipe data record 2 value 1><Line break>
<Tag recipe element 2><List separator>
<Recipe data record 1 value 2><List separator>
<Recipe data record 2 value 2><Line break>
```

ID of the language

Use the "Windows language ID" in decimal notation, e.g. "1033" for English. Additional information is available in the documentation for the Windows operating system.

12.12.4.3 Importing and exporting alarms

Exporting alarms

Introduction

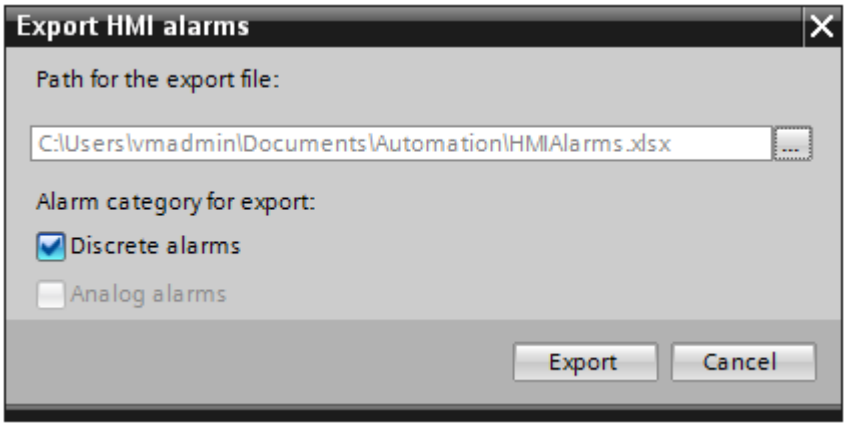
WinCC has an export function for alarms.

Requirements

- The WinCC project for the export is open.
- Alarms have been created in the project.
- The "HMI alarms" editor is open.

Exporting alarms

1. Click the button in "Discrete alarms" or "Analog alarms".
   The "Export" dialog box opens.



2. Click the "..." button and specify in which file the data are saved.
3. Specify whether you want to export "Disrete alarms" or "Analog alarms".
4. Click "Export". The export will start.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

See also

Importing alarms (Page 6801)

Format of the analog alarm data (Page 6802)

Format of the discrete alarm data (Page 6805)

Importing alarms

Introduction

Alarms are identified by their alarm number. The alarm numbers must be unique in the analog and discrete alarm types. Alarms with redundant alarm numbers will be overwritten. An alarm without an existing alarm number is created.

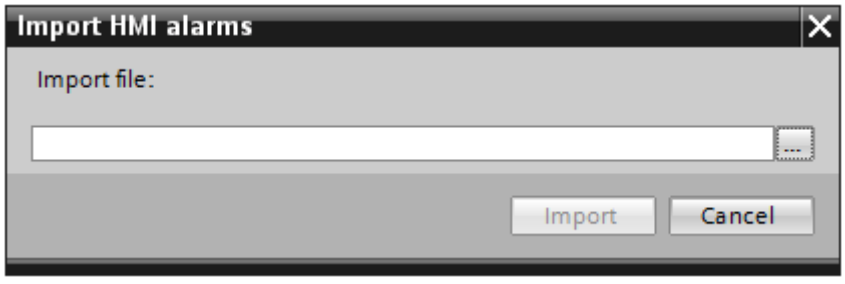
Any empty list entries for existing alarms contained in an xlsx file will be ignored for the purposes of the import. The entries of the existing alarms remain active and will not be replaced by empty ones.

Requirements

- An xlsx file with alarms has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.
- The "HMI alarms" editor is open.

Importing alarms

1. Click the button in "Discrete alarms" or "Analog alarms". The "Import" dialog box opens.



2. Click the "..." button and select the file that you want to import.
3. Click on the "Import" button. The import will start. A progress bar indicates the progress of the import operation.

Result

The corresponding alarms including alarm texts are created in WinCC on the basis of the import data. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign a trigger tag of an incorrect type, such as string, to an alarm. An error will be reported during compilation.

See also

Exporting alarms (Page 6798)

Format of the discrete alarm data (Page 6805)

Format of the analog alarm data

Introduction

This chapter describes the required format of the file for the import of analog alarms. The file containing the analog alarm data must be in "*.xlsx" format.

Structure of the alarm data

The import file in Microsoft Excel consists of a number of worksheets:

- Analog alarms(Analog alarms)
- Limits (Limits)

Each alarm is assigned a separate row in the import file. The import file with the analog alarms must be formatted as follows:

Example of the worksheet "Analog alarms"

	A	B	C	D	E
1	ID	Name	Event text [en-US], Alarm text	FieldInfo [Alarm text]	Class
2	1	Analog_alarm_1	AA1 Error-AC with maximum text length: abcdefghijklmnopqrstuvwxyz		Errors
3	2	Analog_alarm_2	AA2 Warning-AC this text should be bold		Warnings
4	3	Analog_alarm_3	AA3 SDm-AC this text should be italic		SDm
5	4	Analog_alarm_4	AA4 SDo-AC <u>this text should be underlined</u>		SDo
6	5	Analog_alarm_5	AA5 SystemAcknowledgement-AC <blink>this text should be flashing<		System_Ackn
7	25	Analog_alarm_25	Internal AA23 switchDT: Deadband mode in case of violation - value HL		AADT-Interna
8	26	Analog_alarm_26	Internal AA23 switchDT: Deadband mode in case of violation - percent		AADT-Interna
9	31	Analog_alarm_31	Internal AA23 switchDT: Low limit violation static		AADT-Interna
10	32	Analog_alarm_32	Internal AA23 switchDT: High limit violation static		AADT-Interna
11	33	Analog_alarm_33	Internal AA23 switchDT: Low limit violation dynamic		AADT-Interna
12	34	Analog_alarm_34	Internal AA23 switchDT: High limit violation dynamic		AADT-Interna
13	35	Analog_alarm_35	Internal AA23 switchDT: delay 3 seconds High limit violation		AADT-Interna
14	42	Analog_alarm_40	Internal AA23 switchDT: delay 3 seconds Low limit violation LLV		AADT-Interna
15	23	Analog_alarm_41	AA23 DT in event text: <field ref="0" /> Bool, <field ref="1" /> Byte, <fi	<ref id = 0; type = AlarmTag; T	ACAT
16	24	Analog_alarm_42	AA24 DT in event text: <field ref="0" /> Timer, <field ref="1" /> Counte	<ref id = 0; type = AlarmTag; T	ACAT

Table 12-156 Meaning of the entries

List entry	Meaning
ID	The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Name	Name of the analog alarm
Event text [de-DE], Alarm text	Displays the alarm text. The field designation contains a language ID. Alarm texts must be assigned a language ID for import. An expression with a reference ID will be added to the alarm text if the text has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign dynamic parameters to alarm texts.
FeldInfo	Specifies whether the alarm text contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = AlarmTag; Tag = Tag1; DisplayType = Decimal; Length = 5;> Text list: <ref id = 1; type = CommonTextList; TextList = Textlist1; Tag = tag 2; Length = 5;>
Class	The class of an alarm determines whether or not the alarm must be acknowledged. It can also be used to determine how the alarm appears when it is displayed on the HMI device. The alarm class also determines whether and where the corresponding alarm will be logged.
Group	Indicates the allocation to an alarm group. If an alarm belongs to a group with other alarms, it can be acknowledged together with these alarms of the same group in a single operation.
Trigger tag	Specifies the tag monitored for limit value violation.
Delay time value	Specifies the delay time. The alarm is not triggered until the duration of the limit value violation equals the specified delay time.

List entry	Meaning
Delay time unit	Specifies the time unit for the delay.
Report	Enables reporting of the specific alarm on a printer. True or "1" = Reporting enabled. False or "0" = Reporting disabled. Reporting must also be globally enabled in the project.
Info text [de-DE], Info text	The tooltip is an optional property of an alarm. Tooltips can contain additional information about the alarm. A tooltip will be displayed in a separate window on the HMI device when the operator presses the <HELP> key. The field designation contains a language ID.

Example of the worksheet "Limits"

	A	B	C	D	E	F	G
1	Alarm ID	Limit type	Limit value	Limit mode	Deadband mo	Deadband valu	Deadband in percent
2	1	Constant	0	Upper limit	Off	0	False
3	2	Constant	1	Upper limit	Off	0	False
4	3	Constant	2	Upper limit	Off	0	False
5	4	Constant	3	Upper limit	Off	0	False
6	5	Constant	4	Upper limit	Off	0	False
7	25	Constant	50	Upper limit	On both	5	False
8	26	Constant	50	Upper limit	On both	10	True
9	31	Constant	50	Lower limit	Off	0	False
10	32	Constant	50	Upper limit	Off	0	False
11	33	Tag	AASDTdyn	Lower limit	Off	0	False
12	34	Tag	AASDT1dyn	Upper limit	Off	0	False
13	35	Constant	50	Upper limit	Off	0	False
14	36	Constant	50	Lower limit	On both	5	False

Table 12-157 Meaning of the entries

List entry	Meaning
Alarm ID	Alarm number The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Limit mode	Trigger mode Indicates the method used for monitoring the limit value.
Limit type	Specifies the limit that will be monitored. Both a tag and a constant can be used as limit value.
Limit value	Limit value Indicates the tag or constant monitored for limit violation.

List entry	Meaning
Deadband mode	Hysteresis mode Specifies whether and in which cases hysteresis will be used. For "Outgoing" For "Incoming" For "Incoming" and "Outgoing"
Deadband in percent	0 = The value specified for "Hysteresis" is considered to be absolute. 1 = The value specified for "Hysteresis" is referred to as a percentage of the limit value.
Deadband mode	Hysteresis Specifies a constant as a value of the hysteresis.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing alarm of the same name.

See also

Exporting alarms (Page 6798)

Format of the discrete alarm data (Page 6805)

Format of the discrete alarm data

Introduction

This chapter describes the required format of the file for the import of discrete alarms. The file containing the discrete alarm data must be in "*.xlsx" format.

Structure of the alarm data

The import file in Microsoft Excel consists of the worksheets "Discrete alarms" (discrete alarms). Each alarm is assigned a separate row in the import file. Structure of the import file containing the discrete alarms:

Example of the worksheet "Discrete alarms"

	A	B	C	D	E	F
1	ID	Name	Event text [en-US], Alarm text	FieldInfo [Alarm text]	Class	Trigger t
2	1	Discrete_alarm_1	DA1 Error-AC with maximum text length: abcdefghijklmnopqrstuvwxyz äöü\na		Errors	HMI_AC
3	2	Discrete_alarm_2	DA2 Warning-AC this text should be bold		Warnings	HMI_AC
4	3	Discrete_alarm_3	DA3 SDm-AC this text should be italic		SDm	HMI_AC
5	4	Discrete_alarm_4	DA4 SDo-AC <u>this text should be underlined</u>		SDo	HMI_AC
6	5	Discrete_alarm_5	DA5 SystemAcknowledgement-AC <blink>this text should be flashing</blink>		System_Ackn	HMI_AC
7	6	Discrete_alarm_6	DA6 SystemNoAcknowledgement-AC mixed test: Bold, Italic, <l		System_No_A	HMI_AC
8	7	Discrete_alarm_7	DA7 DT in event text: <field ref="0" /> Integer, <field ref="1" /> Real, <field re	<ref id = 0; type = AlarmTag; Tag = Pl	ACAT	HMI_Tri
9	8	Discrete_alarm_8	DA8 DT in event text: <field ref="0" /> S5Time, <field ref="1" /> Timer, <field r	<ref id = 0; type = AlarmTag; Tag = Pl	ACAT	HMI_Tri
10	11	Discrete_alarm_9	DA11 DT in event text: <field ref="0" /> Int, <field ref="1" /> Real, <field ref="2	<ref id = 0; type = AlarmTag; Tag = In	ACAT	HMI_Tri
11	12	Discrete_alarm_10	DA12 DT in event text: <field ref="0" /> UDInt, <field ref="1" /> UInt;	<ref id = 0; type = AlarmTag; Tag = U	ACAT	HMI_Tri
12	13	Discrete_alarm_11	DA13 Textformat: Integer: <field ref="0" /> decimal, <field ref="1" /> binary, <	<ref id = 0; type = AlarmTag; Tag = H	ACAT	HMI_TF

Table 12-158 Meaning of the entries

List entry	Meaning
ID	The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Name	Name of the analog alarm
Event text [de-DE], Alarm text	Displays the alarm text. The field designation contains a language ID. For import, a language ID must be assigned to alarm text. An expression with a reference ID will be added to the alarm text if the text has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign dynamic parameters to alarm texts.
FeldInfo	Specifies whether the alarm text contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = AlarmTag; Tag = Tag1; DisplayType = Decimal; Length = 5;> Text list: <ref id = 1; type = CommonTextList; TextList = Textlist1; Tag = tag 2; Length = 5;>
Class	The class of an alarm determines whether or not the alarm must be acknowledged. It can also be used to determine how the alarm appears when it is displayed on the HMI device. The alarm class also determines whether and where the corresponding alarm will be logged.
Group	Indicates the allocation to an alarm group. If an alarm belongs to a group with other alarms, it can be acknowledged together with these alarms of the same group in a single operation.
Trigger tag	Specifies the tag containing the bit that triggers the alarm.
Trigger bit	Specifies the number of the bit that triggers the alarm.
Acknowledge tag	Specifies the tag containing the bit that is set by the operator upon acknowledgment. Only available if the selected alarm class requires alarm acknowledgment.
Acknowledgment bit	Specifies the number of the bit that is set when the operator acknowledges the alarm.
PLC acknowledgement tag	Specifies the tag containing the bit that acknowledges the alarm of the control program. Only available if the selected alarm class requires alarm acknowledgment.

List entry	Meaning
PLC acknowledgment bit	Specifies the number of the bit that acknowledges the alarm of the control program.
Delay time value	Specifies the delay time. The alarm is not triggered until the duration of the limit value violation equals the specified delay time.
Delay time unit	Specifies the time unit for the delay.
Report	Enables reporting of the specific alarm on a printer. True or "1" = Reporting enabled. False or "0" = Reporting disabled. Reporting must also be globally enabled in the project.
Info text [de-DE], Info text	The tooltip is an optional property of an alarm. Tooltips can contain additional information about the alarm. A tooltip will be displayed in a separate window on the HMI device when the operator presses the <HELP> key. The field designation contains a language ID.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing alarm of the same name.

See also

Exporting alarms (Page 6798)

Format of the analog alarm data (Page 6800)

Importing alarms (Page 6799)

12.12.4.4 Importing and exporting tags

Exporting tags

Introduction

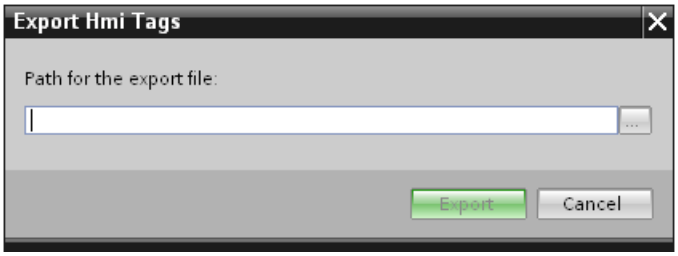
WinCC has an export function for tags.

Requirements

- The WinCC project for the export is open.
- Tags have been created in the project.
- The "HMI tags" editor is open.

Exporting tags

1. Click on the button in the "HMI Tags" tab.
   The "Export" dialog box opens.



2. Click the "..." button and specify in which file the data are saved.
3. Click "Export". The export will start.

Note

The version number of the xlsx file with the exported tags depends on the TIA Portal version. If the xlsx file was exported from a project in WinCC V13 SP1, it has the version number 1.2. If the xlsx file was exported from a project prior to WinCC V13 SP1, it has the version number 1.1.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

See also

Importing tags (WinCC V13 SP1 or higher) (Page 6808)

Format of the tag data (Page 6811)

Importing tags (WinCC V13 SP1 or higher)

Introduction

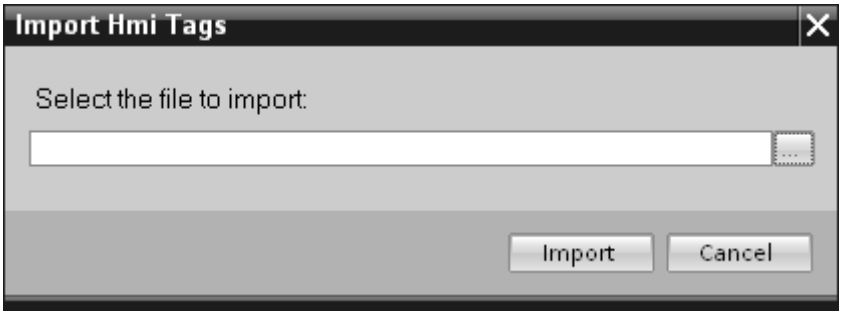
Tags are identified by the tag name. An existing tag is overwritten with the data from the xlsx file if the tag name already exists in the project. A new tag is created if the tag does not yet exist.

Requirements

- An xlsx file with tags has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.

Importing tags

1. Click "HMI tags" in the project navigation.
2. Double-click "Show all tags". The "HMI tags" editor opens.
3. Click the button. The "Import" dialog box opens.



4. Click the "..." button and select the file that you want to import.
5. Click on the "Import" button. The import will start.

Result

The relevant tags have been created in WinCC. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign a tag a trigger tag of the wrong type, for example, string. An error will be reported during compilation.

See also

Exporting tags (Page 6805)

Format of the tag data (Page 6811)

Importing tags (prior to WinCC V13 SP1)

Introduction

Tags are identified by the tag name. An existing tag is overwritten with the data from the xlsx file if the tag name already exists in the project. A new tag is created if the tag does not yet exist.

The version number of the xlsx file depends on the TIA Portal version. If the xlsx file was exported from a project in WinCC V13 SP1, it has the version number 1.2. If the xlsx file was exported from a project prior to WinCC V13 SP1, it has the version number 1.1.

To import files with the version number 1.2 into a project that was created prior to WinCC V13 SP1, you must prepare the files.

Requirements

- An xlsx file with tags has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.

Preparing files for the import

1. Open the corresponding file in Microsoft Excel.
2. Select "File > Properties > Advanced properties".
   The "Properties" dialog is opened.
3. Switch to the "Fit to size" task card.
4. Select "TIA_Version" in the "Properties" area.
5. Change the value in the column from "1.2" to "1.1".
6. Click the "Change" button.
7. Click "OK".
8. Delete the right column "Synchronization" in the file.
9. Save the file.

Importing tags

1. Click "HMI tags" in the project navigation.
2. Double-click "Show all tags". The "HMI tags" editor opens.
3. Click the button. The "Import" dialog box opens.
4. Click the "..." button and select the file that you want to import.
5. Click on the "Import" button. The import will start.

Result

The relevant tags have been created in WinCC. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign a tag a trigger tag of the wrong type, for example, string. An error will be reported during compilation.

Format of the tag data

Introduction

This section describes the format required for the file with tag data used for imports. The tag data file must be in "*.xlsx" format.

Tag data structure

The import file in Microsoft Excel consists of a number of worksheets:

- HMI Tags (HMI tags)
- Multiplexing (multiplex tags)

Each tag is on a separate line in the import file. The import file with the tag data must have the following format:

Example of the worksheet "HMI Tags"

	A	B	C	D	E	F	G	H
1	Name	Path	Connection	PLC tag	DataType	Length	Address	Access Me
2	HMI_Int	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
3	Mux_Tag_1	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
4	Mux_Tag_2	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
5	Mux_11	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
6	Mux_21	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
7	Mux_13	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
8	Mux_12	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
9	Mux_23	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
10	Mux_22	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
11	Mux_Tag_1_Index	Internal Tags	<No Value>	<No Value>	UInt	2	<No Value>	<No Value
12	Mux_Tag_12	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value
13	Mux_Tag_11	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value
14	Mux_Tag_13	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value
15	HMI_UDInt	Internal Tags	<No Value>	<No Value>	UDInt	4	<No Value>	<No Value
16	Gauge_Process	Default tag table	<No Value>	<No Value>	Int	2	<No Value>	<No Value
17	Button_Tag_4	Default tag table	<No Value>	<No Value>	Int	2	<No Value>	<No Value
18	HMI_USInt	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value
19	Data_block_2_PLC_DateTime_2	Default tag table	HMI_connection_1	Data_block_2.PLC_I	Date_And_Time	8	%DB28.DBX598.(	<absolute

Table 12-159 Meaning of the entries

List entry	Meaning
Name	Indicates the configured name of an HMI tag.
Path	Specifies which folders in the project tree contain the tag. The folder structure is represented by "\" : "FolderName1\FolderName2\TagName".
PLC Tag	Indicates whether the tag is linked to a PLC tag.
Connection	Indicates the name of the connection to the PLC.
Data type	Specifies the data type of a tag. The data types allowed depend on the communication driver being used. See the "Communication" section of the documentation for additional information on the data types permitted for the various communication drivers.
Length	Specifies the length of the tag. This entry is only useful for data types with a dynamic length such as strings; it is left empty for all other data types.
Address	Specifies the tag address in the PLC. The tag address must exactly match the one used in WinCC, for example, "%DB1.DBW0". The tag address is empty for internal tags.
Multiplexing	Specifies whether multiplexing is used.
Index tag	Shows the name of the index tag for multiplexing. In Runtime, the system first reads the value of the index tag. It then accesses the tag in the corresponding place in the tag list.
StartValue	Specifies the start value of a tag.
ID tag	The update ID updates the value of a tag with the aid of a function or a PLC job. The update ID must be unique within an HMI device.
Coding	Shows the coding method.

List entry	Meaning
DiplayName [de_DE]	Shows the display name of an HMI tag. The field designation contains a language ID. The field designation contains a language ID. Display names must be assigned a language ID for import. Texts are imported to the corresponding project language.
Acquisition mode	Specifies the tag acquisition mode.
Acquisition cycle	Specifies the tag acquisition cycle. The acquisition cycle must correspond exactly to the one used in WinCC. The value is not language-dependent and should therefore be the same in every language. The default value is "1 s". The acquisition cycle is undefined if the tag acquisition mode is "on demand". User-defined acquisition cycles must be created beforehand as the file will otherwise not be imported.
High High Limit type	Indicates whether the limit value "High high" is monitored by a constant, a tag or not at all.
High High Limit	Displays the limit value "High High".
High Limit type	Indicates whether the limit value "High" is monitored by a constant, a tag or not at all.
High Limit	Displays the limit value "High".
Low Limit type	Indicates whether the limit value "Low" is monitored by a constant, a tag or not at all.
Low Limit	Displays the limit value "Low".
Low Low Limit type	Indicates whether the limit value "Low Low" is monitored by a constant, a tag or not at all.
Low Low Limit	Displays the limit value "Low Low".
Linear scaling	Indicates whether linear scaling is enabled. This entry can only be used for external tags.
End value PLC	Specifies the end value of the PLC tag.
Start value PLC	Specifies the start value of the PLC tag.
End value HMI	Specifies the end value of the HMI tag.
Start value HMI	Specifies the start value of the HMI tag.

Example of the worksheet "Multiplexing"

	A	B	C
1	HMI Tag name	Multiplex Tag	Index
2	Mux_Tag_1	Mux_11	0
3	Mux_Tag_1	Mux_12	1
4	Mux_Tag_1	Mux_13	2
5	Mux_Tag_2	Mux_21	0
6	Mux_Tag_2	Mux_22	1
7	Mux_Tag_2	Mux_23	2
8	Mux_Tag_12	HMI_Array_Mux2	-1
9	Mux_Tag_11	HMI_Array_Mux1	-1
10	Mux_Tag_13	HMI_Array_Mux3	-1

Table 12-160 Meaning of the entries

List entry	Meaning
Name	Indicates the configured name of an HMI tag which uses indirect addressing. The HMI tag must be available in the "HMI Tags" worksheet.
Index	Shows the value which governs which tag is selected.
Multiplex Tag	Displays the tag from the tag list corresponding to the index value.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing tag of the same name.

See also

Exporting tags (Page 6805)

Importing tags (WinCC V13 SP1 or higher) (Page 6806)

12.12.4.5 Importing and exporting text lists

Exporting text lists

Introduction

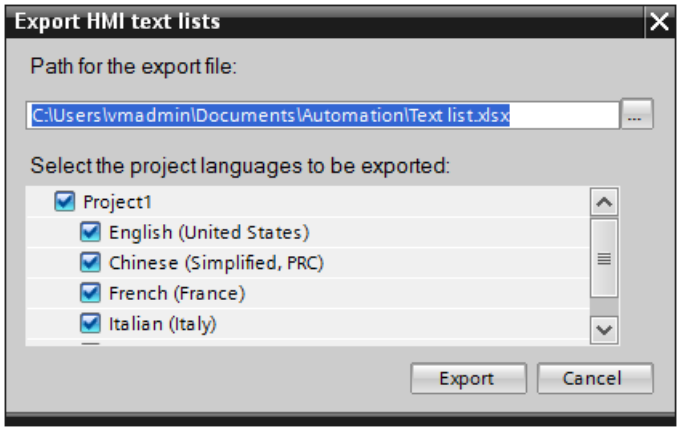
WinCC has an export function for text lists.

Requirements

- The WinCC project for the export is open.
- Text lists have been created in the project.
- The "Text & graphics lists" editor is open.

Exporting text lists

1. Click on the button in the "TextLists" tab.
   The "Export" dialog box opens.



2. Click the "..." button and specify in which file the data are saved.
3. By default, texts are exported in all defined project languages.
   If you do not want to export specific languages, disable the unnecessary languages in the dialog.
4. Click "Export". The export will start.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

Importing text lists

Introduction

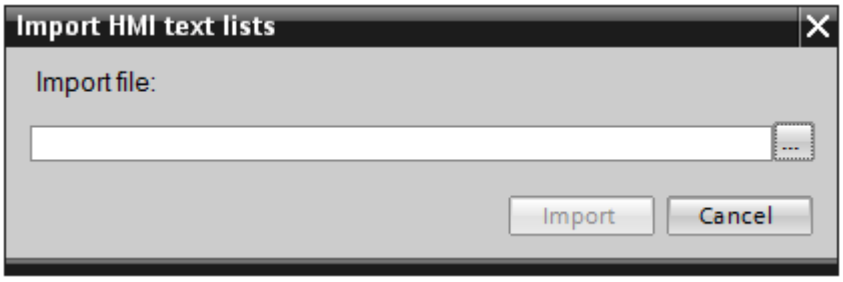
You then import text lists from an xlsx file to WinCC.

Requirements

- An xlsx file with text lists has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.
- The "Text & graphics lists" editor is open.

Importing text lists

1. Click on the button in the "Text lists" tab.
   The "Import" dialog box opens.



2. Select the file you want to import under "File selection".
3. Click on the "Import" button. The import will start.

Result

You have now imported the text lists. The relevant text lists have been created in WinCC. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Format of text list data

Introduction

This section describes the format required for the file with the text lists used for imports. The text list data file must be in "*.xlsx" format.

Tag data structure

The import file in Microsoft Excel consists of two worksheets:

- TextList (Text lists)
- TextListEntry (Text list entry)

Each text list is assigned a separate line in the import file. The import file containing the data must be structured as follows:

Example of the worksheet "TextList"

	A	B	C
1	Name	ListRange	Comment
2	TLValue/Range	Decimal	
3	TLBit	Bit	
4	TLBitnumber	Binary	
5	TLall_1	Decimal	
6	TLall_2	Decimal	

Table 12-161 Meaning of the entries

List entry	Meaning
Name	Shows the name of the text list.
ListRange	Shows the text list range: Number = Bit number (0-31) Range = value/range (...-...) Bit = Bit (0;1)
Comment	Any comments on the text list. You can use up to 500 characters

Example of the worksheet "TextListEntry"

	A	B	C	D	E	F
1	Name	Parent	DefaultEntry	Value	Text	FieldInfos
2	Text_list_entry_1	TLValue/Range	TRUE	0 - 1	Default entry TLValue/Range ->	
3	Text_list_entry_2	TLValue/Range		2 - 3	TLValue/Range = 2-3 ->	
4	Text_list_entry_3	TLValue/Range		1	TLValue Range - single value = 1 ->	
5	Text_list_entry_1	TLBit		0	TLBit = 0 ->	
6	Text_list_entry_2	TLBit		1	TLBit = 1 ->	
7	Text_list_entry_1	TLBitnumber	TRUE	0	Default entry TLBitnumber; ->	
8	Text_list_entry_2	TLBitnumber		0	TLBitnumber - Bitnumber 0 is set ->	
9	Text_list_entry_3	TLBitnumber		1	TLBitnumber - Bitnumber 1 is set ->	
10	Text_list_entry_4	TLBitnumber		2	TLBitnumber - Bitnumber 2 is set ->	
11	Text_list_entry_5	TLBitnumber		3	TLBitnumber - Bitnumber 3 is set ->	
12	Text_list_entry_1	TL1	TRUE	0 - 1	Default entry TL1	
13	Text_list_entry_2	TL1		1 - 3	TL1 Value between 1 - 3 ->	
14	Text_list_entry_3	TL1		4 - 6	TL1 Value between 4 - 6 ->	
15	Text_list_entry_4	TL1		7	TL1 Single value = 7 ->	
16	Text_list_entry_1	TL2	TRUE	0 - 1	<field ref="0" /> ->	<ref id = 0; type = CommonTextList; TextList = TL1; Tag = HMI_TL1control; Leng
17	Text_list_entry_2	TL2		1	TL2 Single value = 1 ->	
18	Text_list_entry_3	TL2		2 - 3	TL2 Range between 2 - 3 ->	
19	Text_list_entry_1	TLMultilined	TRUE	0 - 1	Default entry TLMultilined; last row	
20	Text_list_entry_2	TLMultilined		0 - 3	TLMultilined Value between 0-3\nwith test of"\n"	
21	Text_list_entry_1	TLall_1	TRUE	0 - 1	Default entry TLall_1	
22	Text_list_entry_1	TLall_2	TRUE	0 - 1	Default entry TLall_2	

Table 12-162 Meaning of the entries

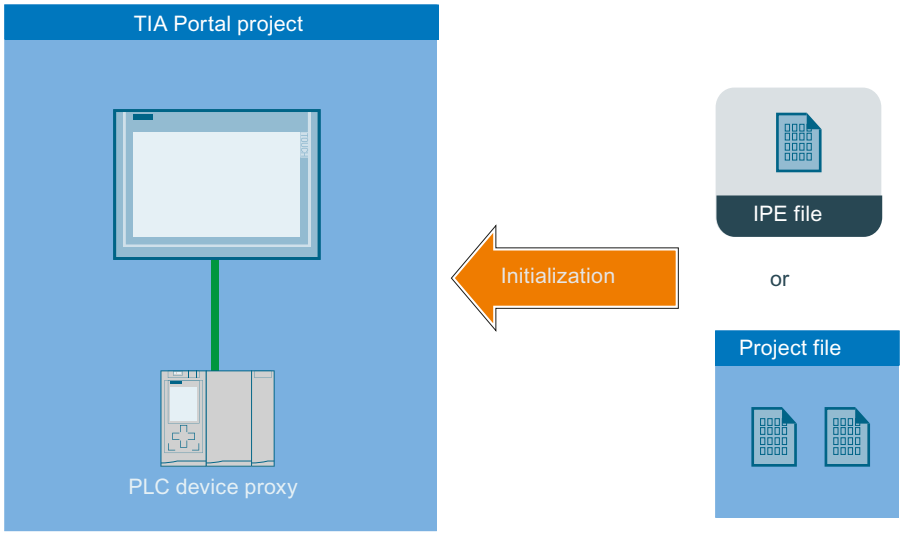
List entry	Meaning
Name	Shows the name of the text list entry.
Parent	Specifies the name of the corresponding text list.
DefaultEntry	Indicates whether the text list entry is a default entry. The default entry is always displayed when the tag has an undefined value.
Value	Specifies the tag integer values or value ranges which are assigned to the text entries in the text list.
Text	Shows the text list entry. The field designation contains a language ID. Text list entries must be assigned a language ID for import. An expression with a reference ID will be added to the text if the text list entry has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign the dynamic parameter to a text list entry.
FeldInfo	Specifies whether the text list contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = CommonTagDisplayFormat; Tag = tag 1; DisplayType = Decimal; DisplayFormat = 9;> Text list: <ref id = 1; type = CommonTextList; TextList = Textliste_1; Tag = tag 2; Length = 5;> PLC tag: <ref id = 0; type = CommonControlTagDisplayFormat; DisplayType = Decimal; DisplayFormat = 9;>

12.12.5 Exchanging controller data from other projects

Inter-project engineering

You can use the IPE (Inter Project Engineering) function to read in controller data from other TIA Portal projects and use it for configuring.

IPE enables you to transfer and merge the HMI configuration and PLC programming communication into various TIA Portal projects.



You can find more information and notes on configuration under: Basics of Inter Project Engineering (IPE)

12.12.6 Using cross-references

12.12.6.1 General information about cross references

Introduction

The cross-reference list provides an overview of the use of objects within the project.

Uses of cross-references

The cross-reference list offers you the following advantages:

- When creating and changing a program, you retain an overview of the objects, tags, and alarms etc. you have used.
- From the cross-references, you can jump directly to the object location of use.
- You can learn the following when debugging:
  - The objects used in a specific screen.
  - The alarms and recipes shown in a specific display.
  - The tags used in a specific alarm or object.
- As part of the project documentation, the cross-references provide a comprehensive overview of all object, alarms, recipes, tags and screens.

12.12.6.2 Displaying the cross-reference list

Introduction

Details on the use of objects can be found in the cross-reference list. You can show cross-references for HMI devices, folders and all editors in the project tree. The detail view also lets you select individual objects of the editors.

Requirement

You have created a project.

Several objects have been created.

Procedure

1. Select the required entry in the project tree or detail view.
2. Select "Cross-references" in the shortcut menu. The cross-reference list is opened in the work area.
3. Open the "Used by" tab to display where the objects shown in the cross-reference list are used.
4. Open the "Used" tab to view the users of the objects displayed in the cross-reference list.
5. You can sort the entries in the "Object" column in ascending or descending order by clicking on the corresponding column header.
6. To go to the location of use for a specific object, click on the displayed link.

Result

The cross-reference list for the selected object is displayed in the work area.

12.12.6.3 Structure of the cross-reference list

Views of the cross-reference list

There are two views of the cross-reference list. The difference between the two views is in the objects displayed in the first column:

- Used by:
  Display of the referenced objects. Here, you can see where the object is used.
- Used:
  Display of the referencing objects. The users of the object are shown here.

The assigned tool tips provide additional information about each object.

Structure of the cross-reference list

Column	Content/meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Numbers	Number of uses
Location of use	Each location of use, for example, an object or event
Property	Function of the referenced objects, for example, tag for data record or process value
Connected to	PLC tag with which the object is connected.
Type	Type of object
Path	Path of object

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

Settings in the cross-reference list

You can make the following settings using the icons in the toolbar for the cross-reference list:

- Update cross-reference list
  Updates the current cross-reference list.
- Making settings for the cross-reference list
  Here, you specify whether all used, all unused, all defined or all undefined objects will be displayed. If the "Undefined objects" option is enabled, references to previously deleted objects are also displayed.
- Collapse entries
  Reduces the entries in the current cross-reference list by closing the lower-level objects.
- Expand entries
  Expands the entries in the current cross-reference list by opening the low-level objects.

Sorting in the cross-reference list

You can sort the entries in the "Object" column in ascending or descending order. Click on the column header to do this.

12.12.6.4 Displaying cross-references in the Inspector window

Introduction

The Inspector window displays cross-reference information about a selected object in the "Info > Cross-references" tab. The Inspector window displays the cross-reference information in tabular format.

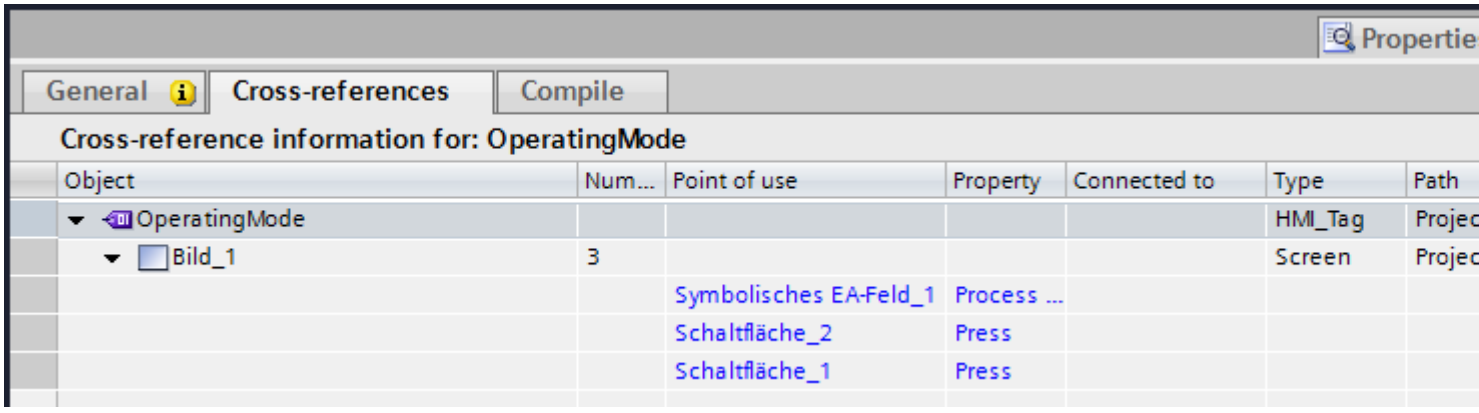
All included elements and their use in the cross-reference list are displayed for structured tags, user data types and instances of a PLC data type.

Requirement

- You have created a project.
- Several objects have been created.

Procedure

1. Select an object in a screen or a tabular editor.
2. Select "Cross-reference information" in the shortcut menu. The cross-references are opened in the Inspector window.



Result

The instances where and the other objects by which the selected object is being used are displayed.

The table below shows the additional information listed in the "About > Cross-reference" tab:

Column	Meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Numbers	Number of uses
Location of use	Each location of use, for example an object or event
Property	Function of the referenced objects, for example tag for data record or process value
Connected to	PLC tag with which the object is connected.
Type	Type of object
Path	Path of object

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

12.12.6.5 Rewiring tags in the screens

Introduction

You can change the references of tags in the properties of screen objects using the "Change object references" dialog.

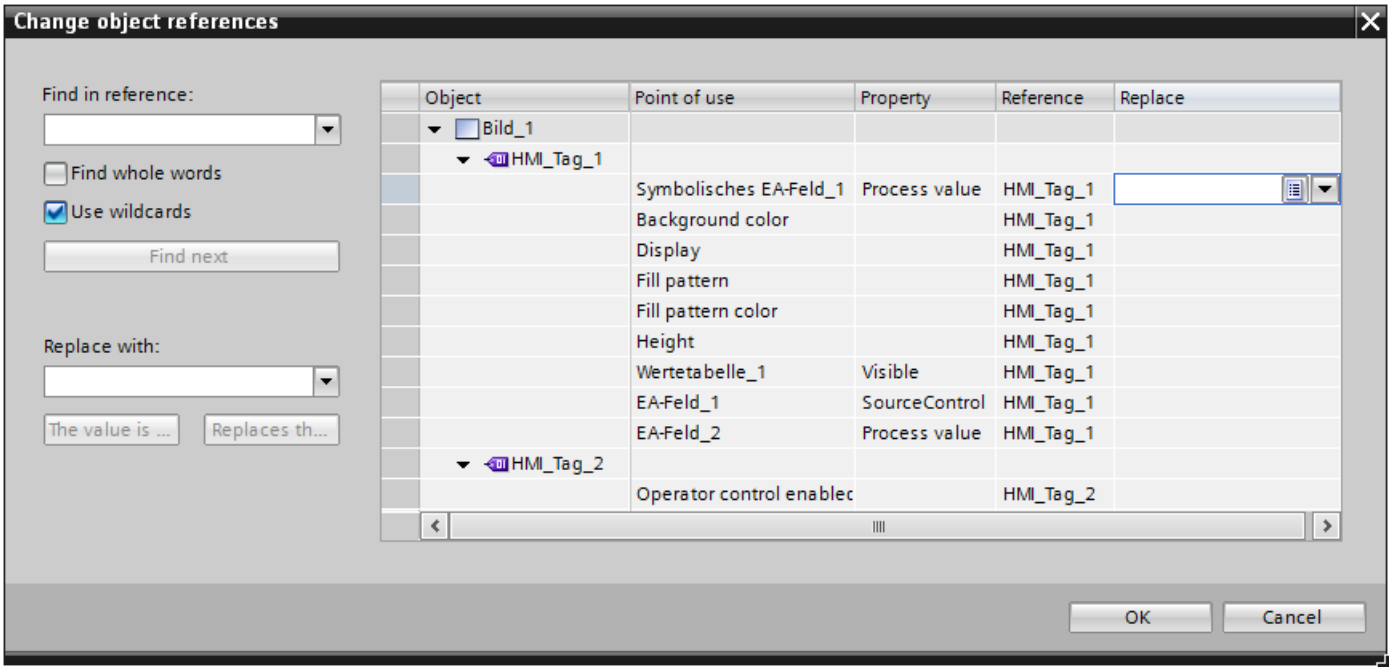
This function lets you to replace a large number of tags in various screen objects.

You search for and replace individual or multiple characters in the tag names.

You can open the "Change object references" dialog with the command in the shortcut menu. You can open the dialog for one or more screen objects or a screen.

How to replace a tag

1. Select the objects in the screen that contains a tag that you want to change.
2. Select the "Change object references" command in the shortcut menu or in the "Edit" menu. The "Change object references" dialog opens. You can see the selected screen objects and the tags used in the "Object" column.



3. Enter the name or part of the name you want to find in the "Find in reference" input box.
4. Start the search with the "Find next" button. If a reference is found, it is highlighted in the "Reference" column.
   Another click on the "Find next" button continues the search process until the list has been completely searched.
   After this, the search process starts again from the top of the list.
5. Enter the name or part of the name you want to replace for the found name in the "Replace with" input box.

6. Click the "Replace" button to replace the selected tag.
   All references found are replaced by clicking the "Replace all" button. In this way, you can replace a large number of references with another
   reference.
7. To select a tag from the list of all tags created in the project, click on the button in the "Replace" column.
8. To select a tag from the object list or to create a new tag, click on the button in the "Replace" column.
9. Confirm your entry with "OK".

The rewiring of the tags is applied in the project.

Additional options for searching

You can narrow down your search by enabling the following options:

- Use wildcards
  Enter * for any number of characters. Example: You want to search for all tags starting with "HMI". Enter "HMI*" in the search key box.
  Enter ? for an individual character.
- Find whole words
  To search only for full names, enable the "Find whole words" option.

Replacing arrays and user-defined tags

Variables whose names contain special characters . [ ] are shown in quotes.

Array type tags and tags of user-defined data types (UDT) contain special characters to separate tags from their elements. However, they are not shown in quotation marks. To replace an array or a UDT with an HMI tag with special characters, enter the name of the HMI tag in quotation marks in the "Replace" input box. If you type the name without quotation marks, a reference to a non-existent array or UDT is created instead of a reference to the desired HMI tag with special characters.

Correcting entries

If an incorrect entry is made in the "Replace with" input box, the existing references can be replaced by references to non-existent tags.

If you detect an incorrect entry before confirming with "OK", you can correct the error. To do this, select the "Replace" column and press the "Remove" button. Any incorrect entries in the "Reference" column are deleted.

If you confirm the dialog with "OK", the affected properties of the screen objects are marked in red. Non-existent tags are highlighted in red in the "Change cross references" dialog.

If the new tag does not yet exist, you must subsequently create the tag.

12.12.7 Managing languages

12.12.7.1 Languages in WinCC

User interface language and project languages

A distinction is drawn between two different language levels in WinCC:

- User interface language
  During configuration, the text in the WinCC menus and dialogs is displayed in the user interface language. The user interface language also affects the labeling of operating elements, the parameters of the system functions, the online help, etc.
- Project languages
  Project languages are all languages in which a project will later be used. Project languages are used to create a project in multiple languages.

The two language levels are completely independent of one another. For example, you can create English projects at any time using a German user interface and vice versa.

Project languages

The following languages are differentiated within the project languages:

- Reference language
  The reference language is the language that you use to configure the project initially. During configuration, you select one of the project languages as the reference language. You use the reference language as a template for translations. All of the texts for the project are first created in the reference language and then translated. While you are translating the texts, you can have them displayed simultaneously in the reference language.
- Editing language
  You produce translations of the texts in the editing language.
  Once you have created your project in the reference language, you can translate the texts into the remaining project languages. Select a project language respectively as an edit language and edit the texts for the appropriate language variant. You can change the editing language at any time.

  Note

  When switching the project languages, the assignment to the keys on the keyboard also changes. For some languages (for example, Spanish), the operating system does not allow you to switch to the corresponding keyboard assignment. In this case, the keyboard assignment is switched to English.

- Runtime languages
  Runtime languages are those project languages that are transferred to the HMI device. You decide which project languages to transfer to the HMI device depending on your project requirements.
  You must provide appropriate controls so that the operator can switch between languages in runtime.

See also

Language settings in the operating system (Page 6826)

Operating system settings for Asian languages (Page 6826)

12.12.7.2 Language settings in the operating system

Introduction

The configuration PC operating system settings influence WinCC language management in the following areas:

- Selection of project languages
- Regional format of dates, times, currency, and numbers
- Displaying ASCII characters

Project language selection

A language is not available as a project language unless it is installed in the operating system.

Regional format of dates, times, currency, and numbers

WinCC specifies a fixed date and time format in the Date - Time field for the selected project language and runtime language.

In order for dates, times, and numbers to be presented correctly in the selected editing language, this language must be set in the Regional Options in the Control Panel.

Displaying ASCII characters

With text output fields, the display of ASCII characters as of 128 depends on the set language and the operating system being used.

If the same special characters are to be displayed on different PCs, the PCs must use the same operating system and regional settings.

See also

Languages in WinCC (Page 6823)

Operating system settings for Asian languages (Page 6826)

12.12.7.3 Operating system settings for Asian languages

Settings on Western operating systems

If you want to enter Asian characters, you must activate the support for this language in the operating system.

The Input Method Editor (IME) is available in Windows for configuring Asian texts. Without this editor, you can display Asian text but not edit it. For more information on the Input Method Editor, refer to the documentation for Windows. To enter Asian characters when configuring, switch to the Asian entry method in the "Input Method Editor".

Switch the operating system to the appropriate language to have language-specific project texts, such as alarm texts, displayed in the simulator in Asian characters.

Settings on Asian operating systems

If you are configuring on an Asian operating system, you must switch to the English default input language to enter ASCII characters, for example, for object names. As the English default input language is included in the basic installation of the operating system, you do not need to install an additional input locale.

Enable language support

1. Open the system controller.
2. Select "Regional and Language Options".
3. On the "Languages" tab, activate the check box "Install files for East Asian languages".
4. Then click on "Details" under "Text Services and Input Languages". The dialog "Text Services and Input Languages" is opened.
5. On the "Settings" tab add the required default input language under the "Installed Services".
6. Select the language of the operating system in the "Language for non-Unicode programs" area in the "Advanced" tab.

See also

Languages in WinCC (Page 6823)

Language settings in the operating system (Page 6824)

12.12.7.4 Setting project languages

Selecting the user interface language

Introduction

The user interface language is used for displaying menu entries, title bars, infotexts, dialog texts and other designations in the WinCC user interface.

You can switch between the installed user interface languages during configuration. The labeling of the operating elements remains in the language you set when you added the object even if you change the user interface language.

Procedure

1. Select "Options > Settings" in the menu.
   The "Settings" dialog box is opened.
2. Select the desired user interface language under "General > General settings".

Result

WinCC will use the selected language as user interface language.

See also

Enable project languages (Page 6828)

Selecting the reference language and editing language (Page 6829)

Languages in WinCC (Page 6823)

Enable project languages

Introduction

The project languages are set in the "Project languages" editor. You define which project language is to be the reference language and which the editing language.

Enable project languages

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The lower-level elements will be displayed.
2. Double-click on "Project languages".
   The possible project languages will be displayed in the working area.
3. Enable the relevant project languages.

Note

Copying multilingual objects

The copies of multilingual objects to a different project only include text objects in the project languages which are activated in the target project. Activate all project languages in the target project to include the corresponding text objects when transferring the copy.

Disabling project languages

1. Disable the languages which are not relevant for the project.

Notice

If you disable a project language, all text and graphic objects you have already created in this language will be deleted from the current project.

See also

Selecting the user interface language (Page 6825)

Selecting the reference language and editing language (Page 6829)

Selecting the reference language and editing language

Introduction

The project languages are set in the "Project languages" editor. You define which project language is to be the reference language and which the editing language. You can change the editing language at any time.

Requirements

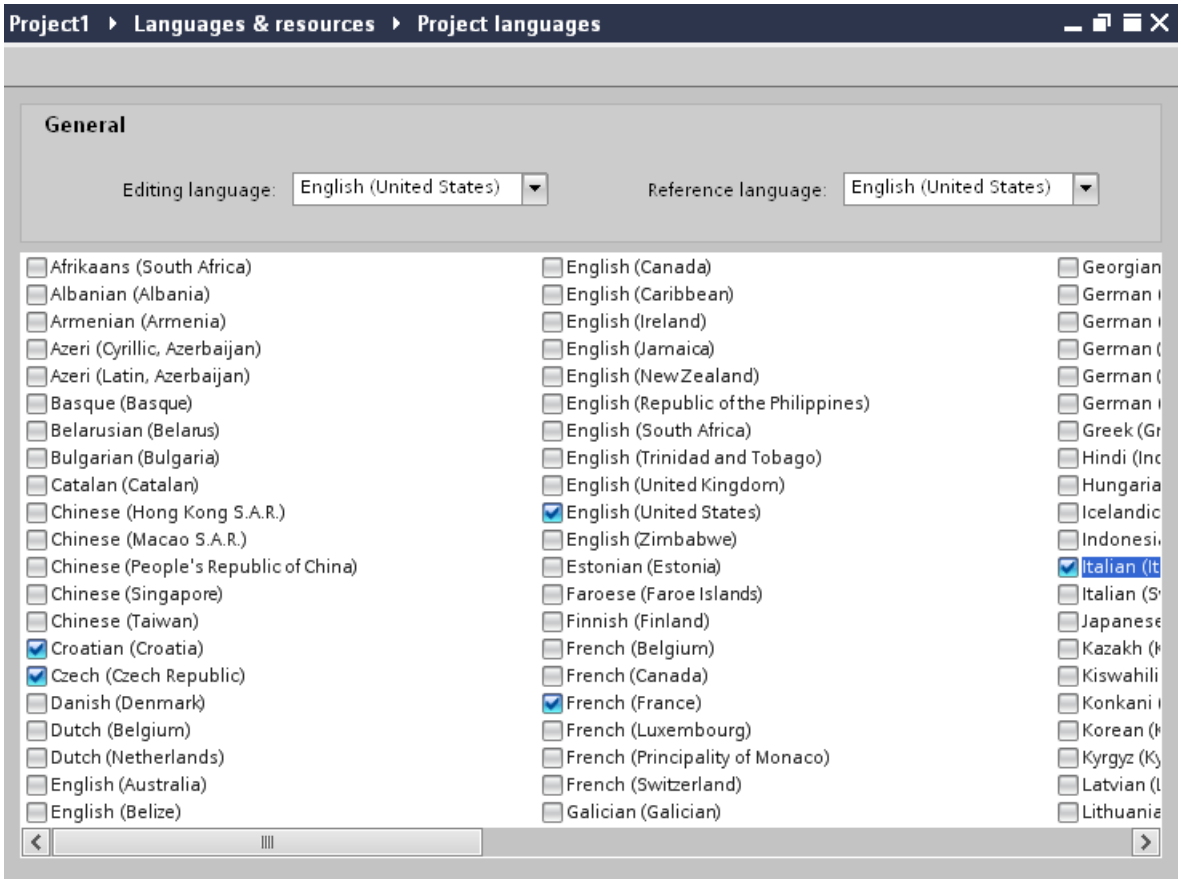
The "Project languages" editor is open.

Several project languages have been activated.

Selecting the reference language and editing language

1. Click the arrow in the drop-down list in the "General > Editing language" section.
2. Click on the required language in the drop-down list, for example, German.
3. Click on the arrow in the drop-down list in the "General > Reference language" section.
4. Click on the required language in the drop-down list, for example, English.

The language selection is displayed in the list box.



Result

You have now selected the editing and reference languages.

If you change the editing language, all future text input will be stored in the new editing language.

See also

Selecting the user interface language (Page 6825)

Enable project languages (Page 6826)

12.12.7.5 Creating one project in multiple languages

Working with multiple languages

Multilingual configuration in WinCC

You can configure your projects in multiple languages using WinCC. There are various grounds for creating a project in multiple languages:

- You would like to use a project in more than one country.
  You create the project in multiple languages but when the HMI device is commissioned, only the language spoken by the operators at the respective site will be transferred to the HMI device.
- The operators of a system speak a range of different languages.
  Example: An HMI device is used in China, but the service personnel understand only English.

Translating project texts

With WinCC, you can enter project texts directly in several languages in various different editors, for example, in the "Project texts" editor. WinCC also allows you to export and import your configuration for translation purposes. This is particularly advantageous if you configure projects containing a large amount of text and want to have it translated.

Language management and translation in WinCC

The following editors are used to manage languages and translate texts in WinCC:

Editor	Short description
Project languages	Selection of project languages, editing language and reference language.
Languages and fonts	Management of runtime languages and fonts used on the HMI device.
Project texts	Central management of configured texts in all project languages.
Graphics	Graphic library for managing graphics and their language-specific versions.

See also

Project text basics (Page 6832)

Translating texts directly (Page 6833)

Translating texts using reference texts (Page 6834)

Exporting project texts (Page 6835)

Importing project texts (Page 6837)

Project text basics

Texts in different languages in the project

Texts that are output on display devices during processing are typically entered in the language in which the automation solution is programmed. Comments and the names of objects are also entered in this language.

If operators do not understand this language, they require a translation of all operator-relevant texts into a language they understand. You can therefore translate all the texts into any language. In this way, you can ensure that anyone who is subsequently confronted with the texts in the project sees the texts in his/her language of choice.

User texts and system texts

In the interests of clarity, a distinction is drawn between user texts and system texts:

- User texts are texts created by the user.
- System texts are texts created automatically and which are a product of configuration in the project.

The project texts are managed in the project text editor. This can be found in the project tree under "Languages & Resources > Project texts".

Examples of multilingual project texts

You can, for example, manage the following types of text in more than one language:

- Display texts
- Alarm texts
- Comments in tables
- Labels of screen objects
- Text lists

Translating texts

There are two ways of translating texts.

- Translating texts directly
  You can enter the translations for the individual project languages directly in the "Project texts" editor.
- Translating texts using reference texts
  You can change the editing language for shorter texts. You can enter the new texts in the editing language while the texts of the reference language are displayed.

See also

Working with multiple languages (Page 6829)

Translating texts directly (Page 6833)

Translating texts using reference texts (Page 6834)

Exporting project texts (Page 6835)

Importing project texts (Page 6837)

Translating texts directly

Translating texts

If you use several languages in your project, you can translate individual texts directly. As soon as you change the language of the software user interface, the translated texts are available in the selected language.

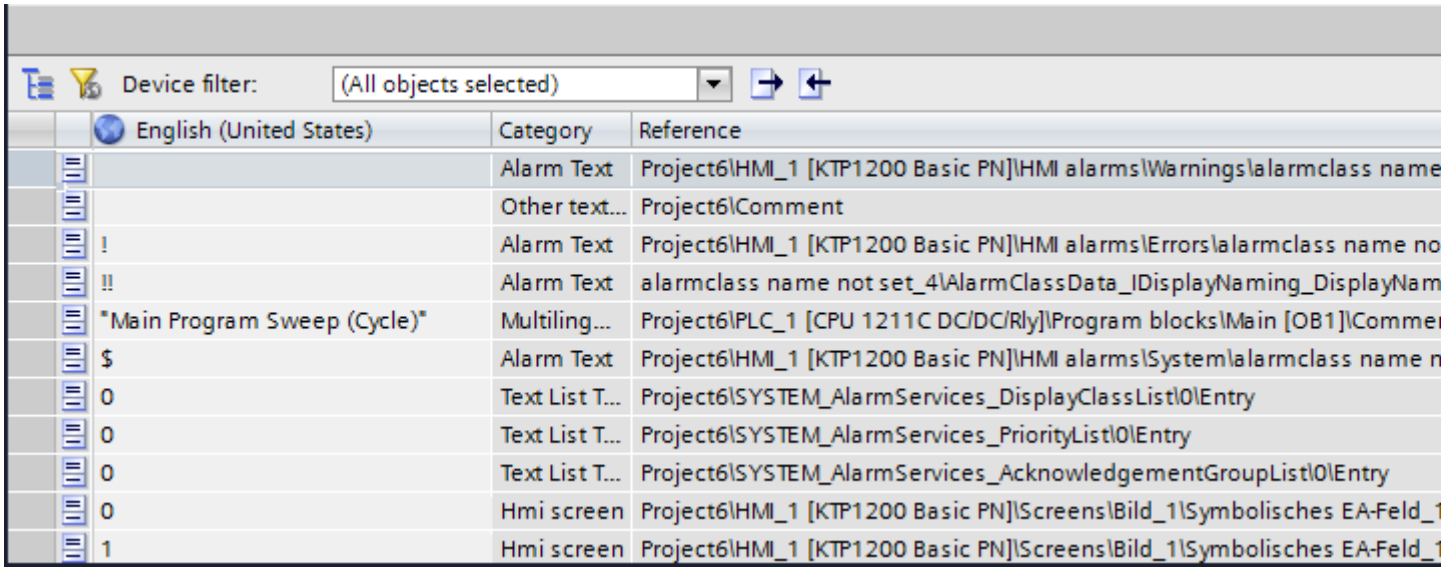
Requirements

- You are in the project view.
- A project is open.
- You have selected at least two further project languages.

Procedure

Proceed as follows to translate individual texts:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The elements below this are displayed.
2. Double-click on "Project texts".
   A list with the texts in the project is displayed in the work area. There is a separate column for each project language.



3. To group identical texts and translate them simultaneously, click on the "" button in the toolbar.

4. To hide texts that do not have a translation, click on the button in the toolbar.
5. Click on an empty column and enter the translation.

Result

You have translated individual texts in the "Project texts" editor. The texts will then be displayed in the runtime language.

See also

Working with multiple languages (Page 6829)

Exporting project texts (Page 6835)

Project text basics (Page 6830)

Importing project texts (Page 6837)

Translating texts using reference texts

Introduction

After changing the editing language, all texts are shown in input boxes in the new editing language. If there is not yet a translation available for this language, the input boxes are empty or filled with default values.

If you enter text again in an input field, this is saved in the current editing language. Following this, the texts exist in two project languages for this input field, in the previous editing language and in the current editing language. This makes it possible to create texts in several project languages.

You can display existing translations for an input box in other project languages. These serve as a comparison for text input in the current editing language and they are known as the reference language.

Requirement

There is at least one translation into a different project language for an input field.

Procedure

To display the translation of an input cell in a reference language, follow these steps:

1. Select "Tasks > Languages & resources" in the task card.
2. Select a reference language from the "Reference language" drop-down list.

Result

The reference language is preset. If you click in a text block, translations that already exist in other project languages are shown in the "Tasks > Reference text" task card.

See also

Working with multiple languages (Page 6829)

Project text basics (Page 6830)

Exporting project texts (Page 6835)

Importing project texts (Page 6837)

Exporting project texts

Project texts are exported for translation. Texts are exported to Office Open XML files ending in ".xlsx". These files can be edited in Microsoft Excel, for example.

You can exchange the file with the translators and import it back to the project as soon as it has been translated.

Requirements

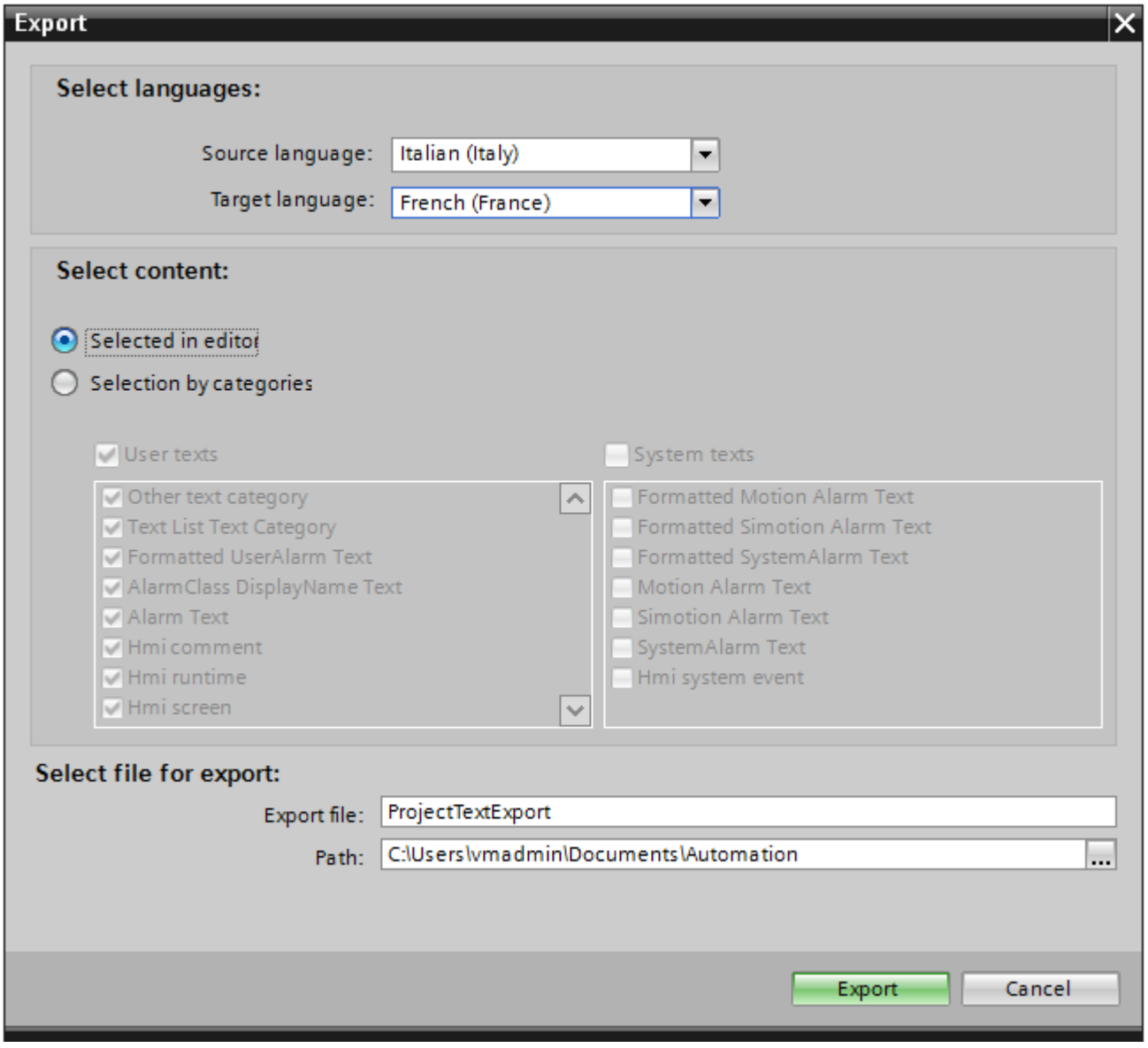
- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Exporting project texts

To export individual project texts, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The child elements are displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Select the texts you want to export.

4. Click on the button. The "Export" dialog opens.



5. From the "Source language" drop-down list, select the language from which you wish to translate, for example Italian.
6. From the "Target language" drop-down list, select the language into which the texts are to be translated, for example, French.
7. Enter a file path and a file name for the export file in the "Export file" input field.
8. Click "Export".

Result

The texts selected in the "Project texts" editor are written to an xlsx file. The xlsx file will be stored in the specified folder.

You can alternatively select and export all project texts from categories. Select "User texts" or "System texts" in the "Export" dialog in line with the type of texts you wish to export. In this case, export can additionally be limited by categories.

Note

Project texts in library objects cannot be exported.

See also

Working with multiple languages (Page 6829)

Project text basics (Page 6830)

Translating texts directly (Page 6831)

Translating texts using reference texts (Page 6832)

Importing project texts (Page 6837)

Importing project texts

Edit the xlsx file or send it to a translator. Import the texts once they have been translated. The foreign languages will be imported to the relevant object in the project.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Importing project texts

To import a project text file, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The lower-level elements will be displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Click on the button. The "Import" dialog opens.
4. Select the path and file name of the import file from the "Import file" field.
5. Activate the "Import source language" check box if you have made changes to the source language in the export file and would like to overwrite the entries in the project with the changes.
6. Click on "Import".

Result

You have imported the project texts.

See also

Working with multiple languages (Page 6829)

Project text basics (Page 6830)

Translating texts directly (Page 6831)

Translating texts using reference texts (Page 6832)

Exporting project texts (Page 6833)

12.12.7.6 Using language-specific graphics

"Graphics" editor

Introduction

You use the "Graphics" editor to manage the configured graphic objects in different language versions. Multilingual projects sometimes also require language-specific versions of the graphics, for example, if

- the graphics contain text;
- cultural aspects play a role in the graphics.

Opening the "Graphics" editor

Double-click on "Languages and resources" in the project tree.

Work area

The work area displays all configured graphic objects in a table. There is a separate column in the table for each project language. Each column in the table contains the versions of the graphics for one particular language.

In addition, you can specify a default graphic for each graphic to be displayed whenever a language-specific graphic for a project language does not exist.

Preview

The preview shows you how the graphics will look on various devices.

See also

Storing an external image in the graphics library (Page 6840)

Storing an image in the graphics library (Page 6839)

Languages in WinCC (Page 6823)

Storing an image in the graphics library

Introduction

You use the "Graphics" editor to import graphics you want to use in screens in the "Screens" editor. It also allows you to manage language-specific versions of graphics. A preview shows the graphic displays on various HMI devices.

Requirement

- The language-dependent versions of a graphic are available.
- Multiple languages have been enabled in the "Project languages" editor.
- The "Graphics" editor is open.

Inserting graphics

1. Click "Add" in the "Graphics library" table. A dialog box opens.
2. Select the required graphic file.
3. Click "Open" in the dialog box.
   The graphic will be imported to the project and displayed in all cells in this row in the "Graphics" editor.
4. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
5. Select "Add graphic" from the shortcut menu. A dialog box opens.
6. Select the desired graphic file and click "Open."
   The language-dependent version is inserted in the table in place of the reference language graphic.
7. Then, in the "Default graphic" column, import a graphic to be displayed in runtime for those languages for which there is no language-specific graphic.

You can also drag&drop a graphic from Windows Explorer to the relevant position in the "Graphics library" table.

Displaying graphics in the HMI device preview

1. Click on a graphic in the table.
2. Select the required HMI device under "Properties > Graphics settings > Device preview" in the Inspector window.
   The graphic will then be displayed as it will appear in runtime on the selected HMI device.

Result

The graphics added are available in the "Graphics" editor. The graphic assigned to the respective editing language will be displayed during editing. The default screen will be displayed in all editing languages for which no screen has been imported.

The screens assigned to the respective runtime language are displayed during runtime. The default screen is displayed in all runtime languages for which a screen has not been imported.

Note

If you disable a project language, all of the graphic objects you have already created in this language will be deleted from the current project.

See also

"Graphics" editor (Page 6836)

Storing an external image in the graphics library

Introduction

To display graphics that have been created in an external graphics program in your screens, you will first have to store these graphics in the graphics browser of the WinCC project.

Requirement

- Multiple languages have been enabled in the "Project languages" editor.
- The "Graphics" editor is open.
- There is a graphic in the "Graphics" editor.

Creating and adding a new graphic as an OLE object

1. Click "Add" in the "Graphics library" table. A dialog box opens.
2. Navigate to the folder in which the graphic is stored.
3. Click "Open" in the dialog box.
   The graphic will be imported to the project and displayed in all cells in this row in the "Graphics" editor.
4. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
5. Select "Insert object" from the shortcut menu. The "Insert object" dialog box opens.

   Note

   In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

6. Select "Insert object > Create new" and an object type in the dialog.

7. Click "OK." The associated graphic program is opened.
8. Close the graphics program once you have created the graphic.
   The graphic will be stored in the graphic programming software standard format and added to the graphic browser.

Inserting created graphics in WinCC

1. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
2. Select "Insert object" from the shortcut menu. The "Insert object" dialog box opens.

   Note

   In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

3. From the "Insert object" dialog box, select "Create from file."
4. Click the "Browse" button.
5. Navigate to the created graphic and select it.

   Note

   To import graphics files, note the following size restrictions:

   *.bmp, *.tif, *.emf, *.wmf ≤4 MB

   *.jpg, *.jpeg, *.ico, *.gif "*≤1 MB

Result

The OLE objects added are available in the "Graphics" editor.

Versions of the graphics for the current editing language are displayed in the "Screens" editor. The default graphic is displayed in all editing languages for which no screen has been imported.

The graphic is displayed in runtime in the set runtime language. The default graphic is displayed in all runtime languages for which no graphic has been imported.

You can double-click OLE objects in your library to open them for editing in the corresponding graphic editor.

See also

"Graphics" editor (Page 6836)

12.12.7.7 Languages in Runtime

Languages in Runtime

Using multiple runtime languages

You can decide which project languages are to be used in runtime on a particular HMI device. The number of runtime languages that can be available at one time on the HMI device depends on the device. To enable the operator to switch between languages during runtime, you must configure a corresponding operator control.

When runtime starts, the project is displayed according to the most recent language setting. When runtime starts the first time, the language with the lowest number in the "Order for language setting" is displayed.

Setting runtime languages during configuration

In the "Languages and Fonts" editor you can specifiy:

- The project languages to be available as runtime languages for the respective HMI device.
- The order in which the languages are to be switched.

See also

Methods for language switching (Page 6842)

Enabling the runtime language (Page 6843)

Setting the runtime language order for language switching (Page 6845)

Setting the default font for a runtime language (Page 6846)

Selecting the log language (Page 6847)

Languages in WinCC (Page 6823)

Methods for language switching

Introduction

You need to configure language switching if you want to have multiple runtime languages available on the HMI device. This is necessary to enable the operator to switch between the various Runtime languages.

Methods for language switching

You can configure the following methods for language switching:

- Direct language selection
  Each language is set by means of a separate button. In this case, you create a button for each Runtime language.
- Language switching
  The operator switches the languages using a button.

Regardless of the method used, the button names must be translated into each of the languages used. You can also configure an output field that displays the current language setting.

See also

Languages in Runtime (Page 6840)

Selecting the log language (Page 6847)

Enabling the runtime language (Page 6843)

Setting the runtime language order for language switching (Page 6845)

Setting the default font for a runtime language (Page 6846)

Enabling the runtime language

Introduction

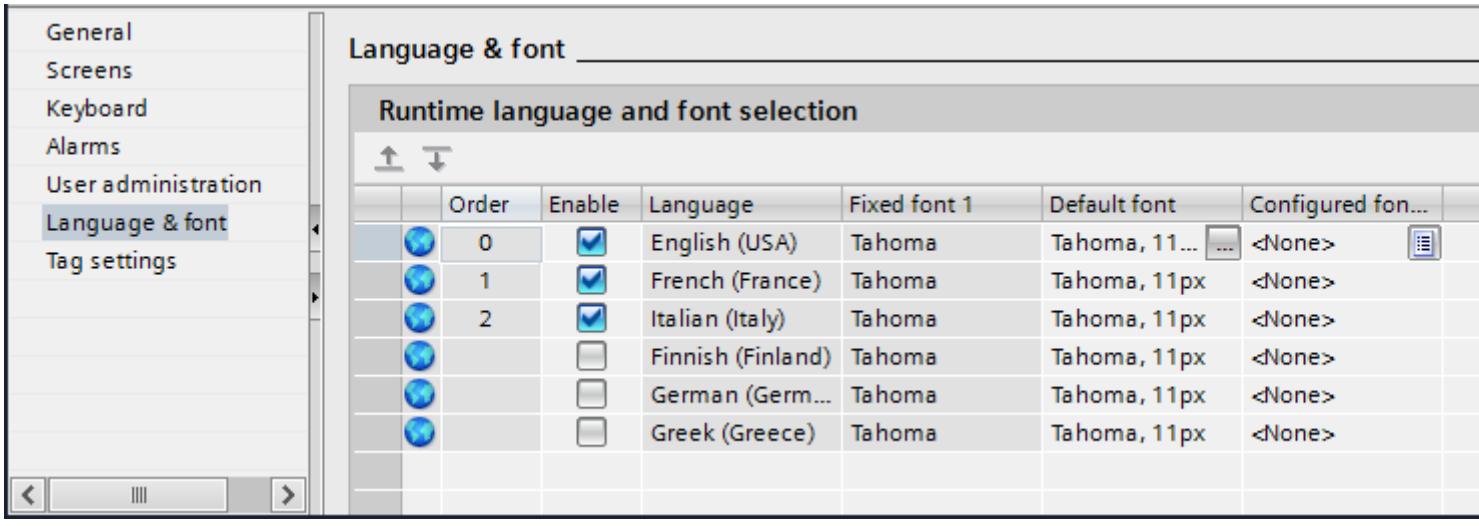
The "Language & Font" editor shows all project languages available in the project. Here you select which project languages are to be available as runtime languages on the HMI device.

Requirements

Multiple languages have been selected in the "Project languages" editor.

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font".
3. Select the following languages:
   - English
   - French
   - Italian



Result

You have now set three runtime languages. A number is automatically assigned to each language in the "Order" column. The enabled runtime languages are transferred with the compiled project to the HMI device.

If the number of languages selected exceeds the number that can be transferred to the HMI device, the table background changes color.

See also

Languages in Runtime (Page 6840)

Selecting the log language (Page 6847)

Setting the runtime language order for language switching (Page 6845)

Methods for language switching (Page 6840)

Setting the runtime language order for language switching

Introduction

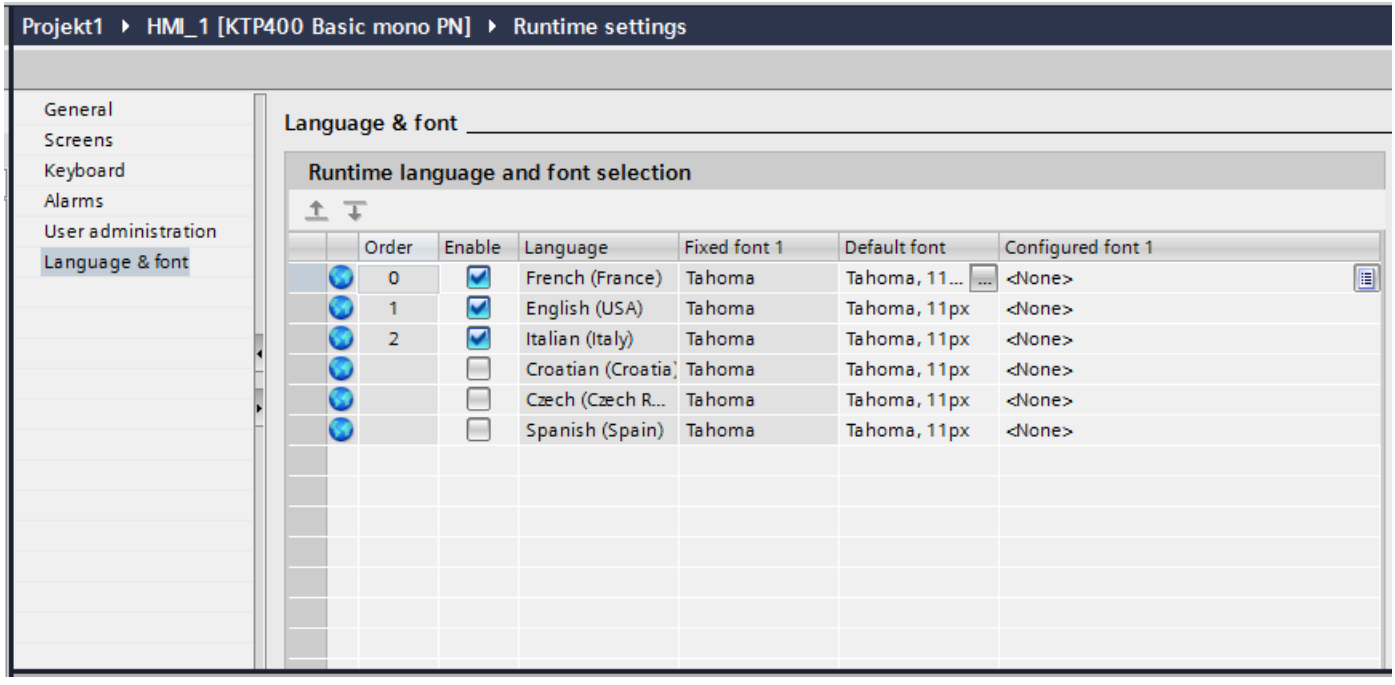
You specify the language order for Runtime language switching. The first time Runtime starts, the project is displayed in the language with the lowest number in the "Order" column.

Requirements

- Multiple languages have been enabled in the "Project languages" editor.
- The "Language & Font" editor is open and three Runtime languages have been set in the following order:
  1. English
  2. Italian
  3. French

Procedure

1. Select the runtime language "English".
2. Click the button. The runtime language "English" is moved down a place. The number will automatically be changed to "1" in the "Order" column.



Result

You have changed the order of Runtime languages. The first time runtime starts, the project will be displayed in the language with the lowest number. If the language is switched, this will happen in numerical order.

See also

Languages in Runtime (Page 6840)

Selecting the log language (Page 6847)

Enabling the runtime language (Page 6841)

Setting the default font for a runtime language (Page 6846)

Methods for language switching (Page 6840)

Setting the default font for a runtime language

Introduction

You can specify the font used to display the texts for each runtime language on the HMI device in the "Language & Font" editor. The default font is used in all texts, such as dialog texts, for which you cannot define a specific font.

WinCC offers only fonts supported by the HMI device.

Requirements

- Multiple languages have been enabled in the "Project languages" editor.
- Three runtime languages have been enabled in the "Language & Font" editor.
  1. Chinese
  2. German
  3. French

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font". The table shows the runtime languages and fonts set.
3. Click in the "French" row in the "Default font" column.
4. Select the font to be used by default if a font cannot be selected for a given text.

Result

The project texts for the runtime language "French" are displayed on the HMI device in the selected font.

These fonts are transferred to the HMI device during a transfer operation.

The default font is also used for the representation of dialogs in the operating system of the HMI device. Select a smaller font as default if the full length of the dialog texts or headers is not displayed.

See also

Languages in Runtime (Page 6840)

Selecting the log language (Page 6847)

Setting the runtime language order for language switching (Page 6843)

Methods for language switching (Page 6840)

Standardizing font for all languages

Introduction

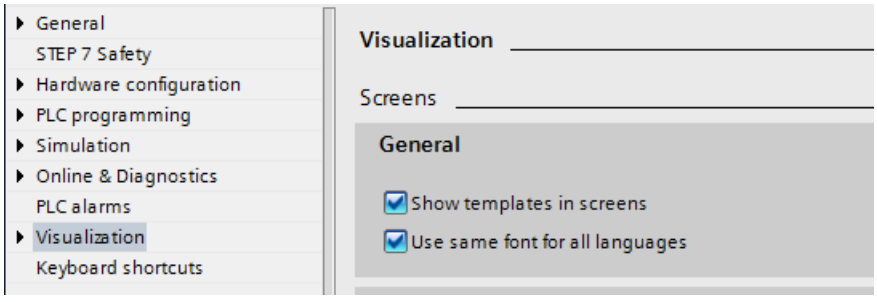
You can standardize the font for all project languages during configuration with the "Use same font for all languages" option.

Requirement

- Multiple languages have been selected in the "Project languages" editor.
- Multiple languages have been selected in the "Language & font" editor.
- The same font is defined for the selected runtime languages under "Configured font".

Procedure

1. In the "Options > Settings > Visualization > General" menu, select the "Use same font for all languages" option.



Result

You have enabled the option "Use same font for all languages". If you change the font of an object in one language during configuration, this font will be applied to all active languages.

Selecting the log language

Introduction

In the "Runtime settings > General" editor, select the language to be used for writing to logs in runtime.

Requirements

- The languages used in your project are activated in the "Project languages" editor, for example "German" and "English" .

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font".
3. Activate the runtime languages, for example, "German" and "English".
4. Specify the "order":
   - 1 German
   - 2 English
5. Click on "Runtime settings > General".
6. Select "German" for "Logs > Log language".

Result

After loading, the project will start in the runtime language "German". The logs are now written in German. During runtime, the operator switches the runtime language to English. The logs will still to be written in German.

See also

Languages in Runtime (Page 6840)

Setting the default font for a runtime language (Page 6844)

Setting the runtime language order for language switching (Page 6843)

Methods for language switching (Page 6840)

Enabling the runtime language (Page 6841)

Specific features of Asian and Eastern languages in Runtime

Introduction

Note the following special considerations for the operation in Runtime of projects for Asian languages.

Note

During configuration, only use the Asian fonts that your configuration computer supports.

Memory requirement for Asian character sets

The memory requirement is greater when using Asian languages. Therefore look out for corresponding error messages when compiling the project.

Font size for Asian character sets

Use at least a font size of 10 points to display the text of projects created for Asian languages in Runtime. Asian characters will become illegible if smaller font sizes are used. This also applies to the default font in the Runtime settings under "Language & font".

Text field length for Asian languages

Make allowances for an appropriate length of the text fields when working on multilingual projects with Asian languages. Field contents may be partially hidden, depending on the font and the font size.

1. Open the "Properties > Appearance" text box in the Inspector window.
2. Under "Fit to size", disable the "Auto-size" option.
3. Verify the proper display in Runtime.

12.12.7.8 Example of multilingual configuration

Example: Configuring a button for language switching

Introduction

In this example, you configure a button that can be used to toggle between multiple runtime languages during runtime.

Requirements

- You have completed the "Configuring a button in multiple languages" example.
- The "Screen_1" screen is open.
- The button on the screen has been selected.

Procedure

1. In the Inspector window, select "Properties > Events > Press".
2. Click on "Add function" in the table.
3. Select the "SetLanguage" system function and the "Switch" setting.

Result

You have assigned the button the function "SetLanguage". Pressing the button during runtime will switch the runtime language. The runtime languages are switched in the order specified by the number sequence in the "Languages and fonts" editor.

See also

Example: Configuring a button in multiple languages (Page 6850)

Example: Configuring a button for language switching for each runtime language (Page 6851)

Languages in WinCC (Page 6823)

Example: Configuring a button in multiple languages

Introduction

In this example, you configure a "Sprache umschalten" button in German and "Switch language" button in English.

Requirements

- The languages "German" and "English" have been enabled in the "Project languages" editor.
- German has been set as editing and reference language.
- You have created and opened the "Screen_1" screen.
- The Inspector window is open.

Procedure

1. Drag-and-drop a button from the "Tools" task card into the screen.
   The button will be added to the screen.
2. In the Inspector window, open "Properties > Properties > General".
3. Enter the text ""Sprache umschalten" under "Text > off".
4. Press <Enter> to confirm. The button is named.
5. Open the "Tasks" task card.
6. Select "English" under "Languages & Resources > Editing language".
7. Enter the label "Switch Language" under "Properties > Properties > General > Text > Off" in the Inspector window.

Result

The button name is configured in German and English language. The button name corresponding with the current Runtime language is shown in Runtime.

See also

Example: Configuring a button for language switching (Page 6847)

Example: Configuring a button for language switching for each runtime language (Page 6851)

Example: Configuring a button for language switching for each runtime language

Introduction

In this example, you configure a "Sprache umschalten" button in German and "Switch language" button in English.

Requirements

- The following languages have been enabled in the "Project languages" editor
  - German
  - English
  - Italian
- All languages have been set as runtime languages in the "Runtime settings > Language & Font" editor.
- You have created and opened the "Screen_1" screen.
- Three buttons have been created on the screen:
  - Button_1 labelled "Deutsch"
  - Button_2 labelled "English"
  - Button_3 labelled "Italiano"
- The Inspector window is open.

Procedure

1. Select "Button_1".
2. In the Inspector window, select "Properties > Events > Press".
3. Click on <Add function> in the table.
4. Select the "SetLanguage" system function.
5. Click on the "Switch" field.
6. Click on the button.
7. Select "Runtime language". The field will be highlighted in red.
8. Select "German" from the drop-down list.
9. Repeat steps 1 - 8 for the other two buttons and select the corresponding runtime language.

Result

You have configured three buttons for switching language in runtime. Each button will switch to a different runtime language. For example, clicking on the "English" button during runtime will switch the runtime language to English.

12.12.8 Replacing devices

12.12.8.1 Basics

Introduction

You can use existing configurations for new devices and optimize these configurations with little manual effort.

All data configured by you is retained in the configuration data. This means you do not need to copy individual objects of one device and paste them to another.

Principle

The following applies when you replace devices:

- Only functions supported by the new device are available. Only configuration data supported by the new device are displayed.
  This affects
  - recipes,
  - objects available on the screens,
  - available system functions,
  - available communication logs, etc.
- The number of supported objects, such as screens or tags, may be limited on the new device. If the existing objects exceed the limitations on the new device, the objects are displayed in full. The objects are, however, highlighted in color in the individual editors. An error is generated when the project data is compiled.
  Manual post processing is required when switching to a device with fewer features.
  Example: Limited number of connections
  All connections will be highlighted in color as invalid if fewer connections are supported on the new device than have been configured. Delete any excess connections.

Note

If you replace a Panel and select a PC Station as your new device, for example, WinCC Runtime Advanced will automatically be moved below the PC Station in the project tree.

See also

Example: Replacing devices (Page 6865)

Screen adjustment options (Page 6858)

Key assignment when replacing devices (Page 6854)

Device-specific functions (Page 6853)

Engineering system (Page 7007)

Basic Panel (Page 7009)

Panel (Page 7016)

Mobile Panel (Page 7020)

Multi Panel (Page 7026)

Comfort Panel (Page 7029)

WinCC Runtime Advanced (Page 7034)

12.12.8.2 Device-specific functions

Device-specific functions

Functions dependent on the device

Functions dependent on the device are implemented as follows:

- Colors
  The color is changed automatically when you switch from a device with full color display to one with a smaller color range.
  If you change the color manually and then change back again to a device with a larger range of colors, the reduced range of colors will be retained.
- Fonts
  Any configured font not available on a device will be replaced by a similar one or by the configured default font. The default font depends on the device selected.
- Character sets with different font sizes
  Avoid using too many different font sizes when configuring the following devices:
  - OP 73
  - OP 77A
  - TP 177A

  A character set is downloaded to the device for each font size. Check the Inspector window during compilation to see how much of the device memory is being used by the character sets.

- Font size
  Use small Windows fonts to display the text on devices. If you use large Windows fonts, then, depending on the size of the display, the text will not be displayed in full.
  Using font sizes of 28 pixels or more for the OP 77A and TP 177A devices will affect device performance.
  The character scope is much greater for Asian languages. The use of different font sizes therefore has serious implications on the memory requirements of all devices.
  Use the same font type for all large characters throughout the project to ensure effective and efficient configuration.
- Screens and screen objects
  If the new device supports a different resolution than the previous device when you replace a device, there are several ways to adjust the screens.
  Adjust the size of the screens to the new device in the menu under "Options > Settings > Visualization > Fit to size screen".

See also

Example: Replacing devices (Page 6865)

Basics (Page 6850)

Key assignment when replacing devices

Introduction

The devices available each have different function keys. The functions configured for these keys will be mapped to the available function keys of the new device if the device is replaced.

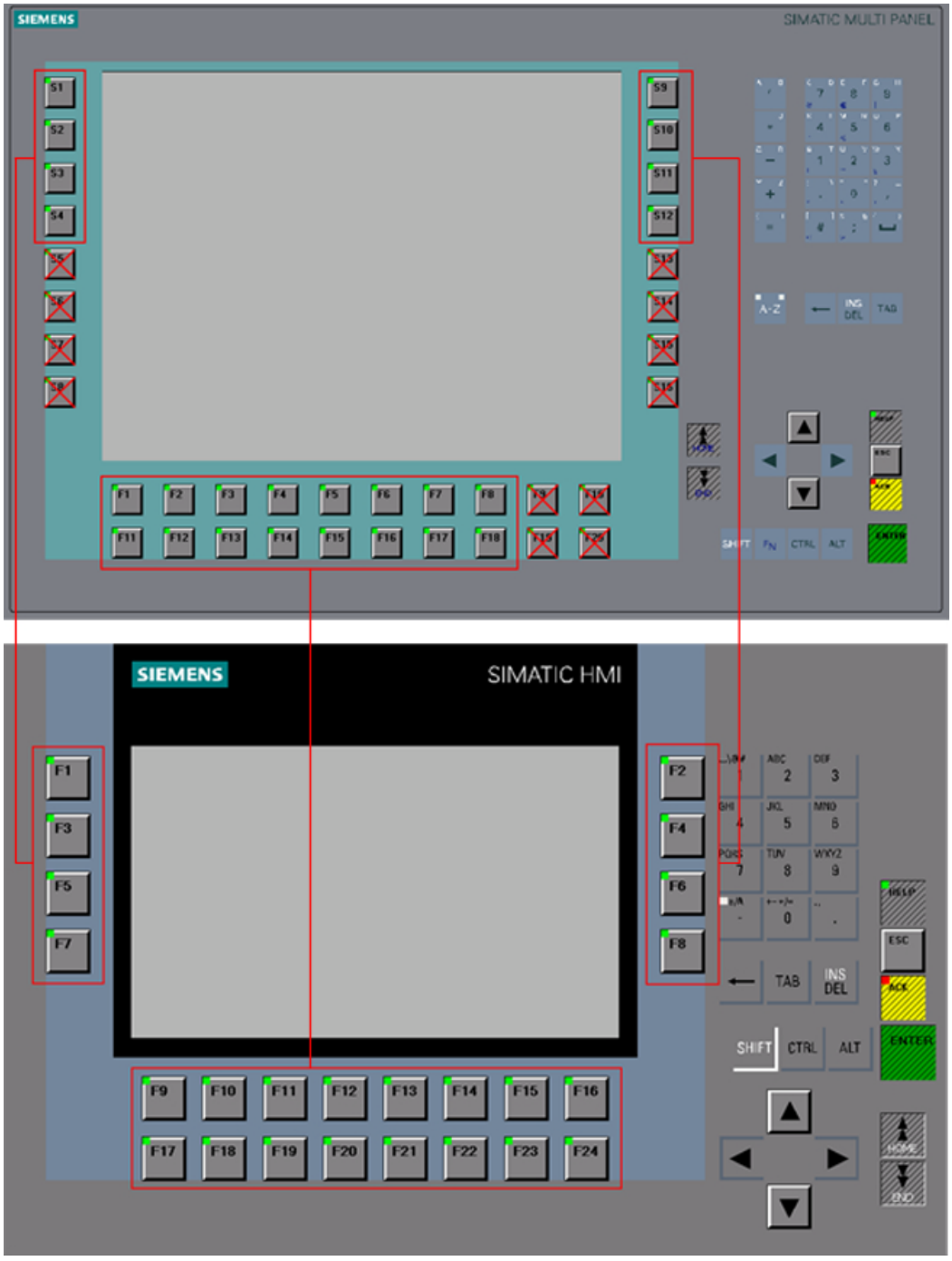
Function key mapping

The function keys to the left and right of the display are mapped from top to bottom to the new device. If the new device has fewer keys, the keys it does not have are not mapped. The function keys below the display are mapped from left to right to the new device. If the new device has fewer keys, the keys it does not have are not mapped.

Example: Exchange MP 377 with KP700 Comfort

You have configured a function on key S3 on the MP 377. This function is triggered by F5 if the panel is replaced with an KP700 Comfort.

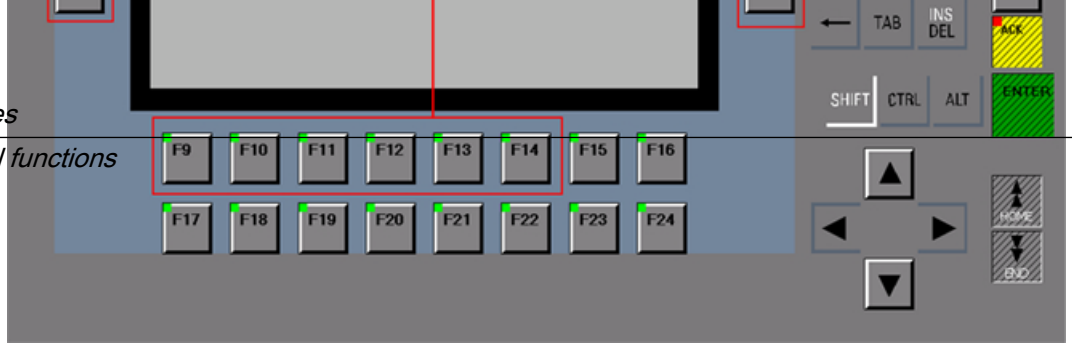
If you use the S8 key on an MP 377, this function is no longer available when the panel is replaced with a KP700 Comfort.



Mapping K-keys

K-keys are only mapped to the same K-keys in the new device, e.g. K5 to K5.

If the new device has no K keys, configurations for the K keys are lost.



Mapping of control keys and cursor keys

The following keys are mapped only to the same keys of the new device:

- HELP
- ESC
- ACK
- ENTER
- PAGE UP
- PAGE DOWN
- CURSOR UP
- CURSOR DOWN

Exception

"HELP" is triggered by "SHIFT+ ESC" on OP 73.

See also

Key assignment when replacing devices (Page 6856)

Basics (Page 6850)

Screen adjustment options (Page 6858)

Key assignment when replacing devices

Introduction

The devices available each have different function keys. The functions configured for these keys will be mapped to the available function keys of the new device if the device is replaced.

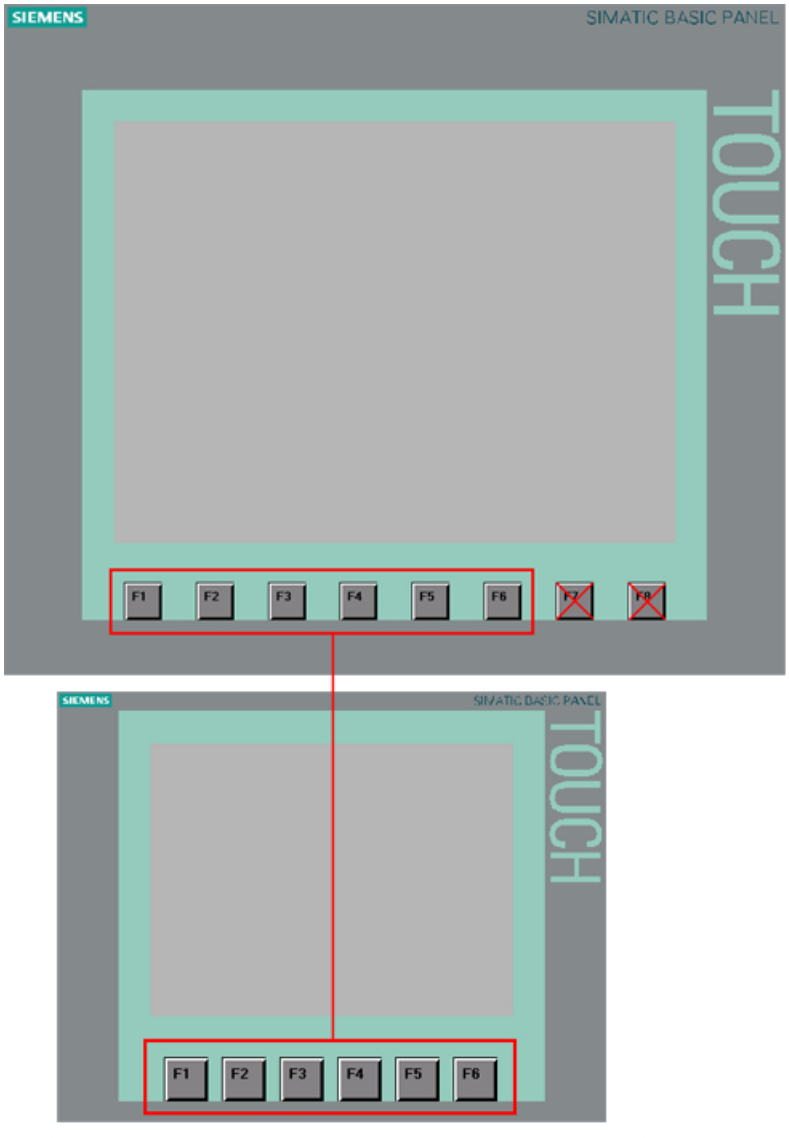
Function key mapping

The function keys below the display are mapped from left to right to the new device. If the new device has fewer keys, the keys it does not have are not mapped.

Example: Replacing a KTP1000 Basic with a KTP600 Basic

You have configured a function for F2 in KTP1000 Basic. This function is triggered by F2 following replacement with a KTP600 Basic.

If you have used F7 in a KTP1000 Basic, this function will no longer be available if the panel is replaced with a KTP600 Basic.



Mapping of control keys and cursor keys

The following keys are mapped only to the same keys of the new device:

- HELP
- ESC
- ACK

- ENTER
- PAGE UP
- PAGE DOWN
- CURSOR UP
- CURSOR DOWN

See also

Key assignment when replacing devices (Page 6852)

12.12.8.3 Adjusting screens to the new device

Screen adjustment options

Introduction

Select fit to size for screens before you replace a device. Fit to size is particularly important when switching devices with different display resolutions.

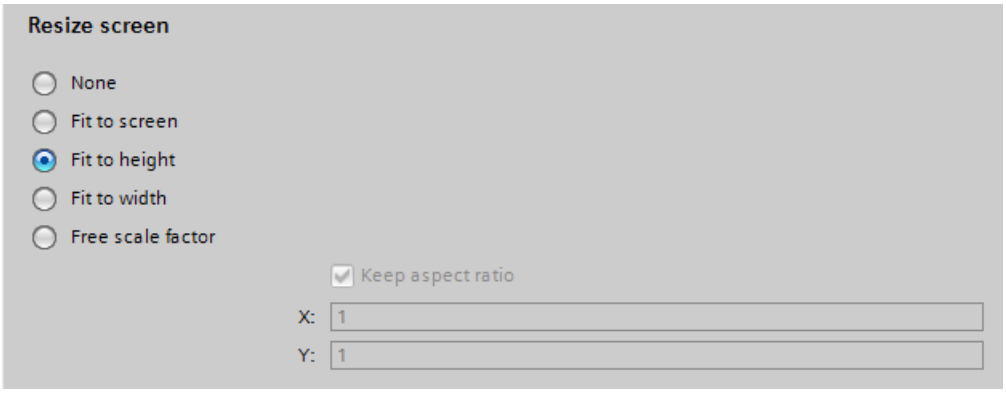
Object adjustment to content can be prevented for objects such as graphic views or text fields.

Note

The objects are distorted if you replace a device with a landscape format display with a device with a portrait format display. The difference in display format can, for example, result in object labels being cut off and content not being fitted to the object. You must therefore adjust the screens to the new device once you have replaced devices.

Screen adjustment when replacing devices

Adjust the size of the screens to the new device in the menu under "Tools > Settings > Visualization > Resize screen".



Select one of the following settings.

None (default setting)

The screens are not scaled. The objects in the screen retain their position and size. Use this setting as first test for checking of a possible exchange result because there are no rounding losses during forth and back exchange.

This option may result in objects being outside the configurable area if the display of the new device is smaller than the old one.

Fit to screen

The position and object size are adjusted to the new display size. Resizing takes place along the x-axis and the y-axis. Graphics and font size are adjusted accordingly.

Fit to height

The aspect ratio is maintained and the screens are adjusted to the height of the new device.

Use this option when you are replacing a device with display format 4:3, for example, with a device with widescreen.

Fit to width

The aspect ratio is maintained and the screens are adjusted to the width of the new device.

Use this option when you are replacing a device with widescreen, for example, with a device with display format 4:3.

Free scale factor

You select a free scale factor for screen adjustment. You can specify a factor for the x-axis and the y-axis.

Using a free scale factor of < 1 may distort the objects. Object labels may, for example, be cut off and the content may not be fitted to the object.

You must therefore adjust the screens to the new device once you have replaced devices.

Note

The aspect ratio is not adjusted for objects with a fixed aspect ratio, for example, gauge, circle. The objects are displayed on the new device with the same aspect ratio as prior to the replacement of the device.

See also

Example: Adjusting screens (Page 6864)

Specifying the position of screen objects (Page 6863)

Fit objects to contents (Page 6860)

Basics (Page 6850)

Key assignment when replacing devices (Page 6852)

Fit objects to contents

Introduction

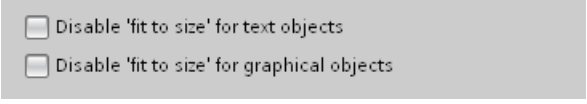
For some objects, you can specify fit to respective content in the Inspector window, for example:

- Text field: fit to text content
- I/O field: fit to text content
- Symbolic I/O field: fit to text content or to text list
- Graphic view: fit to included graphic

Fit to size for text and graphic objects

Disable automatic fit to size of the individual objects in the menu under "Options > Settings > Visualization > Resize screen and screen objects > Fit to content". This results in scaling of the objects as specified under "Options > Settings > Visualization > Resize screen and screen objects".

Select the objects which are not automatically fitted to size.



- If "Disable 'fit to size' for text objects" is activated, automatic fit to size is ignored in the text object properties.
  If you have activated "Fit screen to window height", the text field along with the other objects is scaled in accordance with the height of the new device.
- If "Disable 'fit to size' for graphic objects" is activated, automatic fit to size is ignored in the graphic object properties.
  If you have activated "Fit screen to window width", the graphics view along with the other objects is scaled in accordance with the width of the new device.

Note

The settings have no effect on screen objects whose size cannot be changed, such as alarm indicators or screen objects with a fixed aspect ratio.

"Disable 'fit to size' for text objects" and "Disable 'fit to size' for graphic objects" have no effect if:

- You have activated "Resize screen and screen objects > None".

- You have activated "Fit screen to window width and height" and the new device has the same resolution as the current device.

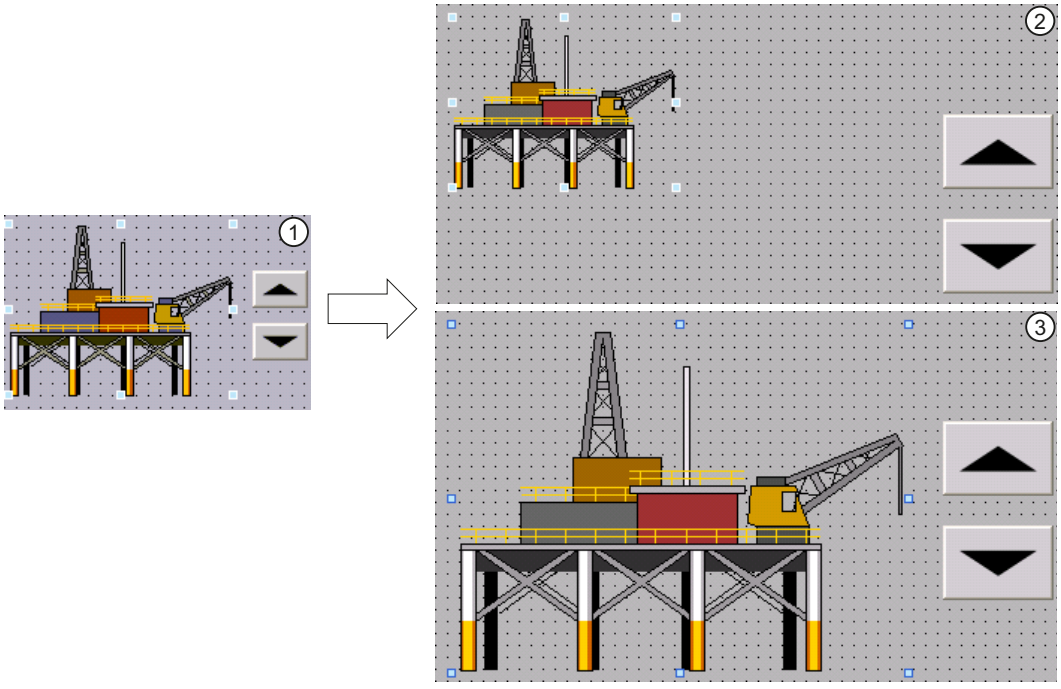
- You have activated "Fit screen to window height" and the new device has the same resolution as the current device.

- You have activated "Fit screen to window width" and the new device has the same resolution as the current device.

- You have selected "Resize screen and screen objects > None" and position "Top left" which means an adaptation is not necessary.

Example

The figure below shows the effects of automatic sizing using a graphic object with two buttons aligned as an example:



① Initial situation:

- Two buttons are aligned on a graphic object.
- The option "Fit object size to graphic" or "Adjust object size to graphic" is activated in the object properties of the graphic object under "Display > Sizing".

② Option 1: The original properties of the graphic object are to be maintained after switching the HMI device.

- Deactivate the option "Disable 'fit to size' for graphical objects" in the settings under "Size adaptation of objects".

Effect: The graphic object retains its original size after switching the HMI device. The alignment to the buttons is lost.

③ Option 2: The graphic object is to be placed relative to the new screen resolution after switching the HMI device.

- Activate the option "Disable 'fit to size' for graphical objects" in the settings under "Size adaptation of objects".

The option "Fit graphic to object size" is activated automatically in the object properties of the graphic object. The two buttons are properly aligned on the graphic object even after switching the HMI device.

See also

Specifying the position of screen objects (Page 6863)

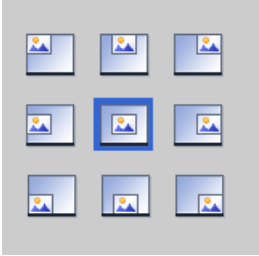
Screen adjustment options (Page 6856)

Example: Adjusting screens (Page 6864)

Specifying the position of screen objects

Introduction

There are various ways to adjust the position of screen objects to the new device.



Select position

Adjust the position of the screen objects to the new device in the menu under "Options > Settings > Visualization > Fit to size screen > Select position".

Example

The following option aligns the objects with the top left edge.

The following object centers the objects in the middle of the screen.

See also

Screen adjustment options (Page 6856)

Fit objects to contents (Page 6858)

Example: Adjusting screens (Page 6864)

12.12.8.4 Example: Replacing devices

Example: Procedures overview

Introduction

In the following example you replace a TP177 B 6" PN/DP with a TP700 Comfort.

Procedures overview

The procedure when replacing a device is as follows:

1. Adjust screens to the new device
2. Replacing devices

Example: Adjusting screens

Introduction

Adapt the screen before you replace a device. Screen adaptation is required as the display formats are different.

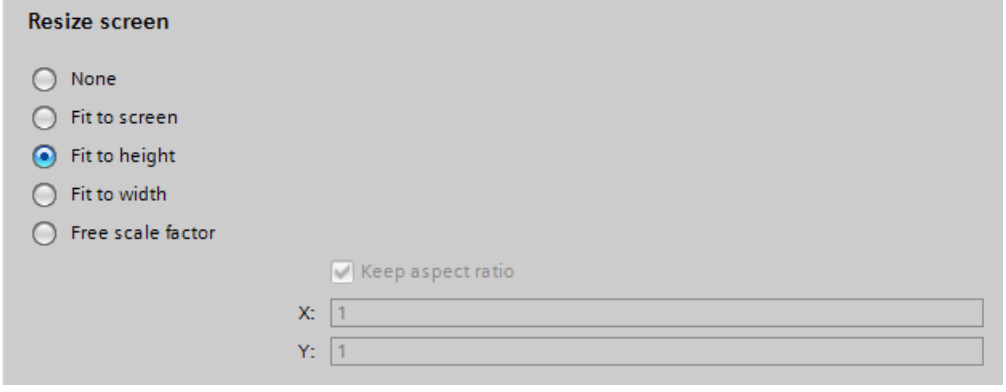
The TP 177B format is 320 x 240 pixels while the format of the TP700 Comfort is 800 x 480 pixels.

Requirements

- A project is open.
- The TP 177B 6" PN/DP device is used in the project.

Adjusting screens

1. Open the "Options > Settings" menu.
2. Click on "Visualization > Resize screen".
3. Activate "Fit to height".



4. Select "Disable fit to size for text objects".
5. Select "Disable fit to size for graphical objects".

Result

You have carried out screen adjustment in preparation for replacing devices.

See also

Example: Replacing devices (Page 6865)

Screen adjustment options (Page 6856)

Specifying the position of screen objects (Page 6861)

Fit objects to contents (Page 6858)

Example: Procedures overview (Page 6861)

Example: Replacing devices

Introduction

The following example shows you how to replace a device.

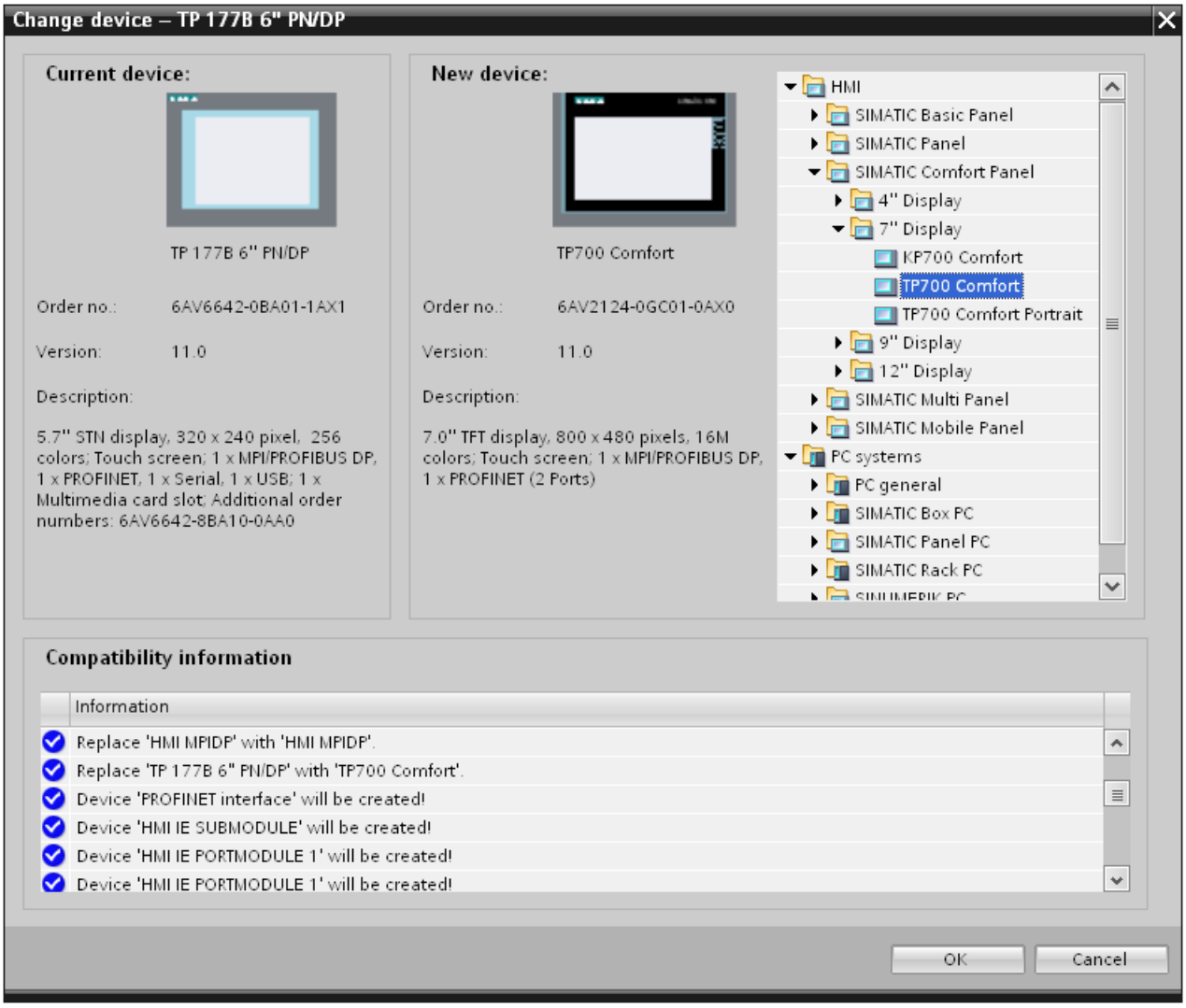
Requirements

- A project has been created and opened.
- The TP 177B 6" PN/DP device is used in the project.

Procedure

1. Double-click on "Devices & Networks" in the project navigation. The editor opens.
2. Click on the "TP 177B" device.
3. Select "Change Device/Version" in the device shortcut menu. A dialog opens.
4. Select the TP700 Comfort device.

5. Select the matching image for the HMI device version that suits your configuration under "Version".
   When you change the image, the new image is automatically transferred to the target system with the next download. All runtime files are deleted during this step.
   Details of hardware differences can be found in the "Compatibility information".



6. Click "OK". Device replacement is started.

Result

You have replaced the TP 177B device used in the project. You now use the TP700 Comfort device.

See also

Example: Adjusting screens (Page 6862)

Device-specific functions (Page 6851)

Basics (Page 6850)
Example: Procedures overview (Page 6861)
Updating the operating system (Page 6996)
Changing the device version (Page 6882)

12.12.9 Copying between devices and editors

12.12.9.1 Basics

Basics

Copying and pasting within an HMI device

You can copy and paste objects, such as display objects, within an HMI device. If the object is already created in the editor, when the object name is inserted a number is automatically attached, in accordance with the following principle:

- "<Object_name>_1" is renamed to "<Object_name>_2".
- "<Object_name>_2" is renamed to "<Object_name>_3".

Copying and pasting between HMI devices

You can also copy and paste between HMI devices. If an object with the same name already exists, you have the following options:

Note

Exception to this basic rule

Copying and pasting of an alarm class that has been generated from project-wide alarm class is handled differently than with this basic rule. When the copied alarm class already exists in the target HMI device within the same project, the "Paste" command is not performed.

Copying user-defined folders

You can create user-defined folders for editors, for example, for HMI tags, screens, etc.

You can copy user-defined folders and paste them into another HMI device. The objects contained in a user-defined folder may exceed the limitations applying to the other HMI device, such as the number of supported screens. After they have been pasted, all the objects are displayed. An error is displayed when the project data is compiled.

System folders cannot be copied.

See also

Unsupported objects and functionalities (Page 6869)

Copy and paste options (Page 6868)

Copy and paste options

Copy and paste options

Copy and pasting individual objects simplifies configuration.

WinCC offers a number of options for copying and pasting objects.

Shortcut menu

To copy and paste objects using the shortcut menu, proceed as follows:

1. Select an object, for example a button.
2. Select "Copy" in the shortcut menu.
3. Move the mouse cursor to the place on the screen where you want to paste the button.
4. Select "Paste" in the shortcut menu.
   The button will be pasted together with all properties already defined.

Drag&drop

To drag-and-drop objects, proceed as follows:

1. Click on "Screens > Start" in the project tree of a device.
2. Drag-and-drop the "Start" screen into the "Screens" folder of another device.
3. A dialog will appear if the second device already contains a screen with the same name.
4. Choose whether to replace or rename the existing screen.

See also

Basics (Page 6865)

Unsupported objects and functionalities

Introduction

When an object is copied, all its properties and settings are transferred to the target HMI device.

Unsupported objects

Objects that are not supported in the target HMI device cannot be pasted.

Note

When you copy a screen containing objects which are not supported by the destination HMI device, the objects remain in the background. When you copy the screen again and the new device supports the objects, they are displayed again.

Invalid objects

The following objects become invalid once they have been pasted into the target HMI device:

- Referenced objects that do not exist in the target HMI device.
- Objects with settings that are not supported in the target HMI device.
- System functions that were configured for objects and that are not supported in the target HMI device.

Invalid objects are highlighted by a color coding. Select a supported object or create a new one. If you retain an invalid object, an error will be displayed when the project data is compiled.

Colors and fonts

Colors and fonts are supported to varying degrees by HMI devices. When unsupported colors and fonts are pasted, they are replaced by supported colors and fonts. When you paste the same object back into the source HMI device, the original settings become active again.

See also

Basics (Page 6865)

12.12.9.2 Copying and pasting

Copying screens

Introduction

You copy one or more screens from the "Screens" folder and paste them into the "Screens" folder of another device.

Type and size of the displays

In the case of HMI devices with keys, the available keys are displayed automatically in the screen. When a screen is copied between HMI devices, the keys are either displayed or hidden; functions configured for function keys are not transferred.

If there is less space for the screen in the target HMI device than in the source HMI device, you can adjust the size of and the spacing between existing objects.

Automatic fit to size for objects

1. Select "Options > Settings > Visualization > Resize screen and screen objects" in the menu.
2. Activate, for example, "Fit to height".

See also

Copying recipes within an HMI device (Page 6870)

Copying objects with linked objects (Page 6871)

Linked objects copied automatically (Page 6872)

Linked objects copied automatically (Page 6872)

Drag & drop from the details view (Page 6873)

Copying recipes within an HMI device

"Recipes" Editor

You can copy recipes, recipe elements and recipe data records within each table. You copy a recipe element to another recipe.

Only WinCC Runtime Professional: You can copy a recipe view element to another recipe view. If a recipe view element of the same name already exists, a conflict dialog is displayed. You can select whether to replace or rename the recipe element. You can copy recipe elements to the first empty row of the "Recipe views" editor, "Elements" tab.

You can copy a recipe data record to another recipe, if the other recipe contains the same number of recipe elements. If the data types differ, the value will be copied to the target data record but it is assigned an error flag.

"Tags" editor

You can drag-and-drop a tag to a recipe element in the "Tag" column. The tag is linked to the recipe element. If a tag is already linked, an error message will be generated.

"Screens" editor

If you drag-and-drop a recipe to a screen, a new recipe display will be created and linked to the recipe.

See also

Copying screens (Page 6868)

Copying objects with linked objects

Introduction

An object is linked to another object in the following situations, for example:

- You specify a tag for an alarm as a trigger tag.
  The alarm is the object. The tag is the linked object.
- You specify a connection for an external tag.
  The tag is the object. The connection is the linked object.

The object is always fully inserted during copying and pasting. Whether or not the linked object is pasted depends on the command used to insert it.

Simple pasting

The linked object is not copied. The linked object is transferred and handled as follows in the target HMI device:

- If an object with the same name exists, the existing object with its settings is used.
- If no object with the same name exists, the name of the object will be displayed. The object becomes invalid.

For some objects, linked objects are pasted automatically during simple pasting.

Extended pasting

Select the "Extended paste" command in the shortcut menu to paste the linked objects as well. If objects of the same name exist in the target HMI device, you need to decide whether or not to overwrite each of these objects.

See also

Copying screens (Page 6868)

Linked objects copied automatically

Copying linked objects

The following table shows the objects for which linked objects are pasted automatically in simple pasting.

Object	Linked object
Screen	Template
Symbolic I/O field	Text list
Graphic I/O field	Graphics list
Graphic view	Graphic
Tag	Alarm
	Cycle
Recipe element	Text list
Scheduler	Triggers

See also

Copying screens (Page 6868)

Linked objects copied automatically

Copying linked objects

The following table shows the objects for which linked objects are pasted automatically in simple pasting.

Object	Linked object
Screen	Cycle
	Template
Symbolic I/O field	Text list
Graphic I/O field	Graphics list
Graphic view	Graphic
Tag	Alarm
	Logging tag
	Cycle
Log	Logging tag
Logging tag	Log type
Recipe element	Text list
Scheduler	Triggers

See also

Copying screens (Page 6868)

Drag & drop from the details view

Introduction

You can improve configuration efficiency with just a few simple measures. Below are a few examples of efficient configuration.

Pasting objects to a screen from the details view

You can drag objects in the details view from various different editors to other editors.

Pasting a symbolic I/O field

1. Open a screen.
2. Click on the "Text and graphics lists" editor in the project tree. All existing text and graphics lists will be shown in the details view.
3. Click on a text list, for example, "Textlist1" in the Details view.
4. Drag-and-drop a text list from the Details view to a screen. A symbolic I/O field has been created and connected to the text list "Textlist1".

Pasting a graphic I/O field

1. Open a screen.
2. Click on the "Text and graphics lists" editor in the project tree. All existing text and graphics lists will be shown in the details view.
3. Click on a graphics list in the Details view, for example "Graficlist1".
4. Drag-and-drop a graphics list from the Details view to a screen. A graphic I/O field has been created and connected to the graphics list "Graficlist1".

Pasting an I/O field

1. Open a screen.
2. Click on the "HMI tags" editor in the project tree. All existing HMI tags will be shown in the Details view.
3. Click on an HMI tag in the Details view, for example "Tag1".
4. Drag-and-drop the HMI tag from the Details view to a screen. An I/O field has been created and connected to the HMI tag "Tag1".

See also

Copying screens (Page 6868)

12.12.9.3 Copying between different RT and ES versions

Introduction

You can copy and paste project data such as screens, objects or tags between projects with different WinCC versions.

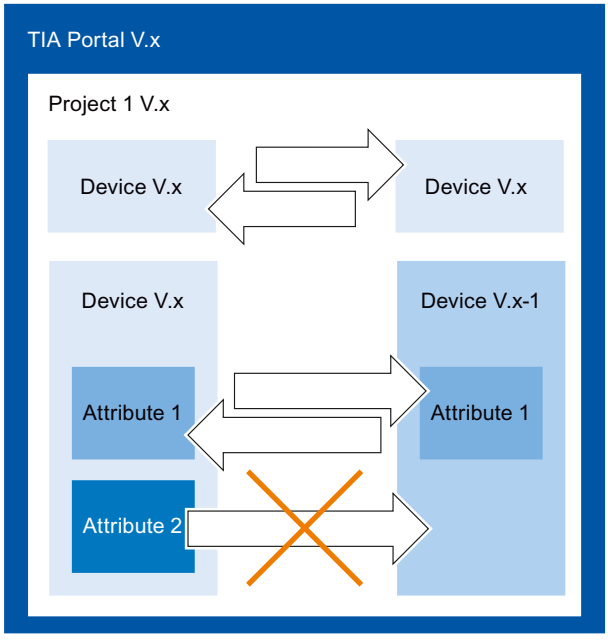
All configurations that are supported in the target version are retained when you copy data between projects of different WinCC versions. Configurations that are not supported by the target version are marked as invalid with a color code.

Copy of control between HMI devices from different device versions

WinCC supports all configurations of a previous WinCC version.

The following applies when copying within different ES versions:

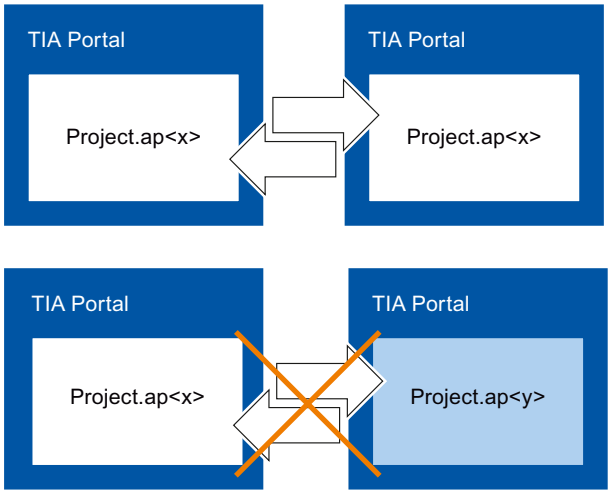
- All the configurations that are also supported in the respective RT version are retained.
- Default settings are defined for configurations that are only supported in the WinCC version of the target project.
- Configurations that are not supported by the respective RT version are marked by color as invalid or are not displayed.
  All properties and settings originally defined in the source HMI device are reactivated when you copy an object back to the source HMI device unchanged.



- The HMI device must be valid for the current Runtime version.

Copying between different ES versions

To copy between two TIA projects, open a second instance of the TIA Portal. You can only copy between projects that have the same ES version. The ES version of a project is indicated by the file extension *.ap<version_number>.



12.12.10 Using WinCC version compatibility

12.12.10.1 Basics on version compatibility

Introduction

Edit existing projects as follows with WinCC:

- You edit, compile and download existing projects with the range of functions of the previous version of WinCC. You can continue to edit these projects afterwards with the previous version of WinCC.
- You upgrade existing projects and use the functions of the current WinCC version.

Note

WinCC functions

While editing a project of a previous WinCC version, you can only access the functions and HMI devices of this previous WinCC version.

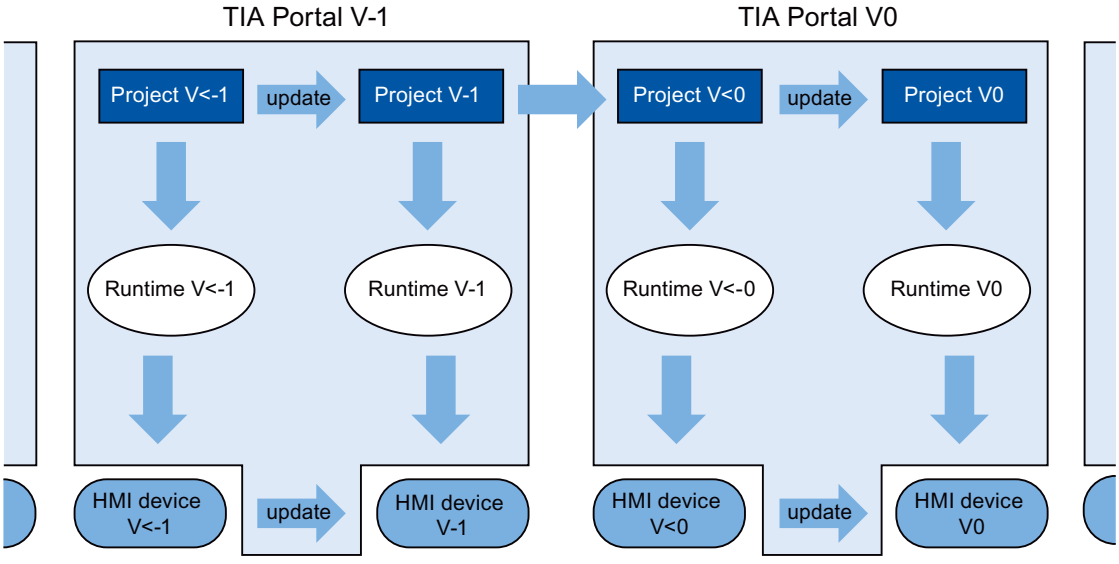
Versions in WinCC

In WinCC, you work with different version types:

- WinCC version
  The WinCC version installed on the configuration PC, for example, WinCC V12.
- Project version
  Projects are created using the WinCC version that is installed on the configuration PC. While you are editing a WinCC project from a previous version in the current WinCC version, the version ID is displayed behind the project name in the project tree.
- Runtime version
  You can configure HMI devices with different runtime versions in WinCC. You specify the runtime version once for an HMI device. The device version must match the Runtime version.
- Device versions
  Depending on the used HMI device, the image is a combination of the operating system and / or runtime software. For each HMI device, WinCC provides various images which can be loaded onto the HMI device, if necessary, according to the configuration. The device version corresponds to a specific image. The device version must match the configuration.

Compatibility of WinCC versions, Runtime versions and device versions

The figure below shows the interaction of the versions in the TIA Portal:



Creating projects

After you create a new project in WinCC, open and edit it in the WinCC version in which you created it.

Saving

To save a project of a predecessor WinCC version in this version again, save it in the usual way. If you manually upgrade the project to your WinCC version, please note that you will no longer be able to open the project in the previous WinCC version.

To save a project of the previous WinCC version in the current version, upgrade the project to your WinCC version. You can then no longer edit the project with a previous version of WinCC.

Compiling, simulating and loading

If you are using a project of a previous WinCC version, you can use your current version to generate Runtime data for this previous version. This also allows you to load HMI devices that are no longer compatible with your WinCC version.

Copying within projects with different WinCC versions

If objects and configurations are also available in the target version, copy these as required via the clipboard or using drag-and-drop.

Opening, editing and saving projects of a previous WinCC version

You can open and edit projects of previous WinCC versions as required. In doing so, you only utilize the functions of the previous WinCC version. Once you have completed editing, you can once again save and edit the project in the previous WinCC version.

Compiling, downloading and simulating projects of a previous WinCC version

You can compile, download and simulate projects of previous WinCC versions as required. Your current WinCC version will provide the Runtimes and device versions for the corresponding WinCC version.

See also

Editing projects of a previous WinCC version (Page 6878)

Upgrading projects (Page 6879)

Changing between device versions (Page 6881)

Changing the device version (Page 6882)

12.12.10.2 Editing projects of a previous WinCC version

Introduction

WinCC provides the option of editing projects of a previous WinCC version. While editing a project of a previous WinCC version, you can only access the functions of this version. In order to use the functions of your current WinCC version for this project, upgrade the project to your WinCC version.

Note

If you upgrade a project to your WinCC version, please note that you will no longer be able to open and edit the project in the previous WinCC version.

Requirement

- A project of a previous WinCC version has been created.
- The current WinCC version is installed on the configuration PC.

Procedure

Proceed as follows to edit a project of a previous WinCC version:

1. Open the project.
2. Edit the project using the functions of the previous WinCC version.
3. Save the project.
4. Compile the project.
5. Download and simulate the project.
6. You can open the project in the previous WinCC version for further editing, if required.

Result

The modified project data can be edited further on a different configuration PC with the previous WinCC version for further processing. The Runtime project was generated and downloaded in the corresponding Runtime version.

See also

Basics on version compatibility (Page 6873)

12.12.10.3 Upgrading projects

Introduction

If the project version is older than the WinCC version, the version ID is displayed in the project navigation. Your WinCC version also contains the previous version that you can use to edit projects as required. In order to use the functions and options of your WinCC version in a project, upgrade the project to your WinCC version. Then change your device or Runtime version(s) to match the new project version.

Note

WinCC version compatibility

If you upgrade a project to your WinCC version, please note that you will no longer be able to edit the project with the previous version.

Requirements

- The project version is a predecessor of your WinCC version.
- You have write access to your project drive.
- The project drive provides sufficient storage capacity for another project of this size.

Procedure

Proceed as follows to upgrade a project to your WinCC version:

1. Select the project in the project navigation.
2. Select the "Upgrade project" command from the shortcut menu of the project.
   A dialog opens.
3. Click "Confirm".
   The project closes and the progress bar is displayed.

An alarm is output when the project has been upgraded.

Result

- The project has been saved on the project drive in the previous WinCC version and with the corresponding file extension.
- The project is displayed on the project drive in the current WinCC version and with the corresponding file extension.
- The project is displayed without a WinCC version ID in the project navigation.

To make use of the functions of the new WinCC version, change your device or Runtime version(s) in accordance with the new project version in the next step.

See also

Basics on version compatibility (Page 6873)

12.12.10.4 Upgrading a global library

Introduction

In order to process objects of a global library in a project, the global library must be available in the product version of the project. You can upgrade each global library from an older product version to the current product version. You are prompted accordingly when you open the global library.

Requirements

- The version of the global library is a predecessor of your WinCC version.
- You have write access to your project drive.
- All types in the library have been released.

Note

Upgrading a user library

If you want to use a user library from an earlier version of WinCC, you must upgrade it. Make sure that all types of the library have been released. The library being upgraded cannot contain a type with the "in progress" status.

Procedure

Proceed as follows to upgrade a global library from TIA Portal V12.x or lower:

1. Open the global library.
   The "Upgrade global library" dialog box opens.
2. Click "OK".

A copy of the global library is created and upgraded. The copy of the global library receives the name extension "_V13". The global library opens.

Result

The global library is stored with the appropriate file extension.

See also

Basics on version compatibility (Page 6873)

Managing libraries
(Page 6776)

12.12.10.5 Changing between device versions

Selection of the device version

When you configure a new HMI device, WinCC automatically selects the latest version of the device.

If you want to use a device version other than the one set in WinCC, transfer an image to the HMI device. WinCC provides the images required for the supported HMI devices.

Information on the device versions used in WinCC is available in the FAQs on the Internet, entry ID 21742389.

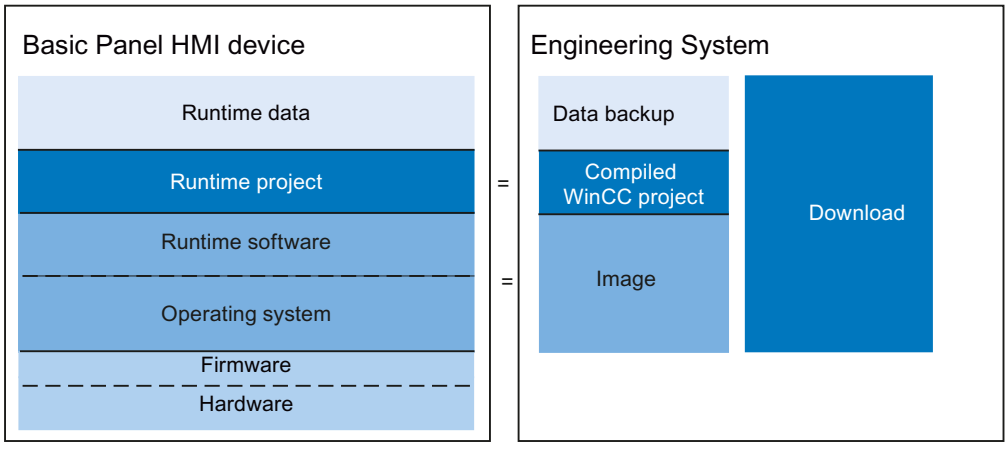
Notice

Changing the device version deletes all data on the HMI device.

Data is deleted on the target system if you change the device version. For this reason, you should save existing Runtime data before changing the device version.

HMI device configuration

The following figure shows the software components of an HMI device:



See also

Basics on version compatibility (Page 6873)

Updating the operating system (Page 6996)

Changing the device version (Page 6882)

12.12.10.6 Changing the device version

Introduction

Depending on the required Runtime version, select the suitable device version for your configuration.

Note

Selection of device versions

The selection of available devices versions depends on the version of the project.

Requirements

- A project has been created and opened.
- The project contains an HMI device.

Procedure

To change the device version, follow these steps:

1. Double-click on "Devices & Networks" in the project tree.
   The editor opens.
2. Select the required HMI device from the device view.
3. Select "Change Device/Version" in the device shortcut menu of the HMI device.
   A dialog opens.
4. Select the required HMI device.
5. Select the required device version under "Version".
6. Confirm your selection with "OK".

Result

You have changed the device version in the WinCC project.

Notice

Changing the device version deletes all data on the HMI device.

All data is deleted from the HMI device when you change the device version and compile/ download the project. You should therefore backup your Runtime data prior to the download.

See also

Basics on version compatibility (Page 6873)

Changing between device versions (Page 6879)

12.12.11 Viewing memory card data

12.12.11.1 Basics

Introduction

WinCC provides you with the possibility of viewing data stored on your memory card. The function supports the use of memory cards of the HMI device and of the CPU.

You have the following options:

Viewing a backup (Page 6883)

Renaming and deleting backups (Page 6885)

Viewing HMI device images (Page 6886)

Deleting HMI device images (Page 6887)

Creating HMI device images on memory card (Page 6888)

See also

Viewing a backup (Page 6883)

Renaming and deleting backups (Page 6885)

Viewing HMI device images (Page 6886)

Deleting HMI device images (Page 6887)

Creating HMI device images on memory card (Page 6888)

12.12.11.2 Working with backups

Viewing a backup

Introduction

The backup of a Basic Panel that is stored on a memory card can also be viewed in the TIA Portal.

Requirements

- WinCC is installed.
- A memory card with a backup is available.
- The card reader is connected to the configuration PC.
- The project view is open.

Backup on the memory card in the card reader

1. Insert the memory card into the card reader.
2. Open "SIMATIC Card Reader" in the project navigation.
3. Select the card reader drive.
   The "Online Card Data" folder is displayed.
4. Open the "Online Card Data" folder
5. Click the backup to open the shortcut menu.
6. Select "Properties".

Backup on the memory card of the PLC

Proceed as follows if the backup is stored on the memory card of the PLC:

1. Connect the PLC with the configuration PC.
2. Click on the PLC in the project navigation.
3. Select "Connect online" from the shortcut menu.
   A connection to the PLC is established.
   Once the PLC is connected, the "Online Card Data" folder is displayed.
4. Open the "Online Card Data" folder.

   Note

   Accessing a password-protected PLC

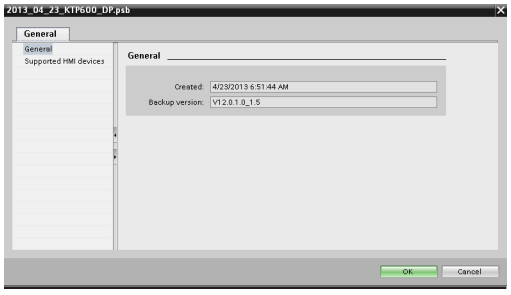
   When you attempt to access a PLC that is protected by a password, you will be prompted to enter the password.

   You need at least read access rights in order to view the data that is stored on the memory card.

5. Click the backup to open the shortcut menu.
6. Select "Properties".

Result

The backup properties are displayed in a separate dialog.



See also

Renaming and deleting backups (Page 6885)

Basics (Page 6881)

Renaming and deleting backups

Introduction

You can rename and delete backups from a memory card in the project navigation of the TIA Portal.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
  The PLC is connected online with the configuration PC.
- A memory card with a backup is available.
- The project view is open.
- The backup is displayed in the project navigation.

Note

Accessing a password-protected PLC

When you attempt to access a PLC that is protected by a password, you will be prompted to enter the password.

You need write access rights to rename or delete memory card data.

Procedure

1. Click on the backup in the project navigation.
2. Open the shortcut menu.

3. Select "Rename" to rename the file.
4. Enter a new name.
5. Select "Delete" to delete the file.

Result

The backup file is now renamed or deleted.

See also

Viewing a backup (Page 6983)
Basics (Page 6983)

12.12.11.3 Working with HMI device images

Viewing HMI device images

Introduction

The HMI device image of a Comfort Panel that is stored on a memory card can be viewed in the TIA Portal.

Requirements

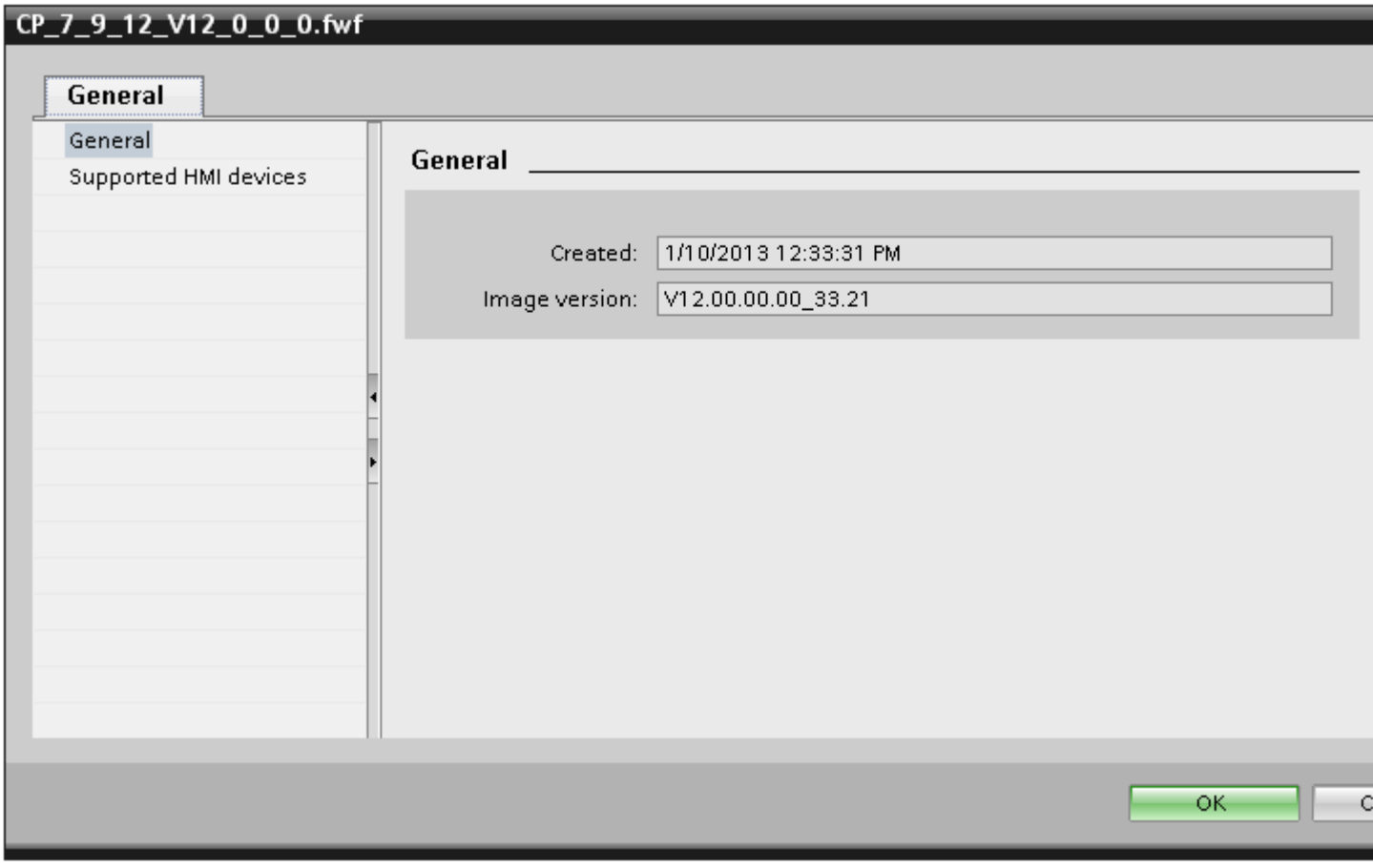
- WinCC is installed.
- The card reader is connected to the configuration PC.
- A memory card with the HMI device image is available.
- The project view is open.

Procedure

1. Insert the memory card into the card reader.
2. Open "SIMATIC Card Reader" in the project navigation.
3. Select the card reader drive.
   The "Online Card Data" dialog is displayed.
4. Open the "Online Card Data" folder.
   The available images of the HMI device are displayed in additional folders.
5. Click the required HMI device image.
6. Select "Properties" in the shortcut menu.

Result

The properties of the HMI device image are displayed in a separate dialog.



See also

Deleting HMI device images (Page 6887)

Creating HMI device images on memory card (Page 6888)

Basics (Page 6881)

Deleting HMI device images

Introduction

You can delete the HMI device image of a Comfort Panel from a memory card in the project navigation of the TIA Portal.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
- A memory card with an HMI device image is available.
- The project view is open.
- The HMI device image is displayed in the project navigation.

Procedure

1. Click the HMI device image in the project navigation.
2. Open the shortcut menu.
3. Select "Delete" to delete the file.

Result

The HMI device image is deleted.

See also

Viewing HMI device images (Page 6884)

Basics (Page 6881)

Creating HMI device images on memory card

Introduction

You may edit the HMI device image of a Comfort Panel without connecting the Comfort Panel to the configuration PC.

Simply create the HMI device image on an external memory card or USB stick and then transfer it from there to the Comfort Panel.

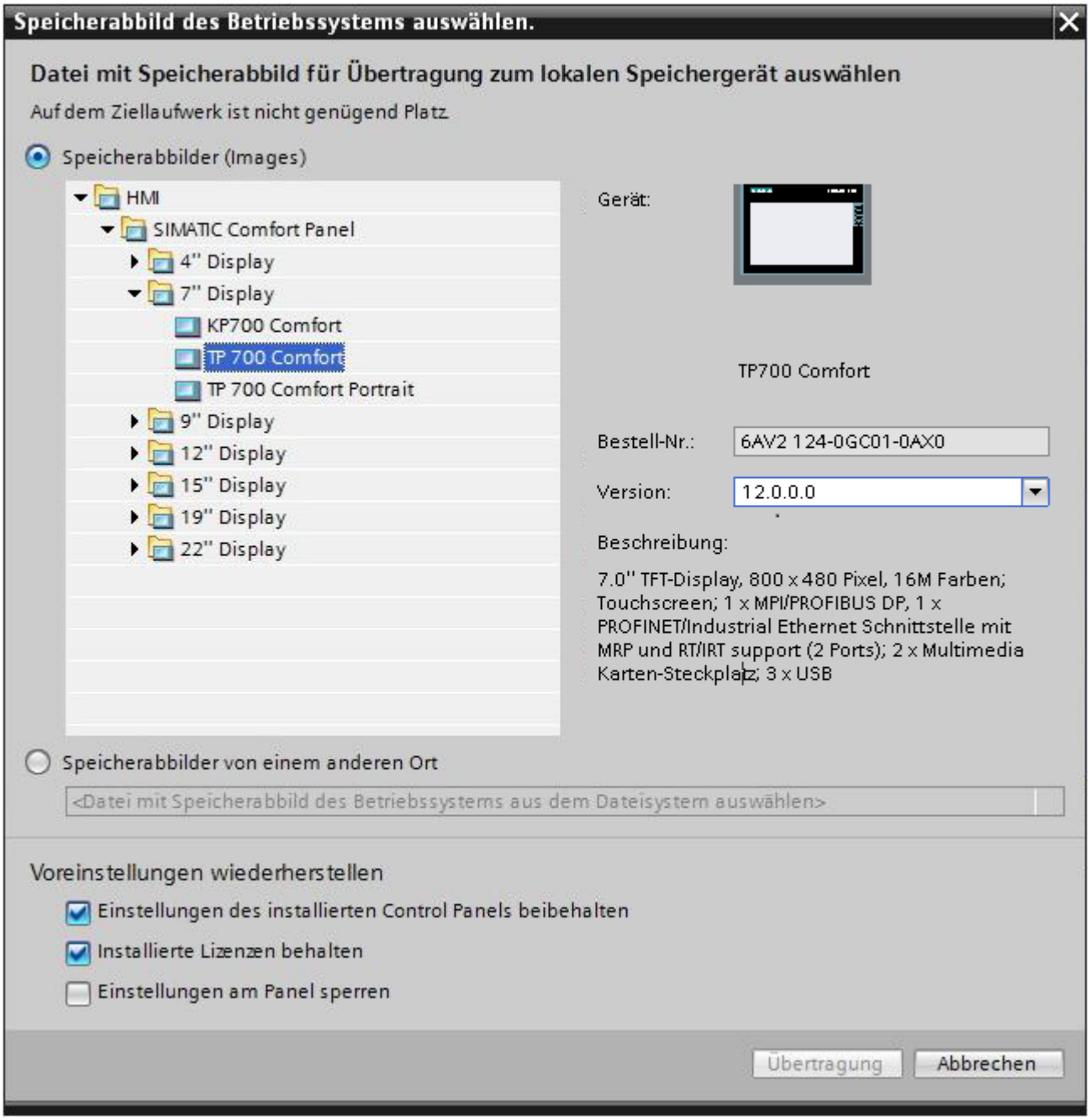
Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
- The project view is open.

Procedure

1. Insert the memory card into the card reader.
2. Click on the memory card in the project navigation.
3. Open the shortcut menu.

4. Select "Create HMI OS image on memory card".
   A dialog opens.



5. Select an HMI device image.
6. Select the settings for the restoration.
   - Activate "Retain installed settings of the Control Panel"
     The settings you have made on the Comfort Panel are retained.
   - Activate "Retain installed licenses":
     The licenses on the Panel are retained.
   - Activate "Lock settings on the Panel"
     You can no longer change the selected settings for "Retain installed settings of the Control Panel" and "Retain installed licenses" on the Comfort Panel.

Result

You have created an HMI device images on memory card. You may use a USB stick instead of a memory card.

See also

Viewing HMI device images (Page 6884)

Basics (Page 6881)

12.12.12 Managing colors centrally

12.12.12.1 Basic principles for color management

Introduction

WinCC provides the option of changing the colors used in a project centrally. The "Change color reference" dialog box contains a hierarchical overview of all color-relevant properties contained in the selected object. In the display, you can navigate within the display and operating objects that are shown and thus get an overview of all colors used. Using search and filter you can specify a color selection and replace it with other colors, if necessary.

Supported objects

You have access to all colors used and configured in the project with the "Change color reference" dialog. Excluded from this are colors that are used as follows:

- In types and instances from a library
- In faceplates
- In scripts
- In designs
- In screens with write protection

12.12.12.2 Finding and replacing colors

Introduction

The scope of the objects displayed in the "Change color references" dialog depends on the location at which the dialog is called.

- If you select an HMI device and call the dialog, all color references used on the HMI device are displayed.
- If you select a display object within a screen and call the dialog, only those color references that are included in the display object are displayed.

Requirements

- You have created a project.
- Screens have been created.

Procedure

1. Select the object that contains the required color references.
2. Select "Change color reference" in the shortcut menu.
   A dialog opens.
3. Select the color that you want to change.
   - Click the color field in the search box.
     The project color selection opens.
   - To select a standard color or a user-defined color, click "More colors".
   - To use a color directly from the selected object, drag a color field from the overview table to the search box.
   - To select similar colors as well, set a tolerance.
4. Filter the displayed table.
5. In the table column "Replace with" select the new color for the individual properties.
6. Click "OK".

Result

The new color references are configured in the selected object.

12.13 Compiling and loading

12.13.1 Establishing a connection to the HMI device

Introduction

To download a WinCC project to an HMI device, a properly configured connection must be set up between the configuration PC and HMI device. The connection cannot be set up, the download is cancelled.

Setting up a connection between the configuration PC and HMI device

1. Check the cable connection between the HMI device and configuration PC.
2. Open the "Devices & Networks" editor in WinCC and start the network view.
3. Select the subnet in the network view and check the settings for the subnet.
4. Select the interface of the HMI device in the network view or device view and check the connection parameters in the Inspector window.
5. Switch on the HMI device and press the "Control Panel" button in the loader.
   The Control Panel opens.
6. Press "Transfer" twice in the Control Panel.
   The "Transfer Settings" dialog box opens.
7. Check the settings and then press "Advanced".
   The [Protocol*] Settings" dialog opens.
   *: The title of the dialog depends on the protocol used, for example, "PROFIBUS Settings".
8. Check the advanced settings and close the dialog with "OK".

Important settings

Check the connection settings and in particular the following parameters:

- Network and station addresses
- Selected transmission rate
- Master on the bus; as a general rule, only one master is permitted.

If using a configurable adapter for the connection, check the adapter settings, for example, transmission rate, master on the bus.

See also

Loading a project (Page 6940)

12.13.2 Basic Panels

12.13.2.1 Runtime settings

Settings in the Runtime software

In WinCC you configure the settings for the Runtime software.

To edit the Runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree.

Display on the target system

In WinCC, configure the visual representation of the generated project in Runtime. The screen resolution is fixed for Basic Panels. Scroll bars will appear if the screen is larger than the configured screen resolution.

To switch off the taskbar, select the Start menu command "Settings > Taskbar". In the "Taskbar Properties" dialog, disable the options "Always on top" and "Auto hide".

Display in Runtime

In the Runtime settings, specify the start screen and default template. You can find additional information in the online help for WinCC under "Working with screens".

Under "General > Screen", you specify whether the HMI device uses the default style of the project or another predefined style.

Comparison of the project version with the PLC

The Project ID area pointer is used to configure the project ID for checking the consistency of the project with the PLC. The area pointer reads the project version from the PLC. You can find additional information in the online help for WinCC under "Communicating with PLCs".

You select the project ID in the Runtime settings of the HMI device under "General > Identification > Project ID".

At startup, Runtime checks whether the project ID of the project matches the PLC project version. Runtime will only start if the two values match.

An area pointer "Project ID" is created for connections whose address you specify in the PLC. The value that you store at this address will become the project ID.

The project ID is not available on all HMI devices.

Logging language

Specify the logging language under "General > Logs". The logs are written in Runtime in the selected language.

Specifying bit selection for text and graphic lists

Under "General > Screens > Bit selection", you specify whether bit selection for text and graphic lists is used on this HMI device.

- When you select the option, the text or graphic is displayed that has been configured for the set bit with the least significance.
- When you disable the option and several bits are set, the text or graphic that has only been configured for the set bit is displayed.

Operation using function keys

To symbolize the function of a function key for keyboard devices, you configure a graphic next to the function key in the display. Under "Screens > Function keys > User-defined pictogram size", you specify whether you want to use a non-standard graphic size.

Configuring operation in Runtime

Under "Keyboard > General > Use screen keyboard", you specify whether the screen keyboard is available on your HMI device.

Under "Keyboard > General > Release button on exit", you specify whether the "Release" event is triggered when the user exits a button without releasing it.

Under "Keyboard > Disable function keys in dialogs", you specify whether the function keys are disabled for the duration of the displayed dialog. Large dialogs can hide areas or graphics that border the function keys or describe the function keys. Enable this property to prevent functions keys from being operated in such a case.

Alarms

Under "Alarms" you specify the layout and properties of alarms. You can find additional information in the online help for WinCC under "Working with alarms".

User management

Under "User management" you specify the settings for password-protected access in Runtime. You can find additional information in the online help for WinCC under "Configuring user administration".

Language & font

Under "Language & font", you specify the configured font for each language. The configured font is available during configuration and is transferred in addition to the HMI device during the transfer. The configured font can be used for displaying the display and operating elements.

The dialog text will be shown in the standard font. Define the default font under Runtime settings > "Language & font". You also specify Runtime languages and the sequence of the language selection in Runtime under "Language & font".

Settings for tags

Under "Settings for tags" you configure the name synchronization of the PLC tags depending on the requirements of your project.

12.13.2.2 Overview of compiling and loading projects

Overview

The project is compiled in the background even as you are configuring it in WinCC. This reduces the time for final compilation. When you start compilation, you create a file that can be run on the corresponding HMI device.

If an error occurs during compilation, WinCC provides support in locating and correcting it.

Once you have corrected any problems, you download the compiled project to the HMI devices on which the project is to run.

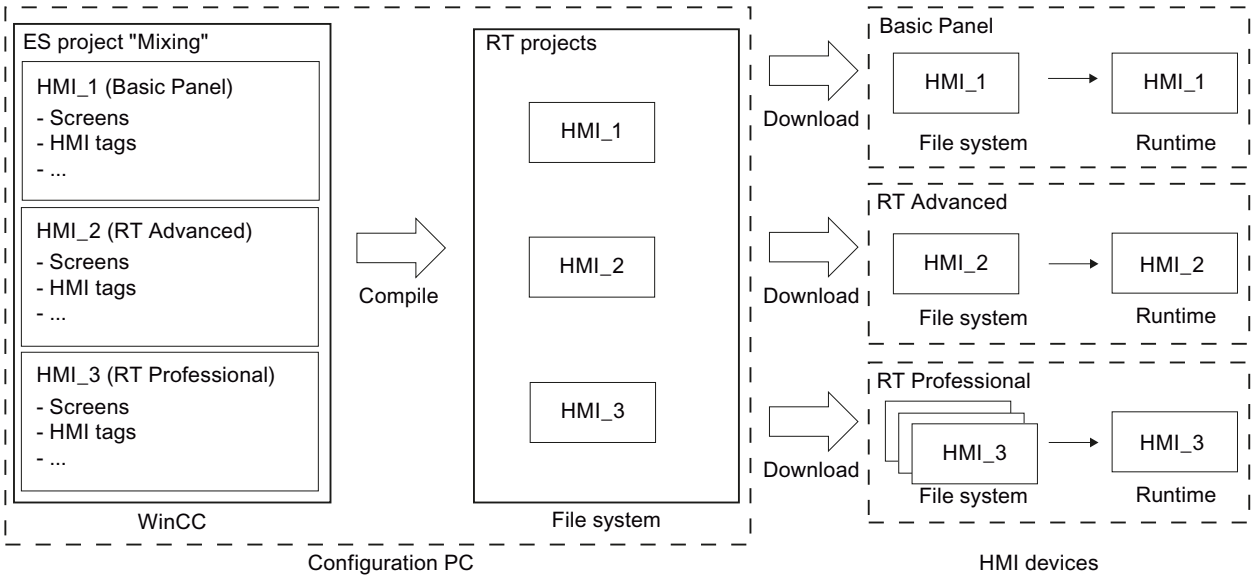
If you are using HMI tags in your project that are connected to PLC tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Definition of terms

The term "project" has two different meanings in the contexts of compilation and loading. "Project" is the WinCC project on the configuration PC. "Project" is also the Runtime project you create by compiling the configuration data of an HMI device and download to the HMI device.

- WinCC project: contains the configuration data of one or more HMI devices
- Runtime project: contains the compiled configuration data of an HMI device

The figure below illustrates the link between WinCC projects and Runtime projects using the example of the "Compile and load" process:



Runtime version

The Runtime version depends on the image of the configured HMI device. The Runtime version of the compiled project is displayed under "Info" in the Inspector window.

12.13.2.3 Compiling a project

Introduction

The changes made to the project are compiled in the background even as you are configuring a project in WinCC. Projects are compiled automatically when you load them. This ensures that the latest version of the project is loaded at all times.

WinCC checks consistency of the project during compilation. The error locations in the project are listed in the Inspector window. You can jump directly to the source of the error from the entry in the Inspector window. Check and correct errors found.

Scope of the compilation

Configuration data is compiled in the background as soon as you start configuring an HMI device. If you compile a project manually, only the changes in the configuration made since the last compilation process are compiled in the background.

You can start complete project compilation manually at any time; this may, for example, be done to test the consistency of the configured data.

Requirement

- A project is open.

Procedure

Proceed as follows to compile a project:

1. If you want to compile several HMI devices at the same time, select all the relevant HMI devices with multiple selection in the project tree.
2. Compile the project:
   - To only compile changes in the project, select the "Compile > Software (only changes)" command from the shortcut menu of the HMI device.
   - To compile all project data, select the "Compile > Software (compile all)" command from the shortcut menu.

Result

The configuration data of all selected HMI devices is compiled. Any errors that occur during compilation are shown in the Inspector window.

12.13.2.4 Simulating projects

Simulation basics

Introduction

You can use the simulator to test the performance of your configuration on the configuration PC. This allows you to quickly locate any logical configuration errors before productive operation.

You can start the simulator as follows:

- In the shortcut menu of the HMI device, or in a screen: "Start simulation"
- Menu command "Online > Simulation > [Start|With tag simulator|With script debugger]"
- Under "Visualization > Simulate device" in the portal view.

Requirements

The simulation/runtime component is installed on the configuration PC.

Field of application

Use the simulator to test various functions of the operator control and monitoring system, such as:

- Checking limit levels and alarm outputs
- Consistency of interrupts
- Configured interrupt simulation
- Configured warnings

- Configured error messages
- Check of status displays
- Interconnection and screen layout

Simulating a project

Introduction

You simulate your project with one of the following two methods:

- Without a connected PLC
  You change the value of area pointers and tags in a tag simulator that is read for the simulation of WinCC Runtime.
- With a connected PLC without a running process
  You simulate your project by running it directly in Runtime. The tags and area pointers become active. This allows you to create an authentic simulation of your configured HMI device in Runtime.

Note

Simulation restrictions

You cannot simulate the following system functions:

- CalibrateTouchScreen

You cannot simulate the Media Player. A static screen appears in the simulation window instead of the Media Player.

File access via scripts is not possible for HMI devices with Windows CE.

Requirement

- Simulation without a connected PLC: Tags have been created
- Simulation with a connected PLC but no active process: A project with tags and area pointers has been created

Procedure

To simulate a project using the tag simulator, follow these steps:

1. Open the project on the configuration PC.
2. Select the "Online > Simulation > With tag simulator" menu command.
   For initial project simulation, the simulator is started with a new, empty table. The project is opened simultaneously in Runtime.
   Toggle between the tag simulator and Runtime using the <Alt +Tab> key combination.
3. To simulate a process value, select the corresponding "tag" from the tag simulator.
   The table lists all configured tags. You can simulate up to 300 tags simultaneously.
4. Select the simulation mode in the "Simulation" column.

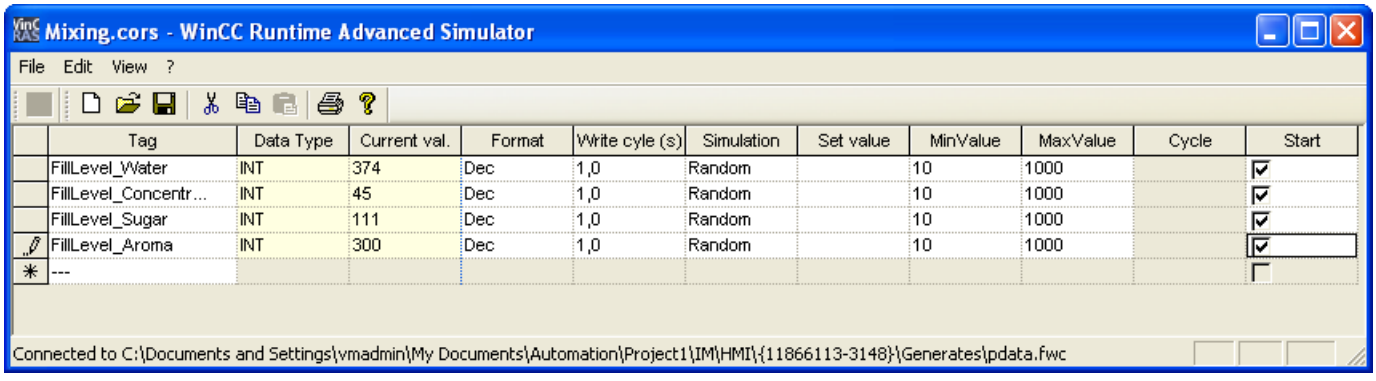
5. Change the value of tags and area pointers in the respective columns.
6. Activate the "Start" check box to start the simulation for this tag.
7. To save the simulation, select the menu command "File > Save" and enter a descriptive name, for example, "Mixing".
   The file name is assigned the extension "*.cors".

Result

The process values are simulated in Runtime. The tag values are created at random, or incremented, depending on the simulation mode.

To specify tag values, change the simulation mode to "<Display>" and enter a value at "Set value".

The following figure shows a tag simulator with four tags whose values can be determined at random in a range of values from 10 to 1000:



Managing simulation data

If you have saved data from a previous simulation, you can open the file at a later point in time and simulate your project again. The tags and area pointers listed in the tag simulator must still be available in the project.

Proceed as follows to open a simulation file:

1. Select the menu command "Online > Simulate Runtime > With tag simulator".
2. Select the menu command "File > Open" in the tag simulator.
3. Select the corresponding simulation file and click "Open".
   The simulator loads the stored data.

Enabling and disabling tags

Start and stop the simulation for each tag separately in order to facilitate the transition from offline to online engineering. Activate "Start" in the corresponding row.

If a tag is activated, the simulation values are calculated and transferred to the WinCC simulator.

Deleting a tag

To delete a tag from the tag simulator, follow these steps:

1. Select the cell that contains the tag name.
2. Select the "Edit > Cut" menu command.
   The tag is removed from the table.

Working with the tag simulator

About the tag simulator

The tag simulator has the following columns:

Column	Description
Tag	Specifies the tags for the simulation.
Data type	Shows the data type of the selected tag.
Current value	Shows the simulated value of the defined tags.
Format	Specifies the selected format in which the tag values are simulated: • Decimal (1, 2, 3, 4, ...) • Hexadecimal (03CE, 01F3, ...) • Binary (0 and 1)
Write cycle	Specifies the selected time interval at which the current tag values are simulated. If you enter "2", for example, the current value of the tag will be shown every 2 seconds.
Simulation	Shows the method by which the tag values are processed during simulation.
Set value	Sets the selected value for the respective tag. The simulation start with the specified value.
minValue maxValue	Specifies the value range of the tag. You set a minimum and maximum value for this range. The default values are -32768 for the minimum and 32767 for the maximum.
Period	Contains the period during which the value of the tag is repeated for the "Increment" and "Decrement" simulation modes.
Start	Starts simulation of the tag based on the previously entered information.

Simulation modes

The simulator has six different simulation modes. The configured tags are supplied with nearly realistic values during the simulation.

Simulation mode	Description
Sinusoidal	Changes the tag value to form a sinusoidal curve. The value is visualized as a periodic, non-linear function.
Random	Provides randomly generated values. The tag value is changed by means of a random function.
Increment	Increases the value of the tag continuously up to a specified maximum value. Begins again at the minimum after the maximum has been reached. The value trend corresponds to a positive saw-tooth curve.

Simulation mode	Description
Decrement	Reduces the value of the tag continuously down to a specified minimum value. Begins again at the maximum after the minimum has been reached. The value curve corresponds to a negative saw-tooth curve.
Shift bit	Shifts a set bit continuously by one position. The previous position is always reset. This lets you test the alarms of an HMI device, for example.
<Display>	The current tag value is displayed statically.

Example: Simulate tags with the "Shift bit" simulation mode

Proceed as follows to simulate tags with the "Shift bit" simulation mode:

1. Open the project you want to simulate.
2. Select the menu command "Online > Simulate Runtime > With tag simulator". The tag simulator opens.
3. In the "Tag" column, select a tag from your project.
4. Select "Bin" in the "Format" column.
5. Enter the value "1" in the "Write cycle" column.
6. Select the "Shift bit" simulation mode in the "Simulation" column.
7. Enter the value "1" in the "Set value" column.
8. Enable the tag with the "Start" check box.

Result

The simulator tests the selected tag bit-by-bit as follows:

Simulation values	Byte for alarms
Set start value	00000001
1. Simulation value	00000010
2. Simulation value	00000100
3. Simulation value	00001000
....	...

In Runtime you see if the desired alarm is output at a given value.

Simulation restrictions

Alarms with dynamic parameters

If you use tags or text lists as external tags for alarms, the dynamic parameters of the alarms are not displayed.

Only internal tags are enabled for the simulation of alarms in the tag simulator .

You can use PLCSim to simulate dynamic parameters.

12.13.2.5 Loading projects

Overview for loading of projects

Overview

Delta data of the project is automatically compiled before you download it to one or several HMI devices. This always ensures that the latest version of the project is transferred.

Loading a project to an HMI device

The following steps are completed prior to downloading:

1. The download settings are verified. The "Extended loading" dialog box is opened automatically during the initial download of a project to an HMI device. You use this dialog to define the protocol and interface or destination path for the project in accordance with the HMI device Runtime used.
   You can call the "Extended loading" dialog at any time with the menu command "Online > Advanced download to device...".
   The "Load preview" dialog opens.
2. The project is compiled. Warnings and errors during compilation are displayed in the Inspector window and in the "Load preview" dialog.
3. The "Load preview" dialog shows you the following information for each HMI device:
   - The individual steps for loading
   - If the image of the target HMI device does not match the image from the configuration, a prompt is displayed asking whether you want to now change the image.

     Notice

     Changing the image deletes all data from the HMI device.

     Data is deleted on the target system if you change the image. For this reason, first back up the following data, if necessary:
     - User administration
     - Recipes
   - Presettings that take effect at loading. You can change the default settings for this download process, if necessary.
   - Warning events (optional). You can download a project while ignoring the "warnings". The functionality may be restricted in runtime.
   - Error events (optional). You cannot load the project. Eliminate the errors and then reload the project.
     WinCC will open the invalid configuration in the corresponding editor if you double-click the error message in the Inspector window. Correct the errors and reload the project.

If you are using HMI tags in your project that are connected to controller tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Loading with S7 routing

Configure the S7 routing settings in the "Devices & Networks" editor in the relevant controller. The settings depend on the device configured.

S7 routing supports the following protocols:

- MPI/PROFIBUS
- Ethernet

Transferring Runtime add-ons

Projects can contain Runtime add-ons in the form of controls or CSP (Communication Support Packages). These Runtime add-ons are automatically transferred with the project.

Loading a project

Introduction

Before a project can run on an HMI device, you must first load it to the HMI device. During loading, you must most importantly specify whether existing data on the HMI device such as "user administration" and "recipe data" is to be overwritten.

If the HMI device supports PROFINET, the name of the HMI device entered in the project tree is used as the device name for the PROFINET communication. The name is written to the HMI device during download. If a device name for the PROFINET communication has already been entered in the HMI device, it will be overwritten.

As a general rule, only one project can be active in Runtime on an HMI device. An HMI device is generally configured to exit Runtime automatically when loading is started. If this is not the case, you will have to exit Runtime manually on the HMI device.

If the image of the target HMI device does not match the image from the configuration, a prompt is displayed asking whether you want to now change the image.

Notice

Changing the image deletes all data from the HMI device.

Data is deleted on the target system if you change the image. For this reason, first back up the following data, if necessary:

- User administration
- Recipes

Controlling the transfer behavior on the HMI device

As a general rule, only one project can be active in Runtime on an HMI device. An HMI device is generally configured to exit Runtime automatically when loading is started. If this is not the case, you will have to exit Runtime manually on the HMI device.

You define how the HMI device reacts when the project is loaded in the "Start Center" under "Settings" on the HMI device:

Transfer mode	Effect
Off	The project cannot be loaded to the HMI device.
Manually	The project can only be loaded to the HMI device if the following requirements are met: • Runtime is not running • The HMI device is in "Transfer" mode.
Automatic	The project can always be loaded to the HMI device. If a transfer is started on the configuration PC and a project is in runtime on the HMI device, the running project is automatically closed. For Mobile Panels, this transfer mode is disabled for security reasons.

Note

Closing Runtime automatically

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the "Start Center".

Please refer to the documentation for the HMI device used for more detailed information on transfer settings.

Requirements

- You have created an HMI device in the project.
- The HMI device is connected to the configuration PC.
- The "Start Center" has been started on the HMI device.
- The protocol by which the project is loaded is set on the HMI device in the "Start Center" under "Settings".
- Transfer mode is set as "Automatically" or "Manually" in the HMI device.

Procedure

Proceed as follows to load a project:

1. To download a project simultaneously to several HMI devices, select the HMI devices by means of multiple selection in the project tree.
2. Select the "Download to device > Software" command from the shortcut menu of an HMI device.

3. If the "Extended loading" dialog is open, configure the "Settings for loading". Make sure that the "Settings for loading" correspond to the "Transfer settings in the HMI device".
   - Select the protocol used, for example, Ethernet or HTTP.
   - Configure the relevant interface parameters on the configuration PC.
   - Make any interface-specific or protocol-specific settings required in the HMI device.
   - Click "Download".

   You can open the "Extended download" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens. The project is compiled at the same time. The result is displayed in the "Load preview" dialog.
4. Check the displayed presettings and change them as necessary.
5. Click "Download".

Result

The project is loaded to all selected HMI devices. Any existing project is replaced. The data for user administration and / or recipes is replaced in accordance with the settings in the "Load preview" dialog.

During the download, you can keep track of the files that are transferred.

If errors or warnings occur during the download, corresponding alarms are displayed under "Info > Load" in the Inspector window.

On completion of the successful download of the project, you can execute it on the HMI device.

Note

If the transfer is interrupted, WinCC automatically ensures that no data is lost and that existing data is deleted on the HMI device only after complete transmission.

12.13.2.6 Runtime start

Starting Runtime on the HMI device

Introduction

On completion of the project download to the HMI device, you can start the project in Runtime. The project is saved in the HMI device to a file with the following extension:

- Basic Panels as well as OP 73, OP 77A and TP 177A: "*.srt"

The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime.

The programs that you can use to start projects on the HMI device are available in the Runtime installation folder.

Note

Closing Runtime automatically

If automatic transfer is activated on the HMI device and a transfer is started on the configuration PC, the running project is automatically terminated.

The HMI device then automatically switches to "Transfer" operating mode.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the "Start Center".

Requirements

- WinCC Runtime is installed on the HMI device.
- The project was downloaded to the HMI device.
- The "Start Center" has been started.

Procedure

The project is stored on a panel in a folder you specify in the HMI device transfer settings. The "Start Center" application is started on a panel. The project loaded is started automatically after expiration of the configured delay.

If the project does not start automatically:

1. Click on "Start" in the "Start Center" to start the loaded project.

Refer to the documentation for the HMI device for additional information on startup of projects.

12.13.2.7 Error messages during loading of projects (Basic)

Possible problems during the download

When a project is being downloaded to the HMI device, status messages regarding the download progress are displayed in the output window.

Usually, problems arising during the download of the project to the HMI device are caused by one of the following errors:

- Incorrect download settings on the HMI device
- Incorrect HMI device type in the project
- The HMI device is not connected to the configuration PC.

The most common download failures and possible causes and remedies are listed below.

The serial download is cancelled

Possible remedy: Select a lower baud rate.

The download is cancelled due to a compatibility conflict

Possible cause	Remedy
The configuration PC is connected to the wrong device, e.g. a controller.	Check the cabling. Correct the communication parameters.

Project download fails

Possible cause	Remedy
Connection to the HMI device cannot be established (alarm in the output window)	Check the physical connection between the configuration PC and the HMI device. Check whether the HMI device is in transfer mode. Exception: Remote control
The default communication driver is not listed in the Windows Device Manager.	Check the device status of the COM connection in the properties window of the Device Manager.

Download over MPI/DP interface fails

Possible cause	Remedy
"Configured mode" is set on the CP, for example, if you are using the SIMATIC NET CD.	Set the CP to "PG mode" using the "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the CP. Set the CP back to "configured mode".
On the programming device/PC panel, the "S7ONLINE" access point is not set to a hardware device such as CP5611 (MPI). The cause may be the installation of "SIMATIC NET CD 7/2001".	Set the access point "S7ONLINE" on the selected device using the "PG/PC Panel" or "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the HMI device. Restore the "S7ONLINE" access point to the original device.

The configuration is too complex

Possible cause	Remedy
The configuration contains too many different objects or options for the HMI device selected.	Remove all objects of a type, e.g. all graphic views.

12.13.2.8 Adapting the project for another HMI device

Introduction

When you download a WinCC project to an HMI device, WinCC checks whether this is compatible with the HMI device type used in the project. If the types of HMI device do not match, you will see a message before the download starts.

The download is aborted.

Adapting the project for the HMI device

You need to adapt the project accordingly to be able to download the project to the connected HMI device.

- Add a new HMI device in the project tree. Select the correct type of HMI device from the HMI device selection.
- Copy the configured components from the previous to the new HMI device.
  Copy large amounts of components directly in the project navigation and details view.
  For example, copy the "Screens" folder to the screens folder of the new HMI device with the help of the shortcut menu.
- Use the detail view to copy entries in the project tree for which the "Copy" command is not available in the shortcut menu.
- Select the "Recipes" entry in the project tree, for example. The recipes are displayed in the detail view.
- Select the recipes in the detail view and drag them to the "Recipes" entry of the new HMI device. The recipes are copied. You can also select multiple objects in the detail view.
- Configure the components that cannot be copied, e.g. connections, area pointers, and alarms.
- Save the project at various points in time.
- Compile the full project.
- When the compilation is successfully completed, download the project to the HMI device.

Linking references

References to linked objects are included in the copying. The references are linked again once the linked objects are copied.

Example:

You copy a screen in which objects are linked to tags. The tag names are entered at the individual objects after the screen is added to the new HMI device. The tag names are marked in red because the references are open. When you then copy the tags and insert them into the new HMI device, the open references are closed. The red marking for the tag names disappears.

For complete references to connected objects in the PLC, you first need to configure a connection to the PLC.

Using the information area

When you compile the project for the HMI device, errors and warnings are displayed in the "Info" tab of the Inspector window. You can use the shortcut menu command "Go to" to go directly to the location where the error or warning can be corrected.

Work through the list of errors and warnings from top to bottom.

When the compilation is successfully completed, download the project to the HMI device.

12.13.2.9 Basics of operating in Runtime

Overview

Overview of operator control of a project

Depending on the HMI device, the following options are available for the input with Basic HMI devices.

- Touch screen
- Function keys
- Mouse and keyboard

Use the touch screen and function keys or mouse and keyboard to operate the Control Panel / Start Center or the project running on your HMI device.

Note

Incorrect operation

A project can contain certain operations that require in-depth knowledge about the specific plant on the part of the operator.
Make sure that your plant is only operated by qualified personnel.

Operating options for an HMI device

Depending on your HMI device, operate your plant as follows:

- Touch screen operation
  The display of the device is touch-sensitive. You operate the operating elements in the display with your finger or a stylus.
- Operation with control keys and function keys
  Control keys and function keys are integrated into the housing of the device.
  - Control keys have a specified function, for example, navigation or the acknowledgment of alarms.
  - Function keys are freely user-assignable and their function is therefore project-specific.
- Mouse and keyboard operation
  Mouse and keyboard are connected via an integrated USB port. You operate the operating objects with mouse and keyboard.

Individually configured operation

The configuration engineer has various options available for setting up operation.

Examples of actions whose execution is always determined on a project-specific basis:

- Screen change
- Reporting
- Changing Runtime language

There are no specific operating elements to execute certain functions. The configuration engineer specifies the project-specific execution. A screen change can be triggered with a button or a function key, for example.

Information on project-specific operations can be found in the system documentation.

Operation with the touch screen

Overview of operation with the touch screen

Use the touch screen to operate the HMI device of the project that is running on your HMI device.

Operating the touch screen

Notice

Damage to the touch screen

Do not touch the touch screen with sharp or pointed objects.

Avoid tapping the touch screen with hard objects, and avoid constantly using motion control.

Both can significantly reduce the useful life or even cause the failure of the touch screen.

Triggering unintended actions

You can trigger unintended actions if you touch several operating elements at the same time. Always touch only one operating element on the screen.

Operating elements are touch-sensitive symbols on the screen of the HMI device.

Special features of operation using the touch screen

Operation with the touch screen is characterized by the following special features:

- Enable
  To enable an operator control, touch the touch screen with your fingers or with a stylus. To generate a double-click, touch the operator control twice in rapid succession.
- Value input
  You enter numbers and letters on the touch screen with a screen keyboard.
- Careful operation
  If you touch multiple operator controls at the same time, you may trigger unintentional actions.

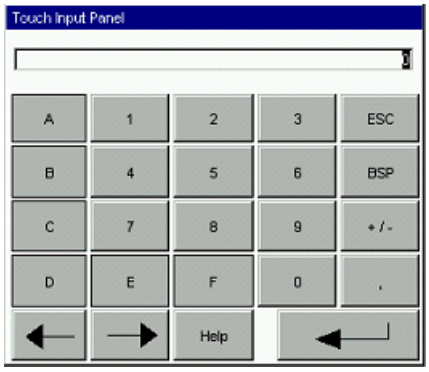
Value input using the screen keyboard

The screen keyboard is displayed as soon as you operate a screen object that requires an input. Depending on the HMI device and the configured operating element, the system displays different screen keyboards for entering numerical or alphanumerical values. The screen keyboard is hidden again when input is complete.

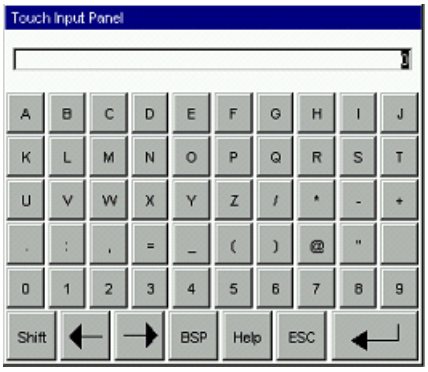
Screen keyboard

Layout

The figure below shows the basic layout of a screen keyboard on a TP1500 Basic.



Numbers



Letters

Operator controls

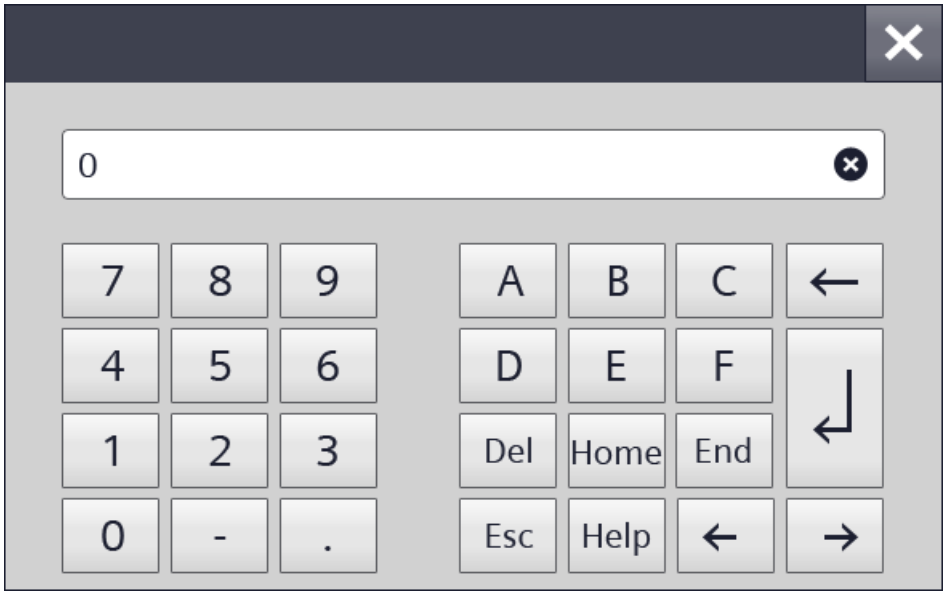
The following keys are available on the screen keyboard of all HMI devices:

Button	Name	Function
	Cursor left	Navigates to the left
	Cursor right	Navigates to the right
	Backspace	Deletes a character
	Escape	Cancels the input
	Enter	Confirms the input
	Help	Displays the infotext This key is only displayed when an infotext has been configured for the operator control.

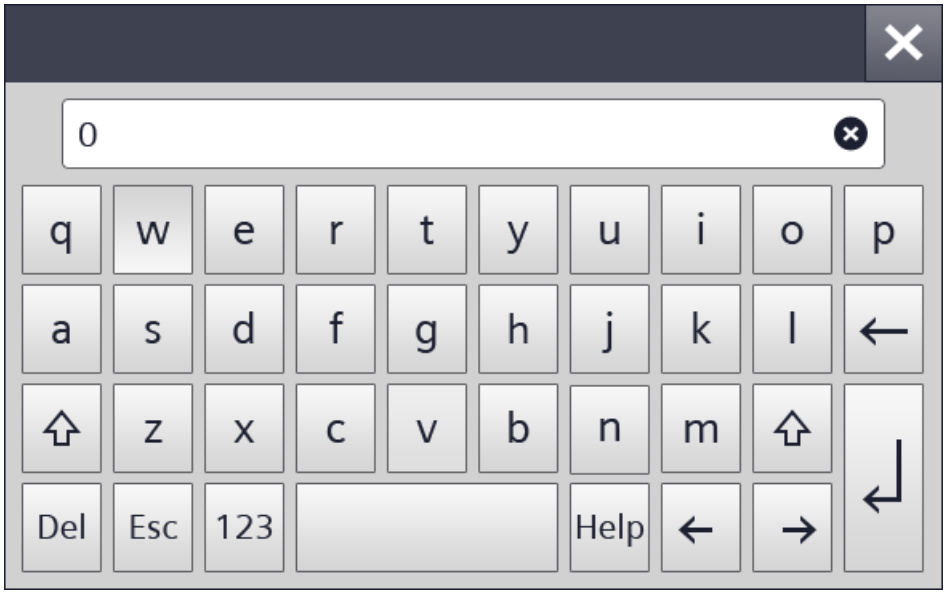
Screen keyboard for 2nd generation Basic Panels

Layout

The figure below shows the basic layout of a screen keyboard on a Basic Panel 2nd Generation.



Numbers



Letters

Operator controls

The following keys are available on the screen keyboard of all HMI devices:

Button	Name	Function
	Cursor left	Navigates to the left
	Cursor right	Navigates to the right
	Backspace	Deletes a character
	Escape	Cancels the input
	Enter	Confirms the input
	Help	Displays the infotext This key is only displayed when an infotext has been configured for the operator control.

Operation with keys

Overview of operation with keys

Introduction

Use the keys of the HMI device to operate the Control Panel / Start Center of your HMI device or the project that is running on your device. Depending on the device, control keys and function keys are available.

For more detailed information, refer to the operating instructions for your HMI device.

Control keys and shortcuts

Introduction

The following tables list the control keys available to operate the project.

Note

The availability of control keys is determined by the HMI device used.

You trigger functions on keyboard HMI devices either with a key or a shortcut. With shortcuts, you keep the first key pressed. Then you press the second key.

Navigating in the display

Key or shortcut		Function	Description
TAB		Tabulator	Selects the next operating element in the tab sequence
SHIFT	TAB	Tabulator	Selects the previous operating element in the tab sequence
◀ ▶ ▼ ▲		Cursor keys	Selects the next operating element to the left, to the right, above or below the current screen object Navigating in the operating element

Operation of operating elements

Key or shortcut		Function	Description
ENTER		ENTER key	• Controls buttons. • Applies and ends an entry • Opens a selection list • Toggles between character mode and standard mode within an input field A single character is selected in character mode. In this mode, you can scroll within the character set using the cursor keys.
SHIFT	▶ ◀ ▲ ▼	Positioning cursor	Positioning the cursor within an operating element, for example, in an I/O field
←		Delete characters	Deletes the character to the left of the current cursor position
INS DEL		Delete characters	Deletes the next character to the right of the current cursor position
ESC		Cancel	• Deletes the input characters of a value and restores the original value • Closes the active dialog.
F N	HOME	Scroll to start	Scrolls to the start page of a list
HOME		Scrolling back	Scrolls the list back by one page
F N	END	Scroll to end	Scrolls to the end of a list.
END		Scrolling up	Scrolls the list up by one page
ALT	▼	Open selection list	Opens a selection list
CTRL	ENTER	Accept value	Accepts the value selected in the selection list without closing the list

Enter shortcut

Key	Function	Purpose
A-Z	Toggle (numbers/letters)	Toggles the assignment from numbers to letters • No LED is lit: The number assignment is enabled. Pressing the key once switches to letter assignment. • The right or left LED is lit: The left or right letter assignment is enabled. Each time the key is pressed, the system toggles between the left letter assignment, the right letter assignment and the number assignment.
SHIFT	Shift (upper/lower case)	Used in shortcuts, for example, for switching to uppercase letters
F_N	Toggle to additional keyboard layout	Some of the keys contain a blue special character in their bottom left corner, for example, the percent sign "%". To input these characters, press the relevant key in combination with the special character key shown on the left.
CTRL	General control function	Used in shortcuts, for example, for navigating trend views
ALT	General control function	Used in shortcuts, for example, for the "Status/Force" screen object

Acknowledge alarms

Key	Function	Purpose
ACK	Acknowledge	Acknowledges the currently displayed alarm or all alarms of an alarm group The LED is lit as long as an unacknowledged alarm is active.

Displaying infotext

Key	Function	Description
HELP	Displaying infotext	For the selected object, opens a window with the configured infotext, for example, for an alarm or an I/O field The LED is lit if an infotext is available for the selected object.

Key or shortcut

Function Keys

The assignment of the function keys (F1, F2, F3, etc.) is defined during configuration.

Function keys with global function assignment

A globally assigned function key always triggers the same action on the HMI device or in the PLC regardless of the screen displayed. The activation of a screen or the closing an alarm window, for example, is such an action.

Function keys with local function assignment

A function key with local function assignment is screen-specific and is therefore only effective within the active screen.

The function of a locally assigned function key can vary from screen to screen.

Within a screen, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Operating function keys

Note

Operating function key after screen change

If you press a function key after a screen change, the associated function may be triggered in the new screen before the new screen is fully displayed.

Navigating in the display (BS)

Introduction

You navigate on the display of your HMI device as follows:

- between configured screen objects
- within screen objects
  When you select a complex screen object, the cursor focus switches to the screen object and follows the tab sequence there.
- in tables of screen objects

Procedure

- To navigate in the specified tab sequence, press the <TAB> key.
- To navigate freely between the operator controls, press the cursor keys.

Depending on the configuration of your HMI device, you can also use function keys or shortcuts for navigation.

When you operate your HMI device with the touch screen or with the mouse, you implicitly navigate by triggering a desired action. For this purpose, touch or click the operator control.

Result

The operator controls receive the cursor focus according to the selected sequence. You can trigger an action on the selected operator control.

For more detailed information, refer to the operating instructions for your HMI device.

Triggering an action

Introduction

Triggering an action at an operator control can mean the following:

- A command is executed.
  Example: Click a button to trigger a script or to execute a predefined function.
- An object is enabled.
  Example: To enter a value, select a table cell with the <Enter> key.

Requirement

- You have navigated to the operator control on which you want to trigger the action.
- The operator control has the cursor focus.

Procedure

- Press <Enter>.
  Or
- Touch the operator control on the touch screen once or twice in rapid succession.
  Or
- Click or double-click the operator control with the mouse.

Result

The following results are possible:

- The requested command is executed.
- The screen keyboard is opened and/or the cursor blinks in the input area of the operator control.
- The element is selected and can be moved.

For more detailed information, refer to the operating instructions for your HMI device.

Entering a value

Introduction

Depending on the input format, you enter numerical or alphanumerical values in an input field.

You enter these values depending on the existing hardware using the screen keyboard, the control keys of the HMI device or an external keyboard.

Requirement

- The object is an input field or table field.
- The operator control is enabled.

Entering a value

1. Enter the desired value.
2. To confirm the value and exit the field, press the <Enter> key.
3. To discard the value and exit the field, press the <Esc> key.

Result

A value is entered or discarded. You navigate as needed to the next operator control.

For more detailed information, refer to the operating instructions for your HMI device.

Moving operator controls

Introduction

In Runtime, you can move the movable operator controls of a screen object with the mouse or using the touch screen, for example, a slider or a scroll bar. Operation with the keyboard is described below.

Requirement

- A movable operator control is selected.

Procedure

- To move the operator control, proceed as follows depending on the operating element:
  - Standard for touch screen: Press the cursor keys.
  - Standard for keyboard devices: Press <SHIFT> and the cursor keys.
  - Switches: Press <ENTER>
  - Slider: Press <PgUp> or <PgDn>

1. To finish the movement, navigate to another screen object or operator control.

Slider procedure

1. To move the operator control, press the cursor keys.
2. To finish the movement, navigate to another screen object or operator control.

Result

The position of the movable operator control and the display in the screen object have changed.

For more detailed information, refer to the operating instructions for your HMI device.

Displaying infotext

Introduction

Depending on the configuration, additional information and operating instructions are available as infotext. The infotext is assigned to an operating element, an alarm or to the open screen. The infotext of an I/O field may contain, for example, information about the value to be entered.

As an alternative to the <Help> key of the HMI device, use the <Help> key of the screen keyboard for input objects.

Requirement

- An infotext is configured for the operating element, the screen or an alarm.

Calling the infotext

1. Enable the desired operating element.
2. Press the <Help> key of the HMI device.
   The infotext for the operating element is displayed.

If you are operating your input object with the touch screen, the screen keyboard opens. If the <Help> key appears, an infotext is configured for the operating element or the current screen.

If there is no infotext for the selected screen object, the infotext for the current screen is displayed, if it has been configured.

Use the scroll bar for long infotexts.

Depending on your configuration, infotext can also be retrieved by means of a configured operating element.

Switching between infotexts

- To switch between the infotexts of the operating elements and the screen, enable the infotext window.

Hiding infotext

- To hide the infotext, press the <Esc> key or press the <Help> key again.

Changing Runtime language

Introduction

The HMI device supports multilingual projects. A corresponding operating element which lets you change the language setting on the HMI device in Runtime has been configured.

The project always starts with the language set in the previous session.

Requirement

- The required language for the project is available on the HMI device.
- The language switching function is linked to an operating element, for example, to a button.

Selecting a language

You can change project languages at any time. Language-specific objects are immediately displayed on the screen in the new language when you switch languages.

You can switch the language in Runtime in one of the following ways:

- Use a configured operating element to switch from one language to the next in a list.
- Use a configured operating element to directly set the required language.

12.13.3 Runtime Advanced and Panels

12.13.3.1 Runtime settings

Runtime settings

Introduction

In the Runtime settings, you enable the options for user administration and other optional functionalities. You also specify which data are included in the transfer to the HMI device.

To edit the Runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree. The options available for selection depend on the HMI device.

Display on the target system

In WinCC, configure the visual representation of the generated project in Runtime:

- Select the full-screen or window mode for the project. The window has a smaller size than the screen. In full-screen mode, the project is zoomed to the full screen. There will be no window and operator controls for this view.
  Click "Screens" in the "Settings" editor to activate the full-screen display at startup. Activate the "Full-screen mode" check box in the "Screen" field.

  Note

  Note: Truncated view

  If the HMI screen does not match the configured size (in pixels), the project opened in full-screen mode only appears on a part of the screen.

- You can hide the taskbar in Windows as required. To do this, select "Start > Settings > Taskbar" and then clear the "Always on top" and "Auto hide" check boxes in the "Taskbar properties" dialog.
- Under "General > Screen", you specify whether the HMI device uses the default style of the project or another predefined style.

Settings for transfer of additional data to the HMI device

- Transferring names of the screen objects in scripts
  If you enable the "Load names" option under "General > Screen", the object names of the screen objects are transferred instead of coded addressing information. You need the object names, if you want to address the screen objects in a script using the object names. When you test a script in the debugger, the code becomes more comprehensible by displaying the object names.
- Displaying comments in scripts
  Under "Screens > Display", you specify whether the comments in scripts are also transferred to the HMI device.
  If you do not transfer the comments, you save storage space on your HMI device.
  If you transfer the comments, the script is more easily understood during testing in the debugger.

User administration settings

Under "User management" you specify the settings for password-protected access in Runtime. You can find additional information in the online help for WinCC under "Configuring user administration".

Alarm settings

Under "Alarms" you specify the layout and properties of alarms. You can find additional information in the online help for WinCC under "Working with alarms".

Settings for compatibility check with the PLC

The "Project ID" area pointer is used to configure the project ID for checking the consistency of the project with the PLC. The area pointer reads the project version from the PLC. You can find additional information in the online help for WinCC under "Communicating with PLCs".

You select the project ID in the Runtime settings of the HMI device under "General > Identification > Project ID".

At startup, Runtime checks whether the project ID of the project matches the PLC project version. Runtime will only start if the two values match.

The project ID is not available on all HMI devices.

Settings for WinCC options

- Remote control
  Under "Services", you configure the requirements for using the Sm@rt options. You can find additional information on this subject in the online help for WinCC under "Options > Sm@rt options".
- GMP
  Under "GMP", you enable GMP-compliant configuration. You can find additional information on this subject in the online help for WinCC under "Options".

OPC settings

Under OPC settings, you specify the security policies and the security mode of the alarm for OPC UA. You also enter the port number that will be used in the server's URL.

Settings for tags

Under "Settings for tags", you configure the settings for name synchronization of the PLC tags depending on the requirements of your project. You can find additional information on this subject in the online help for WinCC under "Synchronization settings".

Configuring display in Runtime

Introduction

In the Runtime settings you specify how screens, alarms and language are displayed.

To edit the Runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree. The options available for selection depend on the HMI device.

Screen settings

In the Runtime settings, specify the start screen and default template. You can find additional information in the online help for WinCC under "Working with screens".

Language settings

- Logging language
  Under "General > Logs", you specify the language in which you want to write the logs. The selected logging language always remains the same, even if the user switches the language in Runtime.
  If you select "Startup language", the logging language remains the same until Runtime is stopped, even if the user has switched the language. However, the next time Runtime is started, the last configured Runtime language is used as the logging language.
- Runtime language
  Under "Language & font", you specify all Runtime languages and their display.

Setting for specific screen objects

- Specify bit selection
  Under "Screens > Bit selection", you specify whether bit selection will be used on this HMI device for the following:
  - Text and graphics (text lists and graphic lists)
  - Color and flashing (appearance analysis)
    When you enable an option, the selection that is configured in the set bit with the least significance is displayed:
    When you disable an option and several bits are set, the selection that has been configured only for the set bit is displayed.
- Tooltips for I/O fields
  Under "Screens > Display", you specify whether configured limit values are displayed as tooltips in Runtime when values are entered in I/O fields.
- Colored objects
  Under "General > Screens > Color depth" you specify the color depth of your HMI device. As needed, you assign screen objects a colored appearance. The range of colors is determined by the color depth supported on your selected HMI device.

Configuring operation in Runtime

Introduction

In the Runtime settings you specify how screens are operated, and how you use keys for operation in Runtime.

To edit the Runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree. The options available for selection depend on the HMI device.

Locking task switching

Depending on the HMI device, you can lock task switching on the HMI device. This will prevent the operator from opening other applications in Runtime.

In the Runtime settings of the HMI device, enable the "Lock task switching" option under "General > Screen".

Note

Stop Runtime

If you lock program switching, always configure in your project the system function "StopRuntime" for an object, e.g. a softkey or button.

Operation using function keys

To symbolize the function of a function key for keyboard devices, you configure a graphic next to the function key in the display. Under "Screens > Function keys > User-defined pictogram size", you specify whether you want to use a non-standard graphic size.

Operation with keyboard

Under "Keyboard > General > Use screen keyboard", you specify whether the screen keyboard is available on your HMI device.

Under "Keyboard > General > Release button on exit", you specify whether the "Release" event is triggered when the user exits a button without releasing it.

Under "Keyboard > Disable function keys in dialogs", you specify for keyboard HMI devices whether the function keys are disabled for the duration of the displayed dialog. Large dialogs can hide areas or graphics that border the function keys or describe the function keys. The functions keys may therefore be unintentionally pressed.

Setting time base

Setting the time base for the time of day

You set the time in the Control Panel of your HMI device. For more detailed information, refer to the operating instructions for your HMI device.

Synchronize date and time with the PLC

You can find additional information on this subject in the online help for WinCC under "Communicating with the PLC".

Printing in Runtime

Print functions

Print functions available in online mode:

- Hardcopy
  You print out the currently displayed screen by means of an operator control that triggers the "PrintScreen" system function.
- Printing alarms
  Every alarm that has occurred and its state changes are reported along on a printer.
- Printing reports
  Reports are output in graphic mode. The use of a serial printer is not recommended because of the accumulated data volume.
  For proper output, the printer must support the paper format and page layout of the report.

Note

The value of a tag in the report is read and output at the moment of printing. A substantial time may elapse between printing out the first and the last page of a report consisting of several pages. This may lead to the same tag on the last page being output with a different value from that on the first page.

12.13.3.2 Overview of compiling and loading projects

Overview

The project is compiled in the background even as you are configuring it in WinCC. This reduces the time for final compilation. When you start compilation, you create a file that can be run on the corresponding HMI device.

If an error occurs during compilation, WinCC provides support in locating and correcting it.

Once you have corrected any problems, you download the compiled project to the HMI devices on which the project is to run. If the configuration PC is not connected to the HMI device, save the compiled project on a data medium of your choice. The compiled project is then transferred from a PC connected to the HMI device to the HMI device.

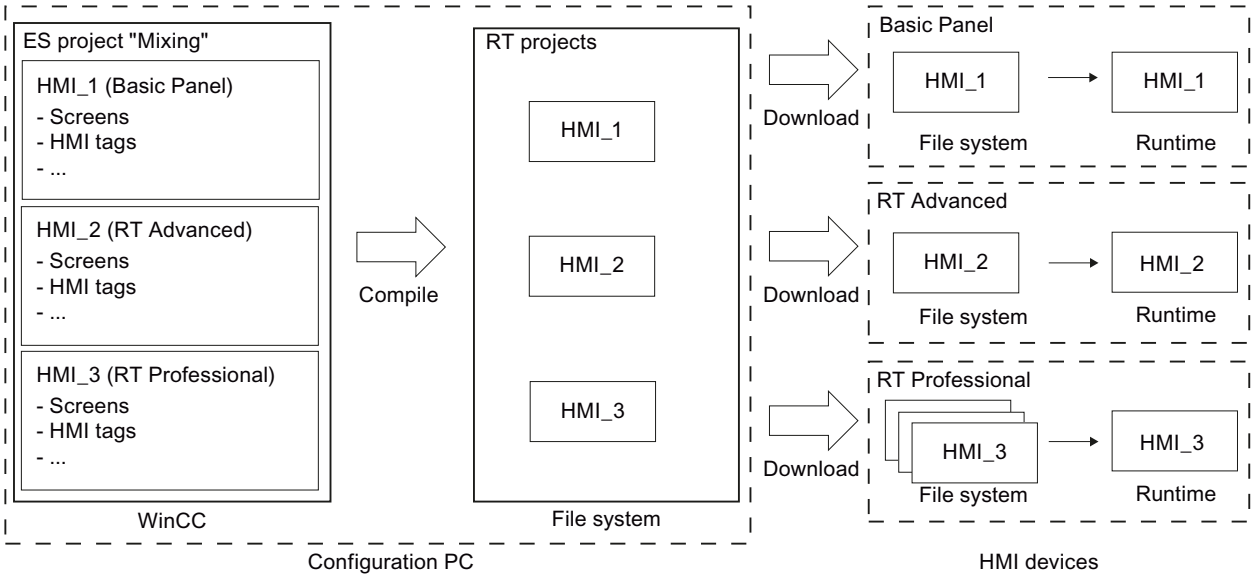
If you are using HMI tags in your project that are connected to PLC tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Project

The term "project" has two different meanings in the contexts of compilation and loading. "Project" is the WinCC project on the configuration PC. "Project" is also the Runtime project you create by compiling the configuration data of an HMI device and download to the HMI device.

- WinCC project: contains the configuration data of one or more HMI devices
- Runtime project: contains the compiled configuration data of an HMI device

The figure below illustrates the link between WinCC projects and Runtime projects using the example of the "Compile and load" process:



Runtime

Runtime is the software for process visualization. In Runtime, you execute the project in process mode.

A distinction is made between two types of Runtime:

1. Runtime on a panel
   Before running a Runtime project on a panel, you have to transfer the Runtime project to the panel before startup.
2. Runtime on a PC
   You can execute the Runtime project directly on the configuration PC if Runtime has been installed on the configuration PC.
   If you want to execute the Runtime project on a different PC, you have to transfer the Runtime project to the PC before startup.

Runtime version

The Runtime version depends on the image of the configured HMI device. The Runtime version of the compiled project is displayed under "Info" in the Inspector window.

Simulation

You test your configuration with a simulation. You can start a simulation without a link to the active process.

In a simulation, you test configured tags or screen changes, for example. During the simulation, the configured tags can be manipulated, activated and deactivated with the help of the tag simulator.

There are two types of simulation:

1. Simulating a panel
   If you created a panel in your project, the panel is displayed in the simulation. With the help of this type of simulation, you can test your configuration on the HMI device without transferring the project to the panel.
2. Simulating Runtime
   Simulating Runtime allows you to test the project directly on the configuration PC.

See also

Generating a "Pack&Go" file (Page 6945)

Starting Runtime Advanced and Panel Runtime (Page 6953)

12.13.3.3 Compiling a project

Introduction

The changes made to the project are compiled in the background even as you are configuring a project in WinCC. Projects are compiled automatically when you load them. This ensures that the latest version of the project is loaded at all times.

WinCC checks consistency of the project during compilation. The error locations in the project are listed in the Inspector window. You can jump directly to the source of the error from the entry in the Inspector window. Check and correct errors found.

Scope of the compilation

Configuration data is compiled in the background as soon as you start configuring an HMI device. If you compile a project manually, only the changes in the configuration made since the last compilation process are compiled in the background.

You can start complete project compilation manually at any time; this may, for example, be done to test the consistency of the configured data.

Requirement

- A project is open.

Procedure

Proceed as follows to compile a project:

1. If you want to compile several HMI devices at the same time, select all the relevant HMI devices with multiple selection in the project tree.
2. Compile the project:
   - To only compile changes in the project, select the "Compile > Software (only changes)" command from the shortcut menu of the HMI device.
   - To compile all project data, select the "Compile > Software (compile all)" command from the shortcut menu.

Result

The configuration data of all selected HMI devices is compiled. Any errors that occur during compilation are shown in the Inspector window.

12.13.3.4 Simulating projects

Simulation basics

Introduction

You can use the simulator to test the performance of your configuration on the configuration PC. This allows you to quickly locate any logical configuration errors before productive operation.

You can start the simulator as follows:

- In the shortcut menu of the HMI device or in a screen: "Start simulation"
- Menu command "Online > Simulation > [Start|With tag simulator|With script debugger]"
- Under "Visualization > Simulate device" in the portal view.

Requirement

The simulation/runtime component is installed on the configuration PC.

Field of application

You can use the simulator to test the following functions of the HMI system, for example:

- Checking limit levels and alarm outputs
- Consistency of interrupts
- Configured interrupt simulation
- Configured warnings

- Configured error messages
- Check of status displays

See also

Simulating a screen (Page 6934)

WinCC Runtime Advanced simulation (Page 6931)

Working with the tag simulator (Page 6934)

WinCC Runtime Advanced simulation

Introduction

There are two different simulation modes for Runtime Advanced simulation:

- Device simulation
- Tag simulation

Both types of simulation simulate the project without a direct process link to the configuration PC. Data such as logs or recipes generated during simulation are not deleted. This data is saved on the configuration PC in the paths configured in the project.

Device simulation

Use device simulation to simulate operator control of the HMI device. Device simulation is, for example, used to test screen switching.

Tag simulation

Use tag simulation to simulate the configured process tags. You can either have tag values generated automatically by a simulation table or define tag values yourself.

See also

Simulation basics (Page 6928)

Simulating a project (Page 6932)

Start debugger (Page 6936)

Simulating a project

Introduction

You simulate your project with one of the following two methods:

- Without a connected PLC
  You change the value of area pointers and tags in a tag simulator that is read for the simulation of WinCC Runtime.
- With a connected PLC without a running process
  You simulate your project by running it directly in Runtime. The tags and area pointers become active. This allows you to create an authentic simulation of your configured HMI device in Runtime.

Note

Simulation restrictions

You cannot simulate the following system functions:

- CalibrateTouchScreen

You cannot simulate the Media Player. A static screen appears in the simulation window instead of the Media Player.

File access via scripts is not possible for HMI devices with Windows CE.

Requirement

- Simulation without a connected PLC: Tags have been created
- Simulation with a connected PLC but no active process: A project with tags and area pointers has been created

Procedure

To simulate a project using the tag simulator, follow these steps:

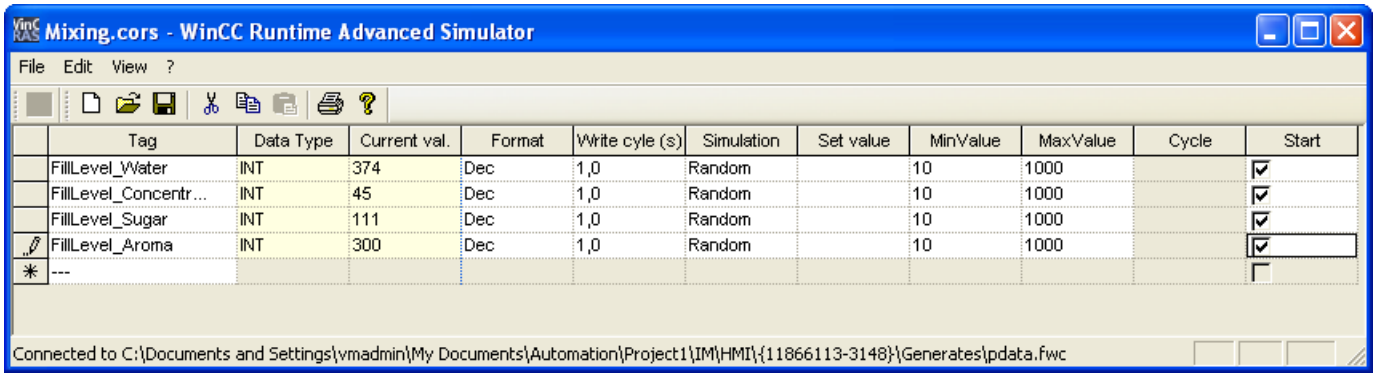
1. Open the project on the configuration PC.
2. Select the "Online > Simulation > With tag simulator" menu command.
   For initial project simulation, the simulator is started with a new, empty table. The project is opened simultaneously in Runtime.
   Toggle between the tag simulator and Runtime using the <Alt +Tab> key combination.
3. To simulate a process value, select the corresponding "tag" from the tag simulator.
   The table lists all configured tags. You can simulate up to 300 tags simultaneously.
4. Select the simulation mode in the "Simulation" column.
5. Change the value of tags and area pointers in the respective columns.
6. Activate the "Start" check box to start the simulation for this tag.
7. To save the simulation, select the menu command "File > Save" and enter a descriptive name, for example, "Mixing".
   The file name is assigned the extension "*.cors".

Result

The process values are simulated in Runtime. The tag values are created at random, or incremented, depending on the simulation mode.

To specify tag values, change the simulation mode to "<Display>" and enter a value at "Set value".

The following figure shows a tag simulator with four tags whose values can be determined at random in a range of values from 10 to 1000:



Managing simulation data

If you have saved data from a previous simulation, you can open the file at a later point in time and simulate your project again. The tags and area pointers listed in the tag simulator must still be available in the project.

Proceed as follows to open a simulation file:

1. Select the menu command "Online > Simulate Runtime > With tag simulator".
2. Select the menu command "File > Open" in the tag simulator.
3. Select the corresponding simulation file and click "Open".
   The simulator loads the stored data.

Enabling and disabling tags

Start and stop the simulation for each tag separately in order to facilitate the transition from offline to online engineering. Activate "Start" in the corresponding row.

If a tag is activated, the simulation values are calculated and transferred to the WinCC simulator.

Deleting a tag

To delete a tag from the tag simulator, follow these steps:

1. Select the cell that contains the tag name.
2. Select the "Edit > Cut" menu command.
   The tag is removed from the table.

Simulating a screen

Introduction

If you have only made changes to one screen, you can temporarily specify this screen as the start screen for simulation. In this way, you can debug changes without having to modify the start screen, or opening the screen on the HMI device.

Requirements

You created a project that contains at least one screen.

Procedure

To define a screen as temporary start screen for simulation, follow these steps:

1. In the project navigation, select the image to display as the start screen in the simulation.
2. Select the "Start simulation" command from the shortcut menu of the screen.

Result

Project simulation is started. Instead of the configured start screen, the simulation window the screen you selected in the project navigation.

See also

Simulation basics (Page 6928)

Working with the tag simulator

About the tag simulator

The tag simulator has the following columns:

Column	Description
Tag	Specifies the tags for the simulation.
Data type	Shows the data type of the selected tag.
Current value	Shows the simulated value of the defined tags.
Format	Specifies the selected format in which the tag values are simulated: • Decimal (1, 2, 3, 4, ...) • Hexadecimal (03CE, 01F3, ...) • Binary (0 and 1)
Write cycle	Specifies the selected time interval at which the current tag values are simulated. If you enter "2", for example, the current value of the tag will be shown every 2 seconds.
Simulation	Shows the method by which the tag values are processed during simulation.

Column	Description
Set value	Sets the selected value for the respective tag. The simulation start with the specified value.
minValue maxValue	Specifies the value range of the tag. You set a minimum and maximum value for this range. The default values are -32768 for the minimum and 32767 for the maximum.
Period	Contains the period during which the value of the tag is repeated for the "Increment" and "Decrement" simulation modes.
Start	Starts simulation of the tag based on the previously entered information.

Simulation modes

The simulator has six different simulation modes. The configured tags are supplied with nearly realistic values during the simulation.

Simulation mode	Description
Sinusoidal	Changes the tag value to form a sinusoidal curve. The value is visualized as a periodic, non-linear function.
Random	Provides randomly generated values. The tag value is changed by means of a random function.
Increment	Increases the value of the tag continuously up to a specified maximum value. Begins again at the minimum after the maximum has been reached. The value trend corresponds to a positive saw-tooth curve.
Decrement	Reduces the value of the tag continuously down to a specified minimum value. Begins again at the maximum after the minimum has been reached. The value curve corresponds to a negative saw-tooth curve.
Shift bit	Shifts a set bit continuously by one position. The previous position is always reset. This lets you test the alarms of an HMI device, for example.
<Display>	The current tag value is displayed statically.

Example: Simulate tags with the "Shift bit" simulation mode

Proceed as follows to simulate tags with the "Shift bit" simulation mode:

1. Open the project you want to simulate.
2. Select the menu command "Online > Simulate Runtime > With tag simulator".
   The tag simulator opens.
3. In the "Tag" column, select a tag from your project.
4. Select "Bin" in the "Format" column.
5. Enter the value "1" in the "Write cycle" column.
6. Select the "Shift bit" simulation mode in the "Simulation" column.
7. Enter the value "1" in the "Set value" column.
8. Enable the tag with the "Start" check box.

Result

The simulator tests the selected tag bit-by-bit as follows:

Simulation values	Byte for alarms
Set start value	00000001
1. Simulation value	00000010
2. Simulation value	00000100
3. Simulation value	00001000
....	...

In Runtime you see if the desired alarm is output at a given value.

Simulation restrictions

Alarms with dynamic parameters

If you use tags or text lists as external tags for alarms, the dynamic parameters of the alarms are not displayed.

Only internal tags are enabled for the simulation of alarms in the tag simulator .

You can use PLCSim to simulate dynamic parameters.

Start debugger

Introduction

Select the "With script debugger" script mode to test scripts. The debugger allows you to set breakpoints in the code, for example, or implement a script step by step.

Requirements

- A debugger that supports VBS is installed on the Engineering Station, e.g. "MS Script Debugger".
- WinCC Runtime is installed on the Engineering Station.
- A project is open.

Procedure

1. Select the "Online > Simulation > With script debugger" menu command.
   The Runtime software searches for debuggers installed on the Engineering Station.
2. If several debuggers are found, click on a selected.

Result

The debugger is connected automatically with the runtime software.

12.13.3.5 Loading projects

Overview for loading of projects

Overview

Delta data of the project is automatically compiled before you download it to one or several HMI devices. This always ensures that the latest version of the project is transferred. When you execute the command "Load (complete)" in Runtime Professional, you need to confirm overwriting of the project.

Loading a project to an HMI device

The following steps are completed prior to downloading:

1. The download settings are verified. The "Extended download to device" dialog box opens automatically during the initial download of a project to an HMI device. You use this dialog to define the protocol and interface or destination path for the project in accordance with the HMI device Runtime used.
   If the HMI device is part of a subnet, for example, you also select the subnet and the first gateway.
   You can open the "Extended download to device" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens.
2. The project is compiled. Warnings and errors during compilation are displayed in the Inspector window and in the "Load preview" dialog.
3. The "Load preview" dialog shows you the following information for each HMI device:
   - The individual steps for loading
   - If the device version of the target HMI device does not match the configured device version, you are queried if you want to change the device version at this time.
     As long as the project with the selected device version of the target HMI device is not executable, you cannot start the download.

     Notice

     Changing the device version deletes all data on the HMI device (does not apply to Runtime Professional V13 SP1)

     Data is deleted on the target system if you change the device version. For this reason, you should first back up the following data:
     - User administration
     - Recipes
     - Licenses

     Resetting to factory settings also deletes the license keys. Back up the license keys before you reset the system to factory settings.

     With Comfort devices, the licenses can only be deleted when resetting to factory settings. Back up the licenses and license keys before you reset the system to factory settings.

   - Presettings that take effect at loading. You can change the default settings for this download process, if necessary.
   - Warning events (optional). You can download a project while ignoring the "warnings". The functionality may be restricted in runtime.

– Error events (optional). You cannot load the project. Eliminate the errors and then reload the project.
WinCC will open the invalid configuration in the corresponding editor if you double-click the error message in the Inspector window. Correct the errors and reload the project.

Note

In Runtime Professional, you can load a project despite errors. In the "Load preview" dialog under "Actions", select the check box "The project has not been compiled without errors. Do you want to continue loading?" and click "Load".

If you are using HMI tags in your project that are connected to PLC tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Note

Does not apply for RT Professional

If the transfer is interrupted, WinCC V13 automatically ensures that no data is lost and that existing data is deleted on the HMI device only after complete transmission.

Loading a project without a connected HMI device

If you cannot establish a direct connection from the configuration PC to the HMI device, save the compiled project on a data medium of your choice. Additional steps depend on the Runtime used:

- Panel Runtime: Copy the compiled project to a PC connected to the HMI device, for example, via a network. Download the project from this PC to the HMI device ("Pack&Go").
- Runtime Advanced: Move the compiled project to the HMI device.
- Runtime Professional: You download the compiled project to a USB stick and copy it on the HMI device via Windows Explorer into the corresponding directory.

Loading with S7 routing

Configure the S7 routing settings in the "Devices & Networks" editor in the relevant PLC. The settings depend on the device configured.

S7 routing supports the following protocols:

- MPI/PROFIBUS
- PN/IE

Transferring Runtime add-ons in WinCC V13

Projects can contain Runtime add-ons in the form of controls or CSP (Communication Support Packages). These Runtime add-ons are automatically transferred with the project.

See also

Loading a project (Page 6940)

Generating a "Pack&Go" file (Page 6945)

Loading a project

Introduction

Before a project can run on an HMI device, you must first load it to the HMI device. During loading, you must most importantly specify whether existing data on the HMI device such as "user administration" and "recipe data" is to be overwritten.

If the HMI device supports PROFINET, the name registered in the project tree is used for the PROFINET communication. The use of the name corresponds to the default settings of the PROFINET interface of the HMI device. For devices with more than one PROFINET interface, the name of the IE CP is automatically added to the device name with a separating period. The name is written to the HMI device during download. If a device name for the PROFINET communication has already been entered in the HMI device, it will be overwritten.

You can find additional information about these settings in the information system in the "Assigning a device name and IP address" section.

If the device version of the target HMI device does not match the configured device version, you are asked whether you wish to change the device version at this time.

Note

If your HMI device is a PC, the device version of the target system is not automatically updated during the download.

Verify that the configured device version corresponds to the one of the target HMI device before you compile and download your project. If necessary, install the matching Runtime version on your HMI device or change the device version manually using the properties of the HMI device.

Notice

Changing the device version deletes all data on the HMI device.

Data is deleted on the target system if you change the device version. For this reason, you should first back up the following data:

- User administration
- Recipes
- Licenses

Resetting to factory settings also deletes the license keys. Back up the license keys before you reset the system to factory settings.

With Comfort devices, the licenses can only be deleted when resetting to factory settings. Back up the licenses and license keys before you reset the system to factory settings.

Note

Transfer to an HMI device with WinAC MP

The "PN/IE" channel is not approved for the project transfer on HMI devices with WinAC MP.

Use the "Ethernet" channel instead of "PN/IE" to transfer data to an HMI device that is running PLC WinAC MP.

Note

Transfer to an HMI device with PC station V1.0 or PC station V2.0

The version of the PC station set in the device settings of the HMI device must match the version installed on the HMI device.

Additional information on configuration settings for "PC Station" on the HMI device is available in the "SIMATIC NET" documentation.

Controlling the transfer behavior on the HMI device

As a general rule, only one project can be active in Runtime on an HMI device. An HMI device is generally configured to exit Runtime automatically when loading is started. If this is not the case, you will have to exit Runtime manually on the HMI device.

You define how the HMI device reacts when the project is loaded in the "Start Center" under "Settings" on the HMI device:

Transfer mode	Effect
Off	The project cannot be loaded to the HMI device.
Manually	The project can only be loaded to the HMI device if the following requirements are met: • Runtime is not running • The HMI device is in "Transfer" mode.
Automatic	The project can always be loaded to the HMI device. If a transfer is started on the configuration PC and a project is in runtime on the HMI device, the running project is automatically closed. For Mobile Panels, this transfer mode is disabled for security reasons.

Note

Closing Runtime automatically

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the "Start Center".

Please refer to the documentation for the HMI device used for more detailed information on transfer settings.

Requirements

- You have created an HMI device in the project.
- The HMI device is connected to the configuration PC.
- The "Start Center" has been started on the HMI device.
- The protocol by which the project is loaded is set on the HMI device in the "Start Center" under "Settings".
- Transfer mode is set as "Automatically" or "Manually" in the HMI device.

Procedure

Proceed as follows to load a project:

1. To download a project simultaneously to several HMI devices, select the HMI devices by means of multiple selection in the project tree.
2. Select the "Download to device > Software" command from the shortcut menu of an HMI device.
3. If the "Extended loading" dialog is open, configure the "Settings for loading". Make sure that the "Settings for loading" correspond to the "Transfer settings in the HMI device".
   - Select the protocol used, for example, Ethernet or HTTP.
   - Configure the relevant interface parameters on the configuration PC.
   - Make any interface-specific or protocol-specific settings required in the HMI device.
   - Click "Download".

   You can open the "Extended download" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens. The project is compiled at the same time. The result is displayed in the "Load preview" dialog.
4. Check the displayed presettings and change them as necessary.
5. Click "Download".

Result

The project is loaded to all selected HMI devices. In WinCC V13, the project with the Runtime add-ons is loaded to the selected HMI devices.

During the download, you can keep track of the files that are transferred.

If errors or warnings occur during the download, corresponding alarms are displayed under "Info > Load" in the Inspector window.

On completion of the successful download of the project, you can execute it on the HMI device.

Note

If the transfer is interrupted, WinCC V13 automatically ensures that no data is lost and that existing data is deleted on the HMI device only after complete transmission.

See also

Error messages during the download of projects (Page 6954)

Overview for loading of projects (Page 6935)

Loading projects to a file

Introduction

Load the Runtime project to the file system of your PC if that is your HMI device. Double-click the Runtime project file to open it in runtime.

During loading, you must most importantly specify whether existing data on the HMI device such as "user administration" and "recipe data" is to be overwritten.

If the HMI device supports PROFINET, the name registered in the project tree is used for the PROFINET communication. The use of the name corresponds to the default settings of the PROFINET interface of the HMI device. For devices with more than one PROFINET interface, the name of the IE CP is automatically added to the device name with a separating period. The name is written during download to the HMI device. If a device name for the PROFINET communication has already been entered in the HMI device, it will be overwritten.

You can find additional information about these settings in the information system in the "Assigning a device name and IP address" section.

Note

Device version

If the device version of the target HMI device does not correspond to that of the configured device version, Runtime cannot be started after loading.

Verify that the device version of the target HMI device conforms to your configuration before you compile and download your project.

If necessary, change the device version manually via the properties of the HMI device.

Notice

Changing the device version deletes all data on the HMI device.

Data is deleted on the target system if you change the device version. It is therefore advisable to generate a backup copy of the Runtime data prior to such actions.

Note

Transfer to an HMI device with PC station V1.0 or PC station V2.0

The version of the PC station set in the device settings of the HMI device must match the version installed on the HMI device.

Additional information on configuration settings for "PC Station" on the HMI device is available in the "SIMATIC NET" documentation.

Controlling the transfer behavior on the HMI device

As a general rule, only one project can be active in Runtime on an HMI device. An HMI device is generally configured to exit Runtime automatically when loading is started. If this is not the case, you will have to exit Runtime manually on the HMI device.

You define how the HMI device reacts when the project is loaded in the "Start Center" under "Settings" on the HMI device:

Transfer mode	Effect
Off	The project cannot be loaded to the HMI device.
Manually	The project can only be loaded to the HMI device if the following requirements are met: • Runtime is not running • The HMI device is in "Transfer" mode.
Automatically	The project can always be loaded to the HMI device. If a transfer is started on the configuration PC and a project is in runtime on the HMI device, the running project is automatically closed. For Mobile Panels, this transfer mode is disabled for security reasons.

Note

Exiting Runtime automatically

Deactivate automatic transfer after the commissioning phase so that the HMI device does not inadvertently go into transfer mode.

Transfer mode can cause undesired reactions in the system.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the "Start Center".

Please refer to the documentation for the HMI device used for more detailed information on transfer settings.

Requirement

- You have created an HMI device in the project.
- The HMI device is connected to the configuration PC.
- The "Start Center" has been started on the HMI device.
- The protocol by which the project is loaded is set on the HMI device in the "Start Center" under "Settings".
  Note: If you load the project to an HMI device with installed "PC Station V2.0", the protocol is set automatically.
- Transfer mode is set as "Automatically" or "Manually" in the HMI device.

Procedure

Proceed as follows to load a project to a file:

1. Select the project in the project tree.
2. Select the "Download to device > Software" command from the shortcut menu of an HMI device.
3. If the "Extended loading" dialog is open, configure the "Settings for loading". Make sure that the "Settings for loading" correspond to the "Transfer settings in the HMI device".
   - Select the "Type of the PG/PC interface > File" dialog.
   - Select "PG/PC interface > File system" in the dialog.
   - Select the file system path under "destination path".
   - Click "Download".
     You can open the "Extended download" dialog at any time with the menu command "Online > Advanced download to device...".
     The "Load preview" dialog opens. The project is compiled at the same time. The result is displayed in the "Load preview" dialog.
4. Check the displayed presettings and change them as necessary.
5. Decide whether to overwrite user administration data and/or recipe data on the HMI device.
6. Click "Download".

Result

The Runtime is loaded to the file system. If errors or warnings occur during the download, corresponding alarms are displayed under "Info > Load" in the Inspector window.

After the successful loading of the Runtime project, you can execute it on the HMI device.

Double-click the Runtime project file on the target system to launch the Runtime project.

Generating a "Pack&Go" file

Introduction

Create a "Pack&Go" file if you cannot connect the HMI device to the configuration PC. The "Pack&Go" file is a ZIP file containing the following data:

- The compiled project
- A program for transferring the project to the HMI device
- Image for the configured HMI device

Typical example of an application: An engineering company creates a project variant for a new HMI device. The project engineer of the engineering company does not have direct access to the plant. The project engineer therefore e-mails the "Pack&Go" file to his local contact. The contact unpacks the "Pack&Go" file on a PC connected to the HMI device via a network. He then transfers the project from the PC to the HMI device.

Requirement

- You have created an HMI device in the project.

Procedure

To create a "Pack&Go" file, follow these steps:

1. Select the HMI device in the project tree.
2. Select the "Pack&Go" command from the menu under "Online > HMI Device maintenance".
3. The "Create Pack&Go file" opens.
4. Verify that the displayed device version corresponds with that of the target HMI device.
5. Select a protocol as transfer mode.
6. Select the storage location under "Pack&Go file" and enter the file name.
7. If necessary, specify if the "Pack&Go" file is split into several files.
8. Click "Create".

Note

To unzip a "Pack&Go" file which is distributed over several files, you need a compression program. The compression program integrated into the operating system is not adequate to unzip distributed files.

Result

The "Pack&Go" file is created and saved to the specified folder in the file system. Now copy the "Pack&Go" file to the PC connected to the HMI device.

See also

Downloading projects from a "Pack&Go" file to the HMI device (Page 6946)

Downloading projects from a "Pack&Go" file to the HMI device

Requirements

- .NET Framework with version 4.5
- ProSave is installed on the PC
- The "Pack&Go" file has been stored in the file system of a PC.
- The PC is connected to the HMI device.
- The device version of the HMI device is consistent with the device version in the project

Note

To unzip a "Pack&Go" file which is distributed over several files, you need a compression program. The compression program integrated into the operating system is not adequate to unzip distributed files.

Note

If the TIA Portal is not installed on your PC, install Microsoft Visual C++ 2010 Redisitributable Package (x86) and .Net Framework 4.5 to run the Pack&Go application.

- You can find the Microsoft Visual C++ 2010 Redistributable Package (x86) in the Download Center of Microsoft at the following URL: http://www.microsoft.com/en-us/download/details.aspx?id=5555
- You can find .Net Framework 4.5 at the following URL: http://www.microsoft.com/en-us/download/details.aspx?id=30653

Procedure

To download the project from a "Pack&Go" file to an HMI device, follow these steps:

1. Unpack the "Pack&Go" file to a folder of your choice in the file system of the PC.
2. From the "PackNGo" subfolder of this directory, run "Siemens.Simatic.Hmi.PackNgo.exe". The "Pack'n Go" dialog opens. The settings for loading are set by default. Exception: You must select the target device if you are using an S7 USB connection.
3. Adjust the settings for loading appropriately if they differ from the interfaces or protocols available on the PC.
4. Specify whether or not to overwrite the user management and recipe data already stored on the HMI device.
5. Click "Transfer".

Result

The connection to the HMI device is set up via the selected path. An alarm is output if the device version used in the project differs from that of HMI device.

If this is the case, update the operating system on the HMI device. To update the operating system on the HMI device, you need ProSave. A prompt for operating system update will appear automatically if ProSave has been installed and the selected connection supports the update of the operating system.

See also

Generating a "Pack&Go" file (Page 6943)

Updating the operating system (Page 6996)

Updating the operating system on the HMI device (Page 6997)

Loading via USB interface

Introduction

When you load a project via a USB port, connect the configuration PC and the HMI device with a USB cable.

Requirements for transfer via USB

The following conditions must be met to ensure successful data transfer using a USB port:

- Use a USB host-to-host cable, USB 2.0 standard.
- You have installed the provided USB driver.
  You can find the driver on the WinCC Product DVD under "Support\DeviceDriver\USB".
- The HMI device being used is based on Windows CE and has a USB port.

You can find more information on the cables used and the manufacturers/suppliers in the Internet at:

- http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/19142034)

Note

Driver installation

To avoid problems when transferring projects, only use the USB driver provided on the WinCC Product DVD.

Note

Comfort HMI devices

Use the mini USB port to load projects via USB with Comfort HMI devices. This driver is installed automatically.

Restrictions

A project can always be transferred to an HMI device. Simultaneous transfer to several HMI devices is not possible.

Installing a USB driver in Windows XP

Introduction

To load a project via the USB port, install the USB drivers provided on the WinCC product DVD.

Note

Driver installation

To avoid problems when transferring projects, only use the USB driver provided on the WinCC product DVD.

Note

Comfort HMI devices

Use the mini USB port to load projects via USB with Comfort HMI devices. This driver is installed automatically.

Requirement

- The WinCC product DVD is available.
- You are using a USB host-to-host cable, USB 2.0 standard.
- The HMI device has a USB port.

Procedure

To install the USB driver under Windows XP, follow these steps:

1. Connect the USB host-to-host cable to the USB port of the configuration PC.
   The USB cable is automatically detected by the hardware detection under Windows XP.
   The driver installation wizard starts automatically.

   Note

   Connect the HMI device only when the driver has been completely installed on the PC.

2. If you are prompted with "Can Windows connect to Windows Update to search for software", select "No, not this time" and click "Next".
3. Select "Install software from a list or specific source" and click "Next".
4. Select "Search removable media".
5. Insert the WinCC product DVD into the DVD drive and click "Next".
   The system will search the WinCC product DVD for the corresponding driver for the USB host-to-host cable.
   The drivers available on the WinCC product DVD are displayed. The wizard highlights the appropriate driver for Windows XP.

6. Check the selection and click "Next".

   Note

   The "Hardware installation" dialog box indicates that the software found has not passed Windows Logo testing. The Logo test does not have any effects on the functioning of the USB driver. Continue with the installation.

7. Click "Hardware Installation > Continue installation".
   The driver is installed.
8. Click "Finish".
   The installation is complete.

Result

The driver for the USB host-to-host cable has been installed. To load the project, connect the HMI device to the USB cable.

Installing a USB driver in Windows 7

Introduction

To load a project via the USB port, install the USB drivers provided on the WinCC product DVD.

Note

Driver installation

To avoid problems when transferring projects, only use the USB driver provided on the WinCC product DVD.

Note

Comfort HMI devices

Use the mini USB port to load projects via USB with Comfort HMI devices. This driver is installed automatically.

Requirement

- The WinCC product DVD is available.
- You are using a USB host-to-host cable, USB 2.0 standard.
- The HMI device has a USB port.

Procedure

To install the USB driver under Windows 7, follow these steps:

1. Connect the USB host-to-host cable to the PC's USB port.
   The USB cable is detected, but the wizard for the driver installation does not find the driver.

   Note

   Connect the HMI device only when the driver has been completely installed on the PC.

2. Open the Device Manager in the Control Panel.
3. Select "Other devices > USB host-to-host cable".
4. Select "USB host-to-host cable" and then select "Update driver software..." in the shortcut menu.
5. When the "How do you want to search for driver software?" prompt appears, select "Search for driver software on the computer".
6. Click "Update driver software > Browse...".
7. Insert the WinCC product DVD as the source for driver installation.
8. Select "Include subfolders" and then click "Next".
   The system will search the WinCC product DVD for the corresponding driver for the USB host-to-host cable.
   The drivers available on the WinCC product DVD are displayed. The wizard highlights the appropriate driver for Windows 7.
9. Check the selection and click "Next".
10. To continue with the installation, click "Hardware Installation > Continue installation".
    The driver is installed.
11. Click "Finish".
    The installation is complete.

Result

The driver required for the USB host-to-host cable has been installed. To load the project, now connect the HMI device to the USB cable.

12.13.3.6 Runtime start

Starting Runtime on the configuration PC

Introduction

You can start a project in Runtime on the configuration PC if Runtime has been installed. The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime.

Note

You can only simulate HMI devices with Runtime Panels on the configuration PC. Select the "Online > Start simulation" menu command.

Note

Exiting Runtime automatically

If automatic transfer is activated on the HMI device and if a transfer is started on the configuration PC, the running project is automatically ended.

The HMI device then switches autonomously to the "Transfer" operating mode.

Deactivate automatic transfer after the commissioning phase so that the HMI device does not inadvertently go into transfer mode.

Transfer mode can cause undesired reactions in the system.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the Start Center.

Requirement

- A project is open on the configuration PC.
- Runtime is installed on the configuration PC.
- There is no project in Runtime on the configuration PC.

Procedure

To start runtime on the configuration PC, follow these steps:

1. Select the desired HMI device in the project navigation.
2. Select the "Online > Start Runtime" menu command.

Result

Runtime is started and the project is displayed on the configuration PC.

See also

Simulating a project (Page 6930)

Starting Runtime Advanced and Panel Runtime

Introduction

You can start the project in Runtime as soon as you have downloaded the project to the HMI device. The project is generally started automatically on the HMI device.

There may be several projects in the file system of the HMI device for HMI devices with Runtime Advanced. You can start one of these projects in Runtime.

The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime. Make sure when defining the Runtime settings "Lock task switching" and "Full screen" that you will be able to stop Runtime again. You can, for example, configure a button with the system function "StopRuntime".

Note

Closing Runtime automatically

If automatic transfer is activated on the HMI device and a transfer is started on the configuration PC, the running project is automatically terminated.

The HMI device then automatically switches to "Transfer" operating mode.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the "Start Center".

Requirements

- WinCC Runtime Advanced or Panel Runtime is installed on the HMI device.
- The project was downloaded to the HMI device.
- The "Start Center" has been started.

Starting Runtime on a PC

The compiled project is saved in the PC file system with the extension "*.fwc" and is freely accessible.

1. Enter the project file with complete path information in the "Start Center" under "Settings > Configuration path".
2. To start the project automatically after booting the HMI device:
   - Select the required delay time in seconds under "Wait until Autostart".
   - Add the "Start Center" application to the "Autostart" group in the Windows Start menu.
3. Click on "Start" in the "Start Center" to manually start the project in the HMI device.

Starting Runtime on a panel

The project is stored on a panel in a folder you specify in the HMI device transfer settings. The "Start Center" application is started on a panel. The project loaded is started automatically after expiration of the configured delay.

If the project does not start automatically:

1. Click on "Start" in the "Start Center" to start the loaded project.

Refer to the documentation for the HMI device for additional information on startup of projects.

Result

Runtime is started on the HMI device.

See also

Overview of compiling and loading projects (Page 6925)

Loading a project (Page 6938)

Downloading projects from a "Pack&Go" file to the HMI device (Page 6944)

12.13.3.7 Error messages during the download of projects

Possible problems during the download

When a project is being downloaded to the HMI device, status messages regarding the download progress are displayed in the output window.

Usually, problems arising during the download of the project to the HMI device are caused by one of the following errors:

- Wrong operating system version on the HMI device
- Incorrect download settings on the HMI device
- Incorrect HMI device type in the project
- The HMI device is not connected to the configuration PC.

The most common download failures and possible causes and remedies are listed below.

The serial download is cancelled

Possible remedy: Select a lower baud rate.

The download is cancelled due to a compatibility conflict

Possible cause	Remedy
Conflict between versions of the configuration software and the operating system of the HMI device	Synchronize the operating system of the HMI device with the version of the configuration software. To update the operating system on the HMI device, select the "Update operating system" command from the "Online > HMI device maintenance" menu in WinCC. You can also use ProSave. For additional information, refer to the operating instructions for the HMI device.
The configuration PC is connected to the wrong device, e.g. a controller.	Check the cabling. Correct the communication parameters.

Project download fails

Possible cause	Remedy
Connection to the HMI device cannot be established (alarm in the output window)	Check the physical connection between the configuration PC and the HMI device. Check whether the HMI device is in transfer mode. Exception: Remote control
The default communication driver is not listed in the Windows Device Manager.	Check the device status of the COM connection in the properties window of the Device Manager.

Download over MPI/DP interface fails

Possible cause	Remedy
"Configured mode" is set on the CP, for example, if you are using the SIMATIC NET CD.	Set the CP to "PG mode" using the "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the CP. Set the CP back to "configured mode".
On the programming device/PC panel, the "S7ONLINE" access point is not set to a hardware device such as CP5611 (MPI). This may be due to the installation of "SIMATIC NET CD".	Set the access point "S7ONLINE" on the selected device using the "PG/PC Panel" or "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the HMI device. Restore the "S7ONLINE" access point to the original device.

The configuration is too complex

Possible cause	Remedy
The configuration contains too many different objects or options for the HMI device selected.	Remove all objects of a specific type, for example all HTML browsers. Alternatively, remove options such as Sm@rtServer or OPC server.

12.13.3.8 Adapting the project for another HMI device

Introduction

When you download a WinCC project to an HMI device, WinCC checks whether this is compatible with the HMI device type used in the project. If the types of HMI device do not match, you will see a message before the download starts.

The download is aborted.

Adapting the project for the HMI device

You need to adapt the project accordingly to be able to download the project to the connected HMI device.

- Add a new HMI device in the project tree. Select the correct type of HMI device from the HMI device selection.
- Copy the configured components from the previous to the new HMI device.
  Copy large amounts of components directly in the project navigation and details view.
  For example, copy the "Screens" folder to the screens folder of the new HMI device with the help of the shortcut menu.

- Use the detail view to copy entries in the project tree for which the "Copy" command is not available in the shortcut menu.
- Select the "Recipes" entry in the project tree, for example. The recipes are displayed in the detail view.
- Select the recipes in the detail view and drag them to the "Recipes" entry of the new HMI device. The recipes are copied. You can also select multiple objects in the detail view.
- Configure the components that cannot be copied, e.g. connections, area pointers, and alarms.
- Save the project at various points in time.
- Compile the full project.
- When the compilation is successfully completed, download the project to the HMI device.

Linking references

References to linked objects are included in the copying. The references are linked again once the linked objects are copied.

Example:

You copy a screen in which objects are linked to tags. The tag names are entered at the individual objects after the screen is added to the new HMI device. The tag names are marked in red because the references are open. When you then copy the tags and insert them into the new HMI device, the open references are closed. The red marking for the tag names disappears.

For complete references to connected objects in the PLC, you first need to configure a connection to the PLC.

Using the information area

When you compile the project for the HMI device, errors and warnings are displayed in the "Info" tab of the Inspector window. You can use the shortcut menu command "Go to" to go directly to the location where the error or warning can be corrected.

Work through the list of errors and warnings from top to bottom.

When the compilation is successfully completed, download the project to the HMI device.

See also

Loading a project (Page 6938)

12.13.3.9 Viewing memory card data

Basics

Introduction

WinCC provides you with the possibility of viewing data stored on your memory card. The function supports the use of memory cards of the HMI device and of the CPU.

You have the following options:

Viewing a backup (Page 6958)

Renaming and deleting backups (Page 6960)

Viewing HMI device images (Page 6960)

Deleting HMI device images (Page 6962)

Creating HMI device images on memory card (Page 6963)

See also

Viewing a backup (Page 6958)

Renaming and deleting backups (Page 6960)

Viewing HMI device images (Page 6960)

Deleting HMI device images (Page 6962)

Creating HMI device images on memory card (Page 6963)

Working with backups

Viewing a backup

Introduction

The backup of a Basic Panel that is stored on a memory card can also be viewed in the TIA Portal.

Requirements

- WinCC is installed.
- A memory card with a backup is available.
- The card reader is connected to the configuration PC.
- The project view is open.

Backup on the memory card in the card reader

1. Insert the memory card into the card reader.
2. Open "SIMATIC Card Reader" in the project navigation.
3. Select the card reader drive.
   The "Online Card Data" folder is displayed.
4. Open the "Online Card Data" folder
5. Click the backup to open the shortcut menu.
6. Select "Properties".

Backup on the memory card of the PLC

Proceed as follows if the backup is stored on the memory card of the PLC:

1. Connect the PLC with the configuration PC.
2. Click on the PLC in the project navigation.
3. Select "Connect online" from the shortcut menu.
   A connection to the PLC is established.
   Once the PLC is connected, the "Online Card Data" folder is displayed.
4. Open the "Online Card Data" folder.

   Note

   Accessing a password-protected PLC

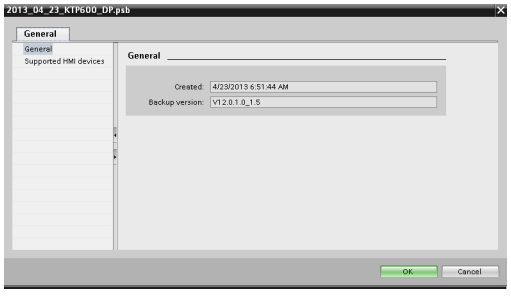
   When you attempt to access a PLC that is protected by a password, you will be prompted to enter the password.

   You need at least read access rights in order to view the data that is stored on the memory card.

5. Click the backup to open the shortcut menu.
6. Select "Properties".

Result

The backup properties are displayed in a separate dialog.



Renaming and deleting backups

Introduction

You can rename and delete backups from a memory card in the project navigation of the TIA Portal.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
  The PLC is connected online with the configuration PC.
- A memory card with a backup is available.
- The project view is open.
- The backup is displayed in the project navigation.

Note

Accessing a password-protected PLC

When you attempt to access a PLC that is protected by a password, you will be prompted to enter the password.

You need write access rights to rename or delete memory card data.

Procedure

1. Click on the backup in the project navigation.
2. Open the shortcut menu.
3. Select "Rename" to rename the file.
4. Enter a new name.
5. Select "Delete" to delete the file.

Result

The backup file is now renamed or deleted.

Working with HMI device images

Viewing HMI device images

Introduction

The HMI device image of a Comfort Panel that is stored on a memory card can be viewed in the TIA Portal.

Requirements

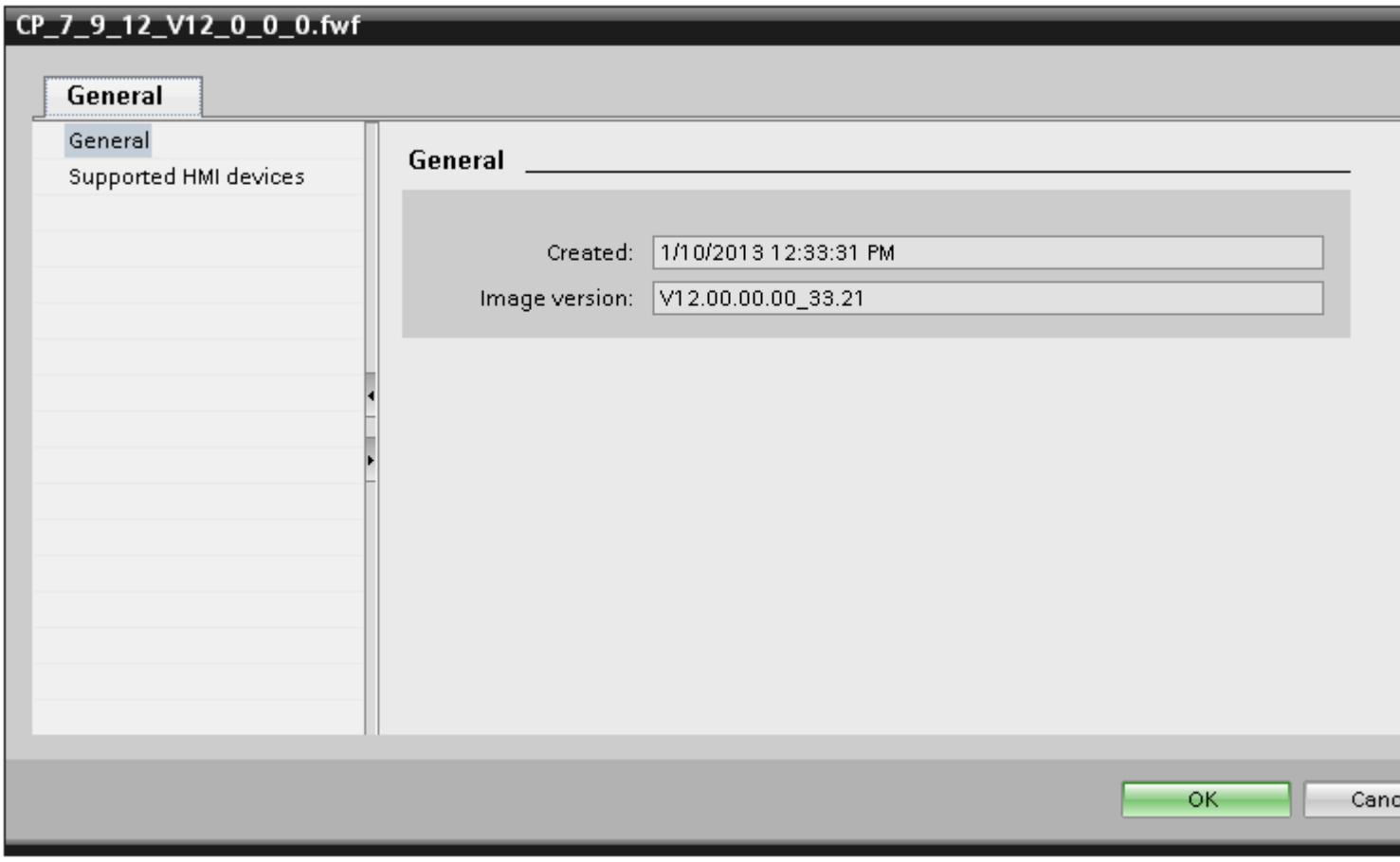
- WinCC is installed.
- The card reader is connected to the configuration PC.
- A memory card with the HMI device image is available.
- The project view is open.

Procedure

1. Insert the memory card into the card reader.
2. Open "SIMATIC Card Reader" in the project navigation.
3. Select the card reader drive.
   The "Online Card Data" dialog is displayed.
4. Open the "Online Card Data" folder.
   The available images of the HMI device are displayed in additional folders.
5. Click the required HMI device image.
6. Select "Properties" in the shortcut menu.

Result

The properties of the HMI device image are displayed in a separate dialog.



Deleting HMI device images

Introduction

You can delete the HMI device image of a Comfort Panel from a memory card in the project navigation of the TIA Portal.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
- A memory card with an HMI device image is available.
- The project view is open.
- The HMI device image is displayed in the project navigation.

Procedure

1. Click the HMI device image in the project navigation.
2. Open the shortcut menu.
3. Select "Delete" to delete the file.

Result

The HMI device image is deleted.

Creating HMI device images on memory card

Introduction

You may edit the HMI device image of a Comfort Panel without connecting the Comfort Panel to the configuration PC.

Simply create the HMI device image on an external memory card or USB stick and then transfer it from there to the Comfort Panel.

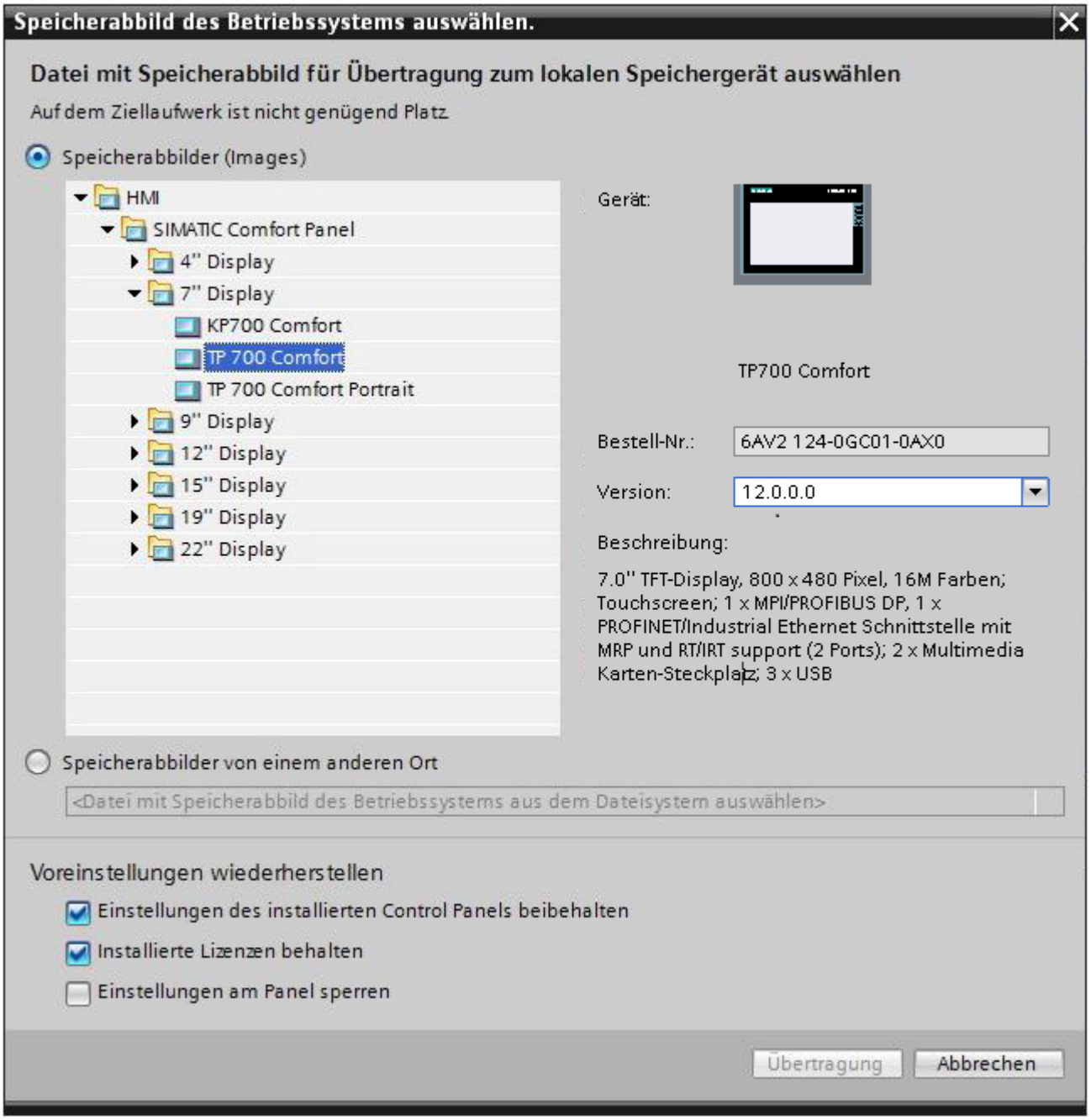
Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
- The project view is open.

Procedure

1. Insert the memory card into the card reader.
2. Click on the memory card in the project navigation.
3. Open the shortcut menu.

4. Select "Create HMI OS image on memory card".
   A dialog opens.



5. Select an HMI device image.
6. Select the settings for the restoration.
   - Activate "Retain installed settings of the Control Panel"
     The settings you have made on the Comfort Panel are retained.
   - Activate "Retain installed licenses":
     The licenses on the Panel are retained.
   - Activate "Lock settings on the Panel"
     You can no longer change the selected settings for "Retain installed settings of the Control Panel" and "Retain installed licenses" on the Comfort Panel.

Result

You have created an HMI device images on memory card. You may use a USB stick instead of a memory card.

12.13.3.10 Basics of operating in Runtime

Overview

Operating options for an HMI device

Depending on your HMI device, operate your plant as follows:

- Operation with the touch screen
  The display of the device is touch-sensitive. You operate the operating elements in the display with your finger or a stylus.
- Operation with control keys and function keys
  Control keys and function keys are integrated into the housing of the device.
  - Control keys have a specified function, for example, navigation or the acknowledgment of alarms.
  - Function keys are freely user-assignable and their function is therefore project-specific.
- Mouse and keyboard operation
  You connect the mouse and keyboard to the HMI device and operate the operating elements in the display as with a PC.

Individually configured operation

The configuration engineer has various options available for setting up operation.

Examples of actions whose execution is always determined on a project-specific basis:

- Screen change
- Reporting
- Changing Runtime language

There are no specific operating elements to execute certain functions. The configuration engineer specifies the project-specific execution. A screen change can be triggered with a button or a function key, for example.

Information on project-specific operations can be found in the system documentation.

Operation with the touch screen

Overview of operation with the touch screen

Use the touch screen to operate the HMI device of the project that is running on your HMI device.

Operating the touch screen

Notice

Damage to the touch screen

Do not touch the touch screen with sharp or pointed objects.

Avoid tapping the touch screen with hard objects, and avoid constantly using motion control.

Both can significantly reduce the useful life or even cause the failure of the touch screen.

Triggering unintended actions

You can trigger unintended actions if you touch several operating elements at the same time. Always touch only one operating element on the screen.

Operating elements are touch-sensitive symbols on the screen of the HMI device.

Special features of operation using the touch screen

Operation with the touch screen is characterized by the following special features:

- Enable
  To enable an operator control, touch the touch screen with your fingers or with a stylus. To generate a double-click, touch the operator control twice in rapid succession.
- Value input
  You enter numbers and letters on the touch screen with a screen keyboard.
- Careful operation
  If you touch multiple operator controls at the same time, you may trigger unintentional actions.

Value input using the screen keyboard

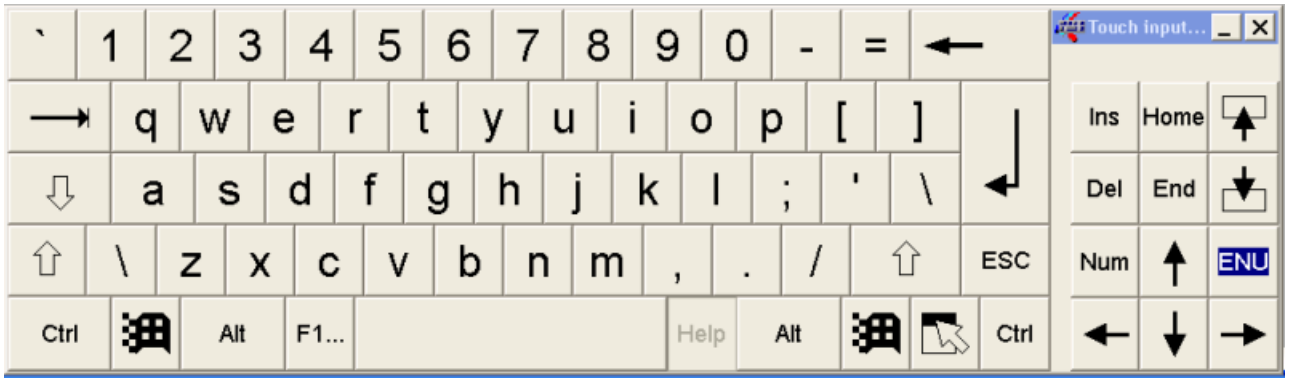
The screen keyboard is displayed as soon as you operate a screen object that requires an input. Depending on the HMI device and the configured operating element, the system displays different screen keyboards for entering numerical or alphanumerical values. The screen keyboard is hidden again when input is complete.

Screen keyboard

Layout

The following figure shows the principal structure of a screen keyboard on a Runtime Advanced HMI device.

Alphanumeric screen keyboard



Note

Language switching

Language switching in the project has no influence on the alphanumerical screen keyboard. The entry of Cyrillic or Asian characters is therefore not possible.

Numerical screen keyboard

Operator controls

The following keys are available on the screen keyboard of all HMI devices:

Button	Name	Function
	Cursor left	Navigates to the left
	Cursor right	Navigates to the right
	Backspace	Deletes a character
	Escape	Cancels the input
	Enter	Confirms the input
	Help	Displays the infotext This key is only displayed when an infotext has been configured for the operator control.

Using multi-touch functions

Multi-touch operation

Introduction

RT Advanced supports multi-touch operation under Windows 7 and Windows 8. You operate objects on the multi-touch screens by touching them with one or two fingers. Multi-touch operation is supported by TIA Portal V13. Versions V12 or older only support single-touch operation.

Requirements

The used monitor must support multi-touch functions.

Scrolling in lists and controls

You can scroll through lists and controls by dragging the screen margin.

You use horizontal dragging to move the content of the screen to the left or right. You use vertical dragging to scroll up or down in the view. An indicator appears during scrolling to indicate your position. When you drag a list diagonally on the screen, the content is moved horizontally and vertically at the same time.

Horizontal scrolling is not supported in the "Trend view" object.

You scroll five times faster up or down on a page when you use two fingers to drag up or down.

Note

Current view is not persistent

The position in the trend window changed by scrolling is not saved.
In case of a screen change, the trend view is reset to the default setting.

Zoom in and out

You enlarge or reduce the view in "Trend view" and "f(x) trend view" objects by pinch-to-zoom with two fingers.

Double tap to switch from the magnified trend view back to the normal view.

The zooming function is limited to the time axis in the "Trend view" object.

If you have enabled the option "Range > Auto-size" during configuration of the value axes in f(x) trend view, the axes are constantly calculated during zooming.

Note

Current view is not persistent

The changes of the zoom factor are not saved.

In case of a screen change, the trend view is reset to the default setting.

Supported gestures

Supported gestures

The following gestures are supported.

Gesture		Function
	Tap	You can select a button, for example, by tapping.
	Drag	Use one finger to scroll horizontally and vertically, for example, in lists. You scroll horizontally and vertically at the same time with a vertical drag.

Gesture		Function
		Use two fingers to scroll up or down a page quickly.
	Scale	Pinch-to-zoom to enlarge or reduce the display of the control.
	Double tap	Double tap the display to reset the view to the default zoom factor 100%.

Note

Operating with three of more fingers

Do not use three or more fingers to operate the device. This can lead to incorrect operations.

Two-hand operation of operator controls

Two-hand operation of operator controls

Introduction

WinCC supports two-hand operation of operator controls with RT Advanced. It ensures safe operation of operator controls which are used to change critical system settings, for example, control tags with machine limits.

Locked and unlocked operator controls

You define specific operator controls as "locked operator controls" for two-hand operation of operator controls. Locked operator controls usually cannot be operated in runtime. The operator can only operate the locked operator controls if he presses a release button at the same time.

In runtime, locked operator controls can only be accessed with the tab sequence if a release button is pressed at the same time.

Locked operator controls and release buttons

You can configure the following operator controls as locked operator controls:

- I/O field as input field or input/output field
- Button (visible)
- Button (invisible)
- Status/Force
- Slider
- Date/time field

You must configure at least one button in the screen as release button. This can be any unlocked button.

Note

You cannot configure operator controls or release buttons in the permanent window.

Templates with locked operator controls

You can also use locked operator controls in screen templates. To lock or unlock operator controls in templates, proceed as you would with operator controls in screens.

You can also implement a release button in the template. Use the same approach as for release buttons in screens.

In a screen created from the template, the release buttons from the template and any release buttons of the screen are active.

Display in runtime

The locked operator controls are displayed as dimmed in runtime. The release buttons are optically highlighted as soon as the operator presses a locked operator control. The locked operator controls are completely visible when they are unlocked by means of the release buttons.

This means you can configure the release buttons as invisible operator controls.

Simulation of projects with multi-touch functions

WinCC supports the simulation of configured multi-touch functions. Requirement is that your monitor supports multi-touch operation.

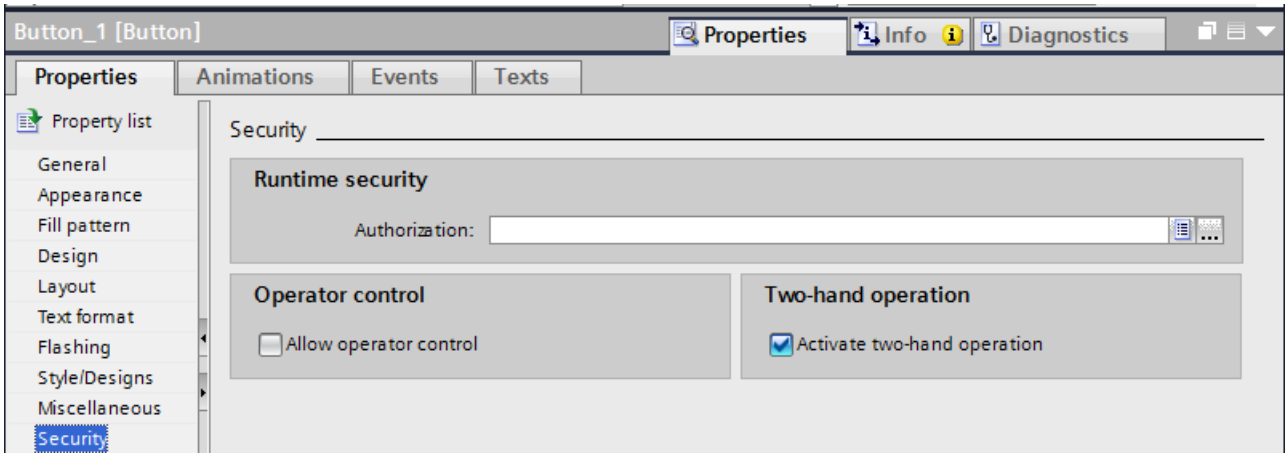
Locking and unlocking operator controls

You can lock and unlock operator controls in projects for multi-touch devices. Locked operator controls can only be operated in runtime if the operator presses a release button at the same time.

You can lock and unlock individual operator controls or several operator controls simultaneously.

Procedure

1. Configure operator controls of the type I/O field,button or slider.
2. Select the required operator control(s).
3. To lock the operator controls, enable the "Activate two-hand operation" option under "Properties > Properties > Security".



4. To unlock the operator controls, disable the "Activate two-hand operation" option under "Properties > Properties > Security".

In runtime, locked operator controls can only be operated if a release button is pressed at the same time.

Defining release button

To use the locked operator controls, you must configure at least one release button (Page 6973) in the same screen.

If no release button is configured, WinCC makes you aware of this fact during compilation of the project.

Configuring release button in screen or template

For the operator to operate locked operator controls on multi-touch devices (Page 6970), at least one release button must be configured in the same screen.

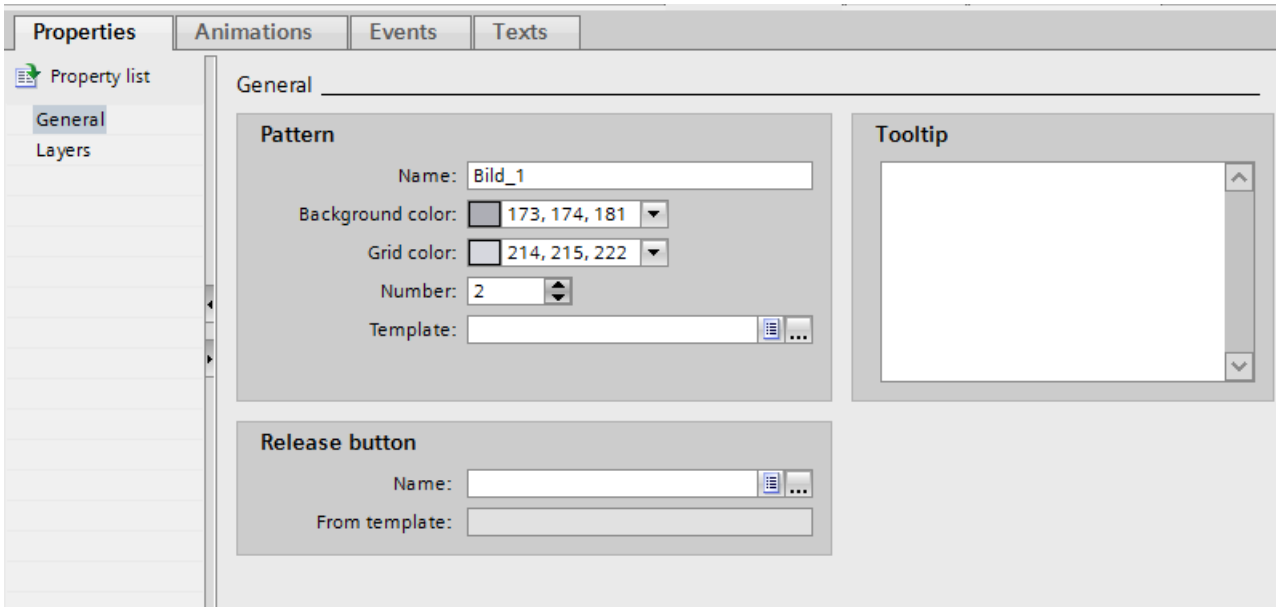
If the screen is based on a template and a release button is already defined in the template, the release button from the template is automatically in effect as release button of the screen. You can then configure an additional release button in the actual screen.

Requirement

At least one unlocked button is configured in the screen or in the template.

Procedure

1. Select the required button of the screen or template under "Release button" under "Properties > Properties > General".



Note

You can also select a button from a group for this purpose. Buttons from faceplates, however, are not available.

The selected button can also be displayed as invisible. It is optically highlighted in runtime when you press a locked operator control.

If the screen is based on a template and a release button is already defined in this template, this button is automatically in effect as release button of the screen.

Locked operator controls can only be operated in runtime if the operator presses one of the configured release buttons at the same time.

Restrictions in Sm@rtServer/Client mode

Multi-touch server and single-touch client

The single-touch client device does not support operation by touch with two or more fingers as well as multi-touch functions. Single-touch client cannot be operated with gestures of the multi-touch Sm@rtServer.

Single-touch gestures work the same way as in WinCC V12.

Single-touch server and multi-touch client

The single-touch server device does not support operation by touch with two or more fingers as well as multi-touch functions. Multi-touch client cannot be operated with gestures of the single-touch Sm@rtServer.

Single-touch gestures work the same way as in WinCC V12.

Operation with keys

Overview of operation with keys

Introduction

Use the keys of the HMI device to operate the Control Panel / Start Center of your HMI device or the project that is running on your device. Depending on the device, control keys and function keys are available.

For more detailed information, refer to the operating instructions for your HMI device.

Control keys and shortcuts

Introduction

The following tables list the control keys available to operate the project.

Note

The availability of control keys is determined by the HMI device used.

You trigger functions on keyboard HMI devices either with a key or a shortcut. With shortcuts, you keep the first key pressed. Then you press the second key.

Navigating in the display

Key or shortcut		Function	Description
TAB		Tabulator	Selects the next operating element in the tab sequence
SHIFT	TAB	Tabulator	Selects the previous operating element in the tab sequence
◀ ▶ ▼ ▲		Cursor keys	Selects the next operating element to the left, to the right, above or below the current screen object Navigating in the operating element

Operation of operating elements

Key or shortcut		Function	Description
ENTER		ENTER key	• Controls buttons. • Applies and ends an entry • Opens a selection list • Toggles between character mode and standard mode within an input field A single character is selected in character mode. In this mode, you can scroll within the character set using the cursor keys.
SHIFT	▶ ◀ ▲ ▼	Positioning cursor	Positioning the cursor within an operating element, for example, in an I/O field
←		Delete characters	Deletes the character to the left of the current cursor position
INS DEL		Delete characters	Deletes the next character to the right of the current cursor position
ESC		Cancel	• Deletes the input characters of a value and restores the original value • Closes the active dialog.
FN	HOME	Scroll to start	Scrolls to the start page of a list
HOME		Scrolling back	Scrolls the list back by one page
FN	END	Scroll to end	Scrolls to the end of a list.

Key or shortcut		Function	Description
END		Scrolling up	Scrolls the list up by one page
ALT	▼	Open selection list	Opens a selection list
CTRL	ENTER	Accept value	Accepts the value selected in the selection list without closing the list

Enter shortcut

Key	Function	Purpose
A-Z	Toggle (numbers/letters)	Toggles the assignment from numbers to letters • No LED is lit: The number assignment is enabled. Pressing the key once switches to letter assignment. • The right or left LED is lit: The left or right letter assignment is enabled. Each time the key is pressed, the system toggles between the left letter assignment, the right letter assignment and the number assignment.
SHIFT	Shift (upper/lower case)	Used in shortcuts, for example, for switching to uppercase letters
FN	Toggle to additional keyboard layout	Some of the keys contain a blue special character in their bottom left corner, for example, the percent sign "%". To input these characters, press the relevant key in combination with the special character key shown on the left.
CTRL	General control function	Used in shortcuts, for example, for navigating trend views
ALT	General control function	Used in shortcuts, for example, for the "Status/Force" screen object

Acknowledge alarms

Key	Function	Purpose
ACK	Acknowledge	Acknowledges the currently displayed alarm or all alarms of an alarm group The LED is lit as long as an unacknowledged alarm is active.

Displaying infotext

Key	Function	Description
HELP	Displaying infotext	For the selected object, opens a window with the configured infotext, for example, for an alarm or an I/O field The LED is lit if an infotext is available for the selected object.

Key or shortcut

Function Keys

The assignment of the function keys (F1, F2, F3, etc.) is defined during configuration.

Function keys with global function assignment

A globally assigned function key always triggers the same action on the HMI device or in the PLC regardless of the screen displayed. The activation of a screen or the closing an alarm window, for example, is such an action.

Function keys with local function assignment

A function key with local function assignment is screen-specific and is therefore only effective within the active screen.

The function of a locally assigned function key can vary from screen to screen.

Within a screen, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Operating function keys

Note

Operating function key after screen change

If you press a function key after a screen change, the associated function may be triggered in the new screen before the new screen is fully displayed.

Direct keys (Advanced)

Introduction

Direct keys on the HMI device are used to set bits in the I/O area of a SIMATIC S7.

Direct keys enable operations with short reaction times that are, for example, a jog mode requirement.

Note

Direct keys are still active when the HMI device is in "offline" mode.

Note

If you operate a function key with direct key functionality in a running project, the direct key function is always executed, independent of the current screen contents.

Note

You can only use direct keys when there is a connection via PROFIBUS DP or PROFINET IO.

Direct keys result in additional basic load on the HMI device.

Direct keys

The following objects can be configured as a direct key:

- Buttons
- Function keys

You can also define image numbers in the case of HMI devices with touch operation. In this way, the project engineer can configure the direct keys on an image-specific basis.

For additional information on configuring direct keys, refer to "WinCC communication".

Operator controls on the Mobile Panel

Operator controls on the Mobile Panel

The following operator controls are available to you depending on your type of mobile panel:

- Function keys
  The number and assignment of the function keys depend on the selected device type. The respective function is project-specifically assigned.
- Handwheel (optional)
  The handwheel can be turned without an end stop, and has no zero position. In a running project you enter incremental values using the handwheel.
- Key switch (optional)
  The key switch permits locking functions, which you can trigger using the HMI device.
- Illuminated pushbutton
  The function of the illuminated pushbutton is specified by the running project.

For more detailed information, refer to the operating instructions for your HMI device, and to the plant documentation.

Navigating in the display (BS)

Introduction

You navigate on the display of your HMI device as follows:

- between configured screen objects
- within screen objects
  When you select a complex screen object, the cursor focus switches to the screen object and follows the tab sequence there.
- in tables of screen objects

Procedure

- To navigate in the specified tab sequence, press the <TAB> key.
- To navigate freely between the operator controls, press the cursor keys.

Depending on the configuration of your HMI device, you can also use function keys or shortcuts for navigation.

When you operate your HMI device with the touch screen or with the mouse, you implicitly navigate by triggering a desired action. For this purpose, touch or click the operator control.

Result

The operator controls receive the cursor focus according to the selected sequence. You can trigger an action on the selected operator control.

For more detailed information, refer to the operating instructions for your HMI device.

Triggering an action

Introduction

Triggering an action at an operator control can mean the following:

- A command is executed.
  Example: Click a button to trigger a script or to execute a predefined function.
- An object is enabled.
  Example: To enter a value, select a table cell with the <Enter> key.

Requirement

- You have navigated to the operator control on which you want to trigger the action.
- The operator control has the cursor focus.

Procedure

- Press <Enter>.
  Or
- Touch the operator control on the touch screen once or twice in rapid succession.
  Or
- Click or double-click the operator control with the mouse.

Result

The following results are possible:

- The requested command is executed.
- The screen keyboard is opened and/or the cursor blinks in the input area of the operator control.
- The element is selected and can be moved.

For more detailed information, refer to the operating instructions for your HMI device.

Entering a value

Introduction

Depending on the input format, you enter numerical or alphanumerical values in an input field.

You enter these values depending on the existing hardware using the screen keyboard, the control keys of the HMI device or an external keyboard.

Requirement

- The object is an input field or table field.
- The operator control is enabled.

Entering a value

1. Enter the desired value.
2. To confirm the value and exit the field, press the <Enter> key.
3. To discard the value and exit the field, press the <Esc> key.

Result

A value is entered or discarded. You navigate as needed to the next operator control.

For more detailed information, refer to the operating instructions for your HMI device.

Moving operator controls

Introduction

In Runtime, you can move the movable operator controls of a screen object with the mouse or using the touch screen, for example, a slider or a scroll bar. Operation with the keyboard is described below.

Requirement

- A movable operator control is selected.

Procedure

- To move the operator control, proceed as follows depending on the operating element:
  - Standard for touch screen: Press the cursor keys.
  - Standard for keyboard devices: Press <SHIFT> and the cursor keys.
  - Switches: Press <ENTER>
  - Slider: Press <PgUp> or <PgDn>

1. To finish the movement, navigate to another screen object or operator control.

Slider procedure

1. To move the operator control, press the cursor keys.
2. To finish the movement, navigate to another screen object or operator control.

Result

The position of the movable operator control and the display in the screen object have changed.

For more detailed information, refer to the operating instructions for your HMI device.

Displaying infotext

Introduction

Depending on the configuration, additional information and operating instructions are available as infotext. The infotext is assigned to an operating element, an alarm or to the open screen. The infotext of an I/O field may contain, for example, information about the value to be entered.

As an alternative to the <Help> key of the HMI device, use the <Help> key of the screen keyboard for input objects.

Requirement

- An infotext is configured for the operating element, the screen or an alarm.

Calling the infotext

1. Enable the desired operating element.
2. Press the <Help> key of the HMI device.
   The infotext for the operating element is displayed.

If you are operating your input object with the touch screen, the screen keyboard opens. If the <Help> key appears, an infotext is configured for the operating element or the current screen.

If there is no infotext for the selected screen object, the infotext for the current screen is displayed, if it has been configured.

Use the scroll bar for long infotexts.

Depending on your configuration, infotext can also be retrieved by means of a configured operating element.

Switching between infotexts

- To switch between the infotexts of the operating elements and the screen, enable the infotext window.

Hiding infotext

- To hide the infotext, press the <Esc> key or press the <Help> key again.

Changing Runtime language

Introduction

The HMI device supports multilingual projects. A corresponding operating element which lets you change the language setting on the HMI device in Runtime has been configured.

The project always starts with the language set in the previous session.

Requirement

- The required language for the project is available on the HMI device.
- The language switching function is linked to an operating element, for example, to a button.

Selecting a language

You can change project languages at any time. Language-specific objects are immediately displayed on the screen in the new language when you switch languages.

You can switch the language in Runtime in one of the following ways:

- Use a configured operating element to switch from one language to the next in a list.
- Use a configured operating element to directly set the required language.

12.13.4 Viewing memory card data

12.13.4.1 Basics

Introduction

WinCC provides you with the possibility of viewing data stored on your memory card. The function supports the use of memory cards of the HMI device and of the CPU.

You have the following options:

Viewing a backup (Page 6983)

Renaming and deleting backups (Page 6985)

Viewing HMI device images (Page 6986)

Deleting HMI device images (Page 6987)

Creating HMI device images on memory card (Page 6988)

See also

Viewing a backup (Page 6983)

Renaming and deleting backups (Page 6985)

Viewing HMI device images (Page 6986)

Deleting HMI device images (Page 6987)

Creating HMI device images on memory card (Page 6988)

12.13.4.2 Working with backups

Viewing a backup

Introduction

The backup of a Basic Panel that is stored on a memory card can also be viewed in the TIA Portal.

Requirements

- WinCC is installed.
- A memory card with a backup is available.
- The card reader is connected to the configuration PC.
- The project view is open.

Backup on the memory card in the card reader

1. Insert the memory card into the card reader.
2. Open "SIMATIC Card Reader" in the project navigation.
3. Select the card reader drive.
   The "Online Card Data" folder is displayed.
4. Open the "Online Card Data" folder
5. Click the backup to open the shortcut menu.
6. Select "Properties".

Backup on the memory card of the PLC

Proceed as follows if the backup is stored on the memory card of the PLC:

1. Connect the PLC with the configuration PC.
2. Click on the PLC in the project navigation.
3. Select "Connect online" from the shortcut menu.
   A connection to the PLC is established.
   Once the PLC is connected, the "Online Card Data" folder is displayed.
4. Open the "Online Card Data" folder.

   Note

   Accessing a password-protected PLC

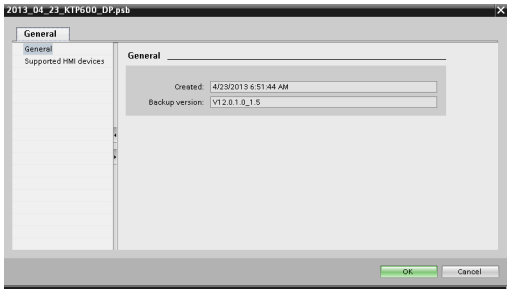
   When you attempt to access a PLC that is protected by a password, you will be prompted to enter the password.

   You need at least read access rights in order to view the data that is stored on the memory card.

5. Click the backup to open the shortcut menu.
6. Select "Properties".

Result

The backup properties are displayed in a separate dialog.



See also

Renaming and deleting backups (Page 6985)

Basics (Page 6981)

Renaming and deleting backups

Introduction

You can rename and delete backups from a memory card in the project navigation of the TIA Portal.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
  The PLC is connected online with the configuration PC.
- A memory card with a backup is available.
- The project view is open.
- The backup is displayed in the project navigation.

Note

Accessing a password-protected PLC

When you attempt to access a PLC that is protected by a password, you will be prompted to enter the password.

You need write access rights to rename or delete memory card data.

Procedure

1. Click on the backup in the project navigation.
2. Open the shortcut menu.

3. Select "Rename" to rename the file.
4. Enter a new name.
5. Select "Delete" to delete the file.

Result

The backup file is now renamed or deleted.

See also

Viewing a backup (Page 6981)

Basics (Page 6981)

12.13.4.3 Working with HMI device images

Viewing HMI device images

Introduction

The HMI device image of a Comfort Panel that is stored on a memory card can be viewed in the TIA Portal.

Requirements

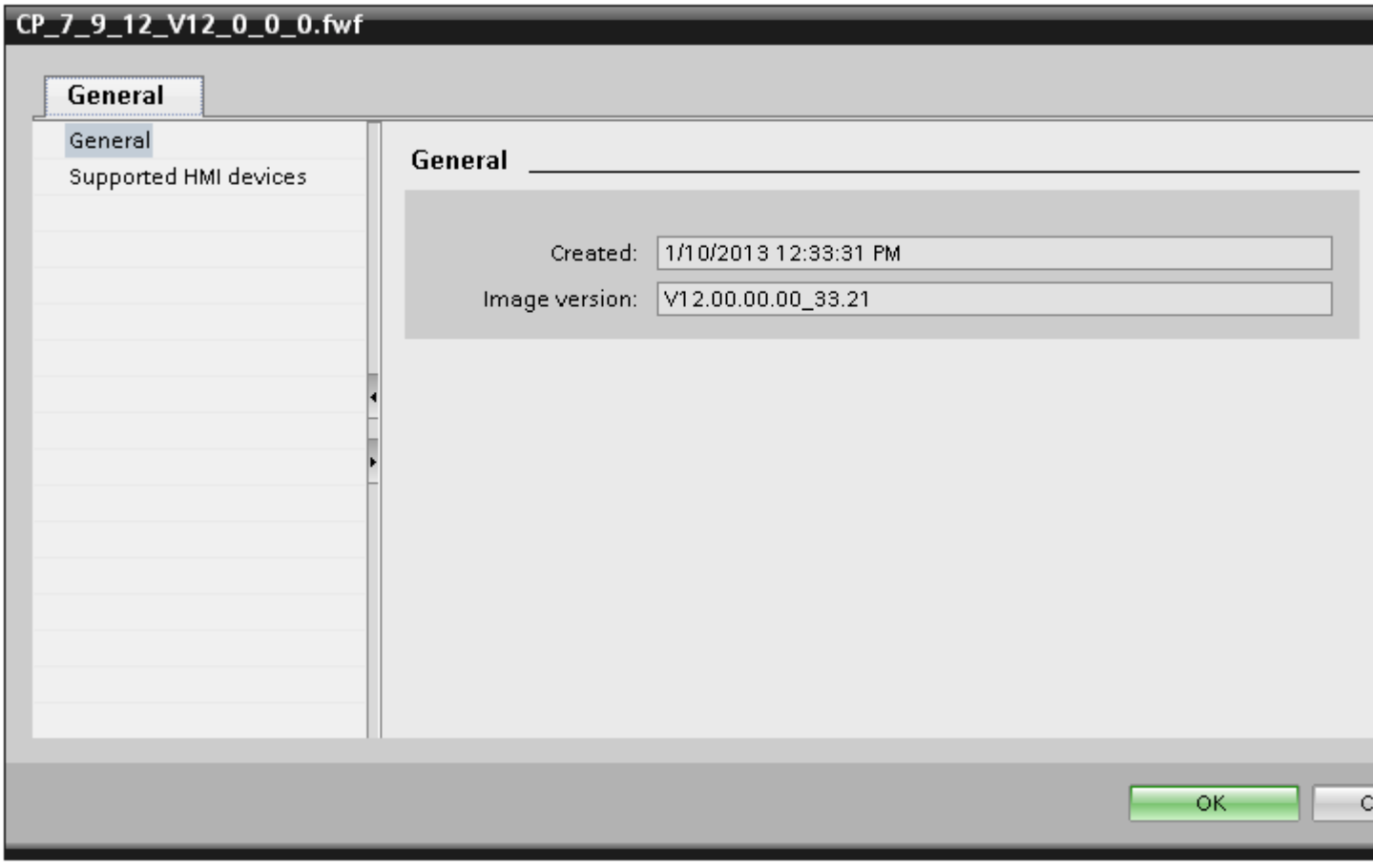
- WinCC is installed.
- The card reader is connected to the configuration PC.
- A memory card with the HMI device image is available.
- The project view is open.

Procedure

1. Insert the memory card into the card reader.
2. Open "SIMATIC Card Reader" in the project navigation.
3. Select the card reader drive.
   The "Online Card Data" dialog is displayed.
4. Open the "Online Card Data" folder.
   The available images of the HMI device are displayed in additional folders.
5. Click the required HMI device image.
6. Select "Properties" in the shortcut menu.

Result

The properties of the HMI device image are displayed in a separate dialog.



See also

Deleting HMI device images (Page 6987)

Creating HMI device images on memory card (Page 6988)

Basics (Page 6981)

Deleting HMI device images

Introduction

You can delete the HMI device image of a Comfort Panel from a memory card in the project navigation of the TIA Portal.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
- A memory card with an HMI device image is available.
- The project view is open.
- The HMI device image is displayed in the project navigation.

Procedure

1. Click the HMI device image in the project navigation.
2. Open the shortcut menu.
3. Select "Delete" to delete the file.

Result

The HMI device image is deleted.

See also

Viewing HMI device images (Page 6984)

Basics (Page 6981)

Creating HMI device images on memory card

Introduction

You may edit the HMI device image of a Comfort Panel without connecting the Comfort Panel to the configuration PC.

Simply create the HMI device image on an external memory card or USB stick and then transfer it from there to the Comfort Panel.

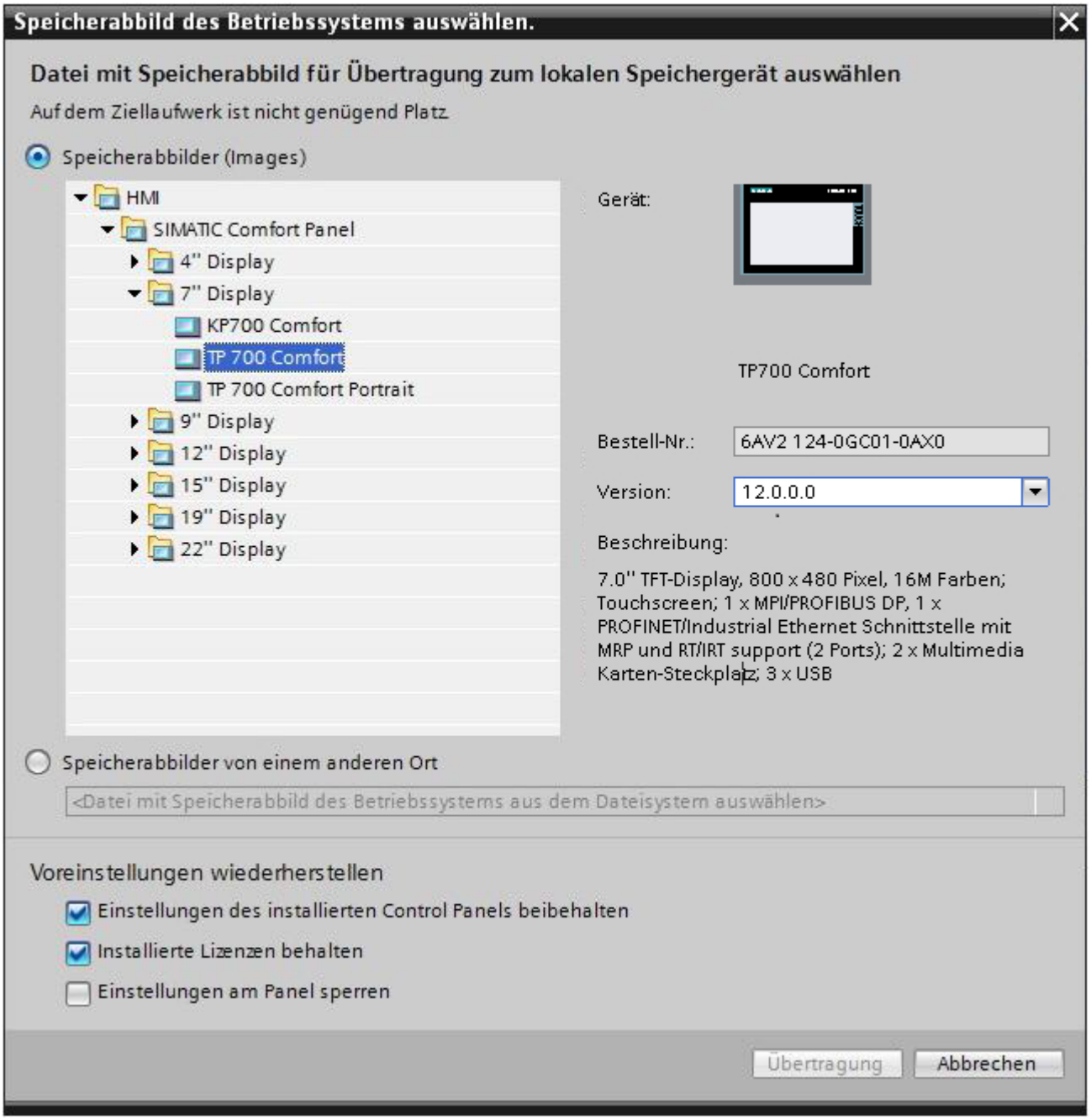
Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
- The project view is open.

Procedure

1. Insert the memory card into the card reader.
2. Click on the memory card in the project navigation.
3. Open the shortcut menu.

4. Select "Create HMI OS image on memory card".
   A dialog opens.



5. Select an HMI device image.
6. Select the settings for the restoration.
   - Activate "Retain installed settings of the Control Panel"
     The settings you have made on the Comfort Panel are retained.
   - Activate "Retain installed licenses":
     The licenses on the Panel are retained.
   - Activate "Lock settings on the Panel"
     You can no longer change the selected settings for "Retain installed settings of the Control Panel" and "Retain installed licenses" on the Comfort Panel.

Result

You have created an HMI device images on memory card. You may use a USB stick instead of a memory card.

See also

Viewing HMI device images (Page 6984)

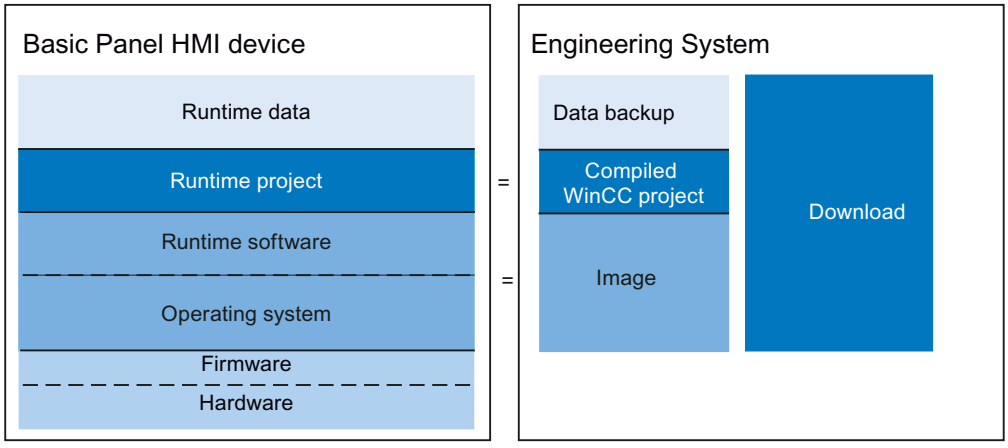
Basics (Page 6981)

12.13.5 Servicing the HMI device

12.13.5.1 Overview of HMI device maintenance (Basic Panels)

Structure

The following figure shows the software components of an HMI device and their relation to the engineering system.



Runtime data

The Runtime data is generated during operation of the plant and stored on the HMI device. This data includes, for example, recipes and data for the user administration. This data is overwritten during loading. If required, save this data before loading a Runtime project.

Runtime project

The Runtime project contains the compiled configuration data for an HMI device. You download the Runtime project from WinCC to the HMI device.

Runtime software and operating system

Together, the Runtime software and operating system of an HMI device form the image. Various images are available for the HMI device. All images of an HMI device are available in WinCC. Depending on the configuration, download the appropriate image along with the Runtime project to the HMI device as required.

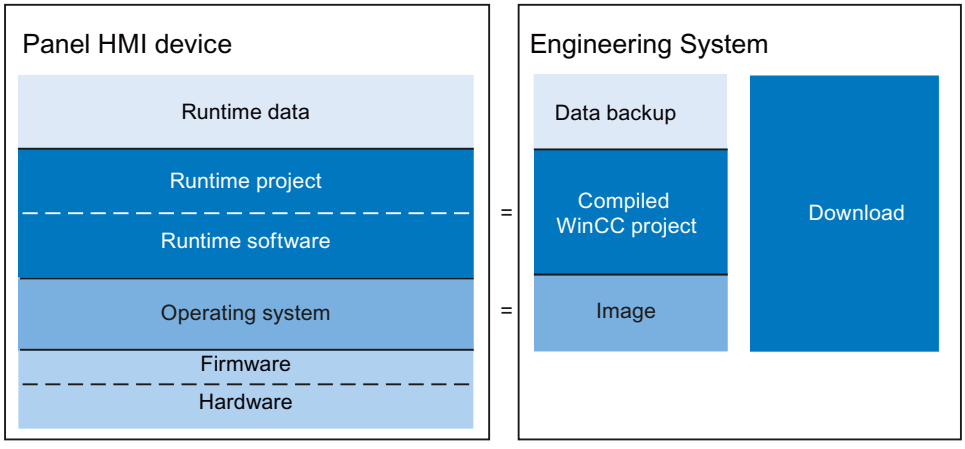
Firmware and hardware

The HMI device is delivered with preconfigured firmware and hardware.

12.13.5.2 Overview of HMI device maintenance tasks (Basic Panels)

Structure

The following figure shows the software components of an HMI device and their relation to the engineering system.



Runtime data

The Runtime data is generated during operation of the plant and stored on the HMI device. This data includes, for example, recipes and data for the user administration. This data is overwritten during loading. If required, save this data before loading a Runtime project.

Runtime project and Runtime software

The Runtime project contains the compiled configuration data for an HMI device. Download the Runtime project along with the Runtime software from WinCC to the HMI device.

Operating system

The HMI operating system is referred to as image. Various images are available for the HMI device. All images of an HMI device are available in WinCC. Depending on the configuration, download the appropriate image along with the Runtime project to the HMI device as required.

Firmware and hardware

The HMI device is delivered with preconfigured firmware and hardware.

12.13.5.3 ProSave

Introduction

The "ProSave" service tool is installed by default when WinCC is installed. The ProSave functions are accessed in WinCC with the menu "Online > HMI Device maintenance".

Functional scope

ProSave provides all of the functions you need to manage data on the HMI device:

- Data backup and restoration of backed-up data
- Operating system update for HMI devices with Windows CE or below
- Transferring License Keys
- Installing and uninstalling drivers and options as well as information on installed options and options that can be installed on an HMI device
- Communication settings (transferred from WinCC)

See also

Backup of HMI data (Page 6993)

Updating the operating system (Page 6996)

Transferring license keys (Page 6998)

Installing and uninstalling an option (Page 7001)

Overview of HMI device maintenance (Basic Panels) (Page 6988)

12.13.5.4 Backup of HMI data

Introduction

Regular backups of HMI device data keep downtimes to a minimum, for example, when you replace a device. You simply transfer the backup data to the HMI device, restoring the original status.

Data backup with WinCC

If an HMI device is connected to the configuration PC, you can back up and restore HMI device data from the configuration PC using WinCC.

Scope of data backup

Which data is backed up and restored depends on the type of HMI device:

- Complete backup
  Depending on the HMI device: Runtime, firmware, operating system, configuration, recipes, user administration, settings

  Note

  License Keys can only be stored only in the following HMI devices:
  - HMI devices of the 177 series (except TP 177A)
  - HMI devices of the 277 series
  - HMI devices of the 377 series
  - Mobile Panel Wireless
  - Comfort Panel

  The License Keys are not saved for the following HMI devices:
  - OP 77A°
  - all other Windows CE HMI devices
- Firmware/configuration
- Recipes only
- Only recipes in CSV format
- User administration only

A backup file with the extension *.psb is generated when you backup the data of an HMI device.

As a general rule, you can backup the data to any storage medium. If the HMI device is networked, you can also backup the data to a server.

Note

Scope of data backup for Windows CE HMI devices

Data backup secures the contents of the flash memory. Alarm logs and process value logs are generally saved on the external storage medium. Alarm logs and process value logs are therefore not backed up. If necessary, back up the contents of the memory card separately.

Please note the following when performing a complete data file backup and restore operation for Windows CE devices:

- A full backup includes all options installed. As a rule, the backup includes all options data that is still available after "POWER OFF".
- All data on the device, including License Keys and the operating system, are permanently deleted when you perform complete data restoration.
- If the data restoration was interrupted, execute the command "Reset to factory settings". Restart data restoration.

Note

Use interfaces with high bandwidths, e.g. USB or Ethernet to backup and restore data.

Note

For Windows CE devices, you can also backup the data directly to a CF or PC card, independent of ProSave and WinCC. For additional information, refer to the relevant operating instructions.

See also

Backing up and restoring data of the HMI device (Page 6994)

Overview of HMI device maintenance (Basic Panels) (Page 6988)

12.13.5.5 Backing up and restoring data of the HMI device

Note

Use the restore function for project data only on operating devices which were configured using the same configuration software.

Requirement

- The HMI device is connected to the configuration PC
- The HMI device is selected in the project tree.
- If a server is used for data backup: The configuration PC has access to the server

Backup of the data of the HMI device

Proceed as follows to backup the data of the HMI device:

1. Select the "Backup" command from the "Online > HMI Device maintenance" menu.
   The "Create backup" dialog box opens.
2. Select the type of the PG/PC interface and the target device, and click "Create".
   The "SIMATIC ProSave" dialog box opens.
3. Select the data to backup for the HMI device under "Data type".
4. Enter the name of the backup file under "Save as".
5. Click "Start Backup".

This starts the data backup. The backup operation takes some time, depending on the connection selected.

Restoring the data of the HMI device

Proceed as follows to restore the data of the HMI device:

1. Select the "Restore" command from the "Online > HMI Device maintenance" menu.
   The "Restore backup" dialog box opens.
2. Select the type of the PG/PC interface and the target device, and click "Load".
   The "SIMATIC ProSave" dialog box opens.
3. Enter the name of the backup file under "Save as".
   Information about the selected backup file is displayed under "File information".
4. Click "Start Restore".

This starts the restoration. This operation takes some time, depending on the connection selected.

Backup/Restore from the "Backup/Restore" dialog in the Start Center of the HMI device

The "Backup/Restore" function is enabled for MMC, SD memory cards and USB mass storage devices. The storage media supported depend on the HMI device.

For more information on this topic, refer to the device manual of the employed HMI device.

See also

Backup of HMI data (Page 6991)

Overview of HMI device maintenance (Basic Panels) (Page 6988)

12.13.5.6 Updating the operating system

Introduction

Update the HMI device image if the version is incompatible with the configuration. The image version matches the device version.

Update the operating system and Runtime software of the HMI device with the help of the device version. While loading the project, you may be prompted to run an automatic update of the device version, depending on the protocol used.

Loading will then continue. Loading of the project is otherwise aborted. In this case, perform the update of the device version manually.

Note

A device version can only be updated on HMI devices that are not PC-based.

Updating the device version

Connect the HMI device to the configuration PC to update the device version. If possible, use the interface providing the highest bandwidth for this connection, e.g. Ethernet. An update of the device version via serial connection may take up to an hour.

Note

Transfer of operating systems for MP 377 via PROFIBUS

The size of the image and the baud rates available with PROFIBUS mean image transfer with an MP 377 via PROFIBUS can take up to an hour.

Updating the operating system via USB or Ethernet.

"Reset to factory settings"

If the operating system on the HMI device is no longer operational, update the operating system and reset the HMI device to the factory settings.

Note

Resetting to factory settings via Ethernet for Basic Panels

You require the following to reset the HMI device to the factory settings via Ethernet:

- the MAC address of the HMI device
- the available IP address
- the programming device/PC interface of the configuration PC set to Ethernet TCP/IP

The programming device/PC interface is configured using the control panel of the configuration PC. Select" "S7ONLINE (STEP7) -> TCP/IP" in the "Access point of the application" field.

See also

Updating the operating system on the HMI device (Page 6997)

Overview of HMI device maintenance (Basic Panels) (Page 6988)

12.13.5.7 Updating the operating system on the HMI device

If possible, you should use the interface with the highest bandwidth for this connection, such as Ethernet. Updating the operating system via a serial connection can take up to an hour. When you update the operating system, the Runtime software on the HMI device is also updated and the device version is changed.

Notice

Updating the operating system deletes all data on the HMI device

When you update the operating system you delete data on the target system. For this reason, you should back up the following data beforehand:

- User administration
- Recipes

Resetting to factory settings also deletes the License Keys. Back up the License Keys before you reset the system to factory settings.

Requirement

- The HMI device is connected to the configuration PC.
- The HMI device is selected in the project tree.

Updating the operating system

Proceed as follows to update the operating system:

1. Select the "Update operating system" command from the "Online > HMI Device maintenance" menu.
   The "Update operating system" dialog box opens.
2. Select the type of the PG/PC interface and the target device, and click "Update".
   The "SIMATIC ProSave [OS-Update]" dialog opens. The path to the image is preset.
3. If required, you can select a different path for the image that you want to transfer to the HMI device.
4. Click "Update OS".

This starts the update. The update operation can take time, depending on the connection selected.

Resetting the HMI device to factory settings

To reset the HMI device to factory settings, proceed as follows:

1. Switch off power to the HMI device.
2. Connect the HMI device to the Engineering Station.
3. Select the "Update operating system" command from the menu under "Online > HMI Device maintenance" on the configuration PC in WinCC.
   The "Update operating system" dialog box opens.
4. Select the type of the PG/PC interface and the target device, and click "Update".
   The "SIMATIC ProSave [OS-Update]" dialog opens. The path to the image is preset.
5. If required, you can select a different path for the image that you want to transfer to the HMI device.
6. Activate "Reset to factory settings".
7. Click "Update OS".
8. To reset to factory settings, switch on the power to the HMI device again.
   This operation can take time.

Result

The operating system of the HMI device is operational and up-to-date.

See also

Managing licenses (Page 6999)

Backing up and restoring data of the HMI device (Page 6992)

Updating the operating system (Page 6994)

Overview for loading of projects (Page 6935)

Overview of HMI device maintenance (Basic Panels) (Page 6988)

12.13.5.8 Transferring license keys

Introduction

You need a license for certain WinCC Runtime options you may want to install on an HMI device. Usually, the necessary licenses supplied as "License Keys" on a data medium, e.g. USB stick. The "License Keys" can also be made available on a license server.

Use "Automation License Manager" to transfer the "License Keys" to or from an HMI device. The "Automation License Manager" is included automatically when you install WinCC.

Notice

Backing up License Keys

In the following cases you have to backup the "License Keys" in order to prevent deletion of the "License Keys":

- Prior to the update of the operating system of a Windows CE HMI device
- Prior to restoring the data from a full backup
  "License Keys" on an HMI device are backed up depending on the HMI device configuration. For more information, refer to the operating instructions of the corresponding HMI device.

See also

Managing licenses (Page 6999)

12.13.5.9 Managing licenses

Requirement

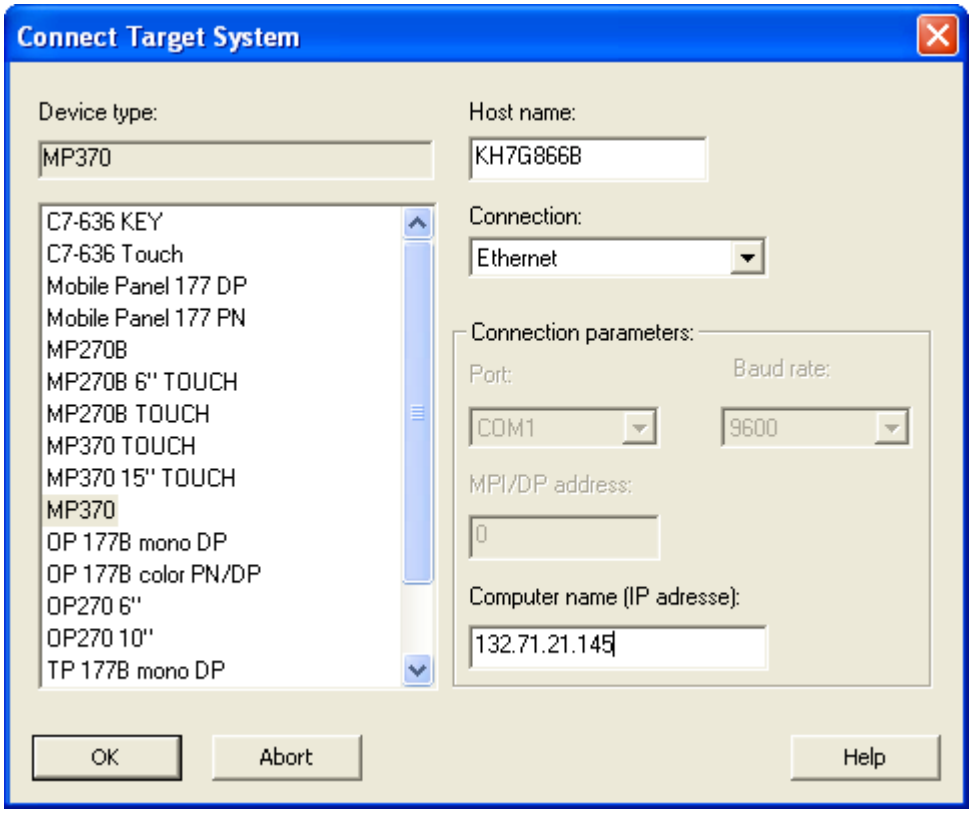
- The HMI device is connected to the configuration PC or the PC running the "Automation License Manager".
- If you are using the configuration PC: The HMI device is selected in the project tree.

Procedure

To transfer license keys, follow these steps:

1. Open the "Automation License Manager". Go to the Windows Start menu and start "Automation License Manager" on a PC on which WinCC is not installed.
   The "Automation License Manager" starts.
2. Select the "Connect HMI device" command from the "Edit > Connect target system" menu.
   The "Connect target system" dialog opens.
3. Select the HMI device type in the "Device type" area.
4. Select the "Connection".

5. Configure the "connection parameters" associated with the selected connection.



6. Click "OK".
   The connection to the HMI device is now set up. The connected HMI device is displayed in the left pane of "Automation License Manager".
7. Transfer the "License Keys" to the HMI device:
   - In the left pane, select the drive on which the "License Keys" are located.
     The "License Keys" are displayed on the right pane.
   - Select the "License Keys"
   - Drag-and-drop the "License Keys" to the HMI device.

You can also remove License Keys from the HMI device using drag-and-drop.

Alternative procedure

You can also start the "Automation License Manager" from WinCC on a PC with a WinCC installation: Select the "Authorize/License" command in the "Online > HMI Device maintenance" menu.

Result

The "License Keys" are transferred to the HMI device.

To backup the "License Keys" from the HMI device, drag-and-drop the "License Keys" from the HMI device to an available drive.

See also

Transferring license keys (Page 6996)

Installing and uninstalling an option (Page 7001)

12.13.5.10 Installing and uninstalling an option

Introduction

You can install the following options on an HMI device:

- Additional options supplied with WinCC
- Options purchased in addition to WinCC

Which options you can install depends on the HMI device type.

Requirement

- The HMI device is connected to the Engineering Station, or to the PC with ProSave.
- The HMI device is selected in the project tree.

Procedure

To install an option on the HMI device, proceed as follows:

1. Select the "Options" command from the "Online > HMI Device maintenance" menu.
   The "Load options" dialog opens.
2. Select the type of the PG/PC interface and the target device, and click "Load".
   The "SIMATIC ProSave" dialog box opens.
   All available options and the previously installed options are displayed.
3. To display the installed options on the HMI device, click "Device status".
4. To install an option on the HMI device, select the option under "Available options" and click ">>".
5. To uninstall an option from the HMI device, select the option under "Available options" and click "<<".

Result

The selected options are installed on or uninstalled from the HMI device.

See also

Managing licenses (Page 6997)

ProSave (Page 6990)

12.13.6 Printer driver

12.13.6.1 Validity

Validity

This document includes a description of the enhancements made possible with optional package printer driver.

Install the optional package to implement the extended functionality. The installation provides new printer drivers for several HMI devices.

If necessary you can install individual printer drivers with the tool SIMATIC ProSave on your HMI device. For more details on which printer drivers are available for which HMI devices please refer to the section "Supported HMI devices".

Installation

Please follow the instructions in the "Installation" section to install the "Optional package printer driver V1.4".

Application example

You can find an application example on the subject "Printing with HMI Panels" on the internet at

Auto-Hotspot

12.13.6.2 Supported HMI devices

Contents of the add-on package

The "Add-on package printer driver V1.4" contains the following printer drivers for the specified HMI devices.

The printer driver supports all printer options of the HMI device.

HMI device	Brother P-Touch QL 650 TD [1)]	Brightek WH-AB, WH-C1, WH-E19	Postscript
KP400 Comfort KTP400 Comfort	USB	USB	Yes
KP700 Comfort TP700 Comfort	USB	USB	Yes
KP900 Comfort TP900 Comfort	USB	USB	Yes
KP1200 Comfort TP1200 Comfort	USB	USB	Yes

HMI device	Brother P-Touch QL 650 TD [1]	Brightek WH-AB, WH-C1, WH-E19	Postscript
KP1500 Comfort TP1500 Comfort	USB	USB	Yes
TP1900 Comfort	USB	USB	Yes
TP2200 Comfort	USB	USB	Yes
USB: Connect printer directly to HMI device via USB port [1] Report printout only.			

Note

Please note that Postscript is a well-known and open communication protocol and theoretically, for this reason, statements could be changed by third parties. It is recommended that the printer be connected by USB in plants where security is especially critical.

HMI device	GMW IPP 144-40 GE [2] IPP 144-40 GS [2]
MP 377	USB
MP 277	USB
MP 177	USB
TP 177B 4"	USB
USB: Connect printer directly to HMI device via USB port [2] Alarm report printout only.	

12.13.6.3 Installation

Requirements

ProSave is installed on the PC.

Procedure

1. Double click the "PROSAVE-OPT.exe" file.
2. Select "<Setup>" in the "WinZip Self-Extractor" dialog.
3. Select the language for the installation menu.
4. Select "<Continue>" in the "Optional package printer driver setup" dialog.

5. Confirm the next box with "<Install>".
6. Confirm the "Optional package printer driver setup" message with "<Finish>".

Note

Please follow the instructions in the "Transferring the Options" chapter to transfer the printer driver onto the HMI device.

12.13.6.4 Transferring the Options

Requirements

The HMI device is connected to a PC on which ProSave is installed.

Procedure

1. Restart ProSave on your PC.
2. Select the corresponding HMI device type in the "General" tab.
3. Select the type of connection between the HMI device and the PC.
4. Set the connection parameters.
5. Select the "Options" tab.
6. Select the desired printer driver under "Available options".
7. Set "Transfer" mode on the HMI device.
   If automatic transfer mode is enabled on the HMI device, the device automatically sets "Transfer" mode when a transfer is initiated.
8. Select ">>" on the PC to start the transfer.
9. Follow the ProSave instructions and the instructions which appear on the HMI device. During the installation of the printer driver, a status display appears indicating the progress of the operation.

Result

When the installation is complete, the new printer driver is displayed in ProSave in the "Installed options" area. The new printer driver can now be used on the HMI device.

12.13.6.5 Setting up the printer driver

Postscript

Define PostScript V1.4 as printer on the HMI device

1. Open the Control Panel.
2. Start the "Printer" application.
3. Under "Printer Language" in the "Printer Properties" dialog, select: the entry "PostScript V1.4".

Note

Not all printers support PostScript. Make sure that your printer supports this printer language.

Select the "Draft mode" option to increase the print speed.

HTML

Introduction

The application "HTMLSettings.exe" is available on the desktop of the HMI device after installation of the "HMTL V1.3" printer driver.

Selecting the storage location

1. Open the application "HTMLSettings.exe" on the desktop of your HMI device.
2. Open "Storage Location" in the tab.
3. Under "Primary Location", select "USB-Stick", for example, for the storage location.
4. Select the directory in which the file is to be saved after printing.

Selecting the file name

1. Open the "HTMLSettings.exe" application
2. Open the "HTML File Names" tab
3. Select the settings for the names of the HTML file for the print-out.
4. Click on "Apply" to confirm your settings.
5. Select "Default" to retain the default settings for file name assignment.

Selecting columns for printing reports

1. Open the "HTMLSettings.exe" application
2. Open the "Columns Attributes" tab.
3. Select an attribute in the "Available Columns" category:
4. Add this attribute to your selection with the ">>" button.
5. If required, define whether the selected attribute is to be displayed as an image or as a hyperlink in the HTML file.
6. To deselect attributes from the "Selected Columns" category, select an attribute and click the "<<" button.

Selecting style sheets

1. Open the "HTMLSettings.exe" application
2. Open the "Advanced Options" tab.
3. Under "Enter Style Sheet File Path", select the storage location of the style sheets you wish to use.
4. Select the file to be used.
5. Select the directory for links.
6. Select the directory for images.

Defining HTML V1.3 as printer on the HMI device

1. Open the Control Panel.
2. Start the "Printer" application.
3. Under "Printer Language" in the "Printer Properties" dialog, select: the entry "HTML V1.3".
4. Select "Draft Mode" for rapid storage of the HTML file.

PDF

Introduction

The application "PDFSettings.exe" is available on the desktop of the HMI device after installation of the "PDF V1.3" printer driver.

Selecting the storage location

1. Start the application "PDFSettings.exe" on the desktop of your HMI device.
2. Open "Storage Location" in the tab.

3. Under "Primary Location", select "USB-Stick", for example, for the storage location.
4. Select the directory in which the file is to be saved after printing.

Selecting the file name

1. Open the "PDFSettings.exe" application
2. Open the "PDF File Names" tab
3. Select the settings for the names of the PDF file for the print-out.
4. Click on "Apply" to confirm your settings.
5. Select "Default" to retain the default settings for file name assignment.

Define PDF V1.3 as printer on the HMI device

1. Open the Control Panel.
2. Start the "Printer" application.
3. Under "Printer Language" in the "Printer Properties" dialog, select: the entry "PDF V1.3".
4. Select "Draft Mode" for rapid storage of the PDF file.

12.14 Performance features

12.14.1 Engineering system

Engineering system

The following tables help you assess whether your project still meets the performance specifications of the engineering system.

In addition to the specified limits, allowances must also be made for restrictions imposed by main memory resources. WinCC uses up to 2 GB of RAM, depending on the operating system. It is nonetheless useful to install more than 2 GB of main memory on the PC if running many applications with high memory requirements in parallel.

Project system limits

Different system limits apply for the engineering system, depending on whether a 32-bit or 64-bit operating system is used.

	32-bit operating system	64-bit operating system
Number of HMI devices in the project	20	40
Number of HMI tags [1)]	80,000	260,000
Number of logging tags	8,000	20,000

	32-bit operating system	64-bit operating system
Number of blocks (faceplates, user data types)[3]	10,000	20,000
Number of screens	3,000	6,000
Number of screen objects per screen	3,200	3,200
Number of screen objects	320,000	640,000
Number of alarms [2] [3]	20,000	60,000
Number of texts [3]	300,000	600,000
Number of text lists and graphic lists [3]	10,000	20,000
Number of entries per text list	3,000	3,000
Number of languages	32	32
Number of global libraries [3]	20	20
Number of objects in the project library [3]	300,000	800,000

1) Including logging tags.
2) With an average of 3 texts and a dynamic parameter
3) Including the objects configured in the "Program PLC" area

WinCC system limits for an HMI device

	32-bit operating system	64-bit operating system
Number of HMI tags [1]	80,000	260,000
Number of logging tags	8,000	20,000
Number of logs	500	1,000
Number of screens	1000	2,000
Number of screen objects per screen	3,200	3,200
Number of screen objects	320,000	640,000
Number of function lists	30,000	60,000
Number of animations and local scripts	50,000	100,000
Number of user-defined functions	1,000	2,000
Number of tasks	500	1,000
Number of alarms [2]	20,000	60,000
Number of recipes	1,000	2,000
Number of recipe elements	10,000	20,000
Number of texts	300,000	600,000
Number of text lists and graphic lists	1,000	2,000
Number of entries per text list	5,000	10,000
Number of users	200	200
Number of reports	300	600

1) Including logging tags.
2) With an average of 3 texts and a dynamic parameter

System limits during migration

You can migrate projects which are beyond the specified system limits in one or more areas.

An alarm will be output if migration creates a project whose limits are beyond the specified system limits. You must then adapt the project after migration to within the specified system limits to ensure safe operation in WinCC.

See also

Basic Panel (Page 7009)

Panel (Page 7016)

Mobile Panel (Page 7020)

Multi Panel (Page 7026)

Comfort Panel (Page 7029)

WinCC Runtime Advanced (Page 7034)

12.14.2 Basic Panel

Basic Panel

The following table helps you assess whether your project meets the performance features of the HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of tags in the project	250	500	250 (mono) 500 (color)	500	500	500
Number of PowerTags	--	--	--	--	--	--
Number of elements per array	100	100	100	100	100	100
Number of local tags	--	--	--	--	--	--
Number of structures	--	--	--	--	--	--
Number of structure elements	--	--	--	--	--	--

Alarms

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of alarm classes	32	32	32	32	32	32
Number of discrete alarms	200	200	200	200	200	200
Number of analog alarms	15	15	15	15	15	15
Length of an alarm in characters	80	80	80	80	80	80
Number of process values per alarm	8	8	8	8	8	8
Size of the alarm buffer	256	256	256	256	256	256
Number of queued alarm events	64	64	64	64	64	64

Screens

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of screens	50	50	50	50	50	50
Number of fields per screen	30	30	30	30	30	30
Number of tags per screen	30	30	30	30	30	30
Number of complex objects per screen [1)]	5	5	5	5	5	5
Number of array elements per screen [2)]	100	100	100	100	100	100

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

2) Array elements contained in recipes are included in the count.

Recipes

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of recipes	5	5	5	5	5	5
Number of elements per recipe[1)]	20	20	20	20	20	20
User data length in bytes per data record	--	--	--	--	--	--
Number of data records per recipe	20	20	20	20	20	20
Reserved memory for data records in the internal Flash	40 KB	40 KB	40 KB	40 KB	40 KB	40 KB

1) If arrays are used, each array element counts as one recipe element

Logs

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of logs	--	--	--	--	--	--
Number of entries per log (including all log segments) [1)]	--	--	--	--	--	--
Number of log segments	--	--	--	--	--	--
Cyclic trigger for tag logging	--	--	--	--	--	--
Number of tags that can be logged per log	--	--	--	--	--	--

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of trends	25	25	25	25	25	25

Text lists and graphics lists

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of graphics lists	100	100	100	100	100	100
Number of text lists	150	150	150	150	150	150
Number of entries per text or graphics list	30	30	30	30	30	30
Number of graphic objects	500	500	500	500	500	500
Number of text elements	500	500	500	500	500	500

Scripts

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of scripts	--	--	--	--	--	--

Communication

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of connections	4	4	4	4	4	4
Number of connections based on "SIMATIC HMI HTTP"	--	--	--	--	--	--

Help system

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of characters in a help text	500	500	500	500	500	500

Languages

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of runtime languages	5	5	5	5	5	5

Scheduler

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Time-triggered tasks [1)]	--	--	--	--	--	--

1) Event-triggered tasks are not relevant for the system limits

User administration

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of user groups	50	50	50	50	50	50
Number of authorizations	32	32	32	32	32	32
Number of users	50	50	50	50	50	50

Project

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Size of the project file "*.srt"	1024 KB	1024 KB	1024 KB	1024 KB	1024 KB	1024 KB

See also

Engineering system (Page 7005)

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

12.14.3 Basic Panel 2nd Generation

Basic Panel 2nd Generation

The following table helps you assess whether your project meets the performance features of the HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of tags in the project	800	800	800	800
Number of PowerTags	--	--	--	--
Number of elements per array	100	100	100	100
Number of local tags	--	--	--	--
Number of structures	--	--	--	--
Number of structure elements	--	--	--	--

Alarms

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of alarm classes	32	32	32	32
Number of discrete alarms	1000	1000	1000	1000
Number of analog alarms	25	25	25	25
Length of an alarm in characters	80	80	80	80
Number of process values per alarm	8	8	8	8
Size of the alarm buffer	256	256	256	256
Number of queued alarm events	64	64	64	64

Screens

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of screens	250	250	250	250
Number of fields per screen	100	100	100	100
Number of tags per screen	100	100	100	100
Number of complex objects per screen [1)]	This information is not relevant for Basic Panels 2nd Generation.			
Number of recipe displays per screen	10	10	10	10
Number of trend displays per screen	8	8	8	8
Number of alarm displays per screen	20	20	20	20
Number of user displays per screen	1	1	1	1
Number of system diagnostic displays per screen	5	5	5	5
Number of system functions per screen	150	150	150	150
Number of multiplex tags per screen	100	100	100	100

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of recipes	50	50	50	50
Number of elements per recipe[1)]	100	100	100	100
User data length in KB per data record	32	32	32	32
Number of data records per recipe	100	100	100	100
Reserved memory for data records in the internal Flash	256 KB	256 KB	256 KB	256 KB

1) If arrays are used, each array element counts as one recipe element

Logs

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of logs	2	2	2	2
Number of entries per log (including all log segments) [1)]	10000	10000	10000	10000
Number of log segments	400	400	400	400
Number of tags that can be logged per log	10	10	10	10
Cyclic trigger for tag logging	1 s	1 s	1 s	1 s

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of trends	25	25	25	25

Text lists and graphics lists

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of graphics lists	100	100	100	100
Number of text lists	300	300	300	300
Number of entries per text or graphics list	100	100	100	100
Number of graphic objects	1000	1000	1000	1000
Number of text elements	2500	2500	2500	2500

Scripts

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of scripts	--	--	--	--

Communication

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of connections	4	4	4	4
Number of connections based on "SIMATIC HMI HTTP"	--	--	--	--

Help system

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of characters in a help text	500	500	500	500

Languages

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of runtime languages	10	10	10	10

Scheduler

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Time-triggered tasks [1)]	--	--	--	--

1) Event-triggered tasks are not relevant for the system limits

User administration

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of user groups	50	50	50	50
Number of authorizations	32	32	32	32
Number of users	50	50	50	50

Project

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Size of the project file "*.srt"	10 MB	10 MB	10 MB	10 MB

12.14.4 Panel

Introduction

The following tables of system limits help you assess whether your project conforms to the system limits of a given HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of tags in the project	1000	1000	1000	500	1000	2048
Number of PowerTags	--	--	--	--	-	--
Number of elements per array	50	100	1000	250	1000	1000
Number of local tags	--	--	500	--	500	1000
Number of structures	--	--	--	--	--	999
Number of structure elements	--	--	--	--	--	400

Alarms

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of alarm classes	32	32	32	32	32	32
Number of discrete alarms	500	1000	1000	1000	2000	4000
Number of analog alarms	3	5	50	15	50	200
Length of an alarm in characters	80	80	80	80	80	80
Number of process values per alarm	8	8	8	8	8	8
Size of the alarm buffer	150	256	256	256	256	512
Number of queued alarm events	50	64	64	64	64	250

Screens

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of screens	500	500	500	250	500	500
Number of fields per screen	20	30	30	30	50	200
Number of tags per screen	20	30	30	30	50	200
Number of complex objects per screen [1)]	5	5	5	5	5	10

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of recipes	--	5	100	5	100	300
Number of elements per recipe[1)]	--	20	200	20	200	1000
User data length in bytes per data record	--	--	800	--	800	4000
Number of data records per recipe	--	20	200	20	200	500
Reserved memory for data records in the internal Flash	--	40 KB	32 KB	40 KB	32 KB	64 KB

1) If arrays are used, each array element counts as one recipe element

Logs

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of logs	--	--	--	--	--	20
Number of entries per log (including all log segments) [1]	--	--	--	--	--	10000
Number of log segments	--	--	--	--	--	400
Cyclic trigger for tag logging	--	--	--	--	--	1 s
Number of tags that can be logged per log	--	--	--	--	--	2048

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of trends	--	--	--	25	50	300
Number of measured values per trend	--	--	--	999	999	999

Text lists and graphics lists

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of graphics lists	--	--	--	100	100	400
Number of text lists	150	300	300	300	300	500
Number of entries per text or graphics list	30	30	30	30	30	256
Number of graphic objects	500	1000	1000	1000	1000	1000
Number of text elements	1000	1000	2500	1000	2500	10000

Scripts

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of scripts	--	--	--	--	--	50

Communication

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of connections	2	4	4	4	4	6
Number of connections based on "SIMATIC HMI HTTP"	--	--	--	--	4	8
Maximum number of connected Sm@rtClients (including a service client)	--	--	--	--	2	3

Help system

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of characters in a help text	320	320	320	320	320	320

Languages

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of runtime languages	5	5	5	5	5	16

Scheduler

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Time-triggered tasks [1)]	--	--	10	--	10	48

1) Event-triggered tasks are not relevant for the system limits

User administration

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Number of user groups	25	50	50	50	50	50
Number of authorizations	32	32	32	32	32	32
Number of users	25	50	50	50	50	50

Project

	OP 73	OP 77A	OP 77B	TP 177A	TP 177B OP 177B	TP 277 OP 277
Size of the project files "*.fwc", "*.srt"	256 KB	320 KB	1 MB	6":512 KB	2 MB	4 MB

See also

Engineering system (Page 7005)

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

12.14.5 Mobile Panel

Mobile Panel

The following tables of system limits help you assess whether your project conforms to the system limits of a given HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of tags in the project	1000	2048	2048	2048	2048	2048	2048	2048
Number of PowerTags	--	--	--	--	--	--	--	--
Number of elements per array	1000	1000	1000	1000	1000	1000	1000	1000
Number of local tags	500	1000	1000	1000	1000	1000	1000	1000
Number of structures	--	999	999	999	999	999	999	999
Number of structure elements	--	400	400	400	400	400	400	400

Alarms

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F MobileKTP700 F Mobile
Number of alarm classes	32	32	32	32	32	32	32	32
Number of discrete alarms	2000	4000	4000	4000	4000	4000	4000	4000
Number of analog alarms	50	200	200	200	200	200	200	200
Length of an alarm in characters	80	80	80	80	80	80	80	80
Number of process values per alarm	8	8	8	8	8	8	8	8
Size of the alarm buffer	256	512	512	512	512	1024	1024	1024
Number of queued alarm events	64	250	250	250	250	500	500	500

Screens

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F MobileKTP700 F Mobile
Number of screens	500	500	500	500	500	500	500	500
Number of fields per screen	50	200	200	200	200	50	400	400
Number of tags per screen	50	200	200	200	200	50	400	400
Number of complex objects per screen [1]	5	10	10	10	10	5	20	20

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F MobileKTP700 F Mobile
Number of recipes	100	300	300	300	300	300	300	300
Number of elements per recipe[1)]	200	1000	1000	1000	1000	1000	1000	1000
User data length in bytes per data record	800	4000	4000	4000	4000	262144	262144	262144
Number of data records per recipe	200	500	500	500	500	500	500	500
Reserved memory for data records in the internal Flash	32 KB	64 KB	64 KB	64 KB	64 KB	2 MB	2 MB	2 MB

1) If arrays are used, each array element counts as one recipe element

Logs

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F MobileKTP700 F MobileKTP700 F Mobile
Number of logs	--	20	20	20	20	50	50	50
Number of entries per log (including all log segments) [1)]	--	10000	10000	10000	10000	20000	20000	20000
Number of log segments	--	400	400	400	400	400	400	400
Cyclic trigger for tag logging	--	1 s	1 s	1 s	1 s	1 s	1 s	1 s
Number of tags that can be logged per log	--	2048	2048	2048	2048	2048	2048	2048

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of trends	50	300	300	300	300	300	300	300

Text lists and graphics lists

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of graphics lists	100	400	400	400	400	500	500	500
Number of text lists	300	500	500	500	500	500	500	500
Number of entries per text or graphics list	30	256	256	256	256	500	500	500
Number of graphic objects	1000	1000	1000	1000	1000	4000	4000	4000
Number of text elements	2500	10000	10000	10000	10000	40000	40000	40000

Scripts

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of scripts	--	50	50	50	50	100	100	100

Communication

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of connections	4	6	6	6	6	8	8	8
Number of connections based on "SIMATIC HMI HTTP"	4	8	8	8	8	8	8	8
Maximum number of connected Sm@rtClients (including a service client)	2	2	8": 3 10": 2	8": 3 10": 2	8": 3 10": 2	3	3	3

Mobile Wireless

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of zones	--	--	254	254	--	254	254	254
Number of effective ranges	--	--	--	127	--	--	--	--
Number of assigned transponders - zones	--	--	255	255	--	--	--	--
Number of assigned transponders - effective ranges	--	--	--	127	--	--	--	--
Number of effective ranges (RFID)	--	--	--	--	127	--	--	--
Number of RFID tags that can be assigned to effective ranges (RFID) in a project	--	--	--	--	127	--	--	--

Help system

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of characters in a help text	320	320	320	320	320	500	500	500

Languages

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of runtime languages	5	16	16	16	16	32	32	32

Scheduler

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Time-triggered tasks [1)]	10	48	48	48	48	48	48	48

1) Event-triggered tasks are not relevant for the system limits

User administration

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of user groups	50	50	50	50	50	50	50	50
Number of authorizations	32	32	32	32	32	32	32	32
Number of users	50	50	50	50	50	50	50	50

Project

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Size of the project file "*.fwc"	2 MB	6 MB	6 MB	6 MB	6 MB	12 MB	12 MB	12 MB

See also

Engineering system (Page 7005)

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

12.14.6 Multi Panel

Multi Panel

The following tables of system limits help you assess whether your project conforms to the system limits of a given HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	MP 177	MP 277	MP 377
Number of tags in the project	1000	2048	4096
Number of PowerTags	--	--	--
Number of elements per array	1000	1000	1000
Number of local tags	500	1000	2000
Number of structures	--	999	999
Number of structure elements	--	400	400

Alarms

	MP 177	MP 277	MP 377
Number of alarm classes	32	32	32
Number of discrete alarms	2000	4000	4000
Number of analog alarms	50	200	200
Length of an alarm in characters	80	80	80
Number of process values per alarm	8	8	8
Size of the alarm buffer	256	512	1024
Number of queued alarm events	64	250	500

Screens

	MP 177	MP 277	MP 377
Number of screens	500	500	500
Number of fields per screen	50	200	400
Number of tags per screen	50	200	400
Number of complex objects per screen [1)]	5	10	20

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	MP 177	MP 277	MP 377
Number of recipes	100	300	500
Number of elements per recipe[1)]	200	1000	1000
User data length in KB per data record	0.8	4	128
Number of data records per recipe	200	500	1000
Reserved memory for data records in the internal Flash	32 KB	64 KB	128 KB

1) If arrays are used, each array element counts as one recipe element

Logs

	MP 177	MP 277	MP 377
Number of logs	--	20	50
Number of entries per log (including all log segments) [1)]	--	10000	50000
Number of log segments	--	400	400
Cyclic trigger for tag logging	--	1 s	1 s
Number of tags that can be logged per log	--	2048	2048

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	MP 177	MP 277	MP 377
Number of trends	50	300	400

Text lists and graphics lists

	MP 177	MP 277	MP 377
Number of graphics lists	100	400	500
Number of text lists	300	500	500
Number of entries per text or graphics list	30	256	256
Number of graphic objects	1000	1000	2000
Number of text elements	2500	10000	30000

Scripts

	MP 177	MP 277	MP 377
Number of scripts	--	50	100

Communication

	MP 177	MP 277	MP 377
Number of connections	4	6	6
Number of connections based on "SIMATIC HMI HTTP"	4	8	8
Maximum number of connected Sm@rtClients (including a service client)	2	6": max. 3 10": max. 2	12": max. 3 15": max. 2 19": max: 1

Help system

	MP 177	MP 277	MP 377
Number of characters in a help text	320	320	320

Languages

	MP 177	MP 277	MP 377
Number of runtime languages	5	16	16

Scheduler

	MP 177	MP 277	MP 377
Time-triggered tasks [1)]	10	48	48

1) Event-triggered tasks are not relevant for the system limits

User administration

	MP 177	MP 277	MP 377
Number of user groups	50	50	50
Number of authorizations	32	32	32
Number of users	50	50	50

Project

	MP 177	MP 277	MP 377
Size of the project file "*.fwc"	2 MB	6 MB	12 MB

See also

Engineering system (Page 7005)

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

12.14.7 Comfort Panel

Comfort Panel

The following tables of system limits help you assess whether your project conforms to the system limits of a given HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Note

Graphic objects on Comfort Panels

The use of 32-bit graphic objects increases memory requirements for the project file on Comfort Panels.
The use of jpeg graphic objects reduces memory requirements for the project file on Comfort Panels.

Tags

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of tags in the project	1024	2048	4096	4096	4096
Number of PowerTags	--	--	--	--	--
Number of elements per array	1000	1000	1000	1000	1000
Number of local tags	500	1000	2000	2000	2000
Number of structures	999	999	999	999	999
Number of structure elements	400	400	400	400	400

Alarms

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of alarm classes	32	32	32	32	32
Number of discrete alarms	2000	4000	6000	6000	6000
Number of analog alarms	50	200	200	200	200
Length of an alarm in characters	80	80	80	80	80
Number of process values per alarm	8	8	8	8	8
Size of the alarm buffer	256	1024	1024	1024	1024
Number of queued alarm events	64	500	500	500	500

Screens

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of screens	500	500	750	750	750
Number of fields per screen	50	400	600	600	600
Number of tags per screen	50	400	400	400	400
Number of complex objects per screen [1)]	5	20	40	40	40

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of recipes	100	300	500	500	500
Number of elements per recipe[1)]	200	1000	2000	2000	2000
User data length in KB per data record	32	256	512	512	512
Number of data records per recipe	200	500	1000	1000	1000
Reserved memory for data records in the internal Flash	512 KB	2 MB	4 MB	4 MB	4 MB

1) If arrays are used, each array element counts as one recipe element

Logs

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of logs	10	50	50	50	50
Number of entries per log (including all log segments) [1)]	10000	20000	50000	50000	50000
Number of log segments	400	400	400	400	400
Cyclic trigger for tag logging	1 s	1 s	1 s	1 s	1 s
Number of tags that can be logged per log	100	2048	2048	2048	2048

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of trends	50	300	400	400	400

Text lists and graphics lists

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of graphics lists	100	500	500	500	500
Number of text lists	300	500	500	500	500
Number of entries per text or graphics list	30	500	500	500	500
Number of graphic objects	1000	4000	4000	4000	4000
Number of text elements	2500	40000	40000	40000	40000

Scripts

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of scripts	50	100	200	200	200

Communication

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of connections	4	8	8	8	8
Number of OPC connections including OPC UA	4	8	8	8	8
Number of connections based on "SIMATIC HMI HTTP"	4	8	8	8	8
Maximum number of connected Sm@rtClients (including a service client)	2	3	3	2	1

Help system

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of characters in a help text	500	500	500	500	500

Languages

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of runtime languages	32	32	32	32	32

Scheduler

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Time-triggered tasks [1)]	10	48	48	48	48

1) Event-triggered tasks are not relevant for the system limits

User administration

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of user groups	50	50	50	50	50
Number of authorizations	32	32	32	32	32
Number of users	50	50	50	50	50

Project

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Size of the project file "*.fwc"	4 MB	12 MB	24 MB	24 MB	24 MB

See also

Engineering system (Page 7005)

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

12.14.8 WinCC Runtime Advanced

WinCC Runtime Advanced

The following tables of system limits help you assess whether your project conforms to the system limits of a given HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of tags in the project	6144	12288
Number of PowerTags	128 - 4096	128 –8192
Number of elements per array	1600	1600
Number of local tags	2048	4096
Number of structures	999	999
Number of structure elements	400	400

Alarms

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of alarm classes	32	32
Number of discrete alarms	4000	6000
Number of analog alarms	500	500
Length of an alarm in characters	80	80
Number of process values per alarm	8	8
Size of the alarm buffer	1024	2048
Number of queued alarm events	500	500

Screens

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of screens	500	750
Number of fields per screen	400	600
Number of tags per screen	400	400
Number of complex objects per screen [1)]	40	40

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of recipes	999	1000
Number of elements per recipe[1)]	2000	2000
User data length in KB per data record	256	512
Number of data records per recipe	5000	5000
Reserved memory for data records in the internal Flash	--	--

1) If arrays are used, each array element counts as one recipe element

Logs

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of logs	100	100
Number of entries per log (including all log segments) [1)]	500000	500000
Number of log segments	400	400
Cyclic trigger for tag logging	1 s	1 s
Number of tags that can be logged per log	6144	12288

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of trends	800	800

Text lists and graphics lists

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of graphics lists	500	500
Number of text lists	500	500
Number of entries per text or graphics list	3500	500
Number of graphic objects	2000	4000
Number of text elements	30000	40000

Scripts

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of scripts	200	200

Communication

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of connections	8	8
Number of OPC connections including OPC UA	8	8
Number of connections based on "SIMATIC HMI HTTP"	16	16
Maximum number of connected Sm@rtClients (including a service client)	4 [1]	4 [1]

Help system

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of characters in a help text	320 [1] 500 [2]	320 [1] 500 [2]	320 [1] 500 [2]	320 [1] 500 [2]	320 [1] 500 [2]

1) For devices with HMI device version 12 or lower
2) For devices with HMI device version higher than 12

Languages

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of runtime languages	32	32

Scheduler

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Time-triggered tasks [1)]	48	48

1) Event-triggered tasks are not relevant for the system limits

User administration

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Number of user groups	50	50
Number of authorizations	32	32
Number of users	100	100

Project

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13
Size of the project file "*.fwc"	No limiting	No limiting

See also

Engineering system (Page 7005)

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

12.14.9 General technical specifications

12.14.9.1 Permitted characters

Introduction

The following table shows the restrictions that must be observed when allocating names and passwords.

Permitted characters

Names / passwords	Restriction
Names of objects	Do not use the special characters ? " / \ * < > %
Names and passwords in user view	Do not use the special characters ? " / \ § & $ %
Names of alarm logs	In the storage location file - RDB, file - CSV (ASCII) and file -TXT (Unicode) do not use the characters \ / * ? : " < > |
	For the storage location database, use only the characters a-z A-Z 0-9 _ @ # $ The characters _ @ # $ may not be used as the first character of a name
Name of HMI tags	HMI tag names may not start with the @ character
Names of screens	Do not use the special characters ? " / \ * < >
Names of AuditTrails	Do not use any special characters
Names of connections	Do not use spaces or special characters
Name of VB functions	Do not use spaces, special characters or VB keywords
Name of Windows user groups and Windows users when using SIMATIC Logon	Do not use the special characters / \

12.14.9.2 Recommended printers

Recommended printers

The current list of printers recommended for use with the HMI devices is available on the Internet at:

Link to the current printer list (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&caller=view&extranet=standard&viewreg=WW&nodeid0=10805558&objaction=csopen)

Note

All HMI devices except for a PC and Panel PC support only one printer at their USB port, even if several ports are available.

See also

Printer list (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&caller=view&extranet=standard&viewreg=WW&nodeid0=10805558&objaction=csopen)

12.14.9.3 Printing via print server

Introduction

Print servers enable access to network printers. Print jobs are routed to a corresponding printer via the print server.

Note

Printing takes place via the print server for the following HMI devices:

- xP 177, Mobile Panel 177
- xP 277, Mobile Panel 277, Mobile Panel 277 IWLAN, Mobile Panel 277F IWLAN
- Mobile Panel 277 IWLAN V2, Mobile Panel 277F IWLAN V2, Mobile Panel 277F IWLAN (RFID Tag)
- MP 377
- PC with WinCC Runtime Advanced

Requirements

- The print server deploys the "RAW" mode.

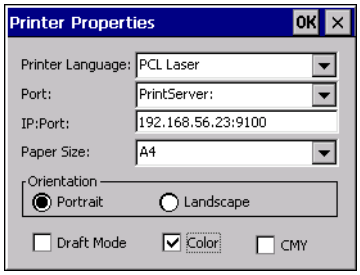
  Note

  For additional information on settings, refer to the relevant print server documentation.

- The device is interconnected via Ethernet with the print server or with a printer featuring an integrated print server.
- External print servers are interconnected with a printer via USB port.
- The IP address and port used are identical in the "Printer Properties" of the HMI device and on the print server.

Procedure

1. Open the "Control Panel" on your HMI device.
2. Select "Printer". The "Printer Properties" dialog box opens.



3. Select a printer from the "Printer Language" selection list.
4. Select the "PrintServer" entry from the "Port" selection list.
5. Enter the IP address and the port used for communication with the printer in the "IP:Port" input field.

   Note

   Use the ":" character as delimiter between the IP address and the port number. For example: "192.168.56.23:**9100**".

6. Select any additional printer settings.

12.14.9.4 Memory requirement of recipes

Introduction

The following calculation of memory requirements of recipes is only valid for Windows CE devices.

Calculation of memory requirements

The memory space required by each recipe (in KB) is derived from the sum of D1 + D2 + D3.

Valid is:

- D1 = (number of entries x 5 + M + 8):1024
  Applies to M:
  M = Accumulated length of all tag names = sum of characters in all tag names (UTF8 coded, max. 255 bytes per tag name) used in the entries.
- D2 = [(number of data records x 12) + 4]:1024
- D3 = [number of data records x (data record length + N) + 4]:1024
  Applies to N:

The total length of the name of the corresponding data record in all languages (max. 255 bytes per language) + overhead per data record (1 byte + number of languages * 3 bytes).

D1, D2 and D3 are rounded to the next higher number.

Memory requirements for using arrays

The memory required by each recipe (in KB) is derived from the sum of D1 + D2 + D3.

Valid is:

- D1 = (number of entries x 5 + M + 8):1024
  Each element of the tag array used counts as a single entry.
  Applies to M:
  M = (length of the array tag name + K) x number of array elements
  Applies to K:
  K = 3: 2 to 9 elements in the array
  K = 4: 10 to 99 elements in the array
  K = 5: 100 to 999 elements in the array
  K = 6: 1000 to 9999 elements in the array
  K = 7: 10000 to 12000 elements in the array
- D2 = [(number of data records x 12) + 4] : 1024
- D3 = [number of data records x (data record length + N) + 4] : 1024
  Applies to N:

The total length of the name of the corresponding data record in all languages (max. 255 bytes per language) + overhead per data record (1 byte + number of languages * 3 bytes).

D1, D2 and D3 are rounded to the next higher number.

Note

If you use both tags and arrays in a recipe, you have to add the results of both formulas to calculate the total memory required.

12.14.9.5 Memory requirements of recipes for Basic Panels, OP 77A, and TP 177A

Introduction

The following calculation of memory requirements of recipes is valid for Basic Panels, OP 77A, and TP 177A devices.

Restrictions

The HMI device provides 39 KB of memory space for recipes. This memory space may not be exceeded. The total memory space for recipes is calculated as follows: Total of all recipes + recipe with highest memory requirement.

Each recipe may not exceed a maximum memory space of 19 KB.

Calculation of memory requirements

The memory space requirement of each recipe (in KB) is calculated based on the three addends D1 + D2 + D3.

Valid is::

- D1 = number of data records x M
  Rule for M (size of a data record):
  M = 1 x number of elements of a byte + 2 x number of elements of 2 bytes + 4 x number of elements of 4 bytes + 8 x number of elements of 8 bytes + K
  Rule for K (size of the string elements):
  K = number of string elements x (string length + 1) x 2

- D2 - data record size
  D2 = 4 + number of languages x 8 + number of languages x (4 + 4 x number of data records + (length of the data record name + 1) x 2 x number of data records) + 8 + 8 x number of data records
  Or rewritten:
  D2 = 12 + 8 x number of data records + number of languages x (12 + number of data records x (4 + (length of the data record name +1) x 2))

- D3 - shared memory
  D3 = 14 + number of elements

Note

Arrays and single elements can be calculated as described above.

12.15 Options

12.15.1 WinCC Audit

12.15.1.1 Basics

GMP compliance

GMP compliant projects with WinCC

Traceability and therefore the documentation of production data is becoming increasingly important in many industrial sectors such as the pharmaceuticals industry, the food and beverage industry. and in the related mechanical engineering sector.

Storage of production data in electronic form offers many advantages compared to paper documents, such as simple acquisition and logging of data.

However, it is also important to ensure that data cannot be falsified and that it can be read at any time.

Industry-specific and general standards for electronic documentation of production data have been developed for this purpose.

The most important set of regulations is the FDA guideline 21 CFR Part 11 for electronic data records and electronic signatures issued by the FDA, the US Food and Drug Administration. The various EU regulations, such as EU 178/2002, also apply for particular industries.

Requirements for production systems in these industries have been developed on the basis of 21 CFR Part 11 and the corresponding layout to comply with GMP (Good Manufacturing Practice). They are also required for other industries.

The following primary requirements are derived from these directives and rules:

- Creation of an Audit Trail or operating trace in runtime
  This document can be use to trace a complete log of which user has run what control function on the machine at what time.
- Important process stages must also be traceable to a specific responsibility, for example with an electronic signature.

GMP-compliant configuration

Introduction

"GMP compliant configuration" means creating projects in accordance with "Good Manufacturing Practice". The requirements are set out in FDA rules "21 CFR Part 11". The FDA is the U.S. Food and Drug Administration.

GMP-compliant configuration means HMI devices have electronic production data documentation functionalities.

GMP relevant and the audit trail

WinCC offers the "Audit" option for implementing GMP compliance. Using the audit option, the "GMP compliant configuration" function can be enabled.

Enable the "GMP compliant configuration" function directly in the runtime settings of the HMI device. GMP relevant functionalities are then added to WinCC. These functionalities are:

- Audit Trail
- Electronic signature
- Option to label tags as "GMP relevant".
- Option to label tags as "GMP relevant" for recipes.
- NotifyUserAction system function
- Logging of tags using checksum
- Logging of alarms using checksum
- Audit trail record for printing logged changes

A license is required to convert the GMP-relevant functions configured in WinCC in runtime.

Depending on the edition of WinCC, use one of the following licenses:

- WinCC Audit for RT Advanced
- WinCC Audit for SIMATIC Panel

If the labeled objects are executed or changed, then it is saved in a special log, the "Audit Trail".

See also

Configuring a checksum for a log (Page 4260)

Evaluating the checksum of log data (Page 4261)

Configuring a checksum for a log (Page 4344)

Audit option

Advanced functions

The Audit option adds functions to WinCC to ensure that your project is GMP compliant.

The following functions are added:

- Audit Trail
  For every HMI device, you can create an Audit Trail .
  Operator actions and system processes that are relevant for the FDA-compliance of the process are recorded in an Audit Trail during runtime.
  - User actions such as changes in the values of GMP relevant tags or recipes or the acknowledgment of alarms.
  - Actions by the system, such as starting up runtime or rejection of logon attempts.
- Electronic signature
  You can set mandatory acknowledgement of important user actions in runtime, such as changing recipe data records or tag values.
  All Audit-relevant user actions must be protected by authorization in the user administration. The user will then only be able to run these actions if an electronic signature and, if configured appropriately, a comment have been input. The electronic signature and the comment are logged in the audit trail.

Extension of the WinCC engineering system

For all HMI devices that support "GMP-compliant configuration", the WinCC engineering system is extended to include the following configuration options when GMP is enabled:

- The entry "AuditTrail" is added to the "Logs" editor.
- A "Good Manufacturing Practice Settings" entry is added to "HMI tags" editor in the inspector window of a "Properties > Properties" tag.
- A "Good Manufacturing Practice" entry is added to "Recipes" editor in the inspector window of a "Properties > Properties" recipe.
- "NotifyUserAction" system function

Scope of logging

Introduction

It is important to ensure that audit-related processes are always logged in runtime in the audit trail in a project with the option "Audit".

Scope of logging

The following operations are Audit-relevant and are automatically saved in the Audit Trail:

- Runtime sequence
  - Runtime start and runtime stop
  - Project information: Version and project name, of the configuration environment, device, and current runtime configuration
  - Failure of the voltage supply of an active Uninterruptible Power Supply (UPS).
- User administration
  - Logon and logoff of users
  - Invalid logon attempts
  - Import of user administration
  - Changes of user administration
- Alarm system
  - All alarms that are acknowledged by the user.
  - All acknowledgment attempts of the user

  Note

  Logging alarm text

  To log alarm texts, select the "Log alarm text in Audit Trail" option in the Audit Trail editor:

  "Audit trail > Properties > Settings" in the "Settings" area
- Log operations
  - Starting, stopping and copying a log
  - Opening and closing all logs
  - Deleting a log
  - Starting a sequence log
  - Long-term logging of a log
- Running specific system functions depending on their functionality and the triggering event

The following audit processes are logged depending on the configuration of the recipes and the tags of the project:

- Change values of GMP-relevant tags by the user
- for GMP-relevant recipes:
  - Storing after changing and creating recipe data records
  - Transfer of recipe data records to the PLC and from the PLC
  - For recipe tags: Changing the setting for the synchronization of the tag values with the PLC ("offline"/"online")
- "NotifyUserAction" system function
  You use the system function "NotifyUserAction" to record user actions that are not automatically recorded by the audit trail.
  You can configure this system function for screen calls, for example You can also configure function lists containing system functions that do not require signature or acknowledgement.

12.15.1.2 Enabling GMP compliant configuration

Introduction

The Audit Trail and "Electronic Signature" functions are qualified as "GMP compliant configuration".

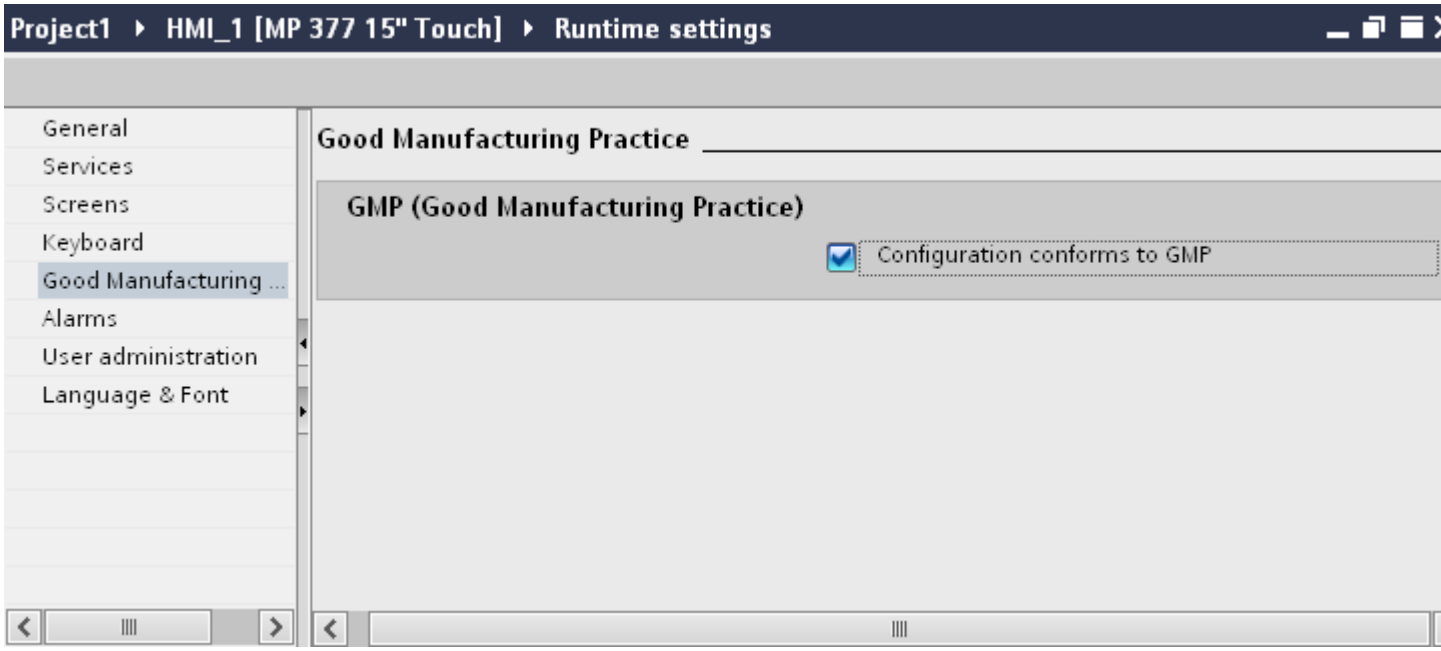
Requirements

- A project is created.
- A GMP compatible HMI device has been created.

Procedure

1. Double-click on the HMI device in the project tree.
2. Double-click "Runtime settings".

3. Click on "GMP".
4. Select "GMP compliant configuration".



Result

The Audit option is now enabled for the HMI device.

The following functions can now be configured:

- Audit Trail log
- "NotifyUserAction" system function
- GMP relevant tags
- GMP relevant recipes

See also

Configuring a checksum for a log (Page 4260)
Evaluating the checksum of log data (Page 4261)
Configuring a checksum for a log (Page 4344)
Evaluating the checksum of log data (Page 4346)

12.15.1.3 Using the Audit trail

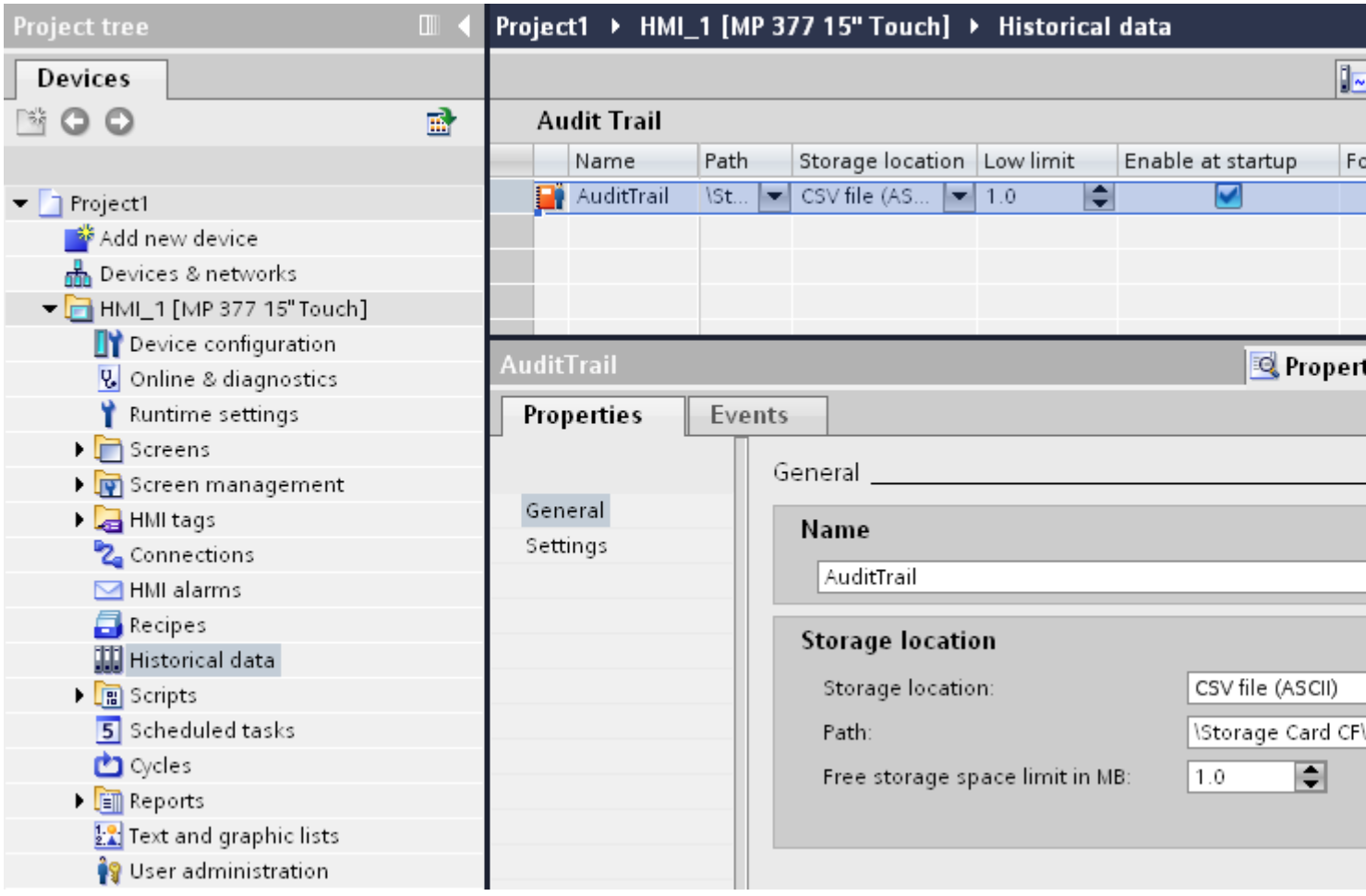
Audit Trail

Introduction

Configure a log in the settings for Audit Trail editor. This log is used to store changes in tag or recipe values made by the user and other user actions in runtime.

"Settings for audit trail" editor

1. In the project view, double-click "AuditTrail" in the "Log" group.
2. Click on the "Settings for audit trail" tab.
3. Change the properties of Audit Trail in the inspector window.



Audit trail work area

You define the settings for the Audit Trail in the "Properties > Properties" inspector window.

You set the name of the log and the storage location and decide whether logging will begin on startup. Also determine if "Forcing" is permitted.

"Forcing" is a function for administrators. It allows the administrator to continue the process even if the maximum log storage space has been exceeded.

Thus, the Audit Trail switches off and must be rebooted using the "StartLogging" system function.

Creating an audit trail

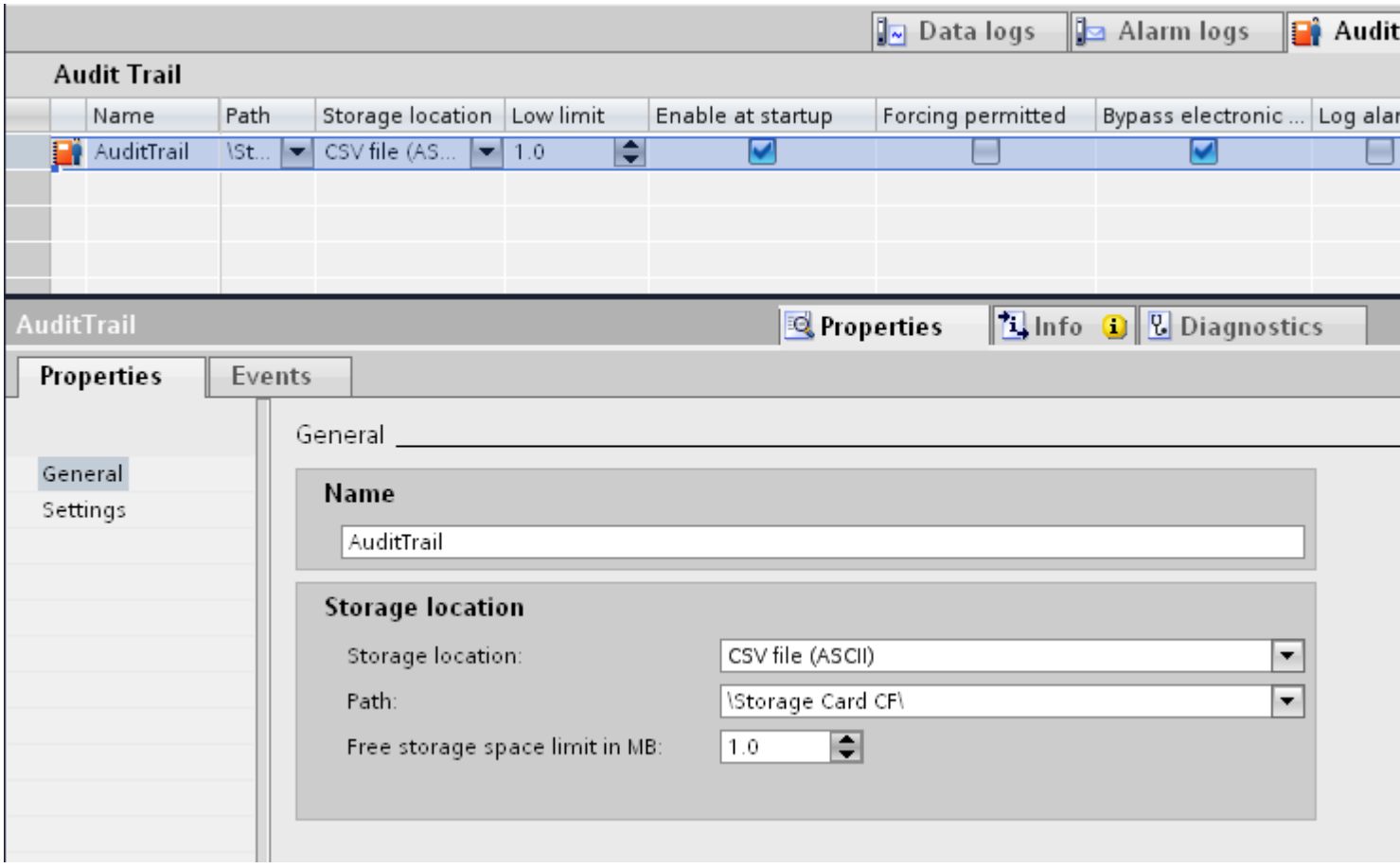
Requirements

"GMP compliant configuration" has been selected on the HMI device.

Procedure

1. Double-click on the HMI device in the project tree.
2. Double-click "Logs".
   The "Logs" editor will open.

3. Change to the "Audit Trail" tab.
   An audit trail has been created.



4. Define the following in the inspector window:
   - Name
   - Storage location
   - Logging with runtime start-up
   - Forcing

   Note

   No audit-related user actions are permitted in GMP relevant projects if there is insufficient storage space available for the audit trail.

   If the check box "Forcing permitted" is activated and, because of the hardware, there is insufficient storage space in runtime, the administrator can interrupt audit trail logging. The administrator can prevent the process from stopping in this way.

   If the administrator enables the "Forcing" function, the interruption of the audit trail by the administrator will be entered in the audit trail as the last entry.

   At the end of "Forcing" restart the audit trail using the "StartLogging" system function.

Configuring a function list

If necessary, configure a function list for the events "Low free storage space" and "Free space critically low".

The "Low free storage space" event is triggered if the amount of free storage space available for the audit trail in runtime is less than the amount configured in "Minimum storage space in MB".

The "Free space critically low" event is triggered if there is no longer sufficient free storage space for the audit trail in runtime. The value depends on the HMI device.

For more information refer to chapter: Configuring the "Low free storage space" event

Result

Audit relevant user actions are entered in the configured audit trail in runtime.

Parameters for the audit trail

Introduction

Configure Audit Trail in the "Logs" editor if you have enabled "GMP compliant configuration" in the runtime settings.

There are two ways of assigning parameters for the audit trail:

- "Settings for audit trail" editor
- "Audit Trail" inspector window

Editor "Audit Trail"

The "Audit Trail" editor is an overview of the Audit Trail created.

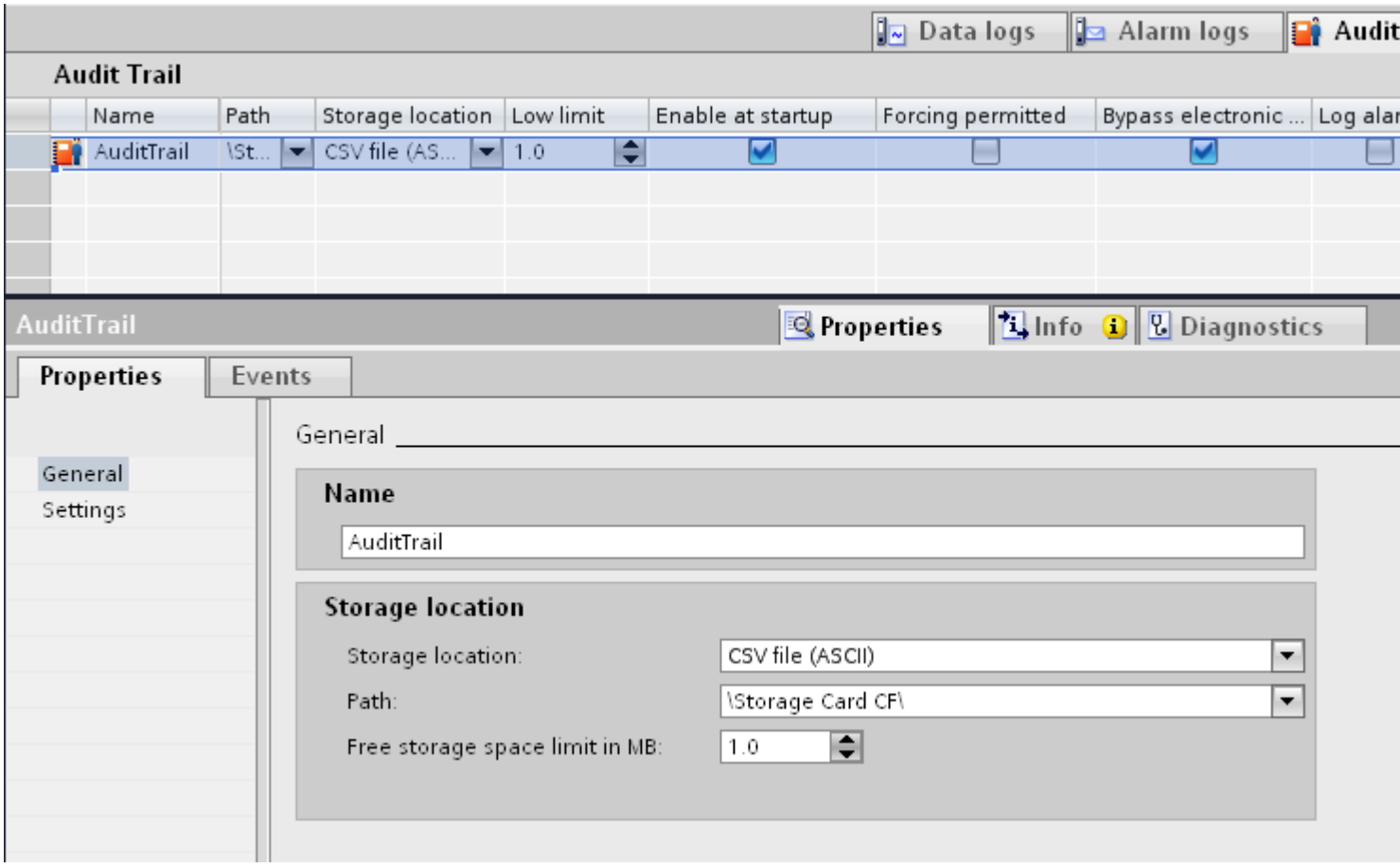
Only one Audit Trail can be created per HMI device.

Parameters that are assigned for the Audit Trail can be seen in the line. You can select or deselect the parameters displayed.

The parameters of an Audit Trail are also displayed and described in more detail in the inspector window.

General inspector window

You can set the following parameters under "Audit trail > Properties > General":



Name

- Under "Name", assign a name for the Audit Trail .
  Special characters are not permitted when assigning the name.

Storage location

- Storage location
  You can choose between:
  - RDB file
  - a CSV file (ASCII)
  - a TXT file (Unicode)

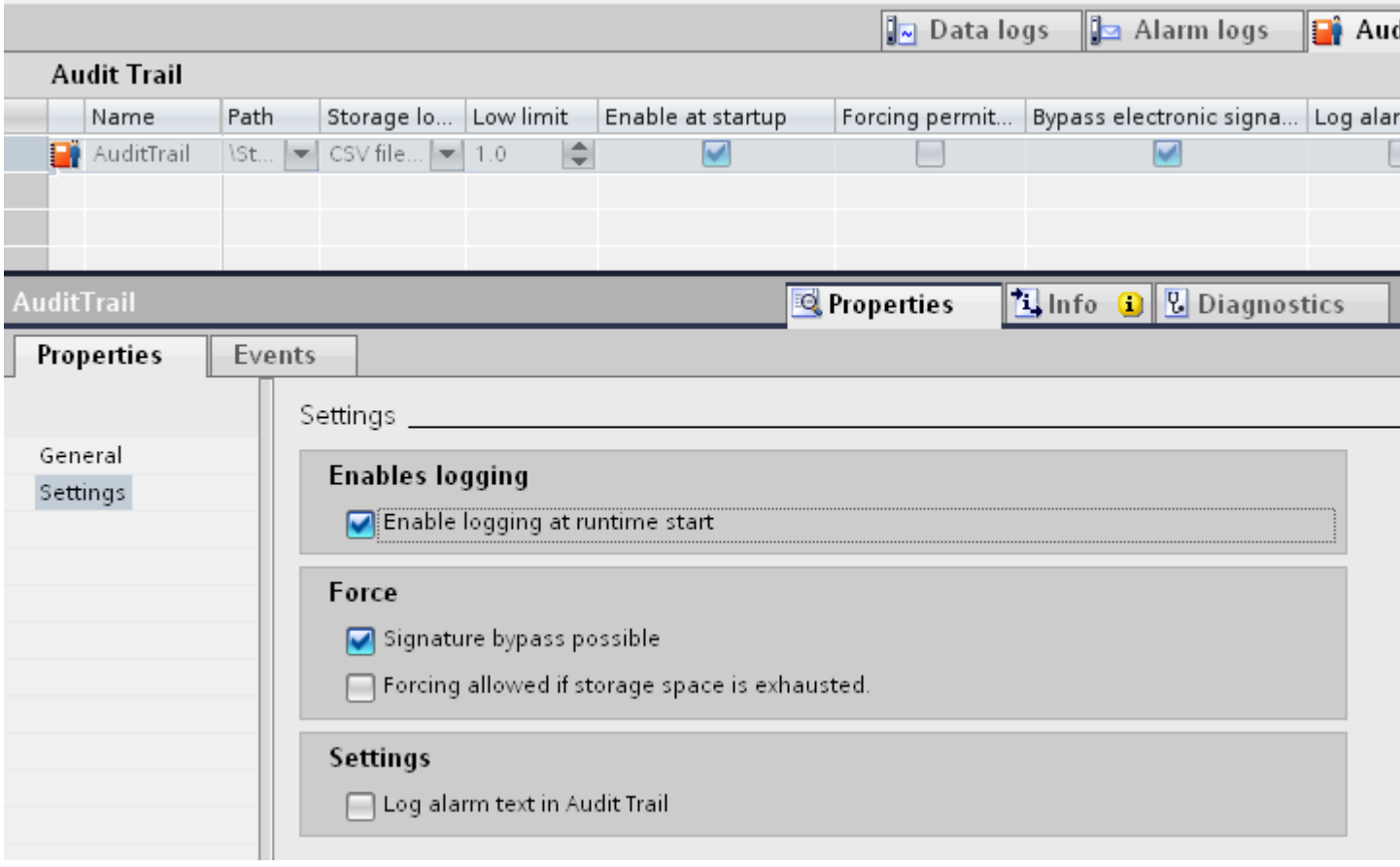
  Note

  Use "TXT (Unicode)" as storage location for logging Asian languages.

- Path
  Depending on the HMI device, enter the storage location of the Audit Trail under "Path".
- Minimum space in MB
  Specify the size of remaining storage space that triggers the "Little available memory" event.

Inspector window settings

You can set the following parameters under "Audit trail > Properties > Properties > Settings":



Logging activation

- Enable logging at runtime start

Forcing

- Bypass electronic signature
  Specifies whether the administrator is permitted to run operator actions without entering an electronic signature or comments.
- Forcing allowed if storage space is exhausted
  This function allows the administrator to interrupt audit trail logging in the following scenarios:
  - There is no free storage space available.
  - The storage medium is missing.
  - Access to required storage medium is not possible.

  You can thus prevent the process from stopping.
  After Forcing was carried out, the audit trail log switches off.
  After the end of "Forcing", the audit trail must be restarted with the system function "StartLogging".

Settings

- Logging alarm texts in the audit trail.

  Note

  Use "TXT (Unicode)" as storage location for logging Asian languages.

  For further details on setting the logging language, refer to the following section:
  Setting the audit trail language

See also

Format (Page 7064)

Setting the audit trail language

Introduction

The logging language for an Audit Trail is set in the runtime settings.

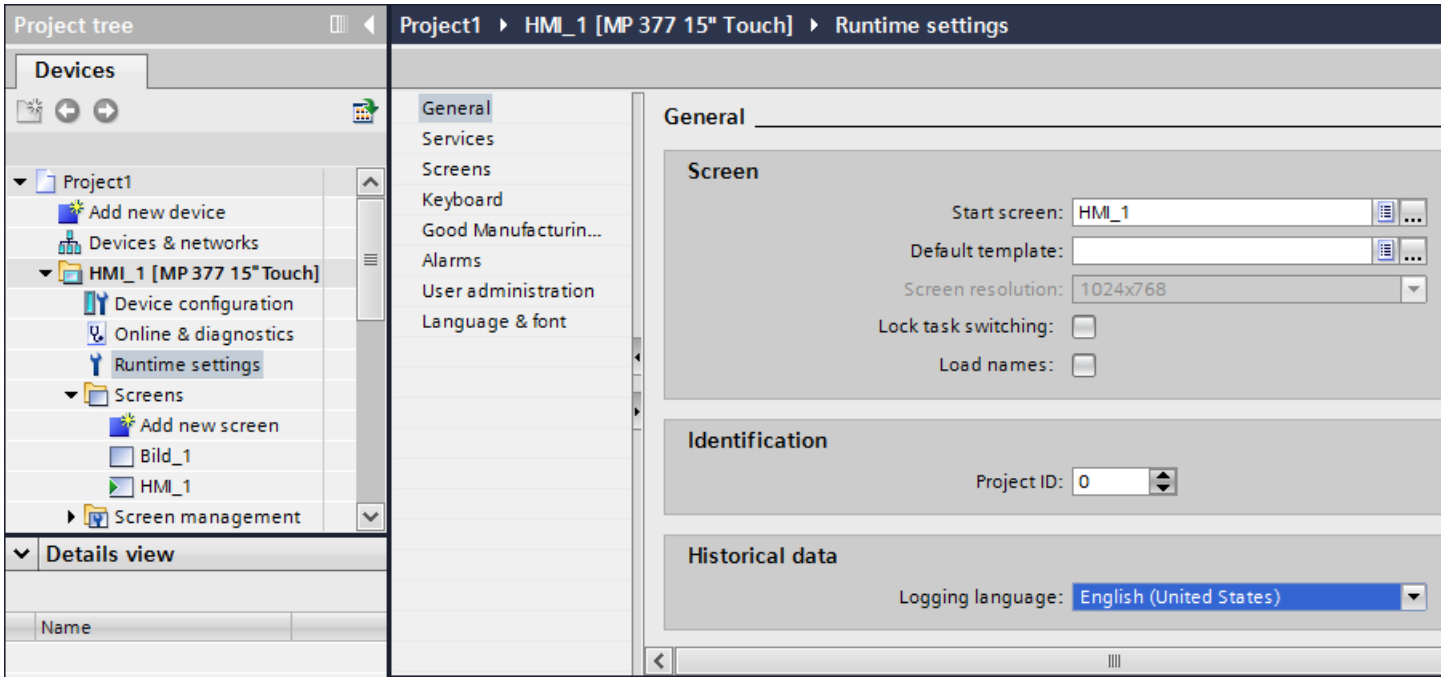
Procedure

1. Double-click on the HMI device in the project tree.
2. Double-click "Runtime settings".

3. Click "General".
4. Select the logging language in the "Logs" area.

Note

If an Asian language was selected, use the "TXT (Unicode)" storage location in the "Audit trail" editor.



Low free storage space

Low free storage space

Description

This event is triggered if the storage space available on the medium to which the Audit Trail is less than the configured minimum.

Free space critically low

Description

This event is triggered if the storage medium to which an Audit Trail is saved provides insufficient storage space due to hardware restrictions.

Configuring the "Low free storage space" event

Requirements

- A GMP- compliant configuration is enabled
- An Audit Trail is created

Procedure

1. Click on the Audit Trail in the "Audit Trail" editor.
2. In the Inspector window, click "Properties > General".
3. In the "Free storage space limit in MB" area, select a value that triggers the "Little free space" event.
4. Click on Events in the Inspector window.
5. Click on the "Low free storage space" event.
6. In the function list, specify a system function to execute when an "Overflow" event is triggered.

Logging the audit trail

Reporting an audit trail

Introduction

You can print a report of the operations saved in an Audit Trail. All recorded actions are included in the printout.

Requirements for reporting

The "Audit trail report" report object is available for the printout of an Audit Trail.

You can configure the report in the "Report" editor. The report object is only available if the "GMP-compliant configuration" option is set in the runtime settings of the HMI device.

If an Audit must be printed in runtime, initially the logging of Audit Trail must be stopped using the "StopLogging" system function.

Whilst an Audit Trail is being printed, no user actions are recorded. Ensure that no GMP-relevant user actions are executed whilst the logging is stopped.

After printing is complete without any errors, restart the audit trail using the "StartLogging" system function.

The header of the report is printed in the current runtime language. Change the runtime language to the logging language accordingly.

The logged data from Audit are printed in the configured runtime logging language.

In order to receive a complete report, the report object can also be used in a report in conjunction with the "Print alarm" and "Print recipe" report objects.

Audit Trail reporting

Introduction

You use the "Audit Trail" report object to configure a report for the output of Audit Trail contents to a printer.

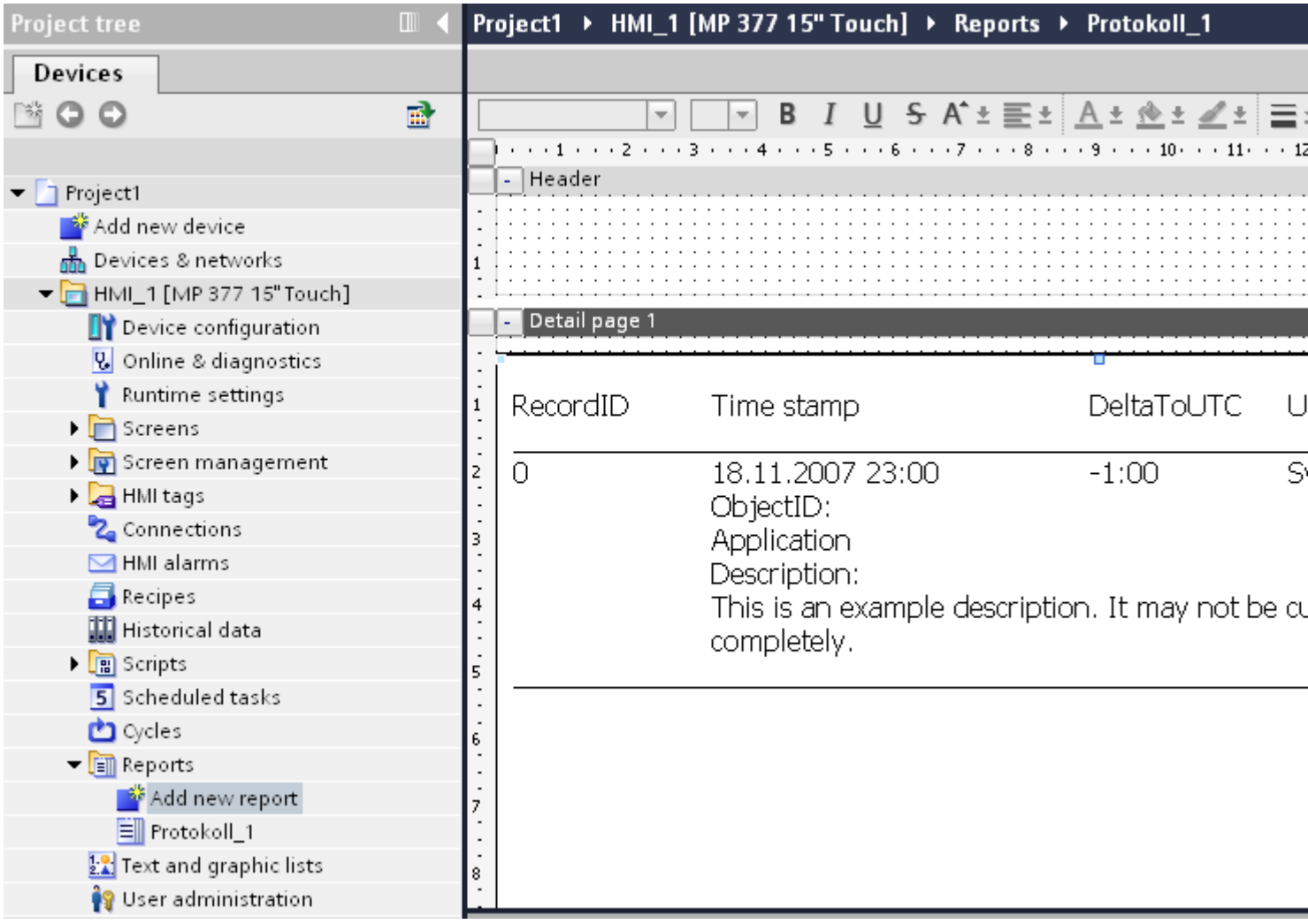
Once the printout is started in runtime, all entries currently contained in the Audit Trail are printed out.

Procedure

To create a report, proceed as follows:

1. Double-click on the "Reports" entry in the project tree.
2. Double-click "Add new report".
   A new report is created and opened in the "Report" editor.

3. Drag & drop the "Audit trail report" object under "Tools > Controls" to the report created.



4. Click on the "Audit trail report" object.
5. Change the object properties of the "Audit trail report" object in the inspector window.

Result

You have created a report for the printout Audit Trail.

Parameters for the audit trail report

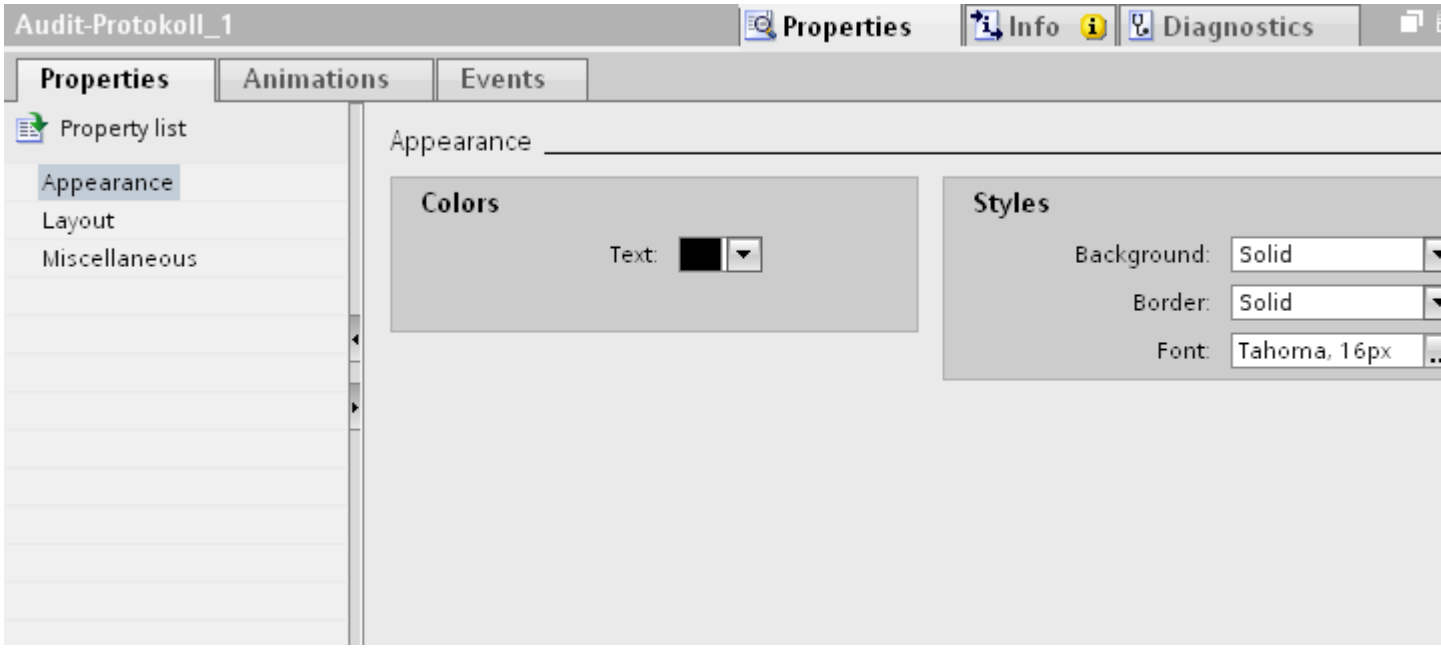
Introduction

The "Audit trail report" parameters can be edited in the inspector window.

Inspector window appearance

Click on the "Audit trail report" object.

Change the appearance of the "Audit trail report" object in the appearance area of the Inspector window under "Properties > Properties > Layout".



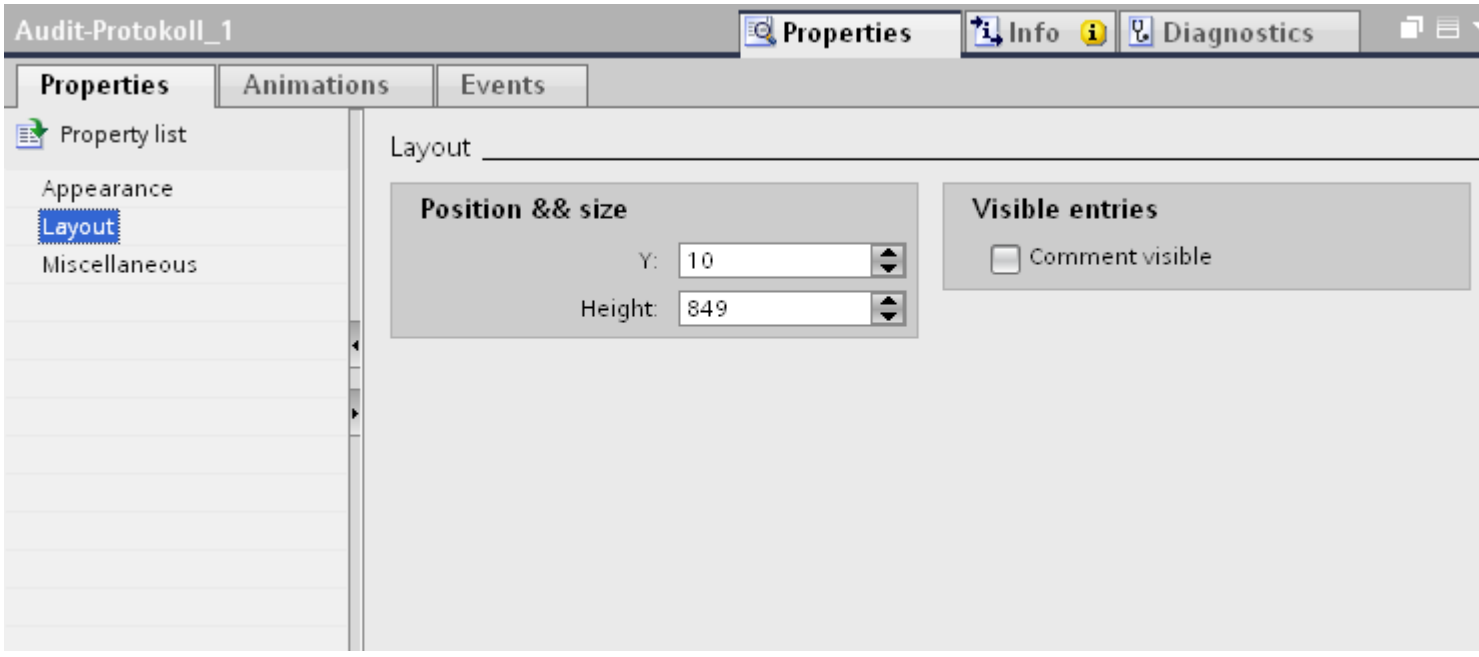
You can configure the foreground color, the background color, the style, and the font settings.

It is recommended to set a font size of 16 px for the output.

Inspector window layout

Click on the "Audit trail report" object.

Change the position and size of the "Audit trail report" object in the appearance area of the Inspector window under "Properties > Properties > Layout".



The "Audit trail report" object always fills the space down to the footer on the report page. If you change the height of the object, then you only change the distance of the object to the header. The report printout can involve a large amount of data.

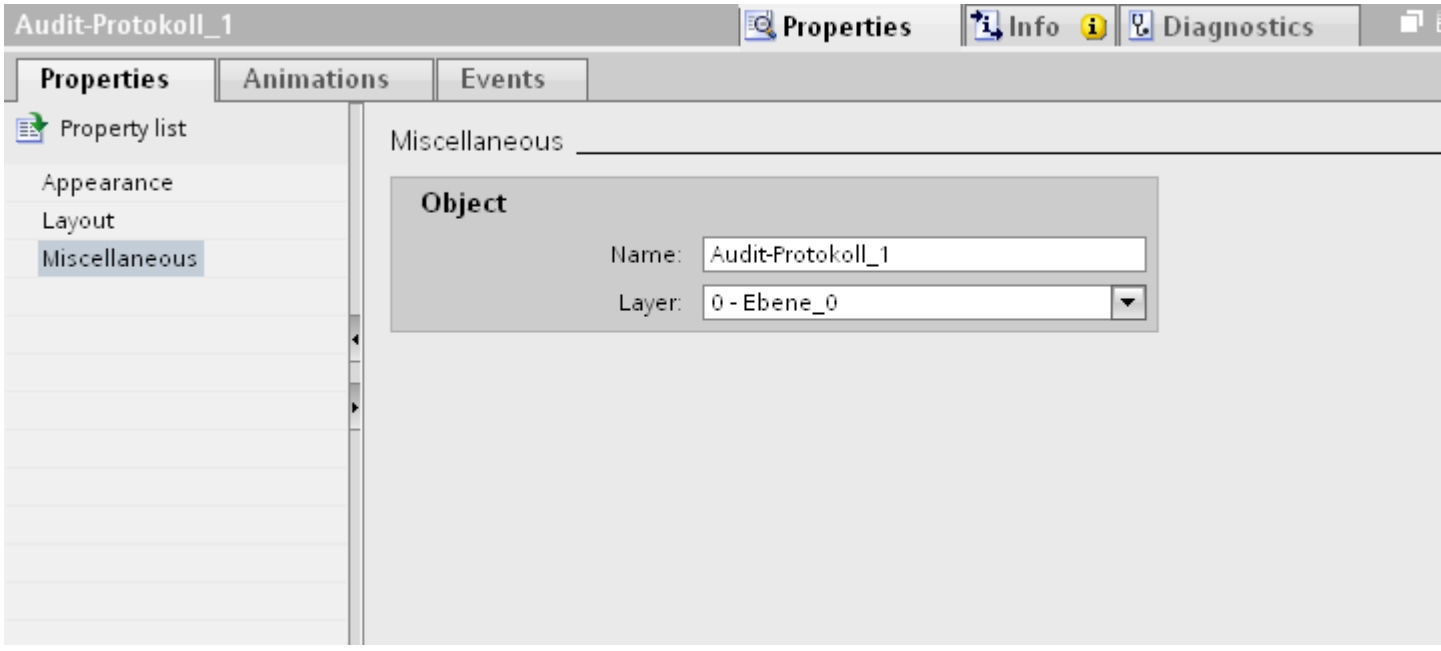
A page break is automatically inserted when the length of the page is exceeded to print out all data.

In the "Visible entries" area, it is determined whether comments are visible on the printed report.

Inspector window miscellaneous

Click on the "Audit trail report" object.

Change the name and layer position of the "Audit trail report" object in the appearance area in the Inspector window under "Properties > Properties > Miscellaneous".



Printing out an audit trail report

Introduction

Since logging of the Audit must be stopped in order to print out an Audit Trail, a few details regarding this procedure must be noted.

The following steps must be taken to print out an audit trail in a report file on a printer:

- Stop logging using the "StopLogging" system function.
- Start the printout using the "PrintReport" system function.
- Check if the printing is successfully completed.
- If needed, move or delete the Audit Trail using the system functions "ArchiveLogFile" or "DeleteLog".
- Start the logging of Audit by selecting the "StartLogging" system function.

Note

Make sure that the Audit Trail has been printed completely before you delete the Audit Trail.

Requirements

- "GMP compliant configuration" has been enabled.
- A report for the printout of an Audit Trail has been configured.
- The screen for the operator control to be configured is open.

Procedure

1. Add a button to the screen and select "Events > Click" in the Properties window.
2. In the function list, assign the "StopLogging" system function to the "Click" event and select your Audit Trail log.
3. Insert an additional button and assign the "PrintReport" system function to the "Click" event of this button.
4. Configure the "StartLogging" system function in the same function list.
5. Assign unique labels to the buttons.
6. Save the project.

Result

You have configured the required buttons and system functions. The operator can perform the operating tasks described in the introduction during runtime to print out an Audit Trail report.

Note

You can also insert the report objects for the output of alarms and recipes in the report for the printout of an Audit Trail. However, since GMP-relevant operations and system processes are not recorded while you are printing, you should preferably print the Audit Trail in a separate operation.

Evaluating an audit trail

Evaluating audit trails

Introduction

The Audit Trail has been saved to the memory card of the HMI device and is also read only.

The Audit Trail is protected by a checksum. This checksum ensures that the entry has not been modified at any later time.

There are two possible ways to evaluate the Audit Trail:

- Use the "Audit Viewer":
  You can easily evaluate the Audit Viewer for external analysis on an Office PC with the help of the Audit Trail.
- Use the "HmiCheckLogIntegrity" DOS program:
  The DOS program makes it possible to carry out an automatic check of the Audit Trail using the return values.

Evaluating Audit Trails in AuditViewer

Introduction

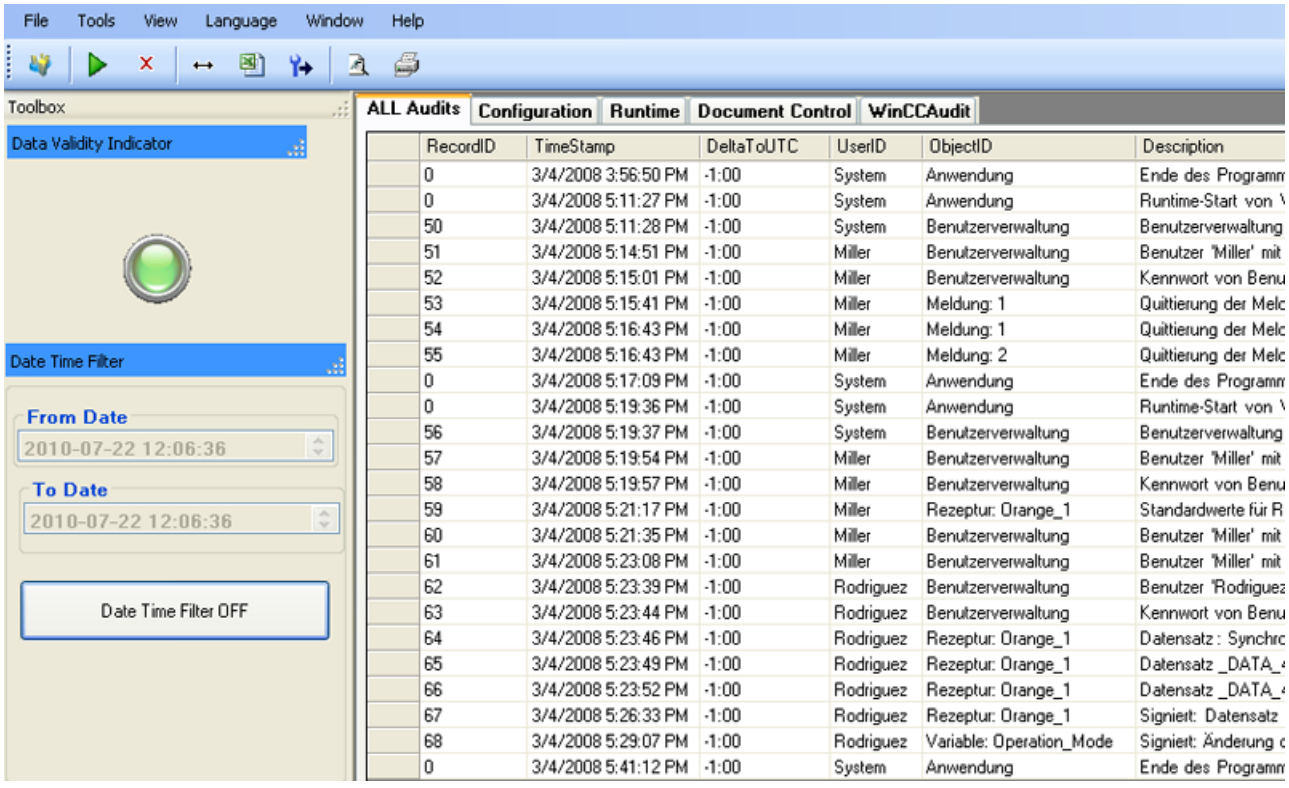
The Audit Viewer allows you to evaluate all Audit Trail data in a table.

Requirements

- Audit Viewer is installed
- The Audit Traillog is located on the computer which has Audit Viewer installed.

Procedure

1. Start the Audit Viewer on the configuration PC:
   "Start > SIMATIC > Audit Viewer > Audit Viewer"
   This path may be different on your operating system version.
2. Click the button.
3. Load the Audit Trail:



The "Data Validity Indicator" is lit up in green to indicate that the loaded Audit Trail has not been manipulated.

Each entry in the Audit Trail is time-stamped to allow precise tracking of operator actions. In addition to system events, such as the attempt to import a password list, the system also records failed logon attempts:

See also

Evaluating the checksum of log data (Page 4261)

Evaluating the checksum of log data (Page 4346)

Evaluating Audit Trails with DOS program

Introduction

Long-term archiving on a server allows an Audit Trail to be checked automatically using return values in a script.

In addition the programmer can integrate the check using the DOS program "HmiCheckLogIntegrity" into the archiving process. "HmiCheckLogIntegrity" then provides the following return values:

- < 0: Different errors, for example, wrong file format or no file exists.
- 1: The checked Audit Trail is valid.
- > 0: The first line that was manipulated will be returned.

Audit Trail logging is only continued if the return value is "1". In both error cases, the administrator or the shift supervisor can be informed.

HmiCheckLogIntegrity

The "HmiCheckLogIntegrity.exe" DOS program is in the installation directory under:

"SIEMENS > Automation > WinCC Runtime Advanced"

Audit trail logging concept

Format

Format - Audit Trail

On an HMI device with "GMP compliant configuration", all events which are relevant to the audit are recorded at runtime in the Audit Trail. You have several format options.

Selection is dependent on the display program and the runtime language used:

- RDB file
  Data is saved with quick access in a proprietary database.
  If you require maximum read performance in Runtime, use the "RDB file" storage location.
- CSV file
  To view and evaluate a CSV file use, e.g. Microsoft Excel on your PC.

  Note

  Double quotation marks or several characters are not permitted as list separators for the storage site "File - CSV (ASCII)". You can find the settings for list separators under "Start > Settings > Control Panel > Regional and Language Options".

- TXT file
  This file format supports all characters that can be used in WinCC. For editing, you will need software that can save files in Unicode, such as Notepad.

  Note

  Use "File - TXT (Unicode)" to log Asian languages.

Audit Trail with checksum

The following files are generated under special circumstances:

- *.keep
  - If a log is started without checksum and will be continued with a checksum.
  - If you update WinCC with a service pack or a new version and the Audit Trail or the log is continued with the checksum.
  - The content of the keep file will remain the same when compared with the original log file.

    Note

    Before you update WinCC with a Service Pack or a new version, exit and save the Audit Trail or the logs using checksum. After WinCC is updated, the audit trail or logs will be continued with new files using checksum.

- *.bak
  - If runtime has determined a serious, irregular problem in the file.

Storage location and medium

Storage location and medium

Depending on the hardware configuration of the HMI device, the data may be logged locally (on the hard disk of a PC or on the storage card of a panel) or, if present, on a network drive.

Note

Logging on network drives

We do not recommend that you log audit trails directly on a network drive. Power supply can be interrupted at any time. This means there is no guarantee for a reliable operation of logs and audit trails.

We recommend you save the logs on your local hard drive, or on a storage medium of the HMI device. Use the system function "ArchiveLogFile" to save the logs long-term on a network drive.

A "GMP compliant configuration" cannot be operated at runtime to the full extent unless it is possible to save all user actions which are relevant to the audit to the Audit Trail. It must be ensured that sufficient storage space is available for the Audit Trail and that the connection to the storage location for the Audit Trail is not disturbed.

Error-handling with insufficient free storage space

If there is insufficient storage space, your project can be configured so that the administrator has an option of continuing the process without logging in the audit trail (forcing).

Error-handling if there is no storage medium or the connection to the server is interrupted

All audit-relevant user actions are blocked if insufficient storage space is available for the Audit Trai, e.g. due to missing storage medium.

Blocking is canceled as soon as the storage location for the Audit Trail is available again. The block can be skipped by "forcing".

Error-handling with long-term logging

If the audit trail must be moved to a server for long-term logging and the connection to the server is interrupted at this time, the following error-handling is required:

The system closes the audit trail and renames it. The system attempts to send the renamed audit trail to the server again in the background.

If disruption in the connection to the server persists, you receive a system alarm telling you that the connection is down. Then the system attempts to send the renamed audit trail every 300 seconds.

The attempt to transmit the data is repeated until successfully completed. The data is also transmitted after a restart of the HMI device.

"Forcing"

If the storage space for the audit trail is insufficient, for example if there is no storage medium or it is full, all audit-relevant user actions are automatically blocked.

In this case the audit trail logging can be configured to give an administrator the option of continuing to operate the system without logging the audit trail (forcing).

The administrator can also be given the option of running the system quickly out of a critical state in case of emergency. In this case the administrator can operate the system without the requirement to input the required electronic signatures or comments.

If the administrator uses the "forcing" function, this is logged as the last entry in the audit trail.

After the end of forcing, the audit trail must be restarted with the system function "StartLogging".

Protection mechanisms

Protective mechanisms to prevent changes to audit trail data

The audit trail data are protected against deliberate or accidental changes:

- The directory in which the audit trail is saved can only be accessed with special rights.
- The audit trail files are write-protected.
- Each data record contains a checksum that can be used to detect a change of its contents. This checksum also ensures that the number of lines has not changed in the audit trail file.

Use the "HmiCheckLogIntegrity" tool, included in the audit option, to check whether an audit trail has been changed:

Evaluating Audit Trails with DOS program

WinCC upgrade

WinCC upgrade

Before you update WinCC with a Service Pack or a new version, you will have to exit and save the Audit Trail or the logs with checksum. After WinCC is updated, the audit trail or logs with checksum will be continued with new files.

Make sure that the logs are started at a defined state with the new version.

Audit trail behavior in runtime

Effects in runtime

The configuration in Audit Trail has the following effects in runtime, depending on the configuration:

- Audit relevant user actions (such as tag changes and recipe changes) are recorded in an audit trail.
- "Enable logging at runtime start" check box enabled:
  The audit trail is started with runtime.
- "Forcing" group, "Allowed if storage space has been exhausted" check box enabled:
  A user with administrator rights can use "forcing" to run operations on the plant even though the audit trail can no longer be logged because of storage space limitations. Interrupting the audit trail prevents the process from being stopped.
  If the check box "Signing may be bypassed" is enabled, the administrator is not required to input electronic signatures, acknowledgments or comments for operator actions that would normally require signing, acknowledgment or comment.
- If the storage space available for the Audit Trail is less than the configured "Minimum storage space in MB", the function list configured for the "Low free storage space" event will be processed.
- If there is insufficient storage space for the audit trail because of hardware limits, the function list configured for the "Free space critically low" event will be processed.

12.15.1.4 Configuring audit functions

Logging tag value changes

Tag value change

Changes to tag values

User actions in runtime are recorded in a Audit Trail once "GMP compliant configuration" has been enabled.

When you configure a GMP compliant project, you specify which tags must meet the requirements of Good Manufacturing Practice (GMP).

If the user changes the value of a GMP-relevant tag in runtime, the value change action is logged in the Audit Trail .

Note

The action of changing the value of GMP-relevant tags made by the PLC, or a system function, is not logged in the Audit Trail.

The system functions used to change the values of GMP-relevant tags which are assigned to events that identify a direct user action will be logged.

In addition, configure the "NotifyUserAction" system function to make a manual entry in the Audit Trail or to prompt the user for an electronic signature, an acknowledgment, or a comment.

With the "NotifyUserAction" system function only the value changes that are directly triggered by the event to which the system function is configured are entered in the Audit Trail. For example, if additional tags are changed by changing the value of one tag, the additional value changes are not logged in the Audit Trail.

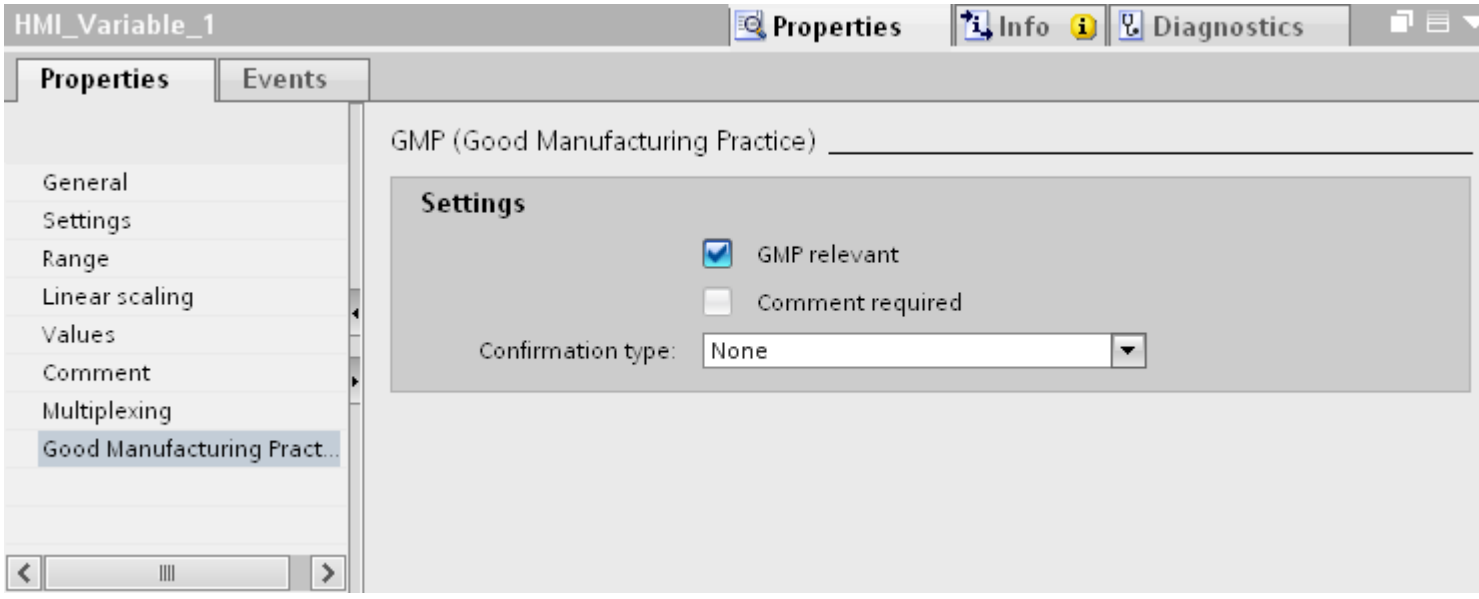
Logging tag value changes

Requirement

- "GMP compliant configuration" has been enabled.
- The tags for which you want to configure the GMP settings are created.
- The property view is open.

Procedure

1. Open the HMI tags editor and select the tag for which you want to make GMP settings.



2. Click "GMP relevant" under "Properties > Properties > GMP" in the Inspector window.
3. Specify how the user must confirm a value change in the "Confirmation type" selection field:
   - "Electronic signature"
     If the user's electronic signature is required.
   - "None"
     If the value change is to be logged in the Audit Trail without user confirmation.
   - "Acknowledgment"
     If user acknowledgement of the value change is required.
4. Enable the "Comment required" check box if the user is required to input a comment as well as an electronic signature or acknowledgment.
   This check box is only enabled if "Electronic signature" or "Acknowledgment" is specified under "Type of confirmation".

Result

If the user changes the value of a GMP-relevant tag in runtime, the value change is entered in the Audit Trail.

Effects of tag change

Effects in runtime

The configuration has the following effects in runtime depending on the properties of the GMP-relevant tags:

- If the user changes the value of a GMP-relevant tag in runtime, the value change is entered in the Audit Trail.
- Electronic signature
  If "Electronic signature" is specified as the "Type of confirmation", the user must log every user-related value change of the tags using an electronic signature. Otherwise the value change will be rejected.
  The user name used to sign the change is logged in the audit trail.
- Acknowledgement
  If "Acknowledgment" is specified as the "Type of confirmation", the user must acknowledge every user-related value change of the tags. Otherwise the value change will be rejected.
  The acknowledgement is logged in the Audit Trail.
- Comments
  If the "Comment required" check box is enabled, the user must also comment every user-related value change of the tags, in addition to acknowledgment or input of the electronic signature. Otherwise the value change will be rejected.
  The entered comment is logged in the Audit Trail.

Logging recipe data record changes

Recipe data changes

Changes to recipe data

User actions in runtime that are relevant to the quality of the process, such as changes of tag values or recipe values, are recorded in an Audit Trail once "GMP compliant configuration" has been enabled.

You specify during configuration which recipes must meet the requirements of "Good Manufacturing Practice" (GMP).

For GMP-relevant recipes, the following operations during runtime are recorded in the Audit Trail:

- Storing after changing and creating recipe data records
- Transfer of recipe data records to the PLC and from the PLC
- For recipe tags: Changing the setting for the synchronization of the tag values with the PLC ("offline"/"online") if the recipe tags are configured as "GMP-relevant".

Note

Differences in the Audit Trail with recipe display and recipe screen

If you use a recipe screen to save recipe data, enable the "GMP-relevant" property for the recipe tags. If the user changes the value of a GMP-relevant recipe tag in runtime, the changed value is recorded in the Audit Trail. You can also configure for the tag to require the user to confirm the value change with an electronic signature and enter a comment.

If you use a recipe display to edit data records of a GMP-relevant recipe, the Audit Trail includes a record of which recipe data records were saved or sent to the PLC. The value changes to the recipe tags are not logged in the Audit Trail. Use the "ExportDataRecords" system function to save the value change to data records in a csv file.

If you want to make changes to recipe data records in conformance to the FDA, disable "Enable edit mode" in the recipe view. Use the recipe screen and "GMP relevant" recipe tags.

If you export recipe data records in a regulated project, you can assign the recipe data with a checksum. When you later import the recipe data back, you can use the checksum to determine if the recipe data has changed. The following system functions are available for exporting and importing recipe data with a checksum:

- "ExportDataRecordsWithChecksum"
- "ImportDataRecordsWithChecksum"

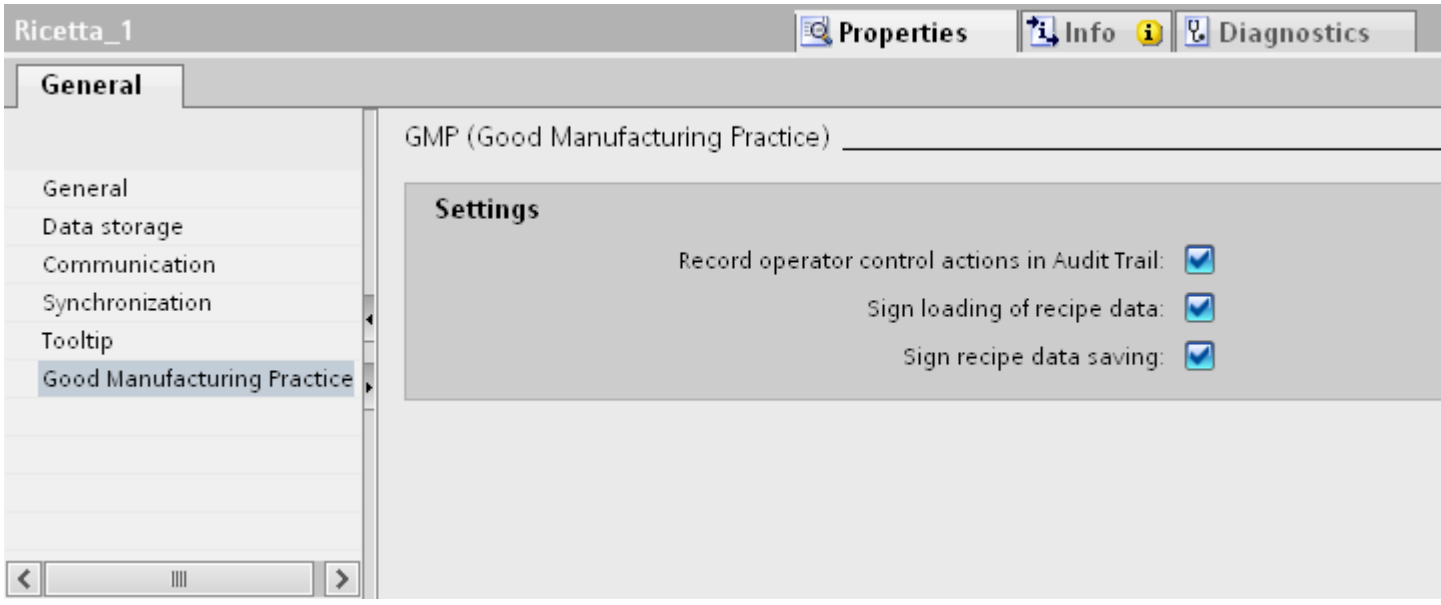
Logging recipe data changes

Requirement

- "GMP compliant configuration" has been enabled.
- The recipes for which you want to configure the GMP settings are created.
- The property view is open.

Procedure

1. Open the recipe editor and select the recipe for which you want to make GMP settings.



2. Click "GMP relevant" under "Properties > Properties > GMP" in the Inspector window.
3. Under "Settings", select the following:
   - "Record operations in audit trail:"
     If all user actions in runtime that affect this recipe are to be recorded in the Audit Trail.
   - "Sign loading of recipe data":
     If the user is required to confirm the transfer of recipe data records to or from the PLC using an electronic signature.
   - "Signature for saving recipe data: "
     If the user is required to confirm recipe data record saving with an electronic signature.

Result

If the user works with the GMP-relevant recipe in runtime, the change is entered in the Audit Trail.

Effects of recipe data change

Effects in runtime

The configuration has the following effects in runtime depending on the properties of the GMP-relevant recipes:

Entries in the Audit Trail

Entries are made in the following cases:

- You create and save new recipe data records of a GMP-relevant recipe at runtime.
- You edit recipe data records of a GMP-relevant recipe and save your changes at runtime.
- Recipe data records are transferred to the PLC or recipe data are read from the PLC.
- The "SetRecipeTags" system function is used to change the setting for synchronization of the tag values with the PLC.
  This concerns the following changes:
  - "offline" to "online"
  - "online" to "offline"

Electronic signature

The user must enter an electronic signature in the following cases depending on the configuration:

- The "Sign loading of recipe data" check box is set:
  Signature for the transfer of recipe data records to the PLC. Signing is always required if the transfer is triggered with the recipe display functions and even if it is triggered with "SetDataRecordToPLC" system functions.
- The "Sign saving of recipe data" check box is set:
  Sign saving of recipe data records. The signature is always required if the save is triggered with the system functions of the recipe display, and even if it is triggered with the "SetDataRecordToPLC" or "SaveDataRecord" system function.
- The user name used to sign the change is logged in the audit trail.

Note

The triggering of the "ImportDataRecords" system function is entered in the Audit Trail but does not require a signature or a comment. In addition, call the "NotifyUserAction" system function to request an electronic signature with or without user comment.

Logging user actions

User actions with GMP-compliant configuration

Introduction

In GMP compliant configuration, user actions and system operations in runtime that are relevant for the quality of the process are recorded in an Audit Trail .

For example, a user logon to the system or the change of a tag value are saved in the log.

In runtime, user actions are saved in an Audit Trail under the following conditions:

- "GMP compliant configuration" has been enabled
- A user is logged on to the system

Logging modes

Automatic logging of user actions

The following user actions are recorded in runtime without the need for additional configuration steps if "GMP compliant configuration" is enabled:

- User Administration
  - Logon and logoff of users
  - Import of user administration
- Alarm system
  - All alarms that are acknowledged by the user.
    If an alarm from an alarm group is acknowledged, an entry is made in the Audit Trail indicating that all other alarms of this group have been acknowledged.
  - All acknowledgment attempts of the user

  Note

  Logging alarm text

  To log alarm texts, select the "Log alarm texts in Audit Trail" option in the Audit Trail editor.

- Log operations
  - Starting and stopping a log
  - Opening and closing all logs
  - Deleting a log
  - Starting a sequence log
  - Copying a log
  - Long-term logging of a log

Configuration-dependent logging

The following processes are logged depending on the configuration of the recipes and the tags of the project:

- Change values of GMP-relevant tags by the user
- for GMP-relevant recipes
  - You create and save new recipe data records of a GMP-relevant recipe at runtime.
  - You edit recipe data records of a GMP-relevant recipe and save your changes at runtime.
  - Transfer of recipe data records to the PLC and from the PLC
  - For recipe tags: Changing the setting for the synchronization of the tag values with the PLC ("offline"/"online")

In addition to logging user actions you can configure tags and recipes to require the user to confirm or acknowledge specific actions with an electronic signature or add a comment to the change.

Manual logging by means of "NotifyUserAction" system function

This system function is used to record actions in the Audit Trail that are not automatically entered in the Audit Trail. This system function is also used to request the user to enter an electronic signature for the action.

Configuring the "NotifyUserAction" system function

Introduction

This system function is used to log user actions that are not entered automatically entered in the Audit Trail. This system function can also used to request an acknowledgment, or an electronic signature for the user's action.

In this example, the system function is assigned to a button. All operations with this button are logged to the Audit Trail.

Requirements

- "GMP-compliant configuration" has been enabled.
- You created the object that is to be assigned the system function.
  A button is used in this example.
- The properties window is open.

Procedure

1. Click the button.
2. Click on "Events" in the Inspector window.
3. Assign the "NotifyUserAction" system function to the "Click" event.

GMP-compliant user administration

SIMATIC Logon

To run a central user and user group administration for several applications or HMI devices, activate SIMATIC Logon.

For more information on user administration and SIMATIC Logon, refer to the following chapter:

Managing users on the server (Page 4543)

Logging system functions

Introduction

If system functions are triggered in runtime, this is recorded in the Audit Trail for some system functions. If specific system functions are used on a GMP-relevant object, the user must confirm the triggering.

Some system functions are not supported when using Audit. If you use these system functions in your project, you are solely responsible for them.

The following table shows which system functions are Audit-relevant and whether the user's signature is required:

System functions and Audit

Function (call in script)	Effect of /Audit
StartLogging (StartLogging)	entered in Audit Trail
StopLogging (StopLogging)	entered in Audit Trail
ClearLog (ClearLog)	entered in Audit Trail
StartNextLog (StartNextLog)	entered in Audit Trail
CloseAllLogs (CloseAllLogs)	entered in Audit Trail
OpenAllLogs (OpenAllLogs)	entered in Audit Trail
LogTag (---)	---
CopyLog (CopyLog)	entered in Audit Trail
ActivateScreen (ActivateScreen)	---
ActivateScreenByNumber (ActivateScreenByNumber)	---
ActivatePreviousScreen (ActivatePreviousScreen)	---
SetBitInTag (SetBitInTag)	entered in Audit Trail when tag is GMP-relevant Signature is mandatory, depending on the tag configuration
ResetBitInTag (ResetBitInTag)	entered in Audit Trail when tag is GMP-relevant Signature is mandatory, depending on the tag configuration

Function (call in script)	Effect of /Audit
InvertBitInTag (InvertBitInTag)	entered in Audit Trail when tag is GMP-relevant Signature is mandatory, depending on the tag configuration
SetBit (SetBit)	entered in Audit Trail when tag is GMP-relevant Signature is mandatory, depending on the tag configuration
ResetBit (ResetBit)	entered in Audit Trail when tag is GMP-relevant Signature is mandatory, depending on the tag configuration
InvertBit (InvertBit)	entered in Audit Trail when tag is GMP-relevant System function must not be applied to tags that require signing or comment.
SetBitWhileKeyPressed (---)	entered in Audit Trail when tag is GMP-relevant Signature is mandatory, depending on the tag configuration
SetDataRecordToPLC (SetDataRecordToPLC)	entered in Audit Trail if the recipe is GMP-relevant Signing required depending on recipe configuration
GetDataRecordTagsFromPLC (GetDataRecordFromPLC)	entered in Audit Trail if the recipe is GMP-relevant
ImportDataRecords (ImportDataRecords)	entered in Audit Trail if the recipe is GMP-relevant
ImportDataRecordsWithChecksum (ImportDataRecord-sWithChecksum)	entered in Audit Trail if the recipe is GMP-relevant
ExportDataRecords (ExportDataRecords)	---
ExportDataRecordsWithChecksum (ExportDataRecord-sWithChecksum)	---
LoadDataRecord (LoadDataRecord)	entered in Audit Trail if the recipe is GMP-relevant Signing required depending on recipe configuration
SaveDataRecord (SaveDataRecord)	entered in Audit Trail if the recipe is GMP-relevant Signing required depending on recipe configuration
SetDataRecordTagsToPLC (SetDataRecordTagsToPLC)	entered in Audit Trail if the recipe is GMP-relevant Signing required depending on recipe configuration
GetDataRecordTagsFromPLC (GetDataRecordTags-FromPLC)	entered in Audit Trail if the recipe is GMP-relevant

Function (call in script)	Effect of /Audit
SetRecipeTags (SetRecipeTags)	entered in Audit Trail if the recipe is GMP-relevant
GetDataRecordName (GetDataRecordName)	---
ClearDataRecordMemory (ClearDataRecordMemory)	Not supported
ClearDataRecord (ClearDataRecord)	entered in Audit Trail if the recipe is GMP-relevant
PrintScreen (PrintScreen)	---
PrintReport (PrintReport)	---
RecipeViewSaveDataRecord (---)	entered in Audit Trail if the recipe is GMP-relevant Signing required depending on recipe configuration
RecipeViewSaveAsDataRecord (---)	entered in Audit Trail if the recipe is GMP-relevant Signing required depending on recipe configuration
RecipeViewNewDataRecord (---)	---
RecipeViewClearDataRecord (---)	entered in Audit Trail if the recipe is GMP-relevant
RecipeViewGetDataRecordFromPLC (---)	entered in Audit Trail if the recipe is GMP-relevant
RecipeViewSetDataRecordToPLC (---)	entered in Audit Trail if the recipe is GMP-relevant Signing required depending on recipe configuration
RecipeViewSynchronizeDataRecordWithTags (---)	entered in Audit Trail if the recipe is GMP-relevant Signing required depending on recipe configuration
RecipeViewRenameDataRecord (---)	entered in Audit Trail if the recipe is GMP-relevant
RecipeViewBack (---)	---
RecipeViewOpen (---)	---
RecipeViewMenu (---)	---
TrendViewScrollForward (---)	---
TrendViewScrollBack (---)	---
TrendViewExtend (---)	---
TrendViewCompress (---)	---
TrendViewBackToBeginning (---)	---
TrendViewStartStop (---)	---
TrendViewSetRulerMode (---)	---
TrendViewBackToBeginning (---)	---
StatusForceGetValues (---)	Not supported
StatusForceSetValues (---)	Not supported
AlarmViewAcknowledgeAlarm (---)	entered in Audit Trail
AlarmViewEditAlarm (---)	---

Function (call in script)	Effect of /Audit
AlarmViewShowOperatorNotes (---)	---
HTMLBrowserBack (---)	Not supported
HTMLBrowserForward (---)	Not supported
HTMLBrowserRefresh (---)	Not supported
HTMLBrowserStop (---)	Not supported
ScreenObjectCursorUp (---)	---
ScreenObjectCursorDown (---)	---
ScreenObjectPageUp (---)	---
ScreenObjectPageDown (---)	---
PressButton (---)	---
ReleaseButton (---)	---
SmartClientViewConnect (---)	Not supported
SmartClientViewDisconnect (---)	Not supported
SmartClientViewReadOnlyOn (---)	Not supported
SmartClientViewReadOnlyOff (---)	Not supported
SmartClientViewRefresh (---)	Not supported
SmartClientViewLeave (---)	Not supported
ShowAlarmWindow (ShowAlarmWindow)	---
ClearAlarmBuffer (ClearAlarmBuffer)	---
ShowSystemAlarm (ShowSystemAlarm)	---
SetAlarmReportMode (SetAlarmReportMode)	---
Logoff (Logoff)	entered in Audit Trail
GetPassword (GetPassword)	---
GetGroupNumber (GetGroupNumber)	---
ExportImportUserAdministration (ExportImportUserAdministration)	Import of user administration is entered in Audit Trail Export is not entered in Audit Trail
Logon (Logon)	entered in Audit Trail
GetUserName (GetUserName)	---
TraceUserChange (---)	---
ShowLogOnDialog (---)	---
LinearScaling (LinearScaling)	entered in Audit Trail when tag is GMP-relevant Signature is mandatory, depending on the tag configuration
InverseLinearScaling (InverseLinearScaling)	entered in Audit Trail when tag is GMP-relevant Signature is mandatory, depending on the tag configuration
IncreaseFocusedValue (---)	---
DecreaseFocusedValue (---)	---
OpenCommandPrompt (OpenCommandPrompt)	Not supported
OpenControlPanel (OpenControlPanel)	Not supported
ActivateCleanScreen (---)	---

Function (call in script)	Effect of /Audit
AdjustContrast (---)	---
CalibrateTouchScreen (CalibrateTouchScreen)	---
OpenScreenKeyboard (OpenScreenKeyboard)	---
OpenTaskManager (OpenTaskManager)	Not supported
BackupRAMFileSystem (BackupRAMFileSystem)	Not supported
SetAcousticSignal (SetAcousticSignal)	---
ShowOperatorNotes (ShowOperatorNotes)	---
AcknowledgeAlarm (AcknowledgeAlarm)	entered in Audit Trail
GoToHome (GoToHome)	---
GoToEnd (GoToEnd)	---
EditAlarm (EditAlarm)	---
DirectKeyScreenNumber (---)	Not supported
DirectKey (---)	Not supported
SetDeviceMode (SetDeviceMode)	entered in Audit Trail
SetDisplayMode (SetDisplayMode)	---
SetConnectionMode (SetConnectionMode)	entered in Audit Trail
SetScreenKeyboardMode (SetScreenKeyboardMode)	---
ChangeConnection (ChangeConnection)	Not supported
SetLanguage (SetLanguage)	---
SetWebAccess (---)	Not supported
StartProgram (StartProgram)	Not supported
ShowSoftwareVersion (ShowSoftwareVersion)	---
SimulateTag (---)	Not supported
StopRuntime (StopRuntime)	entered in Audit Trail
ControlWebServer (ControlWebServer)	Not supported
ControlSmartServer (ControlSmartServer)	Not supported
OpenInternetExplorer (OpenInternetExplorer)	---
SendEMail (SendEMail)	---
UpdateTag (---)	---
ClearAlarmBufferProTool (ClearAlarmBufferProtoolLegacy)	Not supported
Encoding(Encode)	entered in Audit Trail when tag is GMP-relevant Signature is mandatory, depending on the tag configuration
EncodeEx(Encode)	entered in Audit Trail when tag is GMP-relevant Signature is mandatory, depending on the tag configuration

12.15.1.5 Performance features of GMP relevant configuration

Supported HMI devices

Supported HMI devices

The qualification "GMP relevant configuration" can be configured for the following HMI devices:

- TP 277
- OP 277
- MP 277
- Mobile Panels 277
- MP 377
- Comfort Panels
- Panel PCs with WinCC RT Advanced
- WinCC RT Advanced

Note

The qualification "GMP" is not supported by WinCC RT Professional.

Restrictions

Restrictions

The following functions and configurations cannot be used simultaneously with the qualification "GMP relevant configuration":

- "Status/Force" object
- PN direct keys
- DP DirectKey
- Option /Sm@rtServer
- /Sinumerik option
- The functional scope of the HMI devices is only available to a restricted degree in some circumstances because of the limited storage space

- Events of screen objects
  You can set mandatory acknowledgement of important user actions in runtime, such as changing tag values. If you assign an event which has to be acknowledged to a screen object, you may not assign any other events to this graphic object.
  When the event of a screen object is assigned actions which open a user dialog (such as change of a tag value with mandatory acknowledgement), you may not be able to execute these actions at other events.
- Controlling GMP-relevant tags using a slider
  The slider is not suitable for controlling GMP-relevant tags. Any operation of the slider will continuously change the tag value. If this is a GMP-relevant tag, a flood of entries will be generated in the AuditTrail.

12.15.2 WinCC Sm@rtServer

12.15.2.1 Basics

Sm@rt Options

Introduction

Using the Sm@rt Options from WinCC, you can communicate between HMI-systems or to an HMI-System by means of TCP/IP-connections (e.g.LAN).

Use of the Sm@rt Options

- Distributed operator stations with Sm@rtClients for controlling large machines or machines that are spread out over a large area.
- Operator stations with system-wide access to current process data via the communication driver "SIMATIC HTTP Protocol".
- Local servicing solution for the central archiving, analysis and additional processing of process data.
- Provision of current process data for higher-level systems (SCADA, production management systems, office applications).
- Remote control of an HMI-System by means of Internet, Intranet and LAN.
- Sending of E-Mails on the basis of messages and events
- Provisioning of Standard-HTML-Pages in HMI-system with service-and maintenance information as well as diagnostic functions.

User benefits:

- Flexible solution for access to HMI systems and process data from any location
- Reduction of load on the field bus:
  For example, the combination of WinCC Runtime and SIMATIC panels enables a factory control system to have access to process data. No load is placed by the factory level on the sensitive field level with respect to the necessary communication requirements. These requirements are handled by HMI Runtime along with the SIMATIC-panels.
- Expensive on-site service visits to be avoided by using the remote control. Unplanned non-operation periods are reduced and the system productivity is increased.

Note

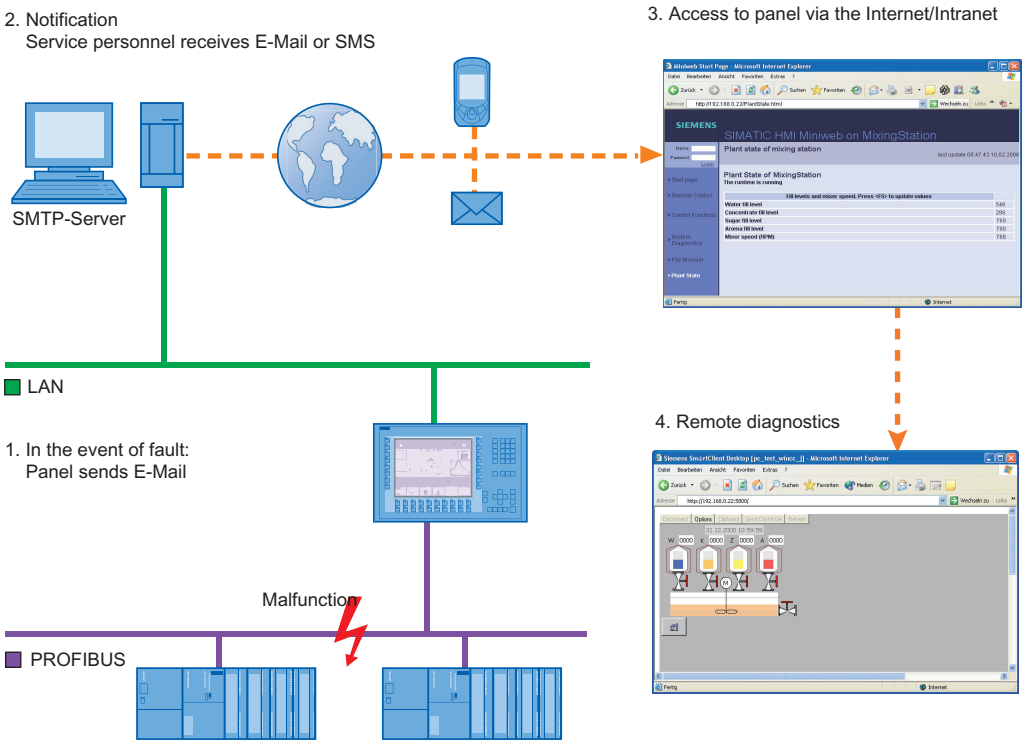
On devices with version V11 or V12, the password "100" is preset for the Sm@rtServer and for the integrated Web server. This password should be changed for security reasons.

On devices with version V13, no passwords are preset.

Application scenarios

E-Mailing and remote diagnostics

A factory has a service contract with an external service company. The HMI device and the service technician's PC are linked together over a TCP/IP-ready network. E-mail delivery of certain alarms to the service technician was configured in the project. The service technician accesses the HMI device via the Internet and executes remote diagnostics.



Application example:

Amongst other things,flow rates are measured in the process control of a cooling unit. Contamination in a feed line reduces the flow of coolant. When the flow rate drops below the configured threshold value, the operating device displays a warning. In addition,this warning is also dispatched as an e-mail to the assigned service technician.

The service technician then establishes a connection with the remote device and takes the appropriate actions.

Advantage: An alarm that reaches the service technician in a timely manner helps to minimize unplanned downtime.

Distributed operator stations

Distributed operator systems, the Sm@rtClients are used for controlling large machines or machines that are spread out over a large area.

The Client-HMI device connects to Sm@rtServer via the Sm@rtClient-Display.

The operator can operate and monitor the system from various locations. The operator sees the same image at each operator station, whereby only one station can be operated at one point of time.

The type of operation is also named as a coordinated operation. Then you must change the configuration only at Sm@rtServer.

Access to the tags via SIMATIC HMI HTTP Protocol

Operator stations with system-wide access

For use of the SIMATIC HMI HTTP Protocol, you can provide tags of HMI-device (HTTP-Server) to another device (HTTP-Client).

Thus the local as well as centrally used HMI-systems have access to the tags of other stations. Cell-concepts or line-concepts can be simply implemented. De-centrally obtained information is available centrally.

This concept also permits the setting of cost-effective and small central servicing. There are additional options for archiving, analysis and additional processing of registered process data, if a PC is used for that.

Remote monitoring and remote control- servicing solution

If you connect the use of the SIMATIC HMI HTTP Protocol and the Sm@rtServer with each other, you can implement a complex servicing solution.

For this,display the interested tags of the HMI-devices on the Service-PC. If necessary, use the PC for the remote monitoring and remote control of a specific HMI-device.

The locally used HMI-devices are connected with each other and the total process is controlled comprehensively.

The concept of the remote servicing is possible by using the Sm@rtClient-Display in the servicing HMI-Application. The operator has access to the respectively desired local HMI-device through flexible configuration of the Sm@rtClient-Display.

Connection to the Office-world

The possibility of data exchange exists between HMI-device and office-applications, e.g. MS Excel, with help of VBA-Macro.

For this, the HMI-device must support the Web-Service(SOAP). A script or macro is called in the external application, which has only read or write access to the concerned tags according to provided syntax.

HMI devices suitable for use

HMI devices suitable for use

The following table shows the HMI-devices that are suitable for the use of Sm@rt Options.

The number of the connections based on "SIMATIC HMI HTTP Protocol" and the number of maximum connected Sm@rtClients depends on the HMI-device. For additional information see the "Performance features" documentation and in the technical manual of your HMI-device.

Technical data subject to change.

HMI device	Sm@rt Options
270 series	Yes
370 series	Yes
OP 177B PN/DP TP 177B PN/DP	Yes
Mobile Panel 177 PN Mobile Panel 277	Yes
In view mode only: • Mobile Panel 277 IWLAN • Mobile Panel 277 IWLAN V2 • Mobile Panel 277F IWLAN (RFID Tag)	Yes
Comfort Panels	Yes
WinCC Runtime Advanced	Yes

Combining options on panels

The following table shows which options and functions on the panels can be combined with each other.

Technical data subject to change.

	SIMATIC HMI HTTP Protocol	Sm@rt Options	HTML browser	WinAC/ MP
SIMATIC HMI HTTP Protocol	--	Yes	Yes	Yes
Sm@rtServer	Yes	--	No	No
HTML browser	Yes	No	--	No
WinAC/ MP	Yes	No	No	--

Settings for Sm@rt Options

Configuration in WinCC

Introduction

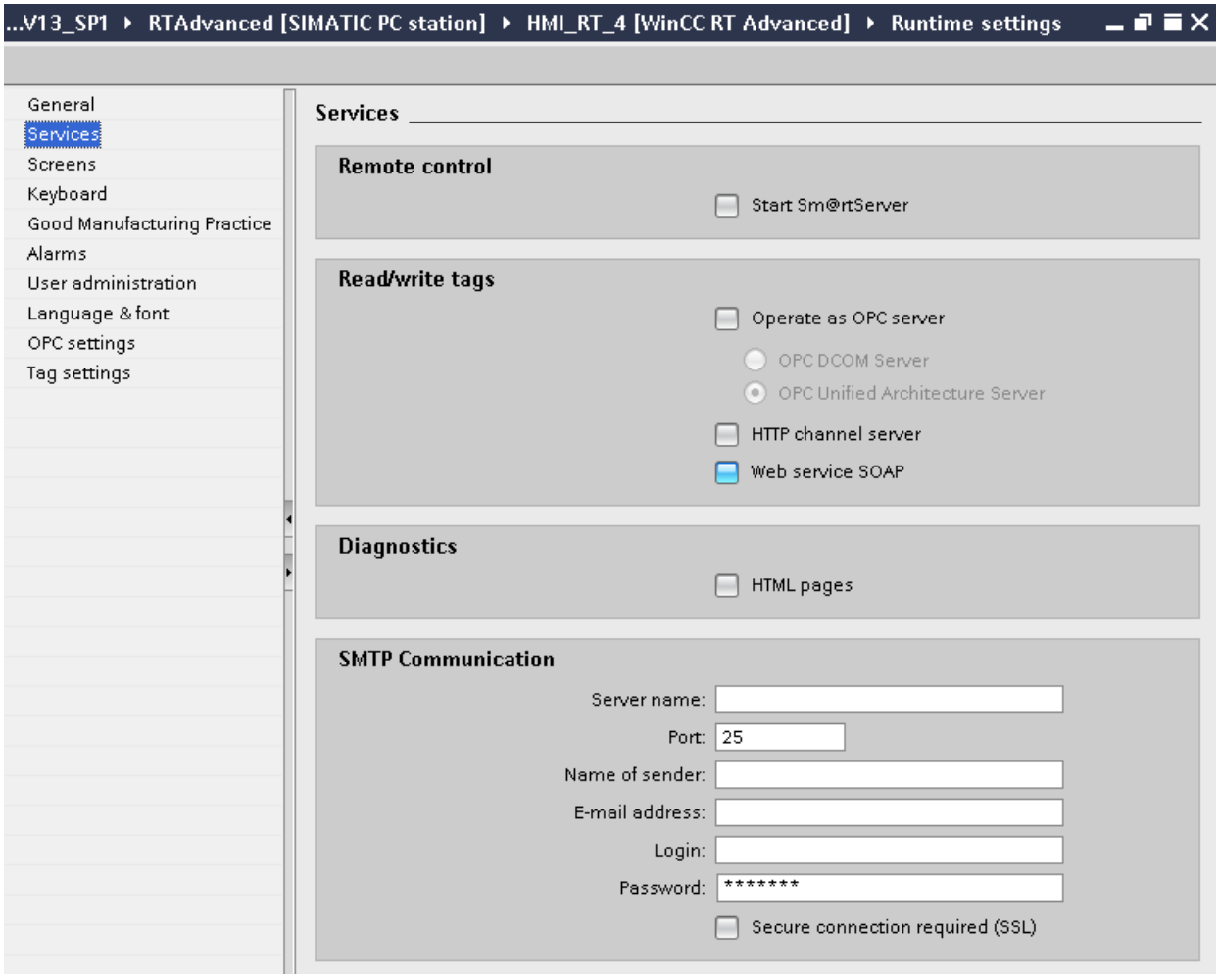
In the "Runtime-Settings" Editor, configure the requirements for using the Sm@rt Options.

As an alternative, configure the settings in the Control Panel of the HMI-device.

Note that the settings on the HMI device have a higher priority than the settings in the WinCC project.

Open

Double-click on the "Runtime-settings" entry in the project tree. In the "Runtime-settings" editor, click on the "Services".



"Remote control" Group.

Enter the settings for the selected HMI device in the work area:

- "Remote control" Group.
  - Start Sm@rtServer
    Configures the HMI device as Sm@rtServer.

"Read/write tags" group for Panels

- Operate as OPC server
  Configures the HMI device as an OPC server.
- HTTP channel server
  Configures the HMI device as HTTP server.
- Web service SOAP
  Activates tag access via SOAP.

"Read/write tags" group for Comfort Panels

- Operate as OPC UA server
  Configures the HMI device as an OPC UA server.
- HTTP channel server
  Configures the HMI device as HTTP server.
- Web service SOAP
  Activates tag access via SOAP.

"Read/write tags" group for RT Advanced

- Operate as OPC server
  - OPC DCOM server
    Configures the HMI device as an OPC DCOM server.
  - OPC UA server
    Configures the HMI device as an OPC UA server.
- HTTP channel server
  Configures the HMI device as HTTP server.
- Web service SOAP
  Activates tag access via SOAP.

"Diagnostics" Group

- HTML pages
  Activates service pages of the HMI device.

"SMTP Communication" Group

Activates service pages of the HMI device. "SMTP Communication" Group

- Server name
  Enter the name of the SMTP server through which you want to send E-mails.
- Port
  Enter the port number. The SMTP port number depends on the outgoing mail server of your service provider. You can obtain the port number from your service provider.

- Sender name
  Enter the sender name. The recipient sees in the E-Mail from which device the E-mail originates, e.g. "HMI device Production line 2". If the function is not supported by the SMTP-Server, delete the entry. You can obtain more detailed information for this from your service provider.
- E-mail address
  If you use an SMTP server that requires a valid e-mail address for authentication, enter it here, for example, "John.Doe@gmx.net.
- Login
  If you use an SMTP server that requires a user name for authentication, enter it here. You can obtain the user name from your service provider.
- Password
  If you are using an SMTP server that requires a password for authentication, enter this password. You can obtain the user name from your service provider.
- The server requires a secure connection (SSL).
  The data are sent via an SSL connection. Your service provider can tell you if your mail server supports an SSL connection.

Configurations on the HMI device

Settings on the HMI device

Introduction

The Sm@rt Options settings in the HMI device are configured in the dialog box "WinCC Runtime Advanced Internet".

Additional tabs may be included in the dialog "WinCC Runtime Advanced Internet". This depends on which options are activated for the network operation in the project.

Note that the settings on the control unit have a higher priority than the settings in the WinCC-project.

Tabs

You have opened the dialog "WinCC Runtime Advanced Internet" with the symbol "WinCC Runtime Advanced Internet" .

The "WinCC Runtime Advanced Internet" dialog of the control panel can contain the following tabs:

- WinCC Runtime Advanced Internet, "Email" tab (Page 7091)
- WinCC Runtime Advanced Internet, "Proxy" tab (Page 7092)
- WinCC Runtime Advanced Internet, "Web Server" tab (Page 7093)
- WinCC Runtime Advanced Internet, "Remote" tab (Page 7094)

Input area plan

At Sm@rtServer and Sm@rtClient, the same input area plan must be set.

The Sm@rtServer uses the Standard-Input area plan of the operating system: (Start>Settings>Control Panel> Country settings>Tab "Entry"). These changes become effective after system restart.

At Sm@rtClient, the same input area plan must be set like at Sm@rtServer. No restart is necessary after the switchover of input area plan at Sm@rtClient.

WinCC Runtime Advanced Internet, "Email" tab

Purpose of the dialog

Specifies the settings for the E-Mailing.

Settings

- SMTP server
  - Use the default of project file
    The SMTP-Server name from the WinCC-Project is used.
  - Enter the name of the SMTP server through which you want to send E-mails.
  - Port
    Enter the port number. The SMTP port number depends on the outgoing mail server of your service provider. You can obtain the port number from your service provider.
- Name
  - Name of sender
    Enter the sender name. The receipient sees in the E-Mail, from which device the E-mail originates, e.g. "HMI device Production line 2".
    If the function is not supported by the SMTP-Server, delete the entry. You can obtain more detailed information for this from your service provider.
  - eMail Adress of sender
    If you use an SMTP server that requires a valid e-mail address for authentication, enter it here, for example, "John.Doe@gmx.net.

Dialog "Advanced Email Settings"

- Authentication
  - Use the default of project file
    The user name and the password from the WinCC-Project are used.
  - Disable authentication
    Authentication is not required.
  - Use panel settings for authentication
    The settings of the HMI-device are used. Enter the user name under "Login" and the password under "Password". You can obtain the user name and the password from your service provider.
- Encryption
  - Use the default of project file
    The setting from the WinCC-Project is used.
  - Enable SSL
    The user data and e-mail are encrypted for transmission.
  - Disable SSL
    The user data and e-mail are sent un-encrypted.

See also

Settings on the HMI device (Page 7088)

WinCC Runtime Advanced Internet, "Proxy" tab

Purpose of the dialog box

Settings for utilizing the Proxy-Server.

Note

Enter the Proxy-Server in "Internet Options" at PC.

Settings

- Use proxy server
  Activate the "Use Proxy Server" if access is given in your network via Proxy-server.
- Proxy
  Enter the name or the address of the proxy-server.
- Port
  Enter the port of the proxy-server.

WinCC Runtime Advanced Internet, "Web Server" tab

Purpose of the dialog

The following is configured:

- The utilization of the integrated web server and of the HTTP-Server
- User and user web authorizations

Settings

Tag access

Governs the access to tags via the "SIMATIC HMI HTTP protocol":

- "Read/write": Read/write access
- "Read only": read access only

Tag authenticate

Governs the authentification in case of access to variables via the SIMATIC HMI HTTP Protocol":

- "No authentication": No authentification is required for access.
- "Authentication required": An authentification is required for access.
  Specify the user name and password when configuring the communication driver "SIMATIC HMI HTTP Protocol".

Enable Remote-Transfer for Projects

This setting enables remote transfer of project files.

Start automatically after booting

(on panel only)

The web server is automatically started after the HMI device boots. As a result, the web server is utilized independent of the runtime.

Note

Start web server automatically on PC

Add a link with the program "Miniweb.exe" in the Autostart-Organizer in order to start the web server automatically after the PC starts. The program is located in the installation index of runtime.

Close with Runtime

The web server is closed along with Runtime.

User Administration

After entering the password, the "UserDatabase-Edit". dialog opens. The "UserDatabase-Edit" dialog is the user administration of the web server.

Start Webserver

Starts the web server.

Close Webserver

Ends the web server.

WinCC Runtime Advanced Internet, "Remote" tab

Purpose of the dialog box

Settings for the Sm@rtServer

Settings

Start automatically after booting

The Sm@rtServer is automatically started after the HMI device boots. Otherwise the Sm@rtServer starts together with the Runtime.

Close with runtime

The Sm@rtServer is closed together with Runtime.

Change settings

Opens the "Sm@rtServer: Current User Properties" dialog for specifying the passwords, authorizations, the screen update mechanism and the behavior when connections are disconnected.

Note

This dialog is titled "Sm@rtServer: Default Local System Properties" on the panel.

The dialog contains the following tabs:

- "Server" tab (Page 7095)
- "Polling" tab (Page 7096)
- "Display" tab (Page 7098)
- "Query" tab (Page 7098)
- "Administration" tab (Page 7099)

Start Remoting

Starts the Sm@rt Server explicitly.

Stop Remoting

Ends the Sm@rt Server explicitly.

"Sm@rtServer Dialog: Current User Properties"

"Server" tab

Purpose of the dialog

Specifying passwords, web authorizations and the disconnection.

Incoming connections

Settings for handling an attempt to establish a connection.

- Accept socket connections
  This setting enables the connection to the HMI device. It is a basic requirement for using the Sm@rtServers from the outside.
  If this check box is deactivated, no remote monitoring and no remote control is possible.
- Encrypt communication
  Allows an encrypted connection between Sm@rtClient and Sm@rtServer.
- Password 1
  First password for remote access. "View only" is disabled as default.
- Password 2
  Second password for remote access. "View only" is selected by default.
  This password can be provided as a reserve password for third-party users (such as service technicians); it can be modified when necessary without significant effort within the organization.
- View only
  If this check box is enabled, read access (monitoring mode) is the only access available when the corresponding password is entered.
  Default: Enabled

Note

On devices with version V11 or V12, the password "100" is preset for the Sm@rtServer and for the integrated Web server. This password should be changed for security reasons.

On devices with version V13, no passwords are preset.

Enable network packets queuing (slower)

This setting enables splitting of data into multiple data packets, which are sent separately over the network. It is useful when multiple clients are connected.

Display or port numbers to use

Here, you select the TCP/IP port in the network where the Sm@rtServer waits for attempts to establish a connection.

- "Auto": The Sm@rtServer automatically searches for the appropriate port by itself.
- "Display": The server uses port 5900 plus display number. For HTTP, the server utilizes Port 5800 plus display number.
- "Ports": You enter the port numbers for the "main" and "HTTP" yourself.

No local input during client session

The keyboard and mouse on the server-HMI device are disabled as long as connections are active.

For example, this setting is useful when an HMI device is being administered from outside.

Remove desktop wallpaper (on PC only)

This setting removes the screen background on the PC, thus saving transmission effort.

Default: Enabled

When last client disconnects (on PC only)

This setting governs the behavior after disconnection of the last client connection:

- "Do nothing": no response.
- "Lock workstation": Server PC is locked.
- "Logoff workstation": Server PC is logged off.

The latter two settings are only useful if the Sm@rtServer is running as a service.

"Polling" tab

Purpose of the dialog

Specifying the screen update and the use of virtual graphics driver.

Polling modes(on PC only)

The settings govern the screen update.

Most changes are recognized automatically by the server. In case of problems, you can enter additional settings here.

The update method cannot be set on the panel. The settings is always "Poll Full Screen".

- Poll foreground window (On PC only)
  Updates the current window.
  It increases the load on the server.
- Poll window under cursor (On PC only)
  Updates the window that is located under the mouse cursor. The window is updated when a change occurs in the operator control element under the mouse cursor.
  Default: Enabled
- Poll full screen (On PC only)
  This setting specifies an update each time the screen changes. This setting provides you the lowest display error but places a maximum load on the server.
- Polling cycle
  Determine the suitable setting for your configuration. Make sure that the selected update cycle is not too short, since it affects the computer load.

Window polling

- Poll console windows Only (On PC only)
  This setting specifies an additional update when changes occur in a console window (MS input requirement).
- Poll on event received only (Only on the PC)
  This setting specifies an additional update each time an entry is made.
  Default: Enabled
- Mirror driver status
  (Only on the PC with a mirror driver installed)
  Provides information about the status of the virtual graphics driver.

Mirror driver options (only on a PC with installed mirror driver)

Enable direct access to display driver's mirror screen

The shared-memory area of the virtual graphics driver is used for the display. The setting improves the performance.

Troubleshooting (on PC only)

- Don't use VNCHooks.DLL while polling full screen
  VNCHooks.dll is used by default for the screen update. If VNCHooks.dll causes problems when using other program, select this setting.
- Don't use mirror display driver even if available
  (Only on the PC with a mirror driver installed)
  Use this setting only for troubleshooting.

"Display" tab

Purpose of the dialog

Setting of the screen display.

Sharing area(on PC only)

- Full desktop
  The entire desktop of the server is accessed.
- Primary display
  The main screen of the multi-monitor configuration is displayed.

Downscale to(on PC only)

Scales the screen to be transferred according your inputs. Servers with Windows CE ignore this setting.

"Query" tab

Purpose of the dialog

Settings for incoming connection attempts.

Note

This dialog is titled "Default Local System Advanced" on the panel.

Query settings

These settings govern acceptance of incoming attempts to establish a connection.

- Query console on incoming connections
  The Sm@rtServer registers the incoming attempts to establish a connection and displays a dialog on the screen in which the connection attempt is accepted or rejected.
- Query timeout
  Set the waiting time.
- Default action
  Select the response to an attempt to establish a connection once the waiting time expires:
  - "Refuse": Reject attempt (operator control mode - single mode)
  - "Accept": Accept attempt (operator control mode – shared mode)

Allow option to accept without authentication

The dialog for handling attempts to establish connections also contains the button "Accept without password" . This gives you the option to accept an attempt to establish connection without a password.

"Administration" tab

Purpose of the dialog

Specifications for session management.

Note

This dialog is titled "Default Local System Advanced" on the panel.

Administration

- Disable empty passwords
  Select this check box in order to allow an empty "Password 1".
  Default: Enabled
- Allow loopback connections
  This setting allows connections to your own HMI device. It is useful and necessary when security software is used for secure (encoded) connections.
- Allow only loopback
  This setting allows only connections to your own HMI device.

Logging

- Log info to SmartServer.log
  (On the panel: Log information to file)
  This setting writes information to the server logbook.
- Log detailed debugging information
  This setting writes expanded information to the server logbook (for locating errors).

With HMI devices operating with Windows CE, the log is only created when the MMC card is inserted. If the MMC card is inserted, the log is created directly on the card.

The log files are created in the following path on a PC: C:\Documents and Settings\All Users \Application Data\Siemens\HmiRTm

Forced write access

This setting governs forced access in an emergency contrary to normal session management.

- Password needed
  If this check box is not selected, every operator can force access in emergencies as follows:
  - Pressing the <Shift> key four times
  - Clicking four times
  - Touching the screen four times

  If this check box is enabled, to force access an attempt must be made to gain access and a password must also be entered.
  In this case, enter the applicable password in the input field underneath. If a password is not entered, it is not possible to force access in an emergency.
  Default: Enabled

Note

On the server, access can only be forced by pressing the <Shift> key four times, clicking four times, or touching the screen four times.

HTTP-Server

- Enable built-in HTTP server
  If this check box is activated, the Java-Applet is automatically downloaded on the PC when the connection is first established.
  The Java applet accesses the Java VM that is installed on the client and enables remote monitoring and remote control using Internet Explorer.
  Default: Enabled
- Enable applet params in URLs
  Forwards all parameters of the URL to the Sm@rtClient application.

Connection priority

These settings govern handling of attempts to establish a connection by non-shared clients.

- Disconnect existing connections
  When an attempt is made to establish a connection by a non-shared client, the attempt is accepted; the existing connections are disconnected (single mode).
- Automatic shared sesssions
  When an attempt is made to establish a connection by a non-shared client, the attempt is accepted; the prior existing connections are retained. Access is controlled using session management in shared mode.
  Default: Enabled

- Active user timeout
  For shared mode, enter the time that must elapse without any actions on the active HMI device before access can be changed.
  Default setting: 10 seconds
- Refuse concurrent connections
  If a non-shared client is already connected to the server, attempts to establish a connection by other non-shared clients are rejected.

See also

Remote control by means of Internet Explorer (Page 7116)

User administration for web server

Introduction

Different web authorizations for the operation and monitoring are allocated to the users in the user administration.

Requirement

- The "WinCC Runtime Advanced Internet" dialog is open.
- The "Web Server" tab is displayed.

Entries and settings

Click "User Administration" in the "Web Server" tab and enter the password.

Note

On devices with version V11 or V12, the password "100" is preset for the Sm@rtServer and for the integrated Web server. This password should be changed for security reasons.

On devices with version V13, no passwords are preset.

The "UserDatabase-Edit" dialog is opened.

The "UserDatabase-Edit" dialog has three tabs:

- Tab "User Manager"
  User administration for creation or deletion of users.
  Create a new user using "New"; Delete a user using "Remove". The recreation and deletion become effective using "Apply".
- Tab "Description"
  You can store a description of or comments on the users selected on the "User Manager" tab.
- Tab "Authorizations"
  Specify the web authorizations for the users selected on the "User Manager" tab. You use "Add" to activate a web authorization and "Remove" to deactivate one.
  By default, the password is initially preset to "100" and all web authorizations are granted to users with "Administrator" rights.
  For read and write access to the file browser, the user must possess the web authorizations "FileBrowserAdministrator" and "FileBrowserUser".

Note

In principle, every user who has access to the control panel can manage users and web authorizations. If necessary, protect the control panel from unwanted access.

List of web authorizations

The following web authorizations exist:

Web authorization	Authorized for:
UserData	Import and export of recipes
UserAdministrator	Import and export of password lists
RuntimeAccess	Starting and stopping of runtime
Engineering	HTTP transfer from ES to the target device
FileBrowserUser	Read access to the file browser
FileBrowserAdministrator	Read/write access to the file browser
RTCommunication	Utilization of the SIMATIC HMI HTTP server
SoapUser	Read/write access via web service (SOAP)

Settings for remote control

Session management for remote control

Introduction

WinCC enables remote monitoring and remote control of HMI devices over a TCP/IP-ready network such as a LAN or the Internet. Remote monitoring and remote control is implemented in different ways:

- Remote control by means of Internet Explorer
- Remote control by means of the Sm@rtClient-application
- Remote control by means of the Sm@rtClient-display in

Only one device can ever have access to the HMI-device. Which device is permitted access is determined by the session management.

Session management options

Session management is used to control access. The client-server connection can be in one of two modes:

- Monitoring mode
- Control mode

Monitoring mode

If the client accesses the server in monitoring mode, the operator can see the current screen of the HMI device and track all changes. He can monitor the server but cannot operate.

In the monitoring mode, all the keys on the client retain their standard functions.

If remote control was started from the Sm@rtClient display, the operator uses the <Tab> key or the cursor keys to go to the next object in the current screen of the client project.

Control mode

If the client accesses the server in operator control mode, the operator can use the mouse and the keyboard to control the server from the client. If an access attempt is made from another client, the assignment of operating permission depends on the settings at the server and at the clients.

In operator control mode, the client keys act on the server screen. Thus, the operator uses the <Tab> key to go to the next object in the current screen of the project running on the server.

If remote control was started from the Sm@rtClient display, the operator can only go to another object or screen in the project on the client by using an additionally configured function or an additional menu command. The operator to this menu command as follows:

- On the Touch-device, in which he touches the screen longer than 1 sec.
- On the keyboard =device, in which he masks on the menu with <Shift+Control> and operates it with <Alt> and the keyboard.

In both operating modes, the Sm@rtServer is set so that the operator at the remotely controlled device, the server, can be prevented from performing any activities.

In an emergency, the operator can exact the user rights on a remotely controlled HMI device as well as on an inactive HMI device. If no password is specified, he must click the user interface four times consecutively, touch the screen four times consecutively, or press the <Shift> key four times consecutively. If a password is specified, he must click once or press a key on the client and then enter the specified password.

Settings for session management

You make settings for session management on the server and on the client in the Control Panel "WinCC Internet Settings".

Configuring Sm@rtServer for remote control

Introduction

The Sm@rtServer has an internal security concept based on passwords and special settings for session management.

Security concept for the Sm@rtServer

Remote monitoring and remote control of the Sm@rtServer from the Sm@rtClient is protected by two functions:

- Encrypt encryption of communication to the server
- Passwords

Encrypt communication to the Sm@rtServer

The "Encrypt communication" function provides a secure connection between Sm@rtClient and Sm@rtServer.

After you have activated the "Encrypt communication" function, the Sm@rtServer sends a Sm@rtServer-specific certificate to the Sm@rtClient.

The Sm@rtClient must support the "Encrypt communication" function for encrypted communication.

The certificate sent by the Sm@rtServer must be accepted to establish communication on the Sm@rtClient.

All connections between Sm@rtClient and Sm@rtServer are then only possible on the basis of the exchanged certificate.

Passwords for the Sm@rtServer

Remote monitoring and remote control of the Sm@rtServer at the Sm@rtClient is protected by two passwords.

The second password is used as a further password for additional access, for example, as a service password.

Note

Passwords for the Sm@rtServer

No default passwords are set.

Therefore, assign the passwords before you use the Sm@rtServer.

Settings on the Sm@rtServer

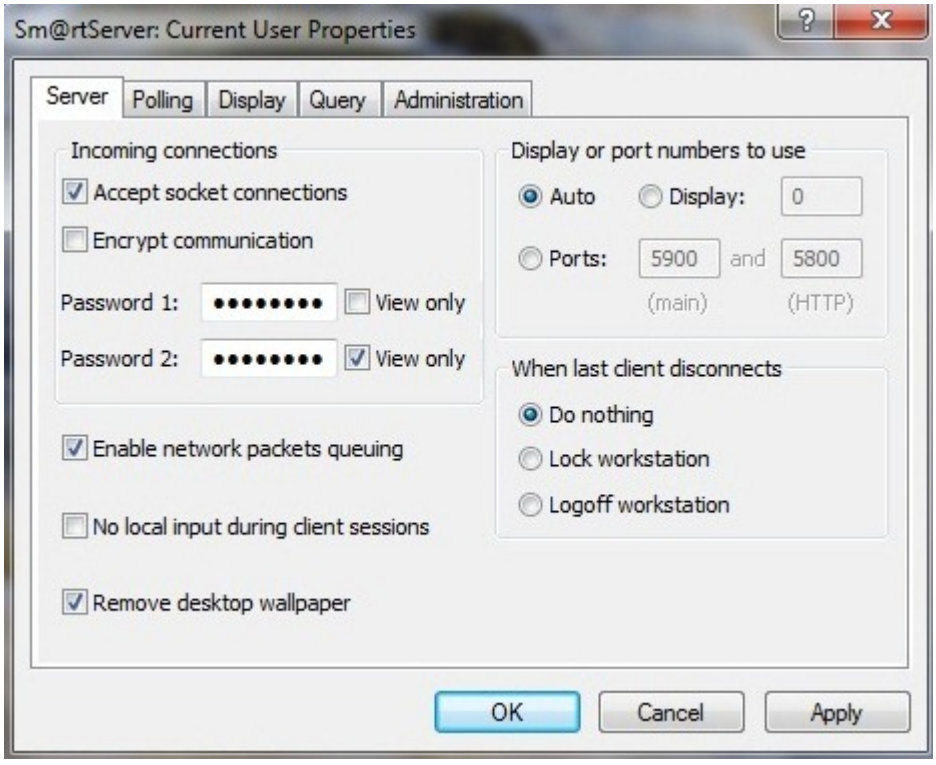
The settings on the server govern which remote operators can access the runtime of the server.

The passwords for access are set on the server. Open "WinCC Runtime Advanced Internet" in the control panel. On the "Remote" tab, click "Change Settings". On the following Server tab of the subsequent dialog, enter the passwords of the Sm@rtClient. For both passwords, you can use "View only" to set the monitoring mode and to exclude operator control mode.

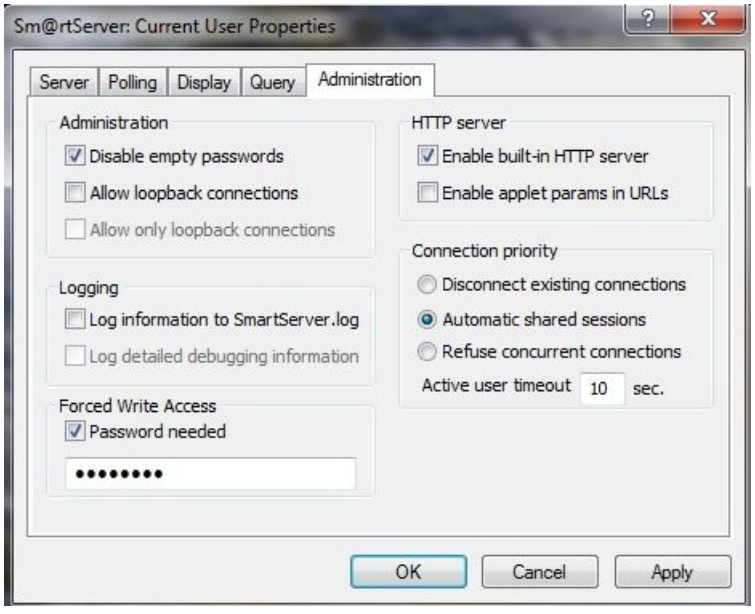
On the panel, this dialog is called "Sm@rtServer: Default Local System Properties" and contains fewer dialog elements than the dialog on the PC.

Control mode

To enable control mode, the "View only" check box must be cleared, at least for "Password 1" (Password 1).



The manner in which the individual remote control station can access the server is set on the "Administration" tab.



- "Disconnect existing connections"
  When an access attempt is made by an non-shared client, the previous connection is automatically disconnected and control is transferred to the new client.
  When an attempt is made to access by a shared client, the behavior is the same as described for "Automatic shared sessions".
- "Automatic shared sessions"
  When an access attempt is made, control is transferred to the new client.
  The condition for transferring control is that no action has been undertaken by the previously active client for a period of time (in seconds) as specified in the "Active user timeout" setting.
- "Refuse concurrent connections"
  When an attempt is made to access by a non-shared client, this access attempt is rejected so long as the operator station that currently has access is still connected to the server.
  When an attempt is made to access by a shared client, the behavior is the same as described for "Automatic shared sessions".

Disabling local operator control of the server

To do so, open the "WinCC Internet Settings" in the Control Panel. On the "Remote" tab, click "Change Settings". On the "Server" tab in the subsequent dialog, select "No local input during client session".

Password for forced access

A password can be specified in "Forced Write Access" for forced access in an emergency.

Sm@rtServer as a service

You can let the Sm@rtServer run as a service. The operator can then also access the service-HMI device from the client-HMI device, for example, the screen saver is active with a password.

Select the check box "Start automatically after booting" in "WinCC Runtime Advanced Internet" on the "Remote" tab.

Sm@rtServer as service in Windows 7

The Sm@rtServer always runs as a service under Windows 7. You cannot stop the Sm@rtServer in Runtime with the Notification Area in the taskbar. You have the following options for stopping the Sm@rtServer:

- Stop using the "ControlSmartServer" system function
  Configure the "ControlSmartServer" system function to a button, for example. Select the "Stop" mode at the system function. You can stop the Sm@rtServer in Runtime by clicking the button.
- Stopping the Sm@rtServer in the Control Panel:
  Open "WinCC Runtime Advanced Internet" in the Control Panel and select the "Remote" tab. Click on the "Stop" button. Sm@rtServer is stopped.
- Stopping Sm@rtServer by stopping Runtime:
  Open "WinCC Runtime Advanced Internet" in the Control Panel and select the "Remote" tab. Activate the "Close with Runtime" option. The Sm@rtServer is stopped at the stop of Runtime.

Configure Sm@rtClient for remote control

Settings on the client PC

- Monitoring mode
  At the client-PC you limit the connection to observation mode, if required. This allows you to prevent unintended control operations.
  - If connection is via the Sm@rtClient application
    In the following "Sm@rtClient Connection" of Sm@rtClient-application, click "Options..." button.
    In the "Sm@rtclient Options" dialog, select the setting"View only (inputs ignored)".
  - If connection is via Internet Explorer
    Click on the button "Options..." and select "View only" in the subsequent dialog.
- Layout
  You can also specify whether or not the HMI-device is to be displayed with the same layout in the Sm@rtClient application. This is useful if you access from a PC on a touch device. In order to suppress the layout, select the "Sm@rtclient Options" setting in the "Suppress Device Layout" dialog.
  In addition, you can use "Scale by" to zoom in or out of the layout. To use "Scale by", the Suppress Device Layout setting must be selected or the HMI-device must not supply any layout. Otherwise, the desktop is always displayed with a zoom setting of 100%.

  Note

  If you scale the display, the performance may be impaired in some situations:
  - Strong scaling, for example from 1600 x 1200 px to 640 x 480 px.
  - Scaling with non-matching scaling factors, for example from a 4:3 resolution to 16:10 resolution.

Configuring of the Sm@rtClient-display

You can configure the Sm@rtClient display in different ways, and thereby set certain inputs: The server name, the password for accessing Sm@rtServer or the restriction to the monitoring mode.

Encrypted communication

You must enable the "Encrypt communication" function for encrypted communication with the Sm@rtServer.

- Support for the "Encrypt communication" function on the Sm@rtClient
  If the Sm@rtClient does not support the "Encrypt communication" function, no communication is possible with a Sm@rtServer which has enabled the "Encrypt communication" function.
  Enable the "Encrypt communication" function on the Sm@rtServer.
- Enabling the "Encrypt communication" function on the Sm@rtServer by the Sm@rtClient
  You can also enable the "Encrypt communication" function on the Sm@rtServer from the Sm@rtClient:
  - Establish a connection to the server.
  - Enable the "Encrypt communication" function in the "Standard VNC Authentication" dialog.
  - Certificates are exchanged between the Sm@rtServer and Sm@rtClient.
  - The certificate sent by the Sm@rtServer must be accepted by the Sm@rtClient.

Sm@rtClient-Application

Dialog "New Sm@rtServer: Connection"

Introduction

This dialog opens when you click the "Sm@rtClient" button in the taskbar.

Purpose of the dialog box

This dialog is used for selecting the server and the connection method.

Sm@rtServer:

Enter the address of the server to which the connection is to be established. The various options for entering the address can be found in "Remote control by means of the Sm@rtClient application (Page 7117)".

Connection profil

Select the type of connection to the server according to the network you are using.

Listening mode

If you enable this function, the Sm@rtClient application is minimized and appears as a button in the Windows taskbar. The Sm@rtClient waits for the Sm@rtServer to establish a connection. To establish the connection, click the "Sm@rtServer" button in the Windows taskbar of the server. Select "Add new client" in the context menu.

Options

The "Sm@rtClient Options" dialog containing the technical settings for the Sm@rtClient-application is displayed.

See also

"Options" dialog, "Globals" tab (Page 7112)

"Options" dialog, "Connections" tab (Page 7110)

"Options" dialog, "Connections" tab

Purpose of the dialog box

Technical settings for the Sm@rtClient-application are specified in this dialog box.

Only change the settings here in special cases.

Note

You can also specify these settings in the Java Applet. Note that some of the dialog elements are named differently there.

Format and encodings

Settings for compressing (encoding) the screen data of the server.

- Use encoding
  Preassigned based on the selection under "Connection profile".
  Select the desired compression or "Raw" (no compression).
- User 8-bit color
  (Only in the Java-Applet): Reduces the color depth at the client to 8 bits (256 colors). The data are then transferred faster. However, incorrect colors may result.
- Custom Compression level
  Allows individual customizing of the compression level in the "Level" input field:
  1 = least compression (faster); 9 = maximum compression (slower).
- Allow JPEG compression
  Allows the use of JPEG compression (involves losses).
  Enter the "Screen quality" in the input field underneath:
  1 = least compression (faster); 9 = maximum compression (slower).
- Allow CopyRect encoding
  (In the Java-Applet: Use CopyRect. Encoding)
  Allows compression while using "similar rectangles".

Restrictions

- Viewonly (inputs ingnored)
  Sets the view mode for this client irrespective of the settings on the server.
- Disable clipboard transfer
  Disables the clipboard that is used to transfer data from one PC to another. Applies only to the copying and pasting of texts.
  This functionality is not available at a Windows CE server.

Display

Settings for the screen display

- Scale by
  Zooms in or zooms out the desktop to be displayed. To use "Scale by", the Suppress Device Layout setting must be selected or the HMI-device must not supply any layout. Otherwise, the desktop is displayed with a zoom setting of 100%.
- Fullscreen Mode
  Displays the desktop to be shown in full-screen mode. If the server screen is larger than the screen of the client, it is scrolled automatically by the mouse movement.
- Suppress Device Layout
  In the Sm@rtClient application window, the entire layout of remote HMI-device is not shown.
- Use CTRL + Cursor Key for Scrolling
  The key combinations <CTRL> + cursor key are used to scroll within the local screen. They are no longer transferred to the server.

Mouse

(In the Java-Applet: Mouse buttons 2 and 3)

Settings for the evaluation of mouse actions

- Emulate 3 Buttons (with 2-button click)
  Emulation of a three-button mouse by a two-button mouse.
- Swap mouse buttons 2 and 3
  (In the Java-Applet: reversed/normal)
  Mouse buttons 2 and 3 are swapped.

Mouse cursor

(In the Java-Applet: Cursor shape updates)

Settings for the display of the cursor

Select the type of transfer of the mouse actions:

- Track remote cursor locally
  The information on the location of the cursor is transferred separately from the screen information. This speeds up the transfer of the cursor. (JavaApplet: Enabled)
- Let remote server deal with mouse cursor
  Tracks the server cursor to the client cursor. This allows more accurate cursor positioning. (YES: Ignore)
- Don't show remote cursor
  The cursor at the server is not included in the transfer. (YES: Disable)

Request shared session

(In the Java-Applet: Share desktop)

Declares this client to be a non-exclusive client.

See also

Dialog "New Sm@rtServer: Connection" (Page 7107)

"Options" dialog, "Globals" tab

Purpose of the dialog box

Technical settings for the Sm@rtClient-application are carried out in this dialog box.

Only change the settings here in special cases.

Note

You can also carry out these settings in the Java Applet. Note that some of the dialog elements are named differently there.

Interface options

- Show toolbars by default
  Displays the toolbar.
- Warn at switching to the full-screen mode
  Outputs a message before the full-screen mode is activated.
- Enable Onscreen keyboard
  Enables the display of the on-screen keyboard
- Number of connection to remember
  The client creates a list of the recently used connections. This setting specifies the number of connections listed.
- Clear the list of saved connections
  The list is cleared.

Local cursor shape

Specifies the appearance of the local cursor. This allows you to better differentiate between the local cursor and remote cursor.

Listening mode

- Accept reverse VNC connections on TCP port
  Specifies the TCP port number. The Sm@rtClient waits for the Sm@rtServer to establish the connection over this TCP port number.

Logging

- Write log to a file
  Writes information in the logbook of the Sm@rtClient-application.
- Verbosity level
  Writes expanded information to the server logbook of the Sm@rtClient-application (for locating faults). The amount of detail in the information is dependent on the verbosity level setting.

Remote control of key devices

Mapping of the function keys

HMI devices are equipped with a variety of function keys.

Example:

From the PC keyboard, you want to remotely control key F20 on the "MP 377 key" of HMI-device.

You can control the F20 key with the PC-keyboard shortcut <SHIFT + F8>.

The table provides you with an overview for controlling the keys.

xP177/277	xP377	PC keyboard shortcut
F13	F13	SHIFT + F1
F14	F14	SHIFT + F2
F15	F15	SHIFT + F3
F16	F16	SHIFT + F4
F17	F17	SHIFT + F5
F18	F18	SHIFT + F6
F19	F19	SHIFT + F7
F20	F20	SHIFT + F8
K1	S1	SHIFT + F9
K2	S2	SHIFT + F10
K3	S3	SHIFT + F11
K4	S4	SHIFT + F12
K5	S5	CTRL + F1
K6	S6	CTRL + F2
K7	S7	CTRL + F3
K8	S8	CTRL + F4
K9	S9	CTRL + F5
K10	S10	CTRL + F6
K11	S11	CTRL + F7
K12	S12	CTRL + F8
K13	S13	CTRL + F9
K14	S14	CTRL + F10
K15	S15	CTRL + F11

xP177/277	xP377	PC keyboard shortcut
K16	S16	CTRL + F12
HELP	HELP	ALT + H
ACK	ACK	ALT + F1

Use and restrictions of Sm@rt Options

Use restrictions

While using the Sm@rt Options, observe the following instructions:

- HMI devices suitable for use
  For additional information, refer to "Usable HMI devices".
- Sm@rtServer and Sm@rtClient
  - If a PC is used as a Sm@rtServer, select the highest-performance platform available.
  - Use only simple projects.
  - Avoid photographs and color gradients in screens.
  - Avoid heavy background loads during operation, for example, those from user-defined functions or logs.
  - The number of the maximum connected Sm@rt Clients depends on the Server-HMI device. For additional information, see the "Performance features" documentation.
  - To improve the performance of the Sm@rtServer, you can disable hardware acceleration of the graphics card .
  - You inevitably loose performance at the HMI when you access the HMI-device using the Sm@rtClient functionality.
- SIMATIC HMI HTTP Protocol
  - Tag exchange via the SIMATIC HMI HTTP Protocol is not suitable for exchanging bulk data.
  - The maximum number of connections depends on the HMI device: For additional information, see the "Performance features" documentation
- Access protection
  To protect access to an HMI-device using different passwords, you must use the first password for the protected, and the second password for the unprotected access e.g. remote control with a password, remote monitoring without password.
- Port
  The web server connects to the network at the port 80. To run it without problems, make sure port 80 is not in use by any other application, such as IIS World Wide Web Publishing Service.

- Integrated HTML-pages
  - The size of the HTML pages must not exceed 100 Kb in case of Windows CE. In case of exceeding of the given value, these pages are spooled on external storage media.
  - If you display tags on HTML-pages using the entry type "Cyclic on use", it can lead to data-inconsistency.
- E-mailing
  The E-mailing function is not suitable for the mass dispatch of E-Mails. It is meant for sending important messages.
- Timeout
  If the connection between server and client is interrupted, the server will register this disconnection only with a certain delay. The delay is based on the Windows standard configuration of TCP/IP Timeout.

Use-requirements in the company network

In order to implement the mentioned scenario, the accesses to the company network must be enabled. If the company network is protected by a Firewall, the system administrator must release the appropriate ports for that.

- Access to the integrated HTML-pages
  The web server connects to the network at the port 80.
- Access to Sm@rtServer for downloading the Java-Applet using the Internet Explorer
  The Sm@rtServer is connected to the network at the Port 5800 for downloading the Java-applet.
- Access to the Sm@rtServer using the Internet Explorer for remote monitoring and remote control
  The Sm@rtServer is connected to the network at the port 5900.

Note

If you change the ports of the Sm@rtServer you must customize the links in the used Html-pages accordingly. Additional information to modify the Html-pages is provided in "Example: "Configure integrated webserver".

12.15.2.2 Remote control via Sm@rtServer

Types of the remote control

Remote control and remote monitoring by means of Sm@rtServer

Introduction

The Sm@rtOptions of WinCC enable the access from HMI device or PC to a remote HMI device via Ethernet.

Requirement

- The License Key "Sm@rtServer" is available at the Server-HMI device.

  Note

  The 14-day license is not supported by Windows CE devices.

- Both devices are linked via a TCP/IP-ready network, that is via a LAN or the Internet.
- "Sm@rtServer" is activated in the WinCC-Project of the Server-HMI device for the "Services in Runtime".
- Additional requirements must be satisfied according to the type of implementation.

Implementing remote access

The Sm@rtServer supports remote monitoring or remote control on the remote device (server).

Remote monitoring or remote control can be implemented on the local device (client) in various ways:

- By means of Internet Explorer
- By means of the Sm@rtClient-application

Access via HTML pages

The Sm@rt Options enable access for remote control with Microsoft Internet Explorer and by means of integrated HTML pages of the server.

Caution

Ethernet communication

In Ethernet-based communication, such as PROFINET IO, HTTP, Sm@rt Options and OPC, it is the end user who is responsible for the security of his data network. The proper functioning of the device cannot be guaranteed in all circumstances; targeted attacks, for example, can lead to an overloading of the device.

Remote control by means of Internet Explorer

Introduction

On the client HMI device, the connection to the remote HMI device is established by means of Internet Explorer.

The window of the Internet Explorer displays only the screen of the remote HMI device, of the server HMI device. If task switching is not disabled at the server HMI device, you can access the complete desktop.

Requirement

- The Client-HMI device is a PC.
- Internet Explorer from V6.0 SP1 is installed.
- The Client-and Server-certificates are installed to ensure the data security when transferred via internet.
- The Java-Applet is installed. The Java applet accesses the Java runtime environment that is installed on the client.

Note

You achieve the best results in Internet Explorer by installing the current Java Runtime Environment (JRETM) of Sun Microsystems. Go to www.java.com to download this program.

Process flow

Enter the address of the remote device in Internet Explorer. The address consists of the server name and the HTTP port number that is set on the server. The default setting is: 5800.

Examples of addressing: "http://MyPanel:5800" or "http://192.168.168.1:5800".

Restrictions

The "Force write access with password" function cannot be implemented using the Java applet.

See also

"Administration" tab (Page 7097)

Remote control by means of the Sm@rtClient application

Introduction

The Sm@rtClient application provides the connection to the remote HMI device on the remote HMI device.

Requirement

- The Client-and Server-certificates are installed to ensure the data security when transferred via internet.
- The Client-HMI device is a PC.

Process flow

The remote control via the Sm@rtClient-application works as follows:

- Start Sm@rtClient application
- Establish connection
- Password input
- Perform operator control or monitoring on the HMI device

Start Sm@rtClient application

You can access the Sm@rtClient application, the program "SmartClient.exe", in various ways:

- By installing WinCC Runtime on the client device you have automatically installed the Sm@rtClient application.
- You have several options if WinCC Runtime is not installed on the client device:
  - Copy the Sm@rtClient application from the WinCC product-DVD from the folder "Support \SmartClient".
  - Copy the Sm@rtClient application from the "...\Siemens\Automation\WinCC RT Advanced" folder of another PC using a floppy disk or the Intranet.

Establish connection

In order to establish the connection to the remote HMI device, call the Sm@rtClient application and enter the IP address of the server.

- IP address or server name:port number
- IP address or server name:display number

Example: "192.168.0.1::5800"

You can also start the Sm@rtClient application with the command line input: "smartclient.exe 192.168.0.1". The logon dialog box opens.

You can include the password in the command line entry to start the Sm@rtClient application: "smartclient.exe 192.168.0.1 /password <password>".

Note

If the Sm@rtServer at the server HMI device does not run as a service, the connection established with the Sm@rtClient application is interrupted automatically as soon as the keyboard shortcut CTRL+ALT+DEL is pressed at the server HMI device or the screen saver is activated. In order for the Sm@rtServer to run as a service, the "Start automatically after booting" check box in the "Remote" tab in the "WinCC Runtime Advacned Internet Settings" dialog must be activated.

Password input

- Password input at the Sm@rtServer
  Instead of the on-screen keyboard, the following message is displayed on the Sm@rtClient if you enter the password directly at the Sm@rtServer: "Remote access by Sm@rt Options is in Progress. Please wait until the input of values has been ended." This measure prevents keyboard input for entering the password from being displayed on the Sm@rtClient.
- Password input at the Sm@rtClient
  The on-screen keyboard is hidden on the Sm@rtServer due to the actions carried out on the Sm@rtClient. Use the local on-screen keyboard for entries at the Sm@rtClient. The local on-screen keyboard will be displayed automatically on the Sm@rtClient or in the Sm@rtClient view. Close the on-screen keyboard manually. Select "Input > Hide Input Panel" to hide the local on-screen keyboard.

Note

The entries with full-screen keyboard are not protected on HMI devices with a screen size of ≤ 6".

Entries in Control Panel Applets which do not use the full-screen keyboard are protected.

Note

Hidden password input is not supported by the on-screen keyboards of third-party products.

Note

You cannot enter special characters with the keyboard shortcut Alt Gr.

Perform operator control or monitoring on the HMI device

In the Sm@rtClient application window, the entire layout of the remote HMI device is shown. Depending on the configuration, you can specify monitoring only or operator control of all keys, including the function keys, with the mouse. In addition, the entire desktop can be accessed in the case of a PC.

For operator control via the keyboard, the following is available:

Keyboard shortcut	Function
<ALT+CTRL+SHIFT+O>	Opens the "Sm@rtClient Options" dialog
<ALT+CTRL+SHIFT+F>	Switches over to full screen mode
<ALT+CTRL+SHIFT+R>	Updates the display
<ALT+CTRL+SHIFT+N>	Opens the "New Sm@rtServer Connection" dialog
<ALT+CTRL+SHIFT+S>	Save as
<ALT+CTRL+SHIFT+T>	Displays and hides the toolbar

Remote control via the Sm@rtClient display during runtime

Introduction

The Sm@rt options of WinCC enables access from the HMI device or PC to a remote HMI device via Ethernet.

Requirement

- The License Key "Sm@rtServer" is available at the Server-HMI device.
- Both devices are linked via a TCP/IP-ready network, that is, via a LAN or the Internet.
- HMI-device is configured as Sm@rtServer. For detailed instructions on this, refer to "Configure Sm@rtServer (Page 7123)".
- The Sm@rtClient-Display in an image is added in the client-HMI device project. For detailed information on this, refer to "Project Sm@rtClient (Page 7125)".

Implementing remote access

The Sm@rtServer supports remote monitoring or remote control on the remote device (server).

On the client HMI device, the connection to the Sm@rtServer is made during runtime by means of the Sm@rtClient display.

On the HMI device only the screen of the server, and not the soft keys, is displayed.

The form of the cursor is not a part of the screen and is therefore not transmitted. Only the coordinates of the cursor are transmitted.

Note

If a soft-key is activated on the client HMI device, then observe the following:

This signal is transferred to the Server-HMI device and becomes effective there, only if no function was configured at the soft-key.

Otherwise, the function projected on the client-HMI device is executed.

Use of direct keys for remote access

You can only operate direct keys locally on the server. Although the key for the direct key can be operated on the Sm@rtClient, no bit is set in the I/O range of the PLC.

Caution

Ethernet communication

In Ethernet-based communication, such as PROFINET IO, HTTP, Sm@rt Options and OPC, it is the end user who is responsible for the security of his data network. The proper functioning of the device cannot be guaranteed in all circumstances; targeted attacks, for example, can lead to an overloading of the device.

Password input

Password input at the Sm@rtServer

Instead of the on-screen keyboard, the following message is displayed on the Sm@rtClient if you enter the password directly at the Sm@rtServer: "Remote access by Sm@rt Options is in Progress. Please wait until the input of values has been ended." This measure prevents keyboard input for entering the password from being displayed on the Sm@rtClient.

Password input at the Sm@rtClient

The on-screen keyboard is hidden on the Sm@rtServer due to the actions carried out on the Sm@rtClient. Use the local on-screen keyboard for entries at the Sm@rtClient. The local on-screen keyboard will be displayed automatically on the Sm@rtClient or in the Sm@rtClient view.

Close the local on-screen keyboard manually. Select "Input > Hide Input Panel" to hide the local on-screen keyboard.

Note

The entries with full-screen keyboard are not protected on HMI devices with a screen size of ≤ 6".

Entries in Control Panel Applets which do not use the full-screen keyboard are protected.

Note

Hidden password input is not supported by the on-screen keyboards of third-party products.

Note

You cannot enter special characters with the keyboard shortcut Alt Gr.

Distributed operator stations

Configuration

Configuration

Multiple HMI devices are used as decentralized, coordinated operator stations that have access to a centralized HMI device connected to the PLC.

The HMI devices are linked via a TCP/IP-network, (LAN or Intranet /Internet).

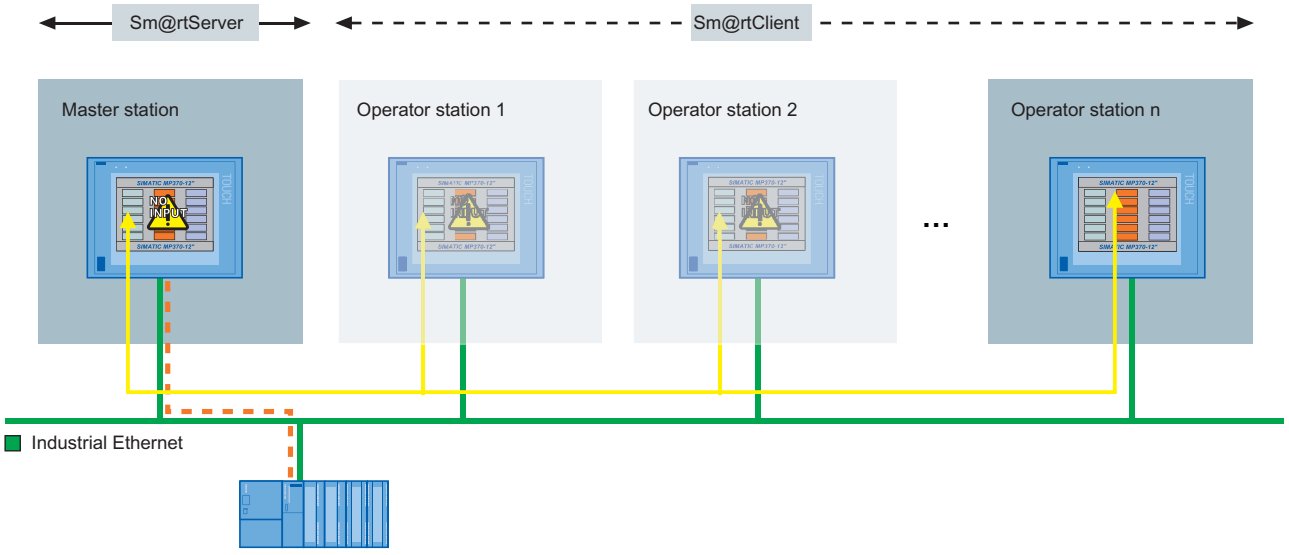


Figure 12-10 Distributed HMI

Only one HMI device, the Sm@rtServer contains the configuration data. The Sm@rtServer is controlled from the HMI devices.

The decentralized operator stations are the Sm@rtClients. These operator stations display the same process screen of the server. A Sm@rtClient-Display is configured in the process screen for the operation and monitoring.

All devices have the same screen resolution.

The operator stations are in shared mode. As soon as a defined period of time elapses without any action on an operator station, another operator station can become active. If Sm@rtClient display is configured accordingly, the user can also log off directly.

Advantages

- Operator control and monitoring can be performed from various locations without significant efforts.
  The project only has to run on one HMI device configured as a server. The same client project runs on all other HMI devices; the Sm@rtClient display object is contained in a screen on these devices. The screen of the server is displayed via the Sm@rtClient display.
- The server is situated remotely from the machine and is thus not exposed to the environmental conditions of the machinery room.
- Coordinated operation is provided by the Sm@rtServer. Additional PLC investments are not required. For example, the load on the field bus is also reduced – the communication load on the bus is removed due to the interlocking mechanisms on the PLC side.

Configure distributed operator stations

Introduction

Operator control of an extensive printing machine need to have the option to exercise control, when necessary, at multiple locations along the machinery. Depending on his current location, the operator must be able to access the process from an operator station in the vicinity.

Requirement

- The HMI-device with the configuration data is connected with the control.
- The server-HMI device and the Client-HMI devices are networked with each other via TCP/IP-Network.
- The License Key "Sm@rtServer" is available.

Configuration steps

The following basic steps are necessary for configuring the distributed operator stations:

Step	
1	Configuring Sm@rtServer (Page 7123)
2	Setting WinCC Runtime Advanced Internet (Page 7124)
3	Project Sm@rtClient (Page 7125)

Configure Sm@rtServer

Configuring Sm@rtServer

Requirement

- The WinCC-Project for the Server-HMI device is configured.

Procedure

Proceed as follows to configure the Sm@rtServer in WinCC:

1. Double-click on the "Runtime-settings" entry in the project tree.
2. In the "Runtime-settings" editor, click on the "Services".
3. Enable "Sm@rtServer" in the group "RemoteControl".
4. Transfer the compiled WinCC-project to the Server-HMI-device.

Result

The Server-HMI device is configured as Sm@rtServer .

Setting WinCC Runtime Advanced Internet

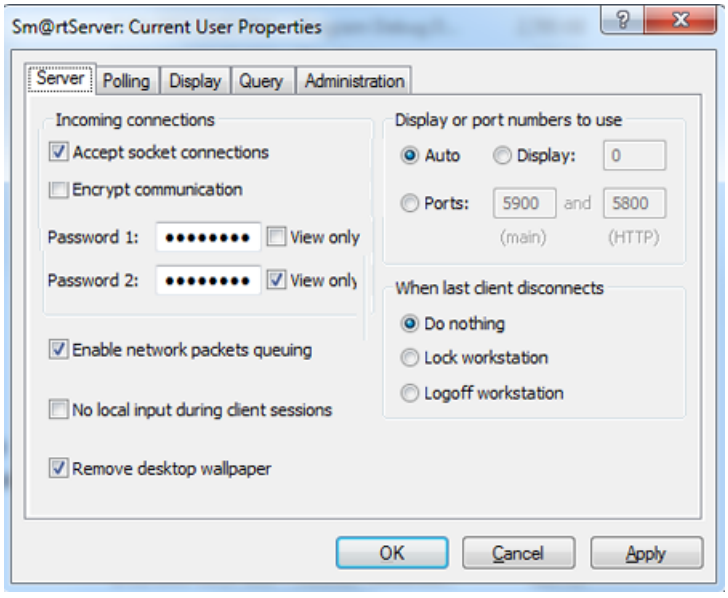
Requirement

- The "Control Panel" opens.
- The "WinCC Runtime Advanced Internet" dialog is open.
- The "Remote" tab is displayed.

Procedure

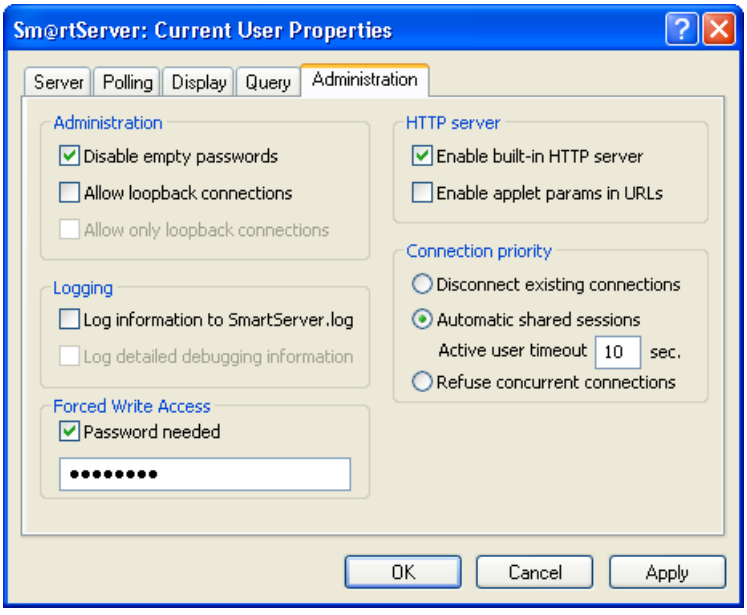
Proceed as follows to change the "WinCC Runtime Advanced Internet" settings on the Sm@rtServer:

1. On the "Remote" tab, select Start automatically after booting".
2. Click on the button "Change settings". The dialogue "Sm@rtServer: Current User Properties" is opened.



3. On the "Server" tab, select "Accept socket connections".
4. Enable "Encrypt communication" to establish an encrypted connection to the server.
5. Enter a password for "Password 1 "and "Password 2". Click "Apply".

6. Click on the "Administration" tab.



7. In the area Connection priority", select Automatic shared sessions". For "Active user timeout" enter time that must elapse without any actions on the active HMI device before access can be changed.
8. Under the "Forced write access", clear Password needed" for the forced access to the HMI device Click "Apply". Click "OK" to close all opened dialogs.

Result

The settings were changed. The changes will be effective after restarting the Sm@rtServer.

See also

Project Sm@rtClient (Page 7125)

Project Sm@rtClient

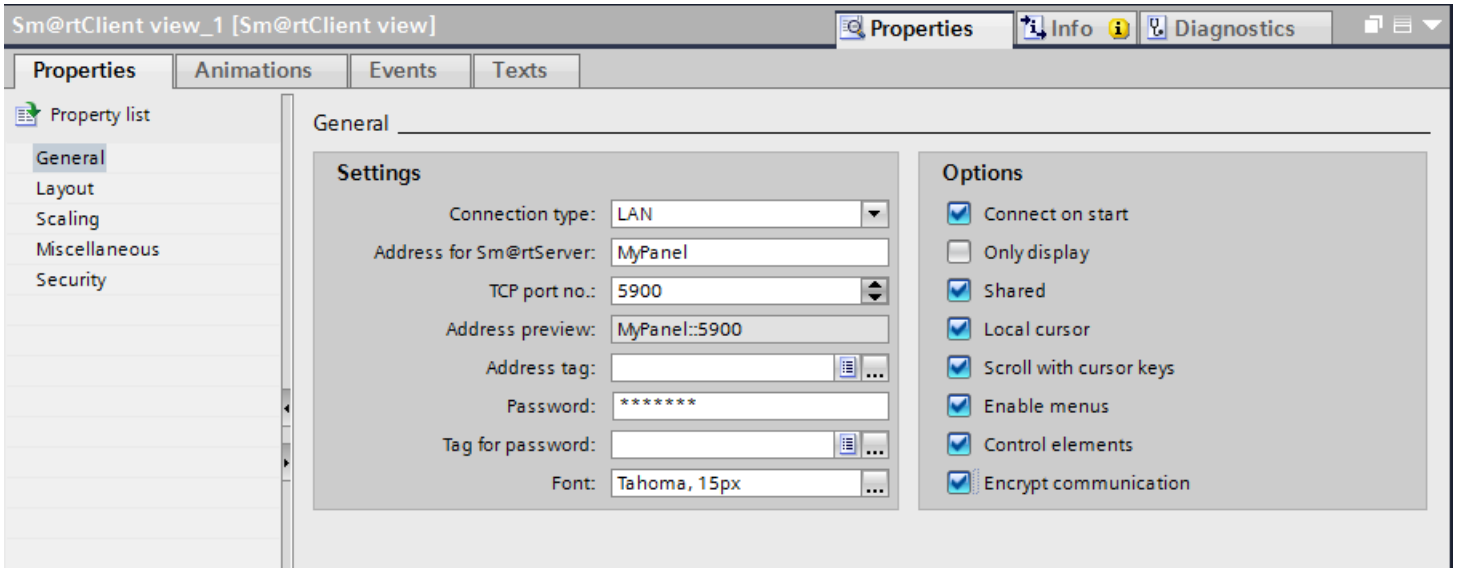
Requirement

- The WinCC project for the client HMI device is configured.
- The "Screens" editor is open.
- The Inspector window is shown.
- The "Tools" task card is open.

Procedure

Proceed as follows to configure the Sm@rtClient:

1. Insert the Sm@rtClient display in the start screen.
2. In the Inspector window, click "Properties > Properties > General".
3. Enter the IP address or the name of the server HMI device in the "Address Sm@rtServer" field.
4. Enter "Password 1" configured on the server in the "Password" field.
5. Activate the "Encrypt communication" function in the "Options" area.
   When the "Encrypt communication" function is enabled, the connection to the server is encrypted through the exchange of certificates.
6. Activate the setting "Allow Menu".
   This provides the operator with the option of logging off using the menu.



7. Transfer the compiled project to all operator stations.

Result

The Sm@rtClient display was added to the start screen in the WinCC project of the Sm@rtClient.

After the Sm@rtServer and the operator stations are started for the first time, a check is carried out as to whether communication is encrypted.

- If communication is encrypted, save the certificate of the Sm@rtServers on the HMI device. Afterwards, you connect the Sm@rtClient with the Sm@rtServer using encrypted communication.
- If communication is not encrypted, you connect the Sm@rtClient with the Sm@rtServer without a certificate.

In order to control the server from an operator station, the operator must wait a specified amount of time following the last action on another HMI device.

If the operator uses the menu of the Sm@rtClient display to log off at the previously used HMI device, he can immediately control the server at the next HMI device.

See also

Remote control via the Sm@rtClient display during runtime (Page 7118)

"Server" tab (Page 7093)

Setting WinCC Runtime Advanced Internet (Page 7122)

12.15.2.3 E-mail notification from runtime

Process flow

Introduction

WinCC Runtime Advanced with the Sm@rtService offers the option of sending messages automatically via email.

The automatic e-mailing feature ensures that all people affected by the machine status (for example, shift engineer and sales manager) are informed in a timely manner.

Contents and triggers for e-mailing

The following events can trigger an e-mail to be sent:

- Alarm of a certain alarm class
- Event in which a standard function has been configured, such as a tag value change, etc.

Such an e-mail can have the following contents:

- Alarm text with process tags (maximum of 256 characters)
- Date/time
- E-mail address for replies

If you use e-mail gateways or SMS gateways, you receive access to standard networks, which requires external service providers. If configured accordingly, in critical situations the operator station sends an SMS to your mobile phone.

Enabling e-mailing and SMS

The HMI device can send e-mails to an SMTP server only. The server sends the e-mails to the addresses configured in the server.

Nothing else is required to send e-mails to addresses in the company network. However, an external service provider is required to access standard networks.

If an SMS communication is to be sent to service personnel, an SMS gateway is required as well.

Settings on the HMI device

The settings for emailing on the HMI device are made in the "Email" tab under "WinCC Runtime Advanced Internet" on the control panel.

The "Sender" entry field is assigned the default value "Automation HMI device." A change is useful if you want the recipient to be able to identify the device from which the e-mail originated, e.g. "HMI device on production line 2"

You can also use an SMTP server that support authentication to send e-mail.

The following authentication modes can be configured:

- Authentication by means of a valid e-mail address
- Authentication by means of user name and password

Data can also be encrypted and sent via an SSL connection. This means the data cannot be manipulated or read.

Specify trigger for E-Mailing

Requirement

- You have to create WinCC project.

Procedure

Proceed as follows to send a message when an alarm is triggered:

1. Double-click on the "HMI-Alarms" entry in the project tree.
2. Click the "Alarm classes" tab in the "HMI-Alarms" editor.
3. Select the alarm class, e.g. "Errors".
4. Enter the E-Mail-Address in the inspector window under "Properties > Properties > General".
5. Create an analog or discrete message with this alarm class.

Result

The trigger to send a message was configured. A message is automatically sent when a message of this alarm class is triggered.

Configure secure e-mail notification from Runtime

Introduction

If you send e-mail via the SMTP protocol, the sender is not verified. To ensure secure transmission of e-mails, you can use SMTP servers that support SMTP AUTH (authentication).

You must log on to the SMTP server to send e-mails. The following authentication modes can be configured:

- Authentication by means of a valid e-mail address
- Authentication by means of user name and password

The data can also be sent via an SSL connection. SSL (Secure Socket Layer) encrypts e-mails and user data for transmission. This means the e-mail cannot be manipulated or read during transmission.

Requirement

- The SMTP server supports SMTP AUTH and STARTTLS. You can obtain more detailed information from your service provider.
- User name and password or a valid e-mail address for logon to the SMTP Server. You can obtain this data from your service provider.
- The SMTP server is available.
- The e-mail address of the service technician is entered in the alarm class.
- An analog or discrete alarm has been created for this alarm class.

Procedure in WinCC

1. Double-click on the "Runtime-settings" entry in the project tree.
2. In the "Runtime-settings" editor, click on the "Services".
3. Enter the sender name to be shown in e-mail under "Sender name". If the SMTP server does not support the function, delete the entry.
4. Assign a port number.
   Port number 25 is assigned by default.
   The port number in WinCC must match the port number in the "WinCC Internet Settings" on the HMI device.

   Note

   You can only configure the port number with the following HMI devices:
   - Comfort Panels
   - RT Advanced

5. Enter the authentication data.
   - Authentication by means of a valid e-mail address
     Type in the e-mail address required for SMTP authentication in the "E-mail address" input field.
   - Authentication by means of user name and password
     Enter the user name and your password. You can obtain the user name and password from your service provider.
6. Enable "The server requires a secure connection (SSL)".

Procedure on the HMI device

Note

Note that the settings on the HMI device have a higher priority than the settings in the WinCC project.

1. Open the "WinCC Runtime Advanced Internet " dialog in the control panel of the HMI device.
2. Click on the "E-mail" tab.
3. Specify the SMTP server.
   - Select "Use the default project file" if you want to use the SMTP server defined in the project.
   - Deactivate "Use the default project file" if you do not want to use the SMTP server defined in the project. Specify the required SMTP server. For HMI devices with Windows CE, specify the computer name or the FQDN (Fully Qualified Domain Name).
4. Assign a port number.
   Port number 25 is assigned by default.
   The port number must match the port number in the WinCC Internet Settings in the HMI device.

   Note

   You can only configure the port number with the following HMI devices:
   - Comfort Panels
   - RT Advanced
5. Enter the sender name given in the e-mail under "Name of the sender". If the SMTP server does not support the function, delete the entry.
6. Enter the authentication data.
7. Enter a valid e-mail address at "eMail adress of sender" if required for authentication.
   - Click "Advanced" if you need a user name and password for authentication. The "Advanced Email Settings" dialog opens.
8. Type in the user name and password in the "Advanced Email Settings" dialog.
   - Enable "Use the default of the project file" to use default user data you have defined in the project.
   - Select "Use panel settings for authentication" if you do not want to use the user data defined in the project. Enter the user name and password.
9. Enable transmission via SSL.
   - To use the project settings, enable "Use the default of project file" and SSL in WinCC.

Result

If a tag such as a mixer speed exceeds configured limits, a corresponding alarm is displayed on the HMI device. The data is sent to the SMTP server via SSL connection. The e-mail is sent to the field service technician after successful logon.

12.15.2.4 Display integrated Service-Pages

Integrated Webserver

Introduction

The operator can display and navigate between web pages during runtime using the web server integrated in the HMI device.

The integrated web server displays the integrated service-pages. Depending on the configuration, own configured HTML-pages or Service-pages of a server accessible over Ethernet are be displayed.

Requirement

- "HTML-Pages" is activated in the WinCC-Project of the Server-HMI device for the "Services in Runtime".

Note

It is always possible on a PC to access HTML-pages in runtime, although the option "HTML-pages" is cleared. Setup always installs the standard pages of the Web Server on the PC. Assign an administrator password to prevent unauthorized access to the pages.

Purpose of the web server

The integrated web server permits HTML pages to be displayed during runtime over one of the following routes:

- Internet Explorer
- HTML browser screen object during runtime (not on Windows CE devices)

The following are displayed:

- internal Service-Pages available by default on the HMI-device
- Other pages that you configure
- Other Internet pages

An operator or service technician can access service-critical information via the HTML pages. The standard HTML pages provide the following options:

- Remote control (if the HMI device is configured as a Sm@rtServer)
- Remote control using Microsoft Internet Explorer
- Starting and stopping of runtime
- Remote access to recipe data records and password lists
- Display of system information
- File management using a file browser
- Downloading of configuration data

- A "DATETIME" tag always returns a date within the range from 1.1.1970 00:00:00 to 31.12.2037 23:59:59.
- The "Export recipes" function requires the following authorizations:
  - PC: "UserData"
  - other HMI devices: "UserData" and "FileBrowserUser"

HTML browser for HTML pages

The HTML-pages are also displayed using the configured "HTML browser" screen object (not on Windows-CE devices).

You can can also arrange for input or activation of an Internet address. As soon as the operator enters or activates an address, the HTML browser opens the relevant page.

The appearance and functionality of the HTML browser screen object depends on the HMI device type. On PCs, the HTML browser corresponds to the Internet Explorer installed.

Note

Note that the HTML browser options during runtime are restricted due to operating device capacities and options.

Service-pages of the web server

Introduction

The operator can use Internet Explorer or the HTML browser screen object during runtime to display service-pages without any additional configuration.

You can also create own service-pages. For detailed information, refer to "Configure In-house service-pages".

Requirement

- "HTML-Pages" is activated in the WinCC-Project of the Server-HMI device for the "Services in Runtime".

  Note

  It is always possible on a PC to access HTML-pages in runtime, although the option"HTML-pages" is cleared. Setup always installs the standard pages of the Web Server on the PC. Assign an administrator password to prevent unauthorized access to the pages.

Service-pages

WINCC Runtime has the following service-pages:

- start.html: Home page
- RemoteControl.html: Remote control (only for Internet Explorer)

- Control.html: Control functions
- StatusDetails.html: System diagnostics
- Browse.html: File browser (only for Internet Explorer)

Home page: Start.html

The start page contains the links to all other pages and displays current information about the project: Mode, software versions, device data, etc.

"Remote control": RemoteControl.html

The "Remote control" page enables operator control of the HMI device for which a page is to be displayed. This page can only be displayed by using the Internet Explorer.

"Control functions": Control.html

The "Control functions" page enables the following options on the HMI device for which a page is to be displayed:

- Starting and stopping of HMI runtime

  Note

  The transfer mode must be set in the Loader-menu on the HMI-device.

- Exporting and importing of recipes

  Note

  After importing recipes with Sm@rtService (HTML pages), restart Runtime. The imported recipes only become active the next time Runtime is started.

- Exporting and importing of password lists

  Note

  The password list must be named "pdata.pwl." It is exported to the following directory:

  On Windows CE-devices: In the "\Flash\simatic\" target directory

  On PCs: the folder that was set in the file "HMILoader.exe".

  The password list is exported and becomes active the next time Runtime is started.

"System diagnostics": StatusDetails.html

The "System diagnostics" page contains system alarms from the alarm buffer.

"File Browser" – Browse.html

The "File Browser" page is used to administer directories and files on the remote device. This page can be displayed with any Internet browser.

Installing the client and server certificates for SSL

Introduction

To ensure data security, data are encoded for transmission over the Internet. Encoding and decoding is performed by appropriate software – the certificates for SSL (Secure Sockets Layer).

- The client certificate for SSL must be installed on devices that are to be used to control a remote device.
- The server certificate for SSL must be installed on HMI devices that are to allow remote control.

Configure access to service-pages

Configure integrated web server

Configure WinCC-Project

Requirement

- The WinCC-Project of the server-HMI-device is configured.

Procedure

Proceed as follows to configure the HMI device in such a way that other HMI devices or PCs can be connected to it:

1. Double-click on the "Runtime-settings" entry in the project tree.
2. In the "Runtime-settings" editor, click on the "Services".
3. Enable the "HTML-Pages" in the "Diagnostics" group.
4. Transfer the compiled WinCC-project to the Server-HMI-device.

Result

The Server-HMI-device is configured as web server.

Setting WinCC Runtime Advanced Internet

Requirement

- The "WinCC Runtime Advanced Internet" dialog is open.
- The "Web Server" tab is displayed.

Procedure

Proceed as follows to change the "WinCC Runtime Advanced Internet" settings on the HMI device:

1. Click "User Administration" in the "Web Server" tab.
2. Open the "UserDatabase-Edit" dialog.
3. Click "Add" in the "User manager" tab to create a new user.
4. Enter a user name and specify a password.
5. Click on "Apply".
6. Click the "Authorizations" tab.
7. Specify on the "Authorizations" tab, which functions can the user carry out on the HTML-pages of an HMI-device. See the section "User administration for web server (Page 7099)" for additional details.
8. Close the "UserDatabase-Edit" dialog.
9. On the "Remote" tab, select the "Start automatically after booting" check box.
10. Click "Change settings" and enable in the "Sm@rtServer dialog: Current User Properties", select the checkbox "Enable connections".
11. Specify a password for "Password2" so that the HMI device can be remotely controlled by the service technician.

Result

A user was created on the HMI-device in the user administration of the web server and configured for the remote control.

The service technician can be connected to the HMI-device by means of the Internet Explorer and the Sm@rtClient Application. After disconnecting the connection, the HMI-device can be operated from his PC.

See also

WinCC Runtime Advanced Internet, "Web Server" tab (Page 7091)

"Server" tab (Page 7093)

Display and remote-control Service-Pages

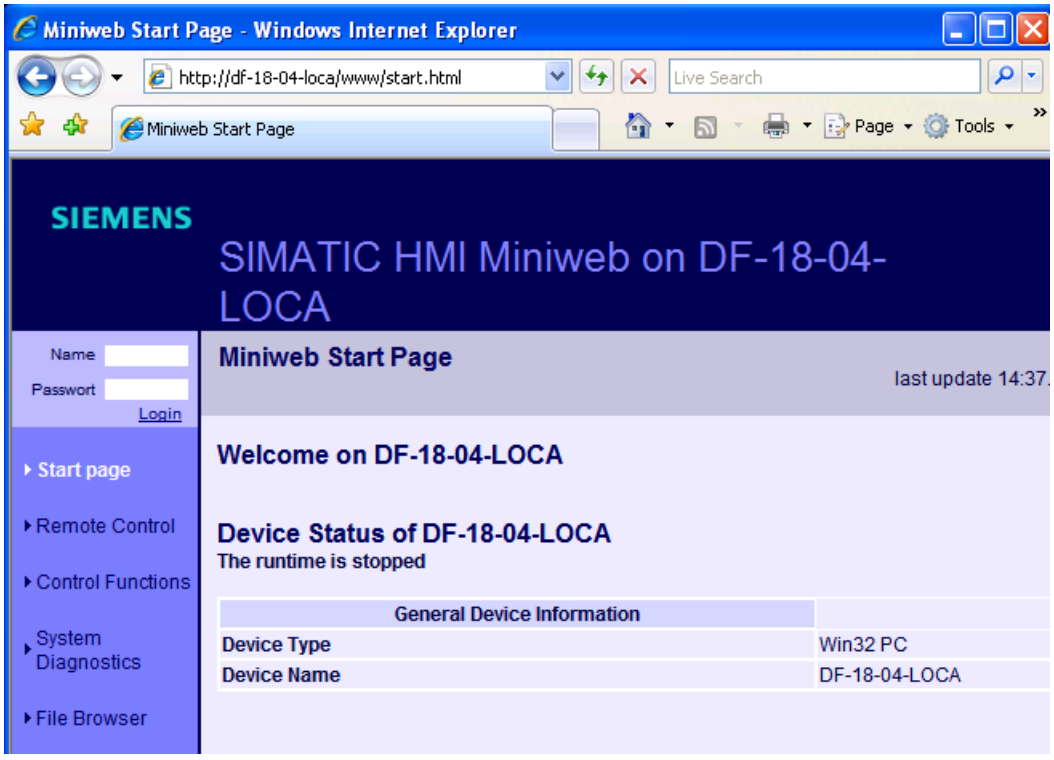
Requirement

- A user is created on the HMI-device in the user administration for the web server.
- The web server is started.
- The Client-and Server-certificates are installed to ensure the data security when transferred via internet.

Procedure

To display and control the service-pages, follow these steps:

1. Start the Internet Explorer on the configuration-PC and connect with the "Homepage" of the HMI-device.



2. For "Name" and for "Password", enter the data of the user configured in the user administration of the web server. Click "Login".
3. Click "System Diagnostics". The system alarms from the alarm buffer are displayed on this page.
4. Click "Remote Control" to remotely control the HMI-device.

Result

The service-pages are displayed. The HMI-device can be operated or monitored via the service-pages.

A keyboard units cannot be operated completely in the Internet Explorer, since only the screen content is displayed. Use the Sm@rtClient-Application to remotely control the keys of HMI-device. The Sm@rtClient-Application can be located under "Start > Program > Siemens Automation > Runtime Systems > WinCC Runtime Advanced > Sm@rtClient"

Create own Service-pages

Basics

Introduction

The basic framework of the service-pages corresponds to a normal HTML-file.

- Declaration of the document type
- Header with data for the title
- Body - content to be displayed.

Variable parameters in Service-pages

You can specify variable parameters in HTML documents. As soon as a page with variable parameters is opened, the parameters are replaced by specific values.

<BODY > Welcome on <MWSL><!-- write(GetVar("[Parameter]")); --></MWSL></BODY>

Available variable parameter

Parameters	Meaning
ProgramMemoryComplete	CE only: Total program memory
ProgramMemoryFree	CE only: Program memory available
ProgramMemoryUsed	CE only: Program memory utilized
FlashComplete	CE only: Total flash memory
ObjStrComplete	CE only: Total available flash memory
ObjStrFree	CE only: Volatile memory available
ObjStrUsed	CE only: Volatile memory utilized
DeviceType	Type of target device as specified in the control panel.
BtLdVer	CE only: Bootloader-version, as specified in the control panel.
BtLdRelDate	CE only: Bootloader release date
ImageVersion	CE only: Image version as it appears on the loader
DramSize	CE only: Size of DRAM
HostName	The name by which the device is logged on/identified in the network.
RtState	Indicates whether Runtime is running on the target device.
SystemMessageTable	Outputs a table containing the current system events.

In the example below, the "HostName" parameter is replaced by the network-name of the device.

<BODY > Welcome on <MWSL><!-- write(GetVar("[HostName]")); --></MWSL></BODY>

Process tag

Process tag values can also be displayed in HTML pages. The syntax is the same as for device tags. Use the tag name as a placeholder for the tag value, , e.g. Tag_1.

<BODY > Welcome on <MWSL><!-- write(GetVar("[Tag_1")); --></MWSL></BODY>

Caution

Data inconsistency caused by HTML pages

Note the information below if the "Cyclic in operation" acquisition mode is set at tag:

1. If this tag is not displayed on the HMI device, the HTML page displays the incorrect tag value in the following situations:
   - The HTML page display a "0" value at its first call. The HTML page only displays the correct value after it is called again or updated.
   - The last value is displayed if the connection to the PLC goes down.
2. The HTML page also displays the correct tag value if the tag is displayed on the HMI device.

Situation 1 is based on standard behavior: If a tag is currently not in use currently and its value is not acquired in "Cyclic continuous" mode, the tag is loaded with its initial value in Runtime. Instead of reading the values from the PLC, however, the HTML page receives these from Runtime.

Link own Service-pages

If a user connects to an HMI device, he is automatically forwarded to the start page http:// <Device name>/www/start.html. This page represents the starting point for the HTML-pages of the web server. Every standard page is accessible from the start page via a link. For this reason, you insert a link for each of your HTML pages in the start page.

Note

When inserting links in the HTML page, you must differentiate between relative and absolute links. Make sure that absolute links start with "/www" to ensure that the document will be searched for in the correct directory. Example: "/www/MyDocument.HTML".

Storage location of the service-pages

If files are to be located during a transfer, they must be in a specific directory:

- on a PC with Windows 7: "C.\ProgramData\Siemens\CoRtHmiRTm\MiniWeb11.x.x \WebContent"
- on a PC with Windows XP: "C:\Documents and Settings\All Users\Application Data \Siemens\CoRtHmiRTm\MiniWeb11.x.x\WebContent"
- On xP 177B: "<ES-Installationspath>\Transfer\11.0\XP177B\WebContent.zip"
- On xP 277: "<ES-Installationspath>\Transfer\11.0\XP277\WebContent.zip"
- On MP 177: "<ES-Installationspath>\Transfer\11.0\MP177\WebContent.zip"
- On MP 377: "<ES-Installationspath>\Transfer\11.0\MP377\WebContent.zip"
- on Mobile Panel 177 PN: "<ES-Installationspath>\Transfer\11.0\XP177B\WebContent.zip"
- On Mobile Panel 277: "<ES-Installationspath>\Transfer\11.0\XP277\WebContent.zip"

- On the Mobile Panel 277 (F) IWLAN: "<ES-Installationspath>\Transfer\11.0\XP277_W\WebContent.zip"
- On the Mobile Panel 277 (F) IWLAN V2: "<ES-Installationspath>\Transfer\11.0\XP277_W2\WebContent.zip"

Create service-page for displaying process values

Requirement

The Ta_1 and Tag_2 tags are created in the WinCc-Project.

Procedure

To create an own Service-page, follow these steps:

1. Copy the "WebContents" ZIP-file in a random work directory on your Configuration-PC and un-zip the ZIP-file.
2. Create a copy of start.html and rename the copy in "tag.hml".
3. Open the "tag.html" in a text editor, e.g. Notepad.
4. Replace the existing table with a new table, in which the process values of "Tag_1" and "Tag_2" tags are displayed. Save the file"tag.html".

```
<font class="ad_headline2">Device Status of <MWSL><!– write(GetVar("HostName")); –></MWSL></font><br>
 <b>The runtime is <MWSL><!– write(GetVar("RtState")); –></MWSL></b><br><br>
<table border="1" class="sph_table" cellspacing="0" width="600">
<tr><th class="sph_th"><b>Display of process tags </b></th></tr>
<tr><td class="sph_td"><b> "Tags1"</b></td><td class="sph_td"><MWSL><!– write(GetVar("Tag_1")); –></MWSL> </td></tr>
<tr><td class="sph_td"><b>"Tags2"</b></td><td class="sph_td"><MWSL><!– write(GetVar("Tag_2")); –></MWSL> </td></tr>
</table>
```

5. Open the "start.html" file and add a hyperlink to page "tag.html". Expand the available navigation bar, in which you supplement the existing table by an entry.

```
<tr>
  <td width="8"></td>
  <td width="7"><img src="/Images/NaviPfeilblau.gif" alt="" border="0"></td>
  <td width="101" class="ad_nav_link"><a href="tag.html" class="ad_nav_link">Process value</a></td>
</tr>
```

6. Save start.html.

Result

You have created the service-page "tag.html". You have added a hyperlink on the start page in order to navigate to the service-page from the start-page.

Transfer Service-pages

Transfer files using the standard-path (Active Sync/CF- card)

Proceed as follows to transfer files via the standard-path:

1. Copy the changed HTML pages and pictures according to "\Flash\Simatic\WebContent". Access then takes place with "http://<device>/www/<HTML page>".

Transfer files via the project transfer

Proceed as follows to transfer the files via the project transfer:

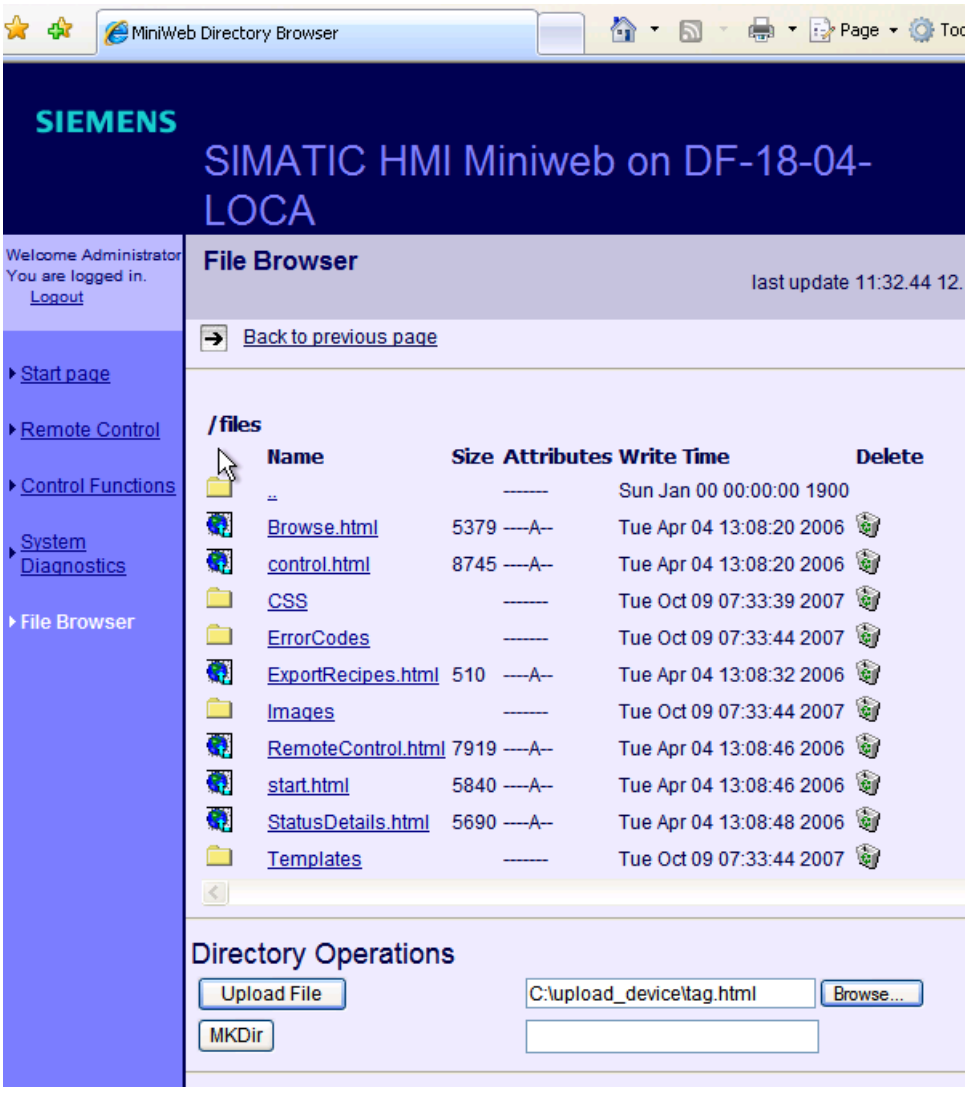
1. Add the changed files to the ZIP-file "WebContents". This file must contain all HTML pages and associated pictures.
   Make sure to provide the correct path information because the files are unpacked in the directories specified in the zip file. Incorrect path information results in errors in direct addressing or due to links.
2. In order to transfer the ZIP-file "WebContents", copy this in a certain directory, e.g. for the transfer to an MP377"<ES-Installationspath>\Transfer\1.3\MP377\WebContent.zip".
3. Transfer the project to the HMI device.
   The ZIP-file "WebContents" is transferred to the Windows CE device where it is unzipped.

Transfer files using the File Browser

Proceed as follows in order to transfer files using the file transfer:

1. Start the Internet Explorer on the configuration-PC and connect with the "Homepage" of the HMI-device.
2. Log-on to the internal Web Server to work with the File Browser.
   For read and write access to the file browser, the user must possess the web authorizations "FileBrowserAdministrator" and "FileBrowserUser" .
3. Click on "Browse" in the File Browser . The file selection dialog opens.

4. Navigate to the file storage location by means of this dialog. Select the desired file and click "Open."



5. Click "Upload File". The file is copied in the directory of the internal web server.

See also

Basics (Page 7135)

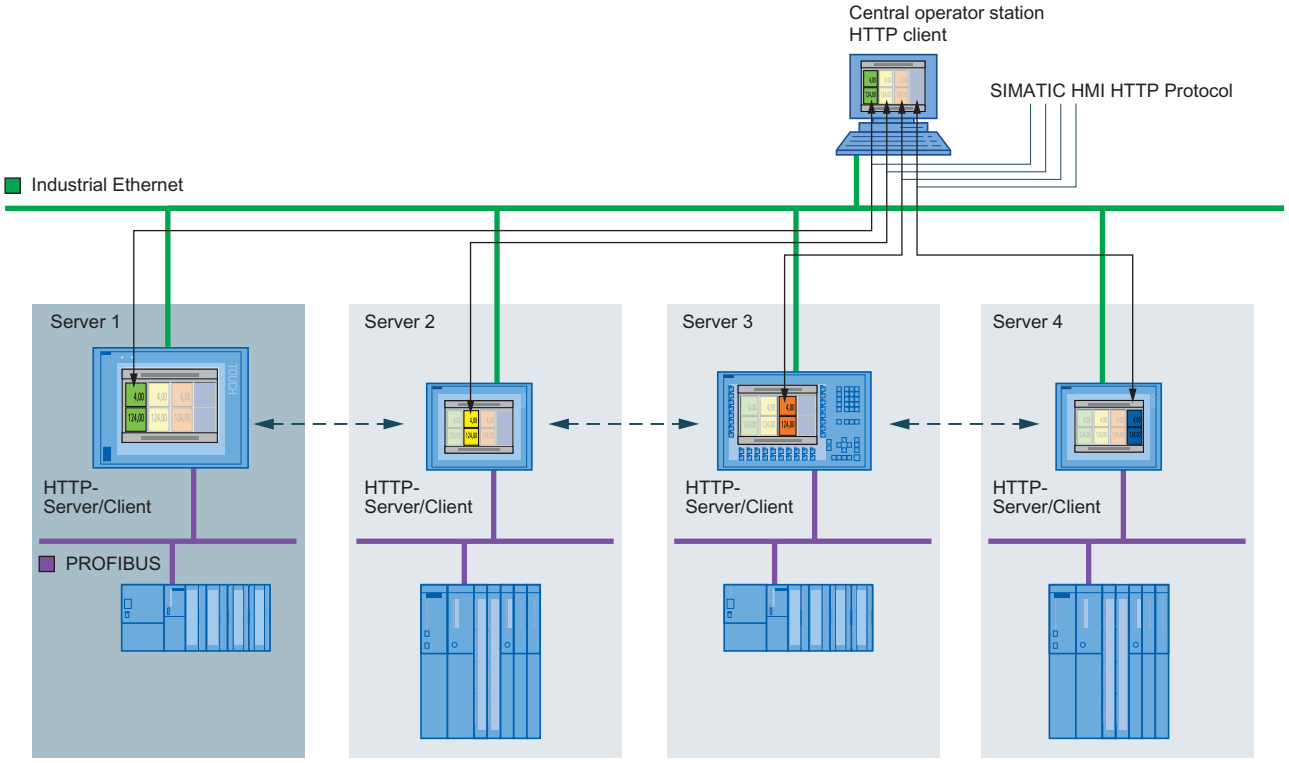
12.15.2.5 Access via SIMATIC HMI HTTP Protocol

Configuration

Configuration

During the communication via SIMATIC HMI HTTP Protocol, an HMI-device accesses the tags of a different HMI-device. The access is "read-only" or "read and write" depending on the configuration of the concerned HMI-device.

The HMI device providing the tags is the HTTP-server; the other HMI-device is the HTTP-client. However, access to tags functions in both directions.



Configure access via SIMATIC HTTP Protocol

Introduction

The tags in service-application should be illustrated in an overview for a configuration from multiple HMI-devices.

The panels in the machine level are used as tag server. The service-application illustrating tags of machines in an overview image runs on a PC.

Requirement

- The HMI-devices are networked via a TCP/IP-network with each other.

Configuration steps

The following basic steps are required to configure the access via "SIMATIC HMI Protocol".

Step	
1	Configure WinCC-Project (Page 7144)
2	Setting WinCC Runtime Advanced Internet (Page 7144)
3	Configuring HTTP connections in the client (Page 7145)
4	Configure the HTTP-Client tags (Page 7147)

Permissible data types (SIMATIC HMI HTTP protocol)

Permitted data types

When configuring tags, the data types listed below can be used.

Data types in the HTTP Protocol	Length	Signs	Range of values
Bool	0	No	true (-1) or false (0)
Char	1 byte	Yes	-128 to 127
Byte	1 byte	No	0 to 255
Int	2 bytes	Yes	-32768 to 32767
UInt	2 bytes	No	0 to 65535
Long	4 bytes	Yes	-2,147,483,648 to 2,147,483,647
ULong	4 bytes	No	0 to 4,294,967,295
Float	4 bytes	Yes	-3.402823E38 to -1.401298E-45 for negative values and 1.401298E-45 to 3.402823E38 for positive values
Double	8 bytes	Yes	-1.79769313486231E308 to -4.94065645841247E-324 for negative values and 4.94065645841247E-324 to 1.79769313486232E308 for positive values
String	1 to 255 byte	——	
DateTime	8 bytes	——	1.1.1970 00:00:00 up to 31.12.2037 23:59:59

Please note that data types may be defined in external controllers which have different names in WinCC. To ensure correct assignment, please observe the tag definition in the external controllers.

Note

It is not possible to access array tags from an HTTP client.

See also

Configure WinCC-Project (Page 7144)

Configure HTTP server

Configure WinCC-Project

Requirement

- The WinCC-Project for the Server-HMI device is configured.

Procedure

Proceed as follows to configure the HTTP-Server:

1. Double-click on the "Runtime-settings" entry in the project tree.
2. In the "Runtime-settings" editor, click on the "Services".
3. Select "SIMATIC HMI HTTP Server" in the group "Read/write tags".
4. Check the data types of the tags. The HTTP-client can access only those tags, whose data type is supported by communication driver "SIMATIC HMI HTTP Protocol". For additional information, refer to "Permissible data types (SIMATIC HMI HTTP Protocol) (Page 7141)".
5. Transfer the compiled WinCC-project to the Server-HMI-device.

Result

The HMI-device is HTTP-server configured.

Setting WinCC Runtime Advanced Internet

Requirement

- The "Control Panel" opens.
- The "WinCC Runtime Advanced Internet " dialog is open.
- The "Web Server" tab is displayed.

Procedure

Proceed as follows to change the "WinCC Runtime Advanced Internet" settings on the HTTP server:

1. Specify the access to tags in case of "Tag access".
   - "Read/write": read and write access
   - "Read only": read access
2. Specify the authentification for access in case of "Tag authenticate":
   - "No authentication": No authentication required.
   - "Authentication required": A password is required for the access. Specify the password for configuring the connection via the SIMATIC HMI HTTP Protocol .
3. Click "User Administration" in the "Web Server" tab. Enter your password. The "UserDatabase-Edit" dialog is opened. For detail instructions, refer to "User administration for Webserver (Page 7099) ".
4. Click "Add" in the "User manager" tab to create a new user. Enter a user name and specify a password. Click on "Apply".
5. Click the "Authorizations" tab.
6. Specify the web-authorizations on the tab "Authorizations". The user must have the Web-authorization "RTCommunication" for utilizing the SIMATIC HTTP Server.
7. Close all open dialog boxes.

Result

The settings were changed. The changes will be effective after the restart of the WebServer.

See also

Settings on the HMI device (Page 7088)

WinCC Runtime Advanced Internet, "Web Server" tab (Page 7091)

User administration for web server (Page 7099)

Configuring HTTP clients

Configuring HTTP connections in the client

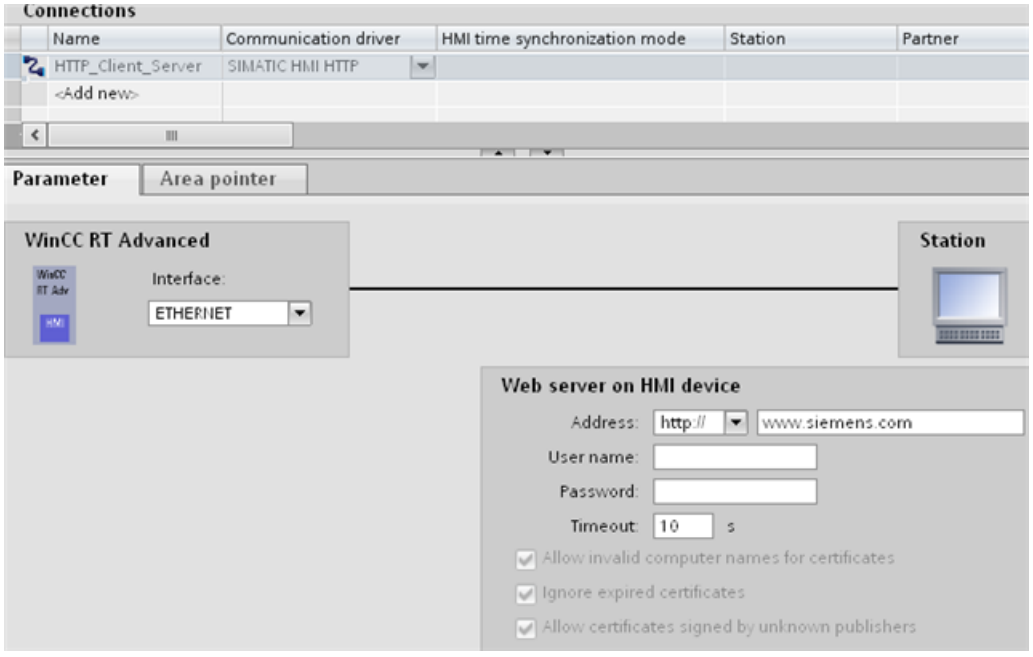
Requirement

The communication driver "SIMATIC HMI HTTP Protocol" is installed.

Procedure

Proceed as follows to create an HTTP-connection:

1. Double-click on the "Connections" entry in the project tree. The "Connections" editor opens.
2. Create a connection. Select "SIMATIC HMI HTTP Protocol" for "Communication driver".



3. Select "Ethernet" for "Interface". Select the protocol type "http://" or "https://" for address.
4. Enter the name of the HTTP-server or its IP address.
   Ask your network administrator for the specific name or parameters of your network.
   If the server has already been commissioned, you can read out the IP address on the server as well:
   - Panel
     Click "Start > Programs > Command Prompt" on the server and enter the "ipconfig" command using the screen keyboard. Press <Enter> to display the IP-address.
   - For PC/Panel PC
     Click on the server on "Start > Run", enter "Cmd", and press <Enter>. The command interpreter is displayed. Enter the "ipconfig" command. Press <Enter> to display the IP-address.
5. If the "HTTPS" protocol type is selected, you can establish how the HTTPS-client verifies the properties of the server-certificate and how it should react in the event of error:
   - "Allow invalid computer names for certificates"
   - "Allow expired certificates"
   - "Allow certificates signed by unknown publishers"
6. If the "Authentication required" option is selected on the HTTP-server, enter the user name and the password.
7. Enter the time for "Timeout" after which disconnection is identified.

Result

A connection was created in the WinCC-Project of the HTTP-Client. For detail information for HTTPS-connection, refer to "Commissioning an HTTP- connection (Page 7148)".

Configure the HTTP-Client tags

Requirement

- An HTTP-connection was created in the WinCC-Project of the HTTP-Client.
- A tag is created in the WinCC-Project of the HTTP-Server. The data type of the tags is supported by SIMATIC HMI HTTP Protocols .

Procedure

To create tags on the HTTP-client, proceed as follows:

1. Open the "HMI- tags" folder in the project tree and double-click the entry "Standard-tag table". The "Tags" editor opens.
2. Enter a clear tag-name for "Name" in the Inspector window under "Properties > Properties > General".
3. Select the HTTP-connection for "Connection".
4. Select the data type for "data type".
   The client does not check any verification of the tag name and the data type. Pay attention that the selected data type here matches the data type of the tags in the HTTP-server. For additional information, refer to "Permissible data types (SIMATIC HMI HTTP Protocol". Array tags are not permitted.
5. Enter the exact name of the tag that is to be communicated with on the HTTP-server in the "Address" field.
   If the tag to be addressed is in a sub-folder, the complete path along with tag name must be given as address, e.g.[folder name]\[Tag name].

Result

A tag was created in the WinCC-Project of the Client-HMI-device. The tag has access to the HTTP-server tag via an HTTP-connection. You can use an "E/A-Field" in an image to display the process value of this tag.

Commissioning an HTTP- connection

Introduction

To establish an HTTP connection, you must perform the following actions:

- In the "Connections" editor of WinCC ES, configure the connection as an "https://" protocol type and define how the HTTPS client should verify the properties of the server certificate and respond to errors.
- Install a valid certificate on the HTTPS client.
  Certificates are necessary for server authentication. Using certificates you can ensure that the server with which the connection is to be developed is actually the server for which it is outputting.

Principle of an HTTPS connection

After runtime start, the HTTPS client establishes a connection to the HTTPS server. The HTTPS server presents its certificate, which the client verifies for authenticity. The session code that can only be read by the HTTPS server is then transmitted. The session code is now available on both sides and enables a symmetrical data encryption.

Note

The certificate contains the current time. The current time can lead to problems if the time zones of the server and client are different. For example, a certificate generated on a server with an Asian time zone only becomes valid on a client with European time zone in the future (8 hours).

Preparation for installing a certificate on the client

With the first HTTPS client access, the HTTPS server generates the certificate itself and then saves it in the "Cert.cer" memory. The file is stored in the following directory:

- On a PC / Panel PC (with Windows XP) in the directory "<Runtime-Verzeichnis>\SystemRoot\SSL"
- On Windows CE-based devices in the directory "Flash\Simatic\SystemRoot\SSL"

The certificate must be stored on the HTTPS client on a storage medium from which it can be launched with a double click. You can select from the following transfer options:

Server	Client	Possible file transfer
with Windows XP (PC, Panel PC)	with Windows XP (PC, Panel PC)	• Diskette • USB stick • LAN (Ethernet) • Internet Explorer (via TCP/IP if service is already running)
with Windows CE (xP 277, MP 377, xP 177B, Mobile Panel 177 PN, Mobile Panel 277, Comfort Panels)	with Windows XP (PC, Panel PC)	• Memory card • ActiveSync (serial)
with Windows XP (PC, Panel PC)	with Windows CE (xP 277, MP 377, xP 177B, Mobile Panel 177 PN, Mobile Panel 277, Comfort Panels)	
with Windows CE (xP 277, MP 377, xP 177B, Mobile Panel 177 PN, Mobile Panel 277, Comfort Panels)	with Windows CE (xP 277, MP 377, xP 177B, Mobile Panel 177 PN, Mobile Panel 277, Comfort Panels)	• Memory card

Installing a certificate on a client with Windows XP

Insert the storage medium on which you have saved the "Cert.cer" file into the HTTPS client or open the directory in which the file is located. Double click on the file and follow the instructions in the Windows dialog.

Tip: The Internet Explorer provides an easy way to install a certificate. Connect to this device via HTTPS (e.g.: https://<my device>). The browser establishes if a certificate has not yet been imported. In this case, the browser asks if you want to install the certificate. Any faults in the certificate are displayed.

Installing a certificate on a client with Windows CE

Insert the memory card on which you have saved the converted "Cert.cer" file into the HTTPS client. WinCC includes the "InstallCert.exe" tool for importing certificates with Windows CE.

You can implement the installation as follows:

- In Explorer:
  Double click the "Cert.cer" file to install the certificate.
- At the command prompt:
  Enter "InstallCert /[command parameter] [filename]".
  - command parameters:
    Parameter /r must be specified because the certificate used in WinCC Runtime Advanced is a root certificate.
    A root certificate is the main certificate and is used to verify the authenticity of all other certificates transferred.
  - filename
    You must specify the certificate file with its complete path (e.g. "\Storage Card\Cert.cer")

A status alarm is output when you completed the installation. Runtime has to be restarted after the installation of a certificate on Windows CE- HMI devices with HTTPS clients. It is necessary to restart Runtime so that an HTTPS connection can be established.

The file "Cert.cer" cannot be opened.

The "Cert.cer" file generated at the HTTPS server cannot be opened on HMI devices based on Windows CE 5.0 by double-clicking the client.

1. Open the Control Panel.
2. Select "Certificates > My Certificates".
3. Click the "Import" button.
   A dialog box opens.
4. Select the "From a File" menu in the file browser and select the "Cert.cer" file.

12.15.2.6 Connection to the Office-world

Configuration

Data access via web service (SOAP)

WinCC provides options for utilization of web-service (SOAP). Web service (SOAP) is based on the Simple Object Access Protocol. Use of this protocol enables an external application to access tags of an HMI device via Ethernet. If the company network is protected by a Firewall, the system administrator must release the appropriate ports.

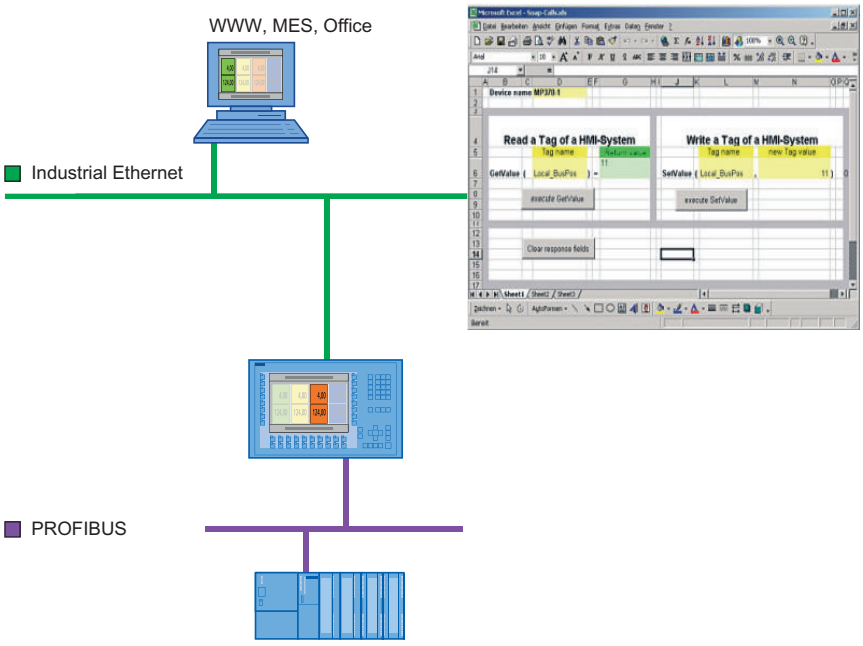


Figure 12-11 Communication with other applications

For example, a device is accessing two HMI devices. The operator sees the values of certain tags and can modify them.

You can use Microsoft Excel, for example, to display tags. You will require the latest version of "MS SOAP Toolkit V2.0" for this purpose. This version is available from Microsoft as a download.

The data access via SOAP is not supported by Windows 7. Use OPC to display the tags in MS Excel. For more information, refer to "Configuring OPC clients (Page 7161)".

Data access to Windows CE HMI-devices

The data access via the Web-service(SOAP) on Windows CE-HMI-devices functions using only the device name and not via the IP-address.

Enter the device name of the HMI-device with the appropriate IP-address in the hosts-file. The hosts-file is given in the directory"%windir%\system32\drivers\etc\", z. B. C:\WINNT \system32\drivers\etc.

The device name, e.g.DEVICEMP377, must be set in the control panel on the HMI device under "System > Device Name". Please change the default device name to ensure that the device name is unique for the network.

Example for the entry in the hosts-file:

192.168.56.198 DEVICEMP377

Replace the IP-address by the device name in the SOAP-client:

objRuntime.mssoapinit http://DEVICEMP377/soap/RuntimeAccess?wsdl

Data access with GetValue, SetValue

Access to a tag in SOAP using GetValue or SetValue functions require a special syntax.

- GetValue: "Sinus_1"
- SetValue: Sinus_1

The tag name must be given in inverted commas for GetValue. The message "Runtime is offline" will otherwise be output when runtime is accessed.

Note

Note that the tag name entry is case-sensitive.

Create VBA-Marko in MS Excel

Introduction

Data access over the network via web service (SOAP) is to be used to permit certain tags of an HMI device to be displayed and reset.

For this purpose, macros are written in Excel, which: 1) obtain the relevant tags on the PC over the network and display them, and 2) transfer reset values back to the HMI device.

The task can be solved using VBA macros "ReadTagValue" and "WriteTagValue," which obtain and display the relevant tags in Excel over an appropriate interface and return them to the HMI device over the network. Note that the tag name entry is case-sensitive.

Requirement

- The SOAP toolkit is installed.
- "Web-Service (SOAP)" is selected in the WinCC-Project under "Services in Runtime".

Procedure

1. Insert the "Control element toolbox" toolbar in your workbook in Microsoft Excel.
2. Create a command button. Label the button "ReadTagValue" and name it "Read value".
3. Double-click this command button.
   The macro editor is displayed. The "Click" event is already preset.
4. Write the "ReadTagValue" macro ("intVarTag_1" designates the actual tag value):

```
'---------------------------------------------------------------
Private Sub ReadTagValue_Click()

Dim objRuntime
Dim intVarTag_1
Dim objWorksheet

Set objWorksheet = Excel.Worksheets("Sheet1")
Set objRuntime = CreateObject("MSSOAP.SoapClient")
objRuntime.mssoapinit "HTTP://servername/soap/RuntimeAccess?wsdl"
objRuntime.ConnectorProperty("AuthUser") = "Administrator"
objRuntime.ConnectorProperty("AuthPassword") = "100"
Var = objWorksheet.Cells(1, 3)
intVarTag_1 = objRuntime.GetValue(Var)
objWorksheet.Cells(1, 1) = intVarTag_1

End Sub
'---------------------------------------------------------------
```

1. Insert a command button. Label the command button "WriteTagValue" and name it "Write value".
2. Double-click this button.
3. Write the "WriteTagValue" macro ("intVarTag_1" designates the return value of the operation):

```
'---------------------------------------------------------------
Private Sub WriteTagValue()

Dim objRuntime
Dim intVarTag_1
Dim objWorksheet

Set objWorksheet = Excel.Worksheets("Sheet1")
Set objRuntime = CreateObject("MSSOAP.SoapClient")
objRuntime.mssoapinit "HTTP://servername/soap/RuntimeAccess?wsdl"
objRuntime.ConnectorProperty("AuthUser") = "Administrator"
objRuntime.ConnectorProperty("AuthPassword") = "100"
Var = objWorksheet.Cells(2,3)
Value = objWorksheet.Cells(2,5)
intVarTag_1 = objRuntime.SetValue(Var,Value)
objWorksheet.Cells(2,8) = intVarTag_1

End Sub
'---------------------------------------------------------------
```

Result

As soon as you call "ReadTagValue_Click" macro by clicking the button"Read-value", the specified intVarTag_1tag is obtained from the HMI device using the specified device address and displayed in the cell (1,1).

As soon as you call Macro "WriteTagValue" by clicking the "Write value" button, the tag name is read from the cell (2,3), and the tag value is transferred from cell (2,5) to the HMI device.

See also

Configuration in WinCC (Page 7085)

12.16 Interfaces

12.16.1 OPC

12.16.1.1 Basics

OPC

Introduction

OPC refers to standardized manufacturer-specific software interfaces for data exchange in automation engineering.

The OPC interfaces provide a standardized environment in which devices and applications from various manufacturers can be linked.

OPC is based on the Windows technology COM (Component Object Model) and DCOM (Distributed Component Object Model).

OPC UA (Unified Architecture) is the technology succeeding OPC. OPC UA is platform-independent and can use different reports as a communication medium.

Principle of operation

In WinCC, you can configure HMI devices as OPC servers or OPC clients. The particular HMI device determines which OPC servers and OPC clients are available.

OPC Specifications

OPC specifies interfaces to gain access to the following objects in WinCC:

- Process values (OPC Data Access 1.0, 2.05a)
- Process values (DataAccess ClientFacet (OPC UA)
- Process values (OPC XML-Data Access 1.01)

You can find additional information about the individual OPC specifications in the Internet at the website of OPC Foundation:.

- www.opcfoundation.org (www.opcfoundation.org)

See also

Supported OPC UA services of the OPC UA client (Page 7170)

Compatibility

Support of the mentioned specifications is checked regularly by the "Compliance Test Tool" (CTT) of the OPC Foundation. Interoperability with OPC products of other manufacturers is ensured through the participation in "OPC Interoperability Workshops".

The test results submitted are published on the website of the OPC Foundation. The results can be called up from there using the search term "OPC Self-Certified Products".

Using OPC in WinCC

Configuration option

HMI devices configured with WinCC feature an OPC interface for data communication between automation devices or automation systems using the OPC communication driver.

You can use an HMI device as an OPC server and/or as an OPC client. As an OPC client, the HMI device can be connected to a maximum of eight OPC servers.

The software ensures up to eight client HTTP connections for an OPC XML DA server on a Multi Panel. For the exchange of data via an XML connection, some OPC XML DA clients create several HTTP connections to the OPC XML DA server.

HMI device	Data exchange over	Operating system	OPC server	OPC client
PC, Panel PC (with WinCC Runtime Advanced)	DCOM or OPC UA binary (TCP/IP)	Windows XP / 7	OPC DA server OPC UA server (DA only)	OPC DA client OPC UA client*
HMI devices with version V11.0 and V12.0: • MP 277 • MP 377 Mobile Panel 277 • Comfort Panels	SOAP	Windows CE	OPC XML DA server	-
Comfort Panel with version V13.0	SOAP or OPC UA binary (TCP/IP)	Windows CE	OPC UA server (DA only)	OPC UA client
*: The OPC UA client exchanges data via TCP/IP.				

A communication via DCOM is only possible between HMI devices of the "PC" or the "Panel PC" type. In order for a control room PC to display the process values of Windows CE HMI devices in the plant, for example, install "OPC-XML-Gateway" on the control room PC using the WinCC Runtime Advanced Setup. The "OPC-XML-Gateway" supports the communication within the SIMATIC HMI devices family between an OPC DA client and an OPC XML DA server.

The area pointer is supported by any OPC connection.

OPC server accessibility

The following table lists the available OPC servers and how you can access them:

OPC server	OPC server name
OPC DA server	OPC.SimaticHMI.CoRtHmiRTm (ProgID)
OPC XML DA server	http://<xxx>/soap/OpcXml (URL) [1] <xxx>: IP address or DNS name of the HMI device
OPC DA server	OPC.Siemens.XML (ProgID) for accessing the OPC XML DA server via the OPC XML gateway from an OPC DA client
OPC UA server	Server URL: opc.tcp://[HostName]:4870
[1]: For the access of an external OPC DA XML client.	

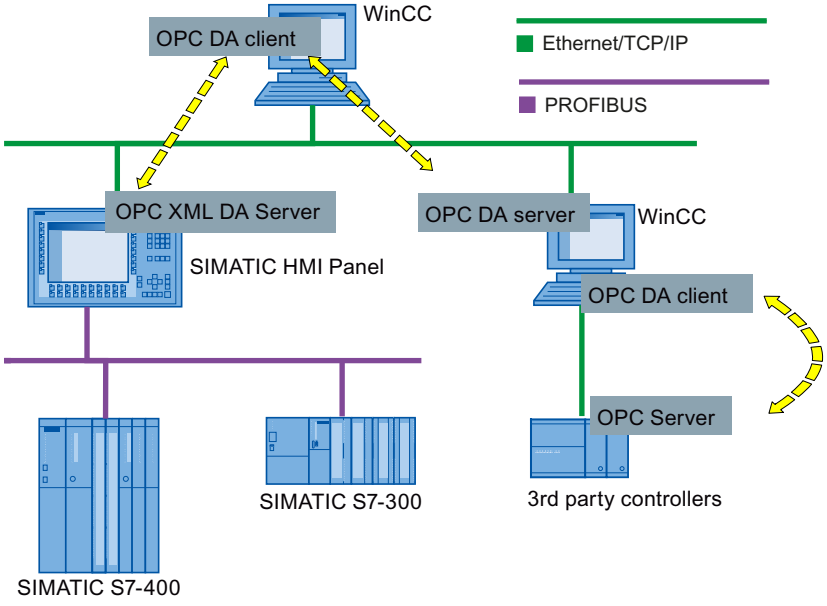
HMI device as OPC client

To use an HMI device as an OPC client, create an "OPC" connection in the WinCC project. The OPC client accesses the tags of the OPC server over this connection. You need a separate connection for each OPC server.

The following exception applies here:

If you want to connect your OPC DA client to multiple panels, you only need one OPC connection to the OPC XML gateway in your project. The multiplexing of this connection to multiple OPC XML connections to individual panels takes place in the OPC XML gateway. You configure the connections to the panels using he OPC XML Manager.

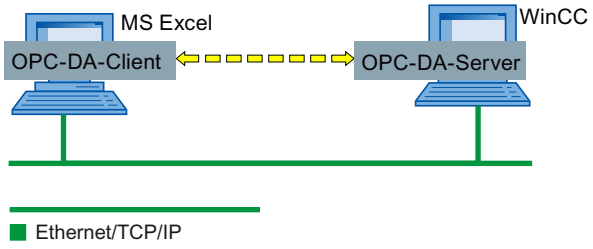
The following figure shows the use of an HMI device as a central operator control and monitoring device:



HMI device as OPC server

An HMI device as OPC server makes the data available to other applications. The applications can run on the same HMI device or on HMI devices in the connected network environment.

The following schematic diagram shows the use of MS Excel as an OPC client that displays process values of the OPC server:



See also

Configuring an HMI device as OPC DA server (Page 7158)

Creating a connection to a WinCC OPC DA server (Page 7161)

OPC XML Gateway (Page 7164)

12.16.1.2 Configuring an OPC server

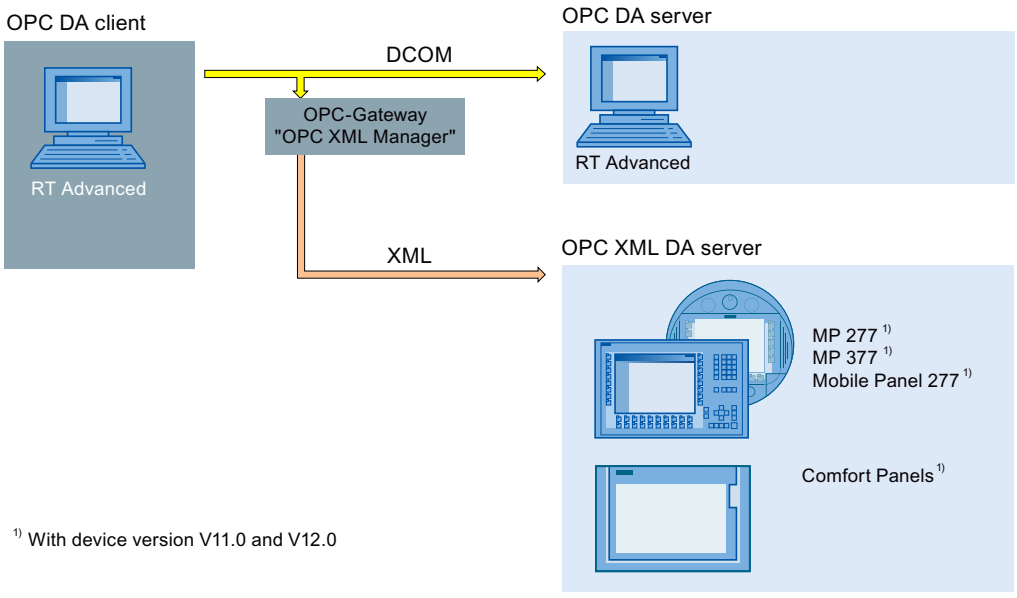
Configuring an HMI device as OPC DA server

Introduction

Which OPC server is used depends on the HMI device:

- HMI devices with RT Advanced as OPC DA server
- HMI devices with version V11.0 and V12.0 as OPC XML DA server
- HMI devices with version V13.0 as OPC XML DA server

The following figure shows the two methods of accessing an OPC server:



Procedure

To configure an HMI device as an OPC server, proceed as follows:

1. Open the "Runtime settings" of the HMI device in the project tree.
2. Select the "Operate as OPC server" option in the "Runtime settings" under "Services > Write/read tags".
3. Save the project.
4. Download the project to the HMI device.
5. Start runtime on the HMI device.

Result

The OPC server is accessible. If an OPC client connects to the OPC server, the OPC server on the HMI device is started.

See also

Loading a project (Page 6938)

Starting Runtime Advanced and Panel Runtime (Page 6951)

Using OPC in WinCC (Page 7154)

Creating a connection to a WinCC OPC DA server (Page 7161)

Configuring an HMI device as an OPC UA server

Introduction

The following HMI devices can be used as OPC UA servers:

- Comfort Panels
- RT Advanced

Procedure

1. Open the "Runtime settings" of the HMI device in the project tree.
2. Select the "Operate as OPC server" option under "Service" in the "Runtime settings".
3. Configure the server settings at "OPC settings > OPC Unified Architecture Server configuration":
   - Change the "Port number", if required.
   - Activate at least one "SecurityPolicy" and the respective "Message security mode".

   Note
   Unsecured communication between client and server possible
   Use the "none" setting only for test and diagnostics purposes.
4. Save the project.
5. Download the project to the HMI device.
6. Start runtime on the HMI device.

Result

The OPC server is accessible.

If an OPC client connects to the OPC server, the OPC server on the HMI device is started.

Configuring DCOM user permissions in Windows

Introduction

The OPC DA client and OPC DA server are DCOM applications, whose security settings must be set in compliance with the DCOM security mechanisms:

- The OPC client needs launch/activation rights and access rights for the OPC DA server.
- The OPC DA server only needs access rights for the OPC DA client

The following must be known on the PCs of the OPC DA server and the OPC DA client respectively:

- The user account for which the OPC DA client is executed

Requirement

You have administrator rights.

Procedure

The procedure for configuring DCOM user rights is described in the document (http://www.opcfoundation.org/DownloadFile.aspx?CM=3&RI=326&CN=KEY&CI=282&CU=14) of the OPC-Foundation.

For additional information on granting user rights, refer to the documentation for Windows XP / 7.

12.16.1.3 Configuring an OPC client

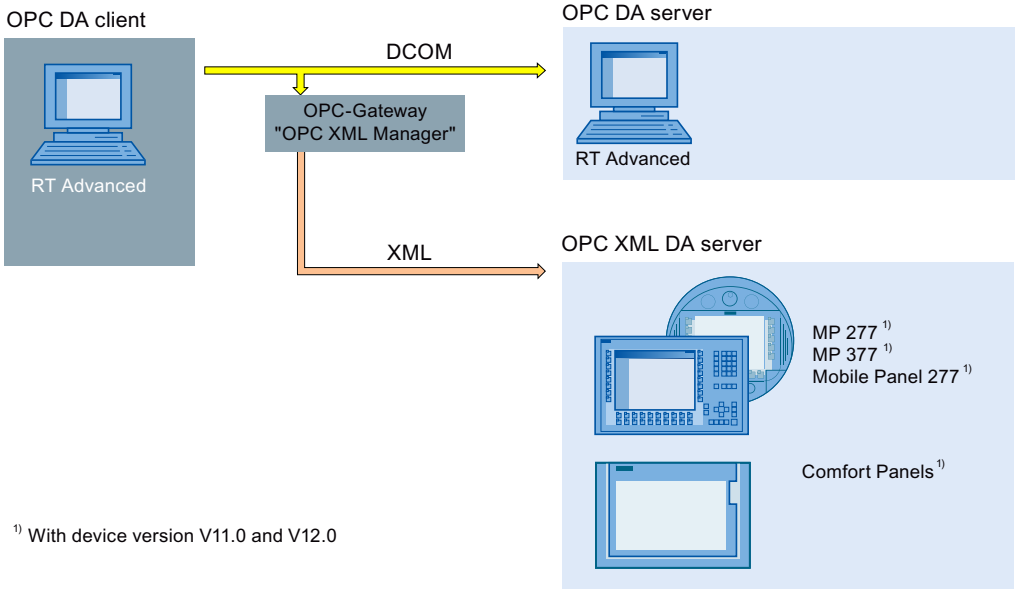
Creating a connection to a WinCC OPC DA server

Introduction

For an HMI device to access the OPC server data of another HMI device, you need to create a connection to this OPC server in the WinCC project.

Depending on the target HMI device used, either an OPC DA server or an OPC XML DA server is employed.

The following figure shows the two methods for accessing the OPC servers of the HMI devices:



1) With device version V11.0 and V12.0

If a connection to one or more OPC XML DA servers is created using an OPC XML gateway, also enter the OPC XML DA servers in the "OPC XML Manager".

Requirement

- An HMI device is configured as an OPC server.
- The project on the HMI device is in runtime.
- For connections to a OPC XML DA server: "OPC-XML-Gateway" is installed on the configuration PC and the HMI device with the OPC client.

Procedure

To create a connection to an OPC server of an HMI device, follow these steps:

1. Open the "Connections" editor on the configuration PC in the WinCC project of the OPC client.
2. Create a new connection and enter a meaningful name.
3. Select the entry "OPC" as the "Communication driver".
4. Enter the communication partner under "Parameters > OPC server" in the work area:
   - If the connection communicates directly with the OPC server or with the OPC server on a PC-based HMI device, select the "OPC.SimaticHMI.CoRtHmiRTm" item from the list.
   - If the connection communicates directly with the OPC server of a Windows CE HMI device, select "OPC.Siemens.XML" from the list.
   - If the OPC server is installed on a remote computer, enter the computer's IP address or name under "Remote computer name".

Result

The OPC connection is configured. To access the data on the WinCC OPC DA server, create tags.

See also

Using OPC in WinCC (Page 7154)

Configuring an HMI device as OPC DA server (Page 7156)

OPC XML Gateway (Page 7164)

Configuring an OPC XML Manager (Page 7165)

Creating a connection to an OPC UA server

Introduction

The OPC UA client can access process data in the hierarchical name space of an OPC UA server.

For the OPC UA client to access the process values of an OPC UA server, the OPC UA server and the OPC UA client authorize each other by exchanging certificates. In addition, you can encode the data transfer.

The OPC UA client usually classifies each certificate of an OPC UA server as a "trustworthy". How an OPC UA server responds to a connection request of the OPC UA client depends on the configuration of the OPC UA server.

In order to establish communication to an OPC UA server, inform yourself from the OPC UA server operator about the following:

- URL of the OPC UA server
- Security settings
- Required certificates

Requirement

URL and security settings of the OPC UA server are known.

Procedure

To create a connection to an OPC UA server, proceed as follows:

1. Open the "Connections" editor on the HMI device.
2. Create a new connection and enter a meaningful name.
3. Select the entry "OPC UA" as the "Communication driver".
4. In the work area, under "Parameters", configure the "OPC server":
   - Specify the "Discovery URL" of the OPC UA server or select the OPC UA server from the list.
   - Select the "Security policy"
   - Select the "Message security mode"

Result

The OPC UA connection is configured. You create tags to access data from the OPC UA server.

See also

Supported OPC UA services of the OPC UA client (Page 7170)

Accessing process values of an OPC server

Requirement

- The OPC server to be addressed is ready-to-operate and in the "running" status
- A connection to the OPC-Server is created

Procedure

To access process values of an OPC-Server via the OPC connection, follow these steps:

1. On the configuration PC in the project navigation, open the "HMI tags" editor under the HMI device that you use as an OPC client.
2. Create a tag with the same data type as the tag on the OPC server.
3. Select the OPC connection for "Connection".
4. Enter the "Address", or select the desired tag on the OPC server via the object list.

Result

If you launch Runtime on the HMI device, the process value from the OPC server will be written to the tag on the HMI device via the OPC connection.

See also

Permitted data types (OPC) (Page 7168)

Access to tags with OPC (Page 7169)

OPC XML Gateway

Usage

The OPC DA client uses the "OPC XML Gateway" so that the OPC DA client can communicate with the OPC XML DA server. The "OPC XML Gateway" compiles the data in the respective "language" of the corresponding standard. The "OPC XML Gateway" communicates exclusively with the OPC XML DA server which runs on an SIMATIC HMI device.

Installation

To install "OPC XML Gateway", activate the "OPC XML Gateway" entry during installation of WinCC runtime advanced in the component selection. To install "OPC XML Gateway" afterwards, re-execute the setup of WinCC Runtime Advanced.

Proxy setting for the OPC XML Gateway

The configuration settings for the OPC XML Gateway can be found in the "SOPCSRVR.ini" file in the section "[Configuration]". The OPC XML Gateway is configured by default on the PC as the HMI device, so that a proxy server configured in the Internet settings of Internet Explorer is ignored:

NOPROXY=1

If this entry is set to "0", the OPC XML Gateway uses a configured proxy server for HTTP connections.

You can find the "SOPCSRVR.ini" file in the folder "C:\Program Files\\Siemens\Automation \WinCC RT Advanced".

Note

When data is requested from an HMI device that cannot be accessed over the configured proxy server, the OPC XML Gateway uses a direct connection after a Timeout. The direct connection is set up again on every request for data and slows down OPC communication significantly.

See also

Configuring an OPC XML Manager (Page 7165)

Using OPC in WinCC (Page 7154)

Configuring an OPC XML Manager

Introduction

The OPC XML DA server, to which the OPC DA client has access, is administered in the "OPC XML Manager". The OPC XML Manager can be found in the Windows Start menu under "SIMATIC > OPC-XML-Gateway > OPC XML Manager".

To enter an OPC server, you need the following information:

- Server prefix
  Any string that is used in the name of the OPC tag. Use an abbreviation of the server, for example. You can find the characters permitted for tag names in the reference.
- Name or IP address of the OPC XML DA server

Requirement

"OPC XML Manager" is open.

Procedure

To configure the OPC XML Manager, proceed as follows:

1. To enter a new OPC XML DA server, click on the button "Add".
   The "Add/Edit Webservice" dialog is opened.
2. Enter the server prefix and the name or IP address of the OPC XML DA server.
3. Close the two dialogs with "OK".

Result

The OPC XML DA server is entered. The OPC DA client can now access the data of the OPC XML DA server via the "OPC XML Gateway".

Editing or removing an OPC server

To edit or delete a configured OPC XML DA server, select the required OPC XML DA server. Then click either "Edit" or "Remove" .

See also

OPC XML Gateway (Page 7162)

12.16.1.4 Security concept of OPC UA

Introduction

The WinCC OPC UA uses the TCP/IP protocol for data exchange. Instance certificates are exchanged for authorization between WinCC OPC UA server and OPC UA client. In addition, you can encode the data transfer.

Security concept

The WinCC OPC UA server and each OPC UA client authorize themselves mutually by exchanging certificates.

By default, the WinCC OPC UA server creates during installation a self signed instance certificate. The server sends this instance certificate to the client. You can alternatively replace this instance certificate with a project-specific instance certificate.

Note

Private key and own certificates

If you have an own certification center, you can create your own certificates and make them available for all communication partners. In this case, delete the instance certificate created by WinCC OPC UA server.

The instance certificate of the client that is allowed to communicate with the server must be stored on both the server and the client in the appropriate directory.

Storage locations of the instance certificates

The instance certificate is stored on the OPC UA server in the following path:

- On PCs: "\ProgramData\Siemens\CoRtHmiRTm\MiniWeb1X.X.X\SystemRoot\SSL"
- On Panels: "\flash\simatic\SystemRoot\SSL"

The instance certificate is stored on the OPC UA client in the following path:

- On PCs: "ProgramData\Siemens\CoRtHmiRTm\OPC\PKI\CA\"
- On Panels: "flash\Simatic\SystemRoot\OPC\PKI\CA\"

If you configure two OPC connections in your project, create the following directories under [...]\OPC\PKI\CA\:

- "Default"
- "!Conn_1"
- "!Conn_2"

Each directory contains the subdirectories "certs", "rejected" and "private". Self-signed instance certificates and private keys are stored under "Default\certs" or "Default\private". The OPC-UA connections only use these certificates.

The client always stores the sent server certificates first in the "rejected" directory and does not allow a connection to the server. To allow connection to the server at the client end, you need to move the certificate in question from the "rejected" directory into the "certs" directory.

The server also initially stores the client certificate in the "rejected" directory under [...] \SystemRoot\SSL. To allow connection to the server at the server end, you need to move the certificate in question from the "rejected" directory into the "certs" directory under [...] \SystemRoot\SSL.

Configuring the OPC UA client for the security mode

If the OPC UA server is running and the OPC UA client was started with an encrypted connection, set up the communication between the OPC UA server and the OPC UA client in security mode.

1. Move the instance certificate sent from the server from the "rejected" directory to the "certs" directory on the OPC UA client.
2. Move the instance certificate sent from the client from the "rejected" directory to the "certs" directory on the OPC UA server.

Once the client and the server have authorized each other, they communicate in security mode.

Note

If you want to use certificates of third-party vendors for the OPC UA client, stop Runtime and copy the third-party certificates and private keys into the directories "!Conn1\certs" and "! Conn1\private". Then copy the certificate from the "certs" directory into the "SSL" directory on the server. Then start Runtime again.

Security settings

The following table lists the security settings supported by the WinCC OPC UA server:

SecurityPolicy	Message Security Mode		
None[1]	None		
Basic128Rsa15 [2]	None [3]	Sign [4]	SignAndEncrypt [5]
1: The certificate exchange is switched off. Every OPC UA client can log on to the WinCC OPC UA server.			
2: Certificate exchange with depth of encryption of 128 bit.			
3: The data packages are exchanged after certificate check unsecured between client and server.			
4: The data packages are signed with the certificates, but not encoded			
5: The data packages are signed with the certificates and encoded			

12.16.1.5 Reference

Permitted data types (OPC)

Permitted data types

The following table lists the data types supported by the WinCC OPC servers:

OPC data type	WinCC data type
VT_BOOL	BOOL
VT_I1	CHAR
VT_UI1	BYTE
VT_I2	SHORT
VT_UI2	WORD
VT_UI4	DWORD
VT_I4	LONG
VT_R4	FLOAT
VT_R8	DOUBLE
VT_DATE	DATE
VT_BSTR	STRING

The following table lists the ranges of values of the OPC data types:

OPC data type	Range of values
VT_BOOL	0 or -1
VT_I1	-128 to 127
VT_UI1	0 to 255
VT_I2	-32768 to 32767
VT_UI2	0 to 65535
VT_I4	-2147483648 to 2147483647
VT_UI4	0 to 4294967295
VT_R4	3.402823466 e-38 through 3.402823466 e+38

OPC data type	Range of values
VT_R8	1.7976931486231e-308 to 1.7976931486231e+308
VT_Date	1 January 100 to 31 December 9999

Special features in communication with the OPC-DA server

The array tag in the OPC-DA server belonging to the area pointer must be of the data type SHORT (VT_I2).

Special features in communication with the OPC-XML server

Array tags are not supported by OPC-XML servers.

Access to tags with OPC

Introduction

When accessing tags with OPC, note the following when configuring tags:

- Permitted characters in tag names
- Permitted cycle times (OPC XML)
- Special features of the data type "STRING" (OPC XML)
- Special features of the data type "Date/Time" (OPC XML)

The OPC XML DA server is tested and enabled for eight connections, each with 2 groups and consisting of 35 tags.

Permitted characters in tag names

Only use the following characters in tag names:

- Letters from "a" to "z" (no umlauts)
- Numbers from "0" to "9"
- Special characters: "-" and "_"

Permitted cycle times (OPC XML)

OPC XML connections are designed for the exchange of small volumes of data:

- For this reason use cycle times that are greater than one second
- In general, only request a small number of tags; a maximum of 30 per screen

Special features of the data type "STRING" (OPC XML)

Only valid ASCII values from 0x20hex to 0x7Fhex are supported in the data type "STRING".

Special features of the data type "Date/Time" (OPC XML)

Values of the data type "Date/Time" are always expected as UTC (Universal Time Coordinated) by the OPC-Gateway. If a tag of the type "Date/Time" is read by the OPC client, the returned value is a time in UTC. If a value is written to the tag, the value is treated as UTC. The time including the time zone and daylight saving time are shown as "local time" on an HMI device.

Example:

The time zone GMT+1 and daylight saving time is set on the HMI device.

OPC DA client (UTC time): 01.01.2005 16:00

Display on the HMI device (OPC server): 01.01.2005 18:00

Supported OPC UA services of the OPC UA client

OPC UA Services support

The OPC UA client supports the following OPC UA services:

- SecurityPolicy - Basic128Rsa15
- SecurityPolicy - Basic256
- SecurityPolicy - None
- DataAccess ClientFacet

You can additional information about OPC UA services in the "OPC UA Part 3 - Address Space Model 1.01 Specification" document under "§5.6".

Explanation of the security settings

The following table lists the security settings supported by the OPC UA client:

SecurityPolicy	Message Security Mode		
None[1]	None		
Basic128Rsa15[2]	None[4]	Sign[5]	SignAndEncrypt[6]
Basic256[3]	None	Sign	SignAndEncrypt
1: The certificate exchange is disabled. Every OPC UA client can log on to the WinCC OPC UA server. This setting can be disabled on each OPC UA server.			
2: Certificate exchange with depth of encryption of 128 bit.			
3: Certificate exchange with depth of encryption of 256 bit.			
4: The data packages are exchanged after certificate check unsecured between client and server.			
5: The data packages are signed with the certificates, but not encoded			
6: The data packages are signed with the certificates and encoded			

Note

Unsecured communication between client and server possible

Use the "none" setting only for test and diagnostics purposes.

For a secure communication between client and server, use in operating mode at least the following settings:

- SecurityPolicy: Basic128Rsa15
- Message Security Mode: Sign

See also

Creating a connection to an OPC UA server (Page 7160)

12.17 Migrating to WinCC in the TIA Portal

12.17.1 Overview of the migration to WinCC in the TIA Portal

Overview of the section "Migration to WinCC in the TIA Portal"

SIMATIC WinCC offers a number of functional changes in the TIA Portal. Some functions differ from the functions that you know from familiar environments such as WinCC V7 or WinCC flexible.

This document provides an overview of the special functions and procedures in SIMATIC WinCC in the TIA Portal.

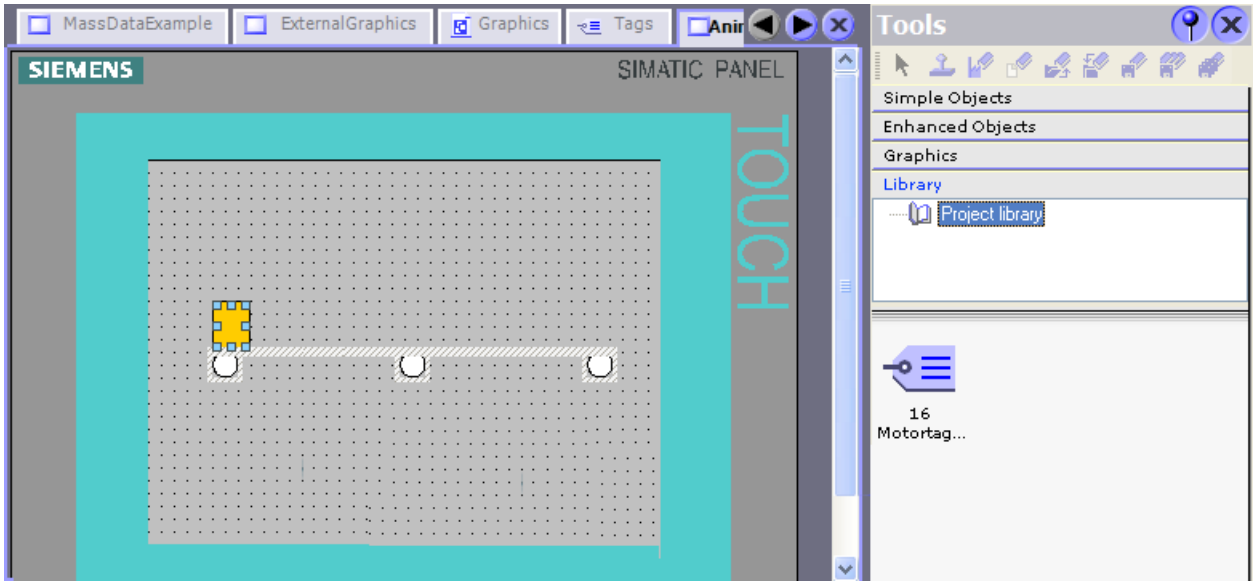
These functions and procedures are fundamentally different from the WinCC V7 and WinCC flexible version, or have a different name.

12.17.2 WinCC flexible

12.17.2.1 Libraries

Libraries in WinCC flexible

Libraries are a collection of pre-configured screen objects. They expand the number of available screen objects and increase engineering efficiency, because library objects are always available for reuse; there is no need to reconfigure them.



WinCC flexible enables you to create two library types:

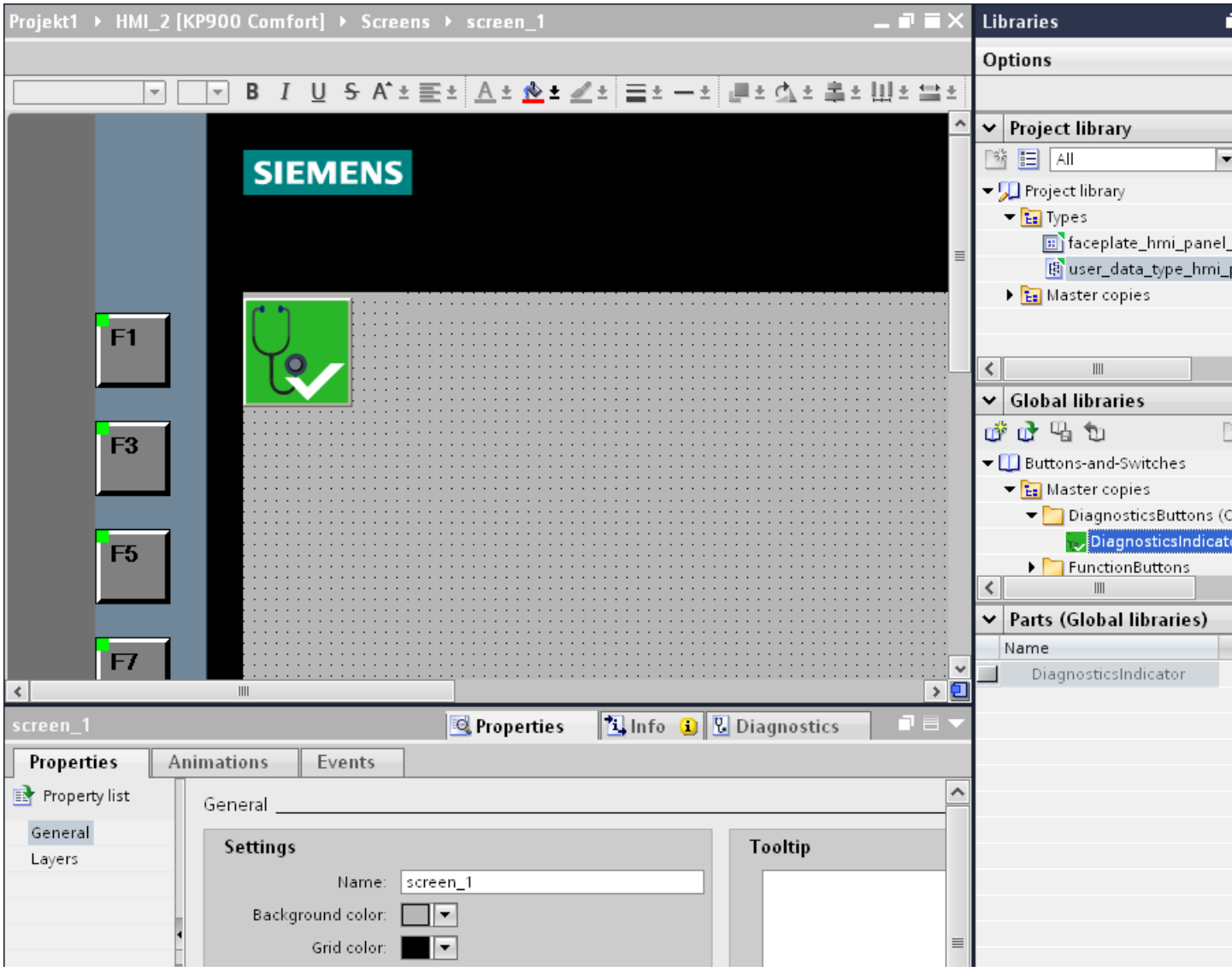
- Project library
- Global library

A library can contain all the WinCC flexible objects, such as screens, tags, graphic objects, or alarms.

How do I configure libraries with WinCC in the TIA Portal?

In WinCC, you also configure the "Project library" and the "Global library".

You can no longer store any system functions in libraries, as was the case in WinCC flexible.



Both the "Project library" and the "Global library" contain the two folders, "Copy templates" and "Types". You can create or use the library objects as a copy template, or as a type.

- Copy templates
  Use copy templates to create independent copies of the library object.
- Types
  Create instances of objects of the "Types" folder and use the instances in your project. The instances are bound to their respective type. Changes to an instance also change all other instances. Types are marked by a green triangle in the "Libraries" task card.
- Managing the library objects
  You can only copy and move library objects within the same library.

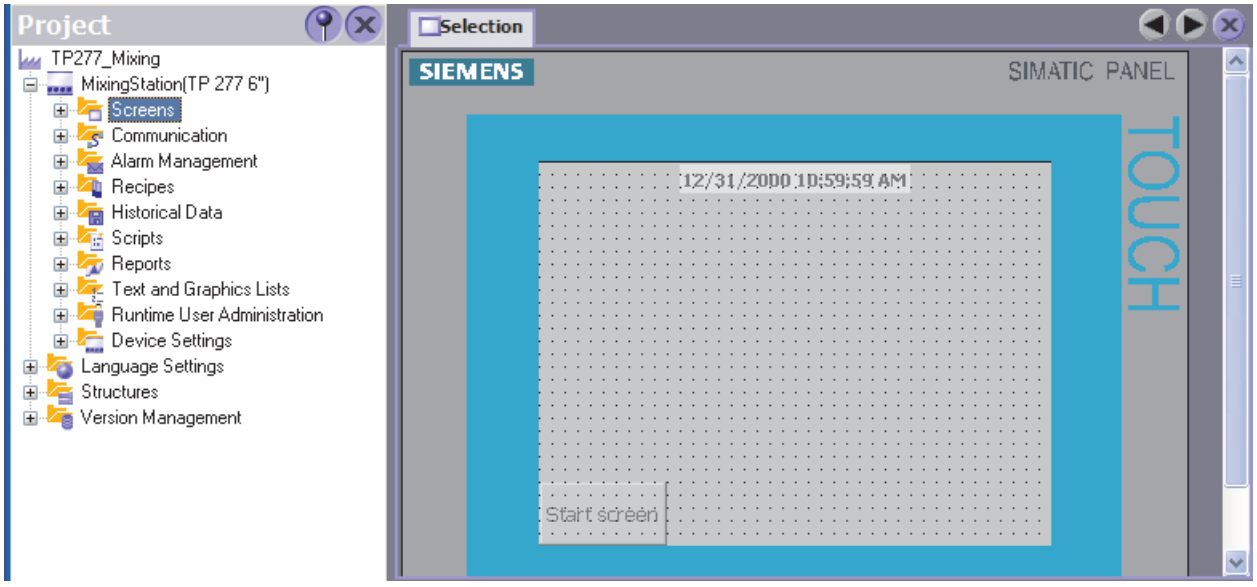
For more detailed information, see:

Auto-Hotspot

12.17.2.2 Screens and templates

Screens and templates in WinCC flexible

In WinCC flexible, you create screens that an operator can use to control and monitor machines and plants. When you create your screens, the object templates provided support you in visualizing your plant, displaying processes and defining process values.



The project has a template for every HMI device. You can centrally configure the function keys and objects for your project in these templates.

Every screen based on this template will contain the function keys and objects that you configured in the template. Changes to an object or of a function key assignment in the template are applied to the object in all the screens, which are based on this template.

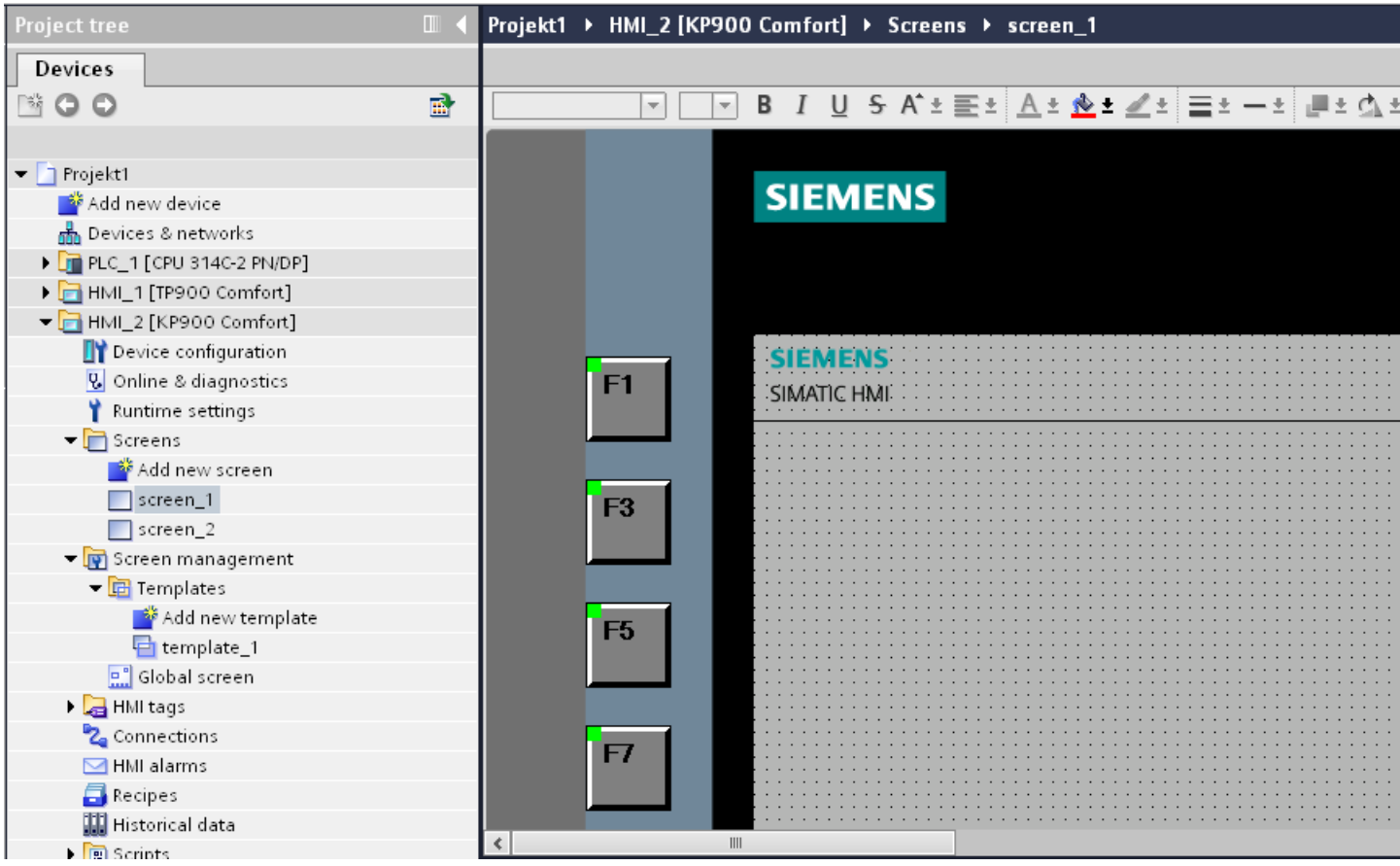
How do I configure screens and templates with WinCC in the TIA Portal?

In WinCC, you also configure "Templates" and a "Global screen" along with the "Screens".

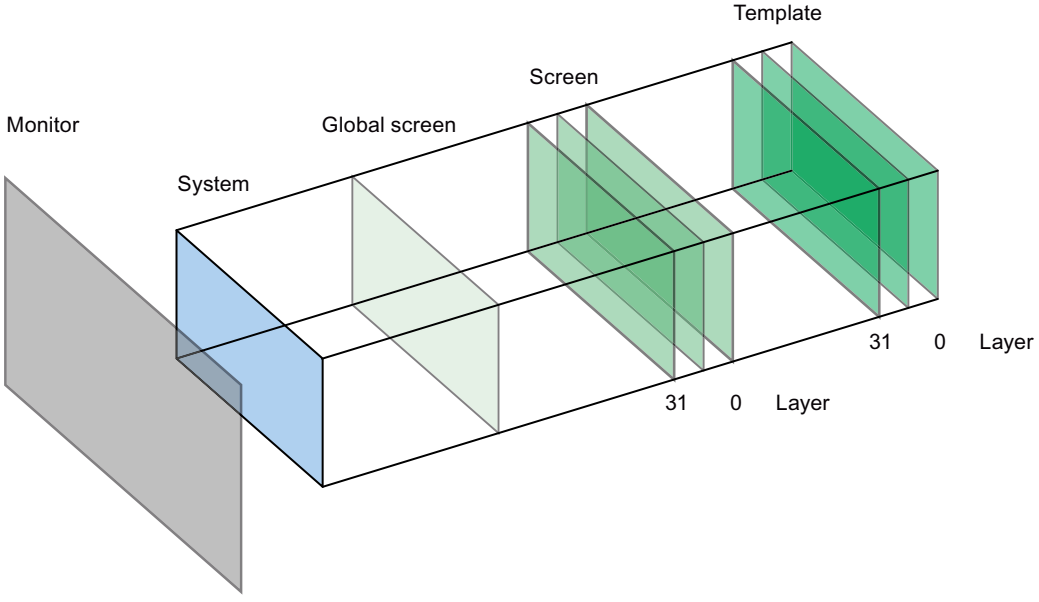
You determine functions and objects in the template which then apply to all screens based on this template. You can create multiple templates in WinCC.

In the "Global screen", define the elements which are independent of the template used for all screens of an HMI device. The "Alarm window" and "Alarm indicator" objects are available for use as global objects. For HMI devices with function keys, configure the function keys in the "Global Screen" editor.

You can also configure a "System Diagnostics Window" in the global screen of Comfort Panels.



Excluding the controls, the screens are displayed in runtime in the following order:



For more detailed information, see:

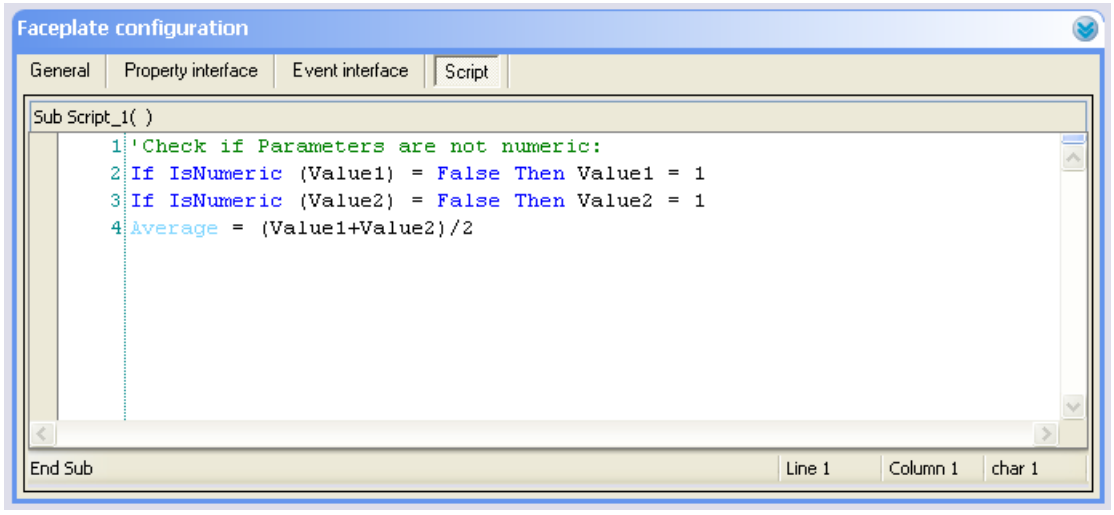
Auto-Hotspot

12.17.2.3 Scripts in faceplates

Scripts in faceplates in WinCC flexible

You configure a script in the "Scripts" tab of the "Faceplate configuration" dialog. This script is only available within the faceplate.

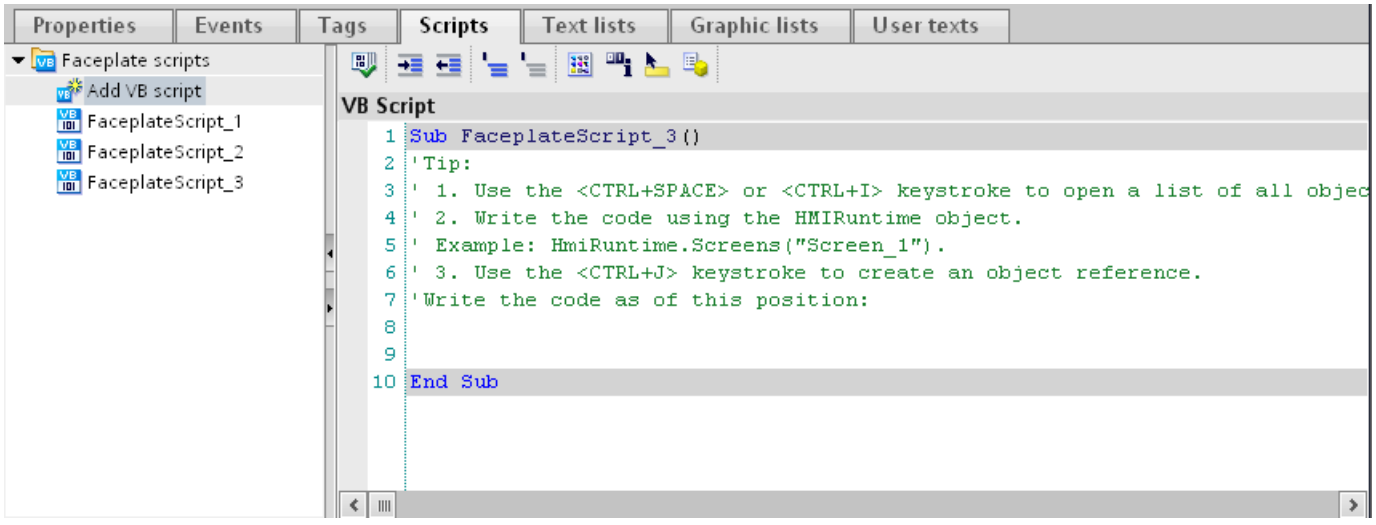
Interconnect the script directly to the events of the objects contained in the faceplate, for example, with the "Click" event of a button. If you use the faceplate in a screen, a faceplate instance is generated.



How do I configure scripts in faceplates with WinCC in the TIA Portal?

In the configuration area of the "Faceplates" editor, you create scripts that you only use within a faceplate type.

In contrast to WinCC flexible, WinCC allows you to configure several scripts for a faceplate.



For more detailed information, see:

Auto-Hotspot

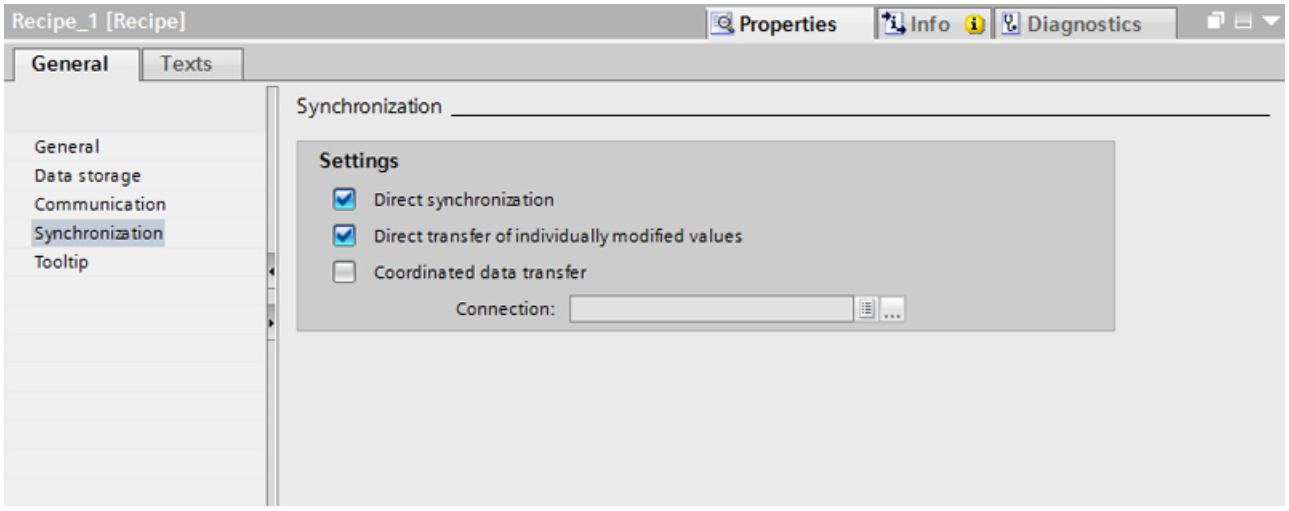
12.17.2.4 Synchronization of recipes

How do I configure a synchronization of recipes with WinCC in the TIA Portal?

You configure the synchronization of recipes in the "Recipes" editor in WinCC.

A few terms have changed in WinCC as compared to WinCC flexible.

Synchronization for Panels and RT Advanced



- To synchronize recipe tags that are configured in I/O fields with the recipe view in Runtime, activate "Synchronize recipe view and recipe tags".
- Deactivate "Direct transfer of individually modified values" to specify that the recipe tags are automatically transferred to the PLC when you edit the I/O fields (teach-in mode).
- Activate "Coordinated data transfer" to monitor the transfer of recipe data in Runtime using area pointers.

For more detailed information, see:

Synchronization of recipe data records with the PLC (Page 4435)

12.17.2.5 Special considerations for converting

Objects with object references in the project library

Two copying methods can be used in WinCC flexible.

- With "simple copy", a WinCC flexible screen including an IO field is copied, for example. Only the object name of a tag configured on the IO field is copied, as this is a reference.
- With "copy", a screen, an IO field contained there and a tag configured on the IO field together with its properties are copied.

These two methods can also be used for storing an object in a library. Project libraries and the objects contained there are migrated during migration and can be used in WinCC.

In WinCC, however, only one copying method is available. With regard to tags, it functions like "simple copy" in WinCC flexible. With regard to graphics, graphics lists and text lists, it functions like "copy" in WinCC flexible.

If you stored objects with references to tags in a library in WinCC flexible, you must reconfigure the referenced objects when using them in WinCC.

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Errors	Errors
System	System
Warnings	Warnings

The display names of the alarm classes can be changed as necessary after migration.

Objects with object references in the project library

Two copying methods can be used in WinCC flexible.

- With "simple copy", a WinCC flexible screen including an IO field is copied, for example. Only the object name of a tag configured on the IO field is copied, as this is a reference.
- With "copy", a screen, an IO field contained there and a tag configured on the IO field together with its properties are copied.

These two methods can also be used for storing an object in a library. Project libraries and the objects contained there are migrated during migration and can be used in WinCC.

In WinCC, however, only one copying method is available. It functions like "simple copy" in WinCC flexible.

If you have stored objects with references to other objects in a library in WinCC flexible, you must reconfigure the referenced objects when using them in WinCC.

Using technology functions 13

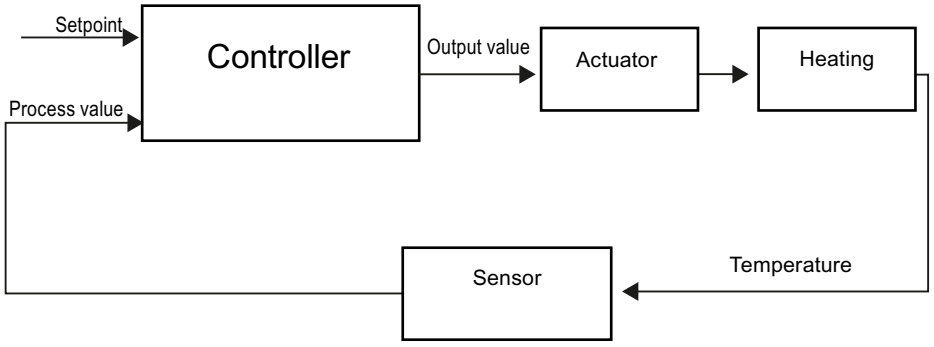
13.1 PID control

13.1.1 Principles for control

13.1.1.1 Controlled system and actuators

Controlled system

Room temperature control by means of a heating system is a simple example of a controlled system. A sensor measures the room temperature and transfers the value to a controller. The controller compares the current room temperature with a setpoint and calculates an output value (manipulated variable) for heating control.



A properly set PID controller reaches this setpoint as quickly as possible and then holds it a constant value. After a change in the output value, the process value often changes only with a time delay. The controller has to compensate for this response.

Actuators

The actuator is an element of the controlled system and is influenced by the controller. Its function modifies mass and energy flows.

The table below provides an overview of actuator applications.

Application	Actuator
Liquid and gaseous mass flow	Valve, shutter, gate valve
Solid mass flow, e.g., bulk material	Articulated baffle, conveyor, vibrator channel
Flow of electrical power	Switching contact, contactor, relay, thyristor
	Variable resistor, variable transformer, transistor

Actuators are distinguished as follows:

- Proportional actuators with constant actuating signal
  These elements set degrees of opening, angular positions or positions in proportion to the output value. The output value has an analog effect on the process within the control range. Actuators in this group include spring-loaded pneumatic drives, as well as motorized drives with position feedback for which a position control system is formed.
  An continuous controller, such as PID_Compact, generates the output value.
- Proportional actuators with pulse-width modulated signal
  These actuators are used to generate the output of pulses with a length proportional to the output value within the sampling time intervals. The actuator - e.g. a heating resistor or cooling apparatus - is switched on in isochronous mode for durations that differ depending on the output value.
  The actuating signal can assume unipolar "On" or "Off" states, or represent bipolar states such as "open/close", "forward/backward", "accelerate/brake".
  The output value is generated by a two-step controller such as PID_Compact with pulse-width modulation.
- Actuators with integral action and three-step actuating signal
  Actuators are frequently operated by motors with an on period that is proportional to the actuator travel of the choke element. This includes elements such as valves, shutters, and gate valves. In spite of their different design, all of these actuators follow the effect of an integral action at the input of the controlled system.
  A step controller, such as PID_3Step. generates the output value.

13.1.1.2 Controlled systems

The properties of a controlled system can hardly be influenced as these are determined by the technical requirements of the process and machinery. Acceptable control results can only be achieved by selecting a suitable controller type for the specific controlled system and adapting the controller to the time response of the controlled system. Therefore, it is is indispensable for the configuration of the proportional, integral and derivative actions of the controller to have precise knowledge of the type and parameters of the controlled system.

Controlled system types

Controlled systems are classified based on their time response to step changes of the output value.

We distinguish between the following controlled systems:

- Self-regulating controlled systems
  - Proportional-action controlled systems
  - PT1 controlled systems
  - PT2 controlled systems
- Non-self-regulating controlled systems
- Controlled systems with and without dead time

Self-regulating controlled systems

Proportional-action controlled systems

In proportional-action controlled systems, the process value follows the output value almost immediately. The ratio between the process value and output value is defined by the proportional Gain of the controlled system.

Examples:

- Gate valve in a piping system
- Voltage dividers
- Step-down function in hydraulic systems

PT1 controlled systems

In a PT1 controlled system, the process value initially changes in proportion to the change of the output value. The rate of change of the process value is reduced as a function of the time until the end value is reached, i.e., it is delayed.

Examples:

- Spring damping system
- Charge of RC elements
- Water container that is heated with steam.

The time constants are often identical for heating and cooling processes, or for charging and discharge characteristics. With different time constants, controlling is clearly more complex.

PT2 controlled systems

In a PT2 controlled system, the process value does not immediately follow a step change of the output value, i.e., it increases in proportion to the positive rate of rise and then approaches the setpoint at a decreasing rate of rise. The controlled system shows a proportional response characteristic with second order delay element.

Examples:

- Pressure control
- Flow rate control
- Temperature control

Non-self-regulating controlled systems

Non-self-regulating controlled systems have an integral response. The process value approaches an infinite maximum value.

Example:

- Liquid flow into a container

Controlled systems with dead time

A dead time always represents the runtime or transport time that has to expire before a change to the system input can be measured at the system output.

In controlled systems with dead time, the process value change is delayed by the amount of the dead time.

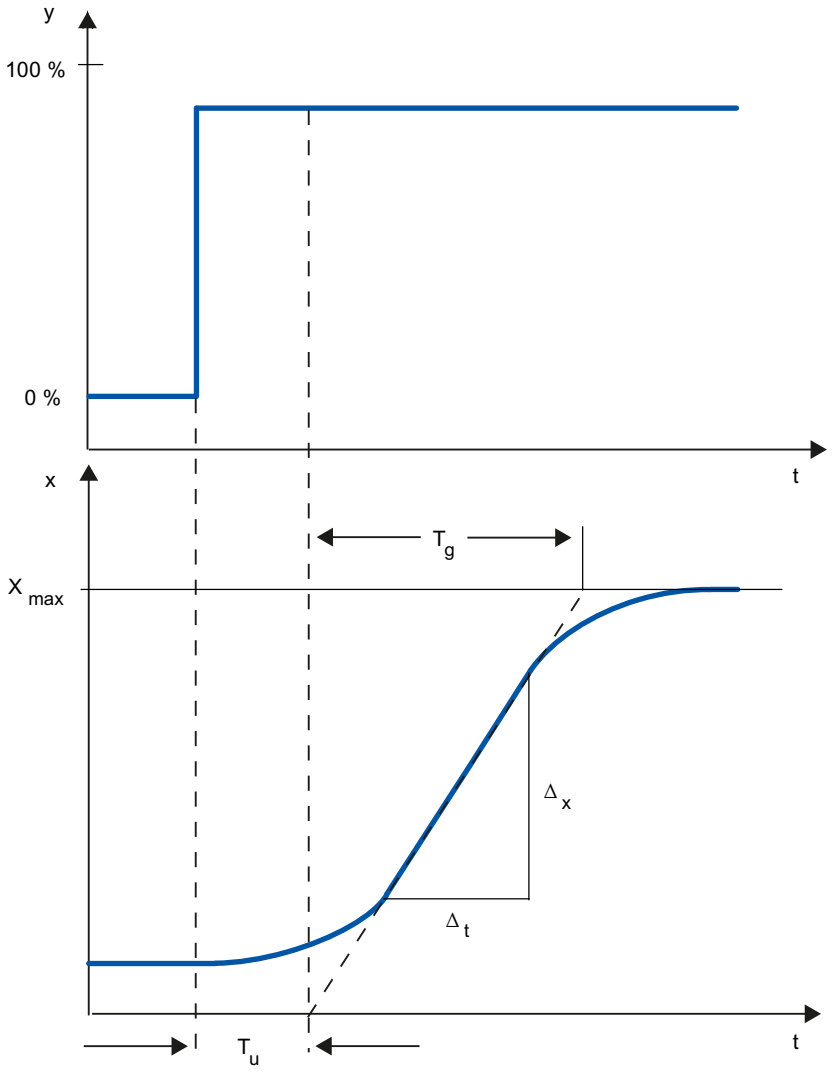
Example:

Conveyor

13.1.1.3 Characteristic values of the control section

Determining the time response from the step response

Time response of the controlled system can be determined based on the time characteristic of process value x following a step change of output value y. Most controlled systems are self-regulating controlled systems.



The time response can be determined by approximation using the variables Delay time T_u, Recovery time T_g and Maximum value X_{max}. The variables are determined by applying tangents

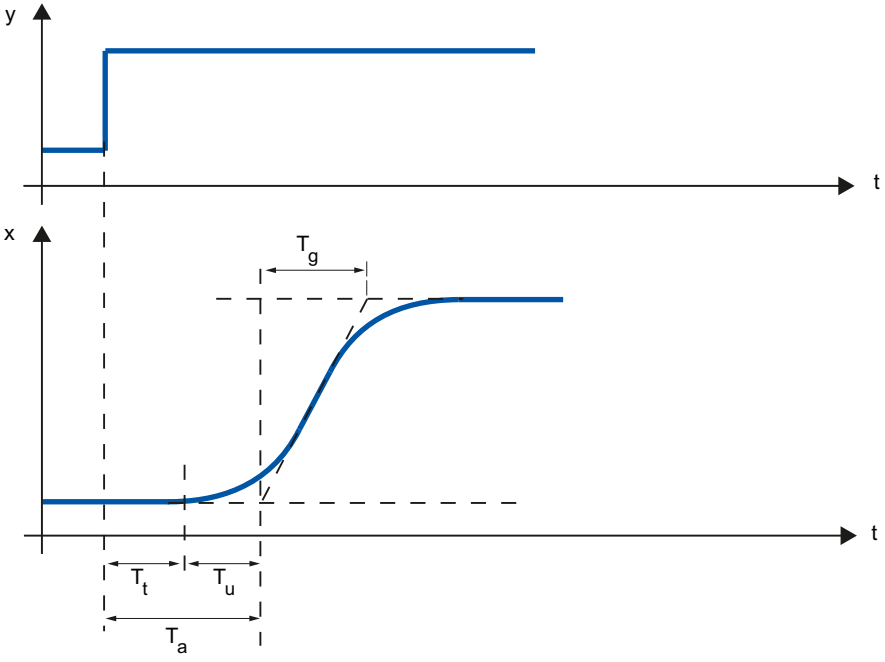
to the maximum value and the inflection point of the step response. In many situations, it is not possible to record the response characteristic up to the maximum value because the process value cannot exceed specific values. In this case, the rate of rise v_{max} is used to identify the controlled system ($v_{max} = \Delta_x/\Delta_t$).

The controllability of the controlled system can be estimated based on the ratio T_u/T_g, or $T_u \times v_{max}/X_{max}$. Rule:

Process type	T_u / T_g	Suitability of the controlled system for controlling
I	< 0,1	can be controlled well
II	0.1 to 0.3	can still be controlled
III	> 0,3	difficult to control

Influence of the dead time on the controllability of a controlled system

A controlled system with dead time and recovery reacts as follows to a jump of the output value.



T_t	Dead time
T_u	Delay time
T_g	Recovery time
y	Output value
x	Process value

The controllability of a self-regulating controlled system with dead time is determined by the ratio of T_t to T_g. T_t must be small compared to T_g. Rule:

$T_t/T_g \leq 1$

Response rate of controlled systems

Controlled systems can be judged on the basis of the following values:

T_u < 0.5 min, T_g < 5 min = fast controlled system

T_u > 0.5 min, T_g > 5 min = slow controlled system

Parameters of certain controlled systems

Physical quantity	Controlled system	Delay time T_u	Recovery time T_g	Rate of rise v_{max}
Temperature	Small electrically heated furnace	0.5 to 1 min	5 to 15 min	Up to 60 K/min.
	Large electrically heated annealing furnace	1 to 5 min	10 to 20 min	Up to 20 K/min.
	Large gas-heated annealing furnace	0.2 to 5 min	3 to 60 min	1 to 30 K/min
	Distillation tower	1 to 7 min	40 to 60 min	0.1 to 0.5° C/s
	Autoclaves (2.5 m^3)	0.5 to 0.7 min	10 to 20 min	Not specified
	High-pressure autoclaves	12 to 15 min	200 to 300 min	Not specified
	Steam superheater	30 s to 2.5 min	1 to 4 min	2°C/s
	Injection molding machines	0.5 to 3 min	3 to 30 min	5 to 20 K/min
	Extruders	1 to 6 min	5 to 60 min	
	Packaging machines	0.5 to 4 min	3 to 40 min	2 to 35 K/min
	Room heating	1 to 5 min	10 to 60 min	1° C/min
Flow rate	Pipeline with gas	0 to 5 s	0.2 to 10 s	Not relevant
	Pipeline with liquid	None	None	
Pressure	Gas pipeline	None	0.1 s	Not relevant
	Drum boiler with gas or oil firing	None	150 s	Not relevant
	Drum boiler with impact grinding mills	1 to 2 min	2 to 5 min	Not relevant
Vessel level	Drum boiler	0.6 to 1 min	Not specified	0.1 to 0.3 cm/s
Speed	Small electric drive	None	0.2 to 10 s	Not relevant
	Large electric drive	None	5 to 40 s	Not relevant
	Steam turbine	None	Not specified	50 min^{-1}
Voltage	Small generators	None	1 to 5 s	Not relevant
	Large generators	None	5 to 10 s	Not relevant

13.1.1.4 Pulse controller

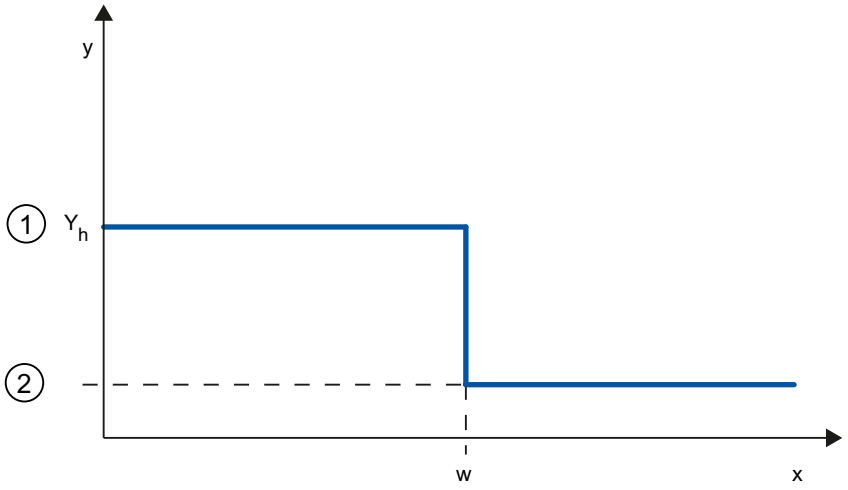
Two-step controllers without feedback

Two-step controllers have the state "ON" and "OFF" as the switching function. This corresponds to 100% or 0% output. This behavior generates a sustained oscillation of process value x around setpoint w.

The amplitude and duration of the oscillation increase in proportion to the ratio between the delay time T_u and recovery time T_g of the controlled system. These controllers are used mainly

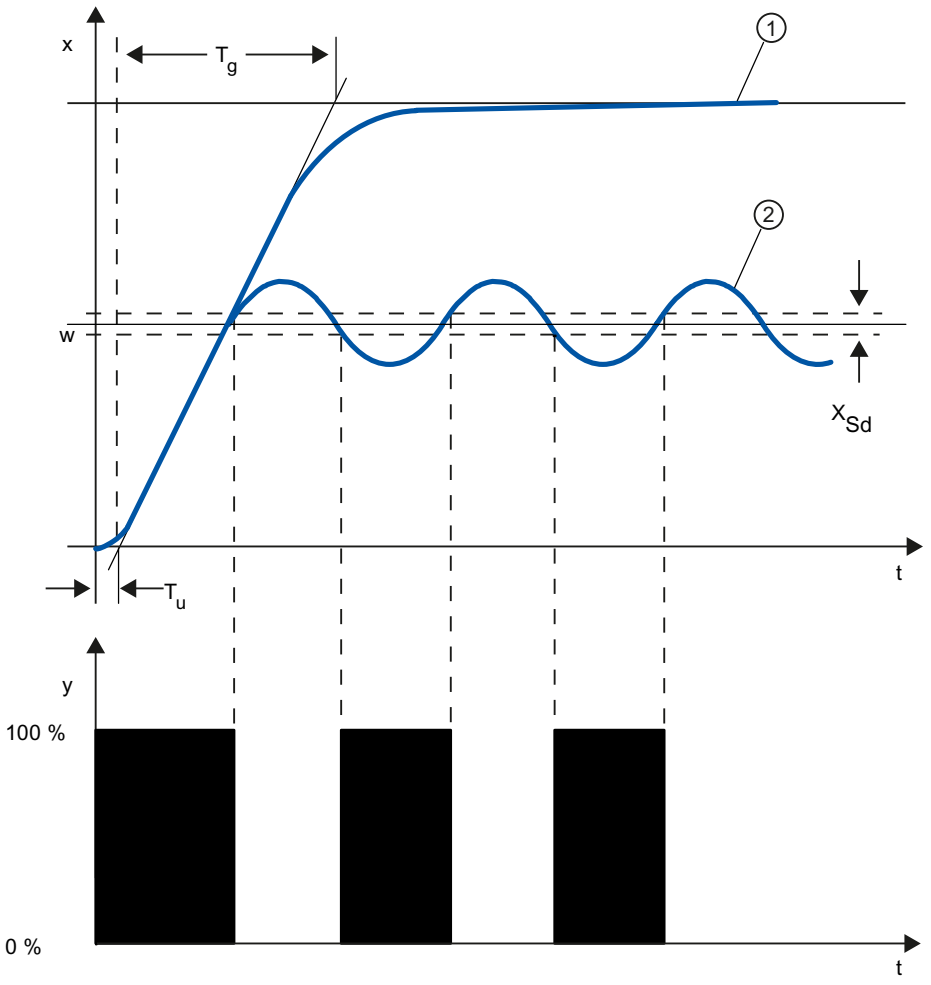
for simple temperature control systems (such as electrically directly heated furnaces) or as limit-value signaling units.

The following diagram shows the characteristic of a two-step controller



①	ON
②	OFF
Y_h	Control range
w	Setpoint

The following diagram shows the control function of a two-step controller



①	Response characteristic without controller
②	Response characteristic with two-step controller
T_u	Delay time
T_g	Recovery time
X_{Sd}	Switching difference

Two-step controllers with feedback

The behavior of two-step controllers in the case of controlled systems with larger delay times, such as furnaces where the functional space is separated from the heating, can be improved by the use of electronic feedback.

The feedback is used to increase the switching frequency of the controller, which reduces the amplitude of the process value. In addition, the control-action results can be improved substantially in dynamic operation. The limit for the switching frequency is set by the output level. It should not exceed 1 to 5 switches per minute at mechanical actuators, such as relays and contactors. In the case of voltage and current outputs with downstream thyristor or Triac controllers high switching frequencies can be selected that exceed the limit frequency of the controlled system by far.

Since the switching pulses can no longer be determined at the output of the controlled system, results comparable with those of continuous controllers are obtained.

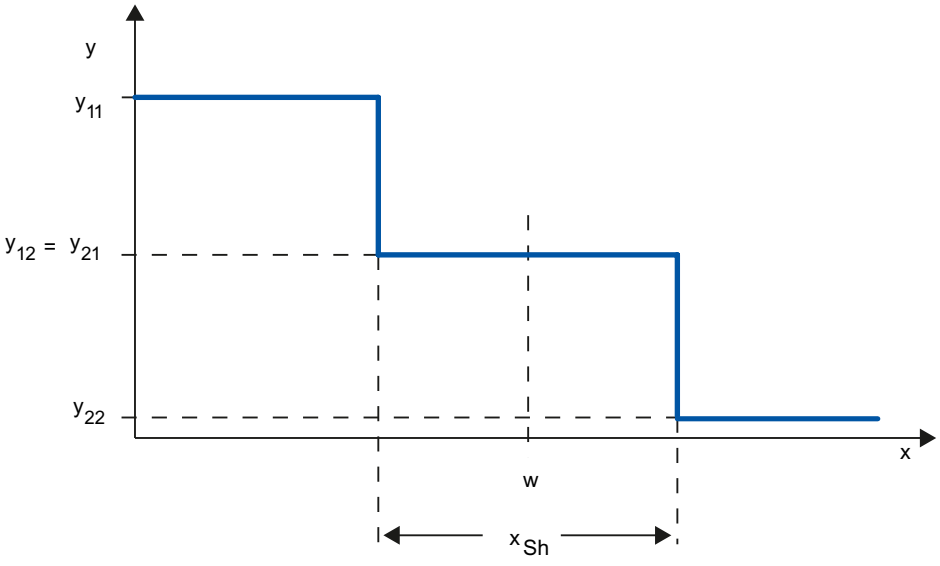
The output value is generated by pulse-width modulation of the output value of a continuous controller.

Two-step controllers with feedback are used for temperature control in furnaces, at processing machines in the plastics, textile, paper, rubber and foodstuff industries as well as for heating and cooling devices.

Three-step controllers

Three-step controllers are used for heating / cooling. These controllers have two switching points as their output. The control-action results are optimized through electronic feedback structures. Fields of applications for such controllers are heating, low-temperature, climatic chambers and tool heating units for plastic-processing machines.

The following diagram shows the characteristic of a three-step controller

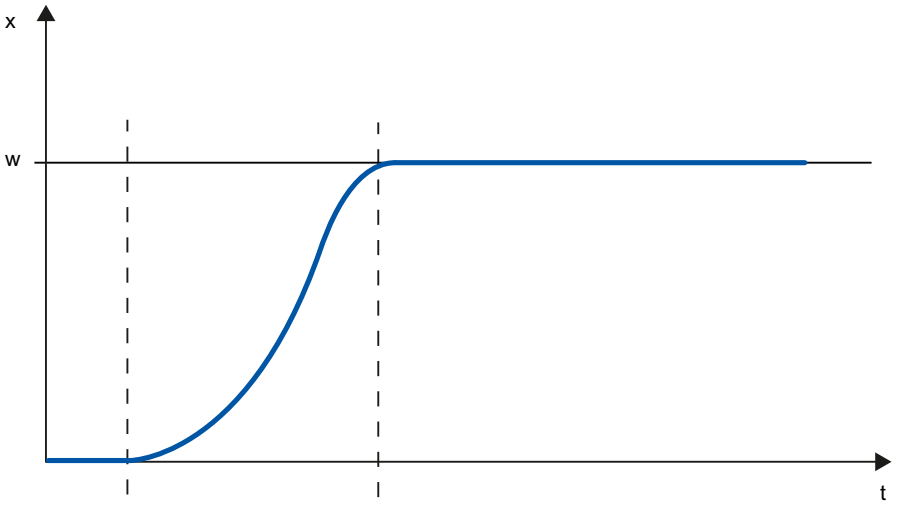


y	Output value, e.g. y11 = 100% heating y12 = 0% heating y21 = 0% cooling y22 = 100% cooling
x	Physical quantity of the process value, e.g., temperature in° C
w	Setpoint
x_{Sh}	Distance between Switching Point 1 and Switching Point 2

13.1.1.5 Response to setpoint changes and disturbances

Response to setpoint changes

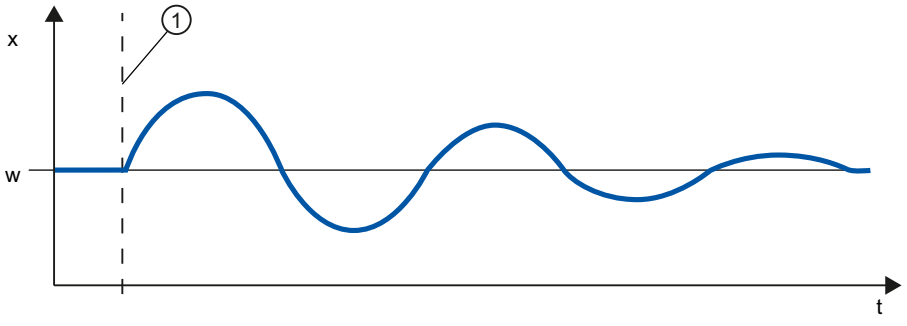
The process value should follow a setpoint change as quickly as possible. The response to setpoint changes is improved by minimizing fluctuation of the process value and the time required to reach the new setpoint.



x	Process value
w	Setpoint

Response to disturbances

The setpoint is influenced by disturbance variables. The controller has to eliminate the resulting control deviations in the shortest time possible. The response to disturbances is improved by minimizing fluctuation of the process value and the time required to reach the new setpoint.



x	Process value
w	Setpoint
①	Influencing a disturbance variable

Disturbance variables are corrected by a controller with integral action. A persistent disturbance variable does not reduce control quality because the control deviation is relatively constant. Dynamic disturbance variables have a more significant impact on control quality because of control deviation fluctuation. The control deviation is eliminated again only by means of the slow acting integral action.

A measurable disturbance variable can be included in the controlled system. This inclusion would significantly accelerated the response of the controller.

13.1.1.6 Control Response at Different Feedback Structures

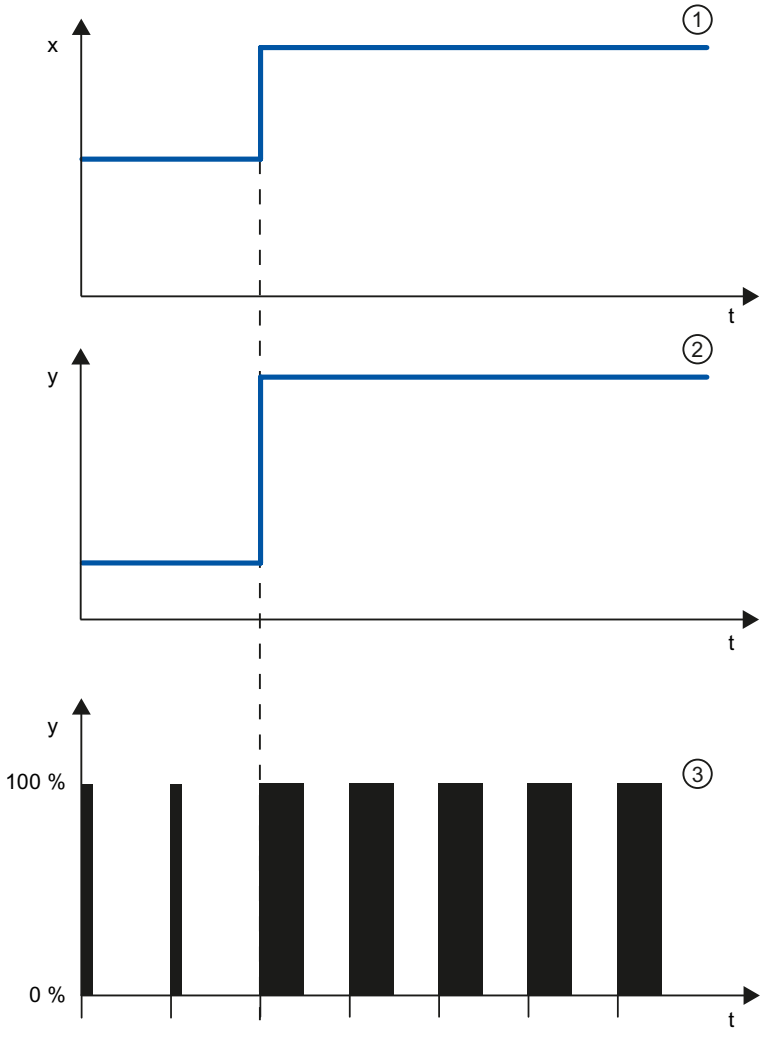
Control behavior of controllers

A precise adaptation of the controller to the time response of the controlled system is decisive for the controller's precise settling to the setpoint and optimum response to disturbance variables.

The feedback circuit can have a proportional action (P), proportional-derivative action (PD), proportional-integral action (PI), or proportional-integral-derivative action (PID).

If step functions are to be triggered by control deviations, the step responses of the controllers differ depending on their type.

Step response of a proportional action controller



① Control deviation
② Output value of a continuous controller
③ Output value of a pulse controller

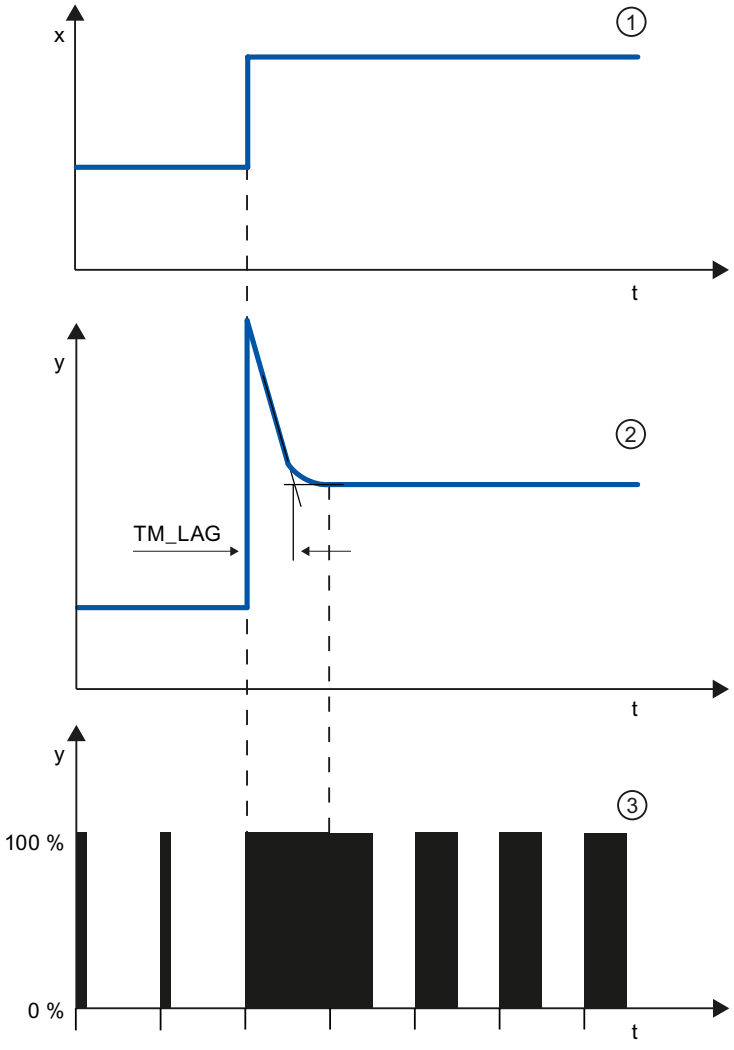
Equation for proportional action controller

Output value and control deviation are directly proportional, meaning:

Output value = proportional gain × control deviation

$y = GAIN \times x$

Step response of a PD-action controller



①	Control deviation
②	Output value of a continuous controller
③	Output value of a pulse controller
TM_LAG	Delay of the Derivative action

Equation for PD-action controller

The following applies for the step response of the PD-action controller in the time range:

$$y = GAIN \cdot X_W \cdot \left(1 + \frac{TD}{TM_LAG} \cdot e^{\frac{-t}{TM_LAG}} \right)$$

t = time interval since the step of the control deviation

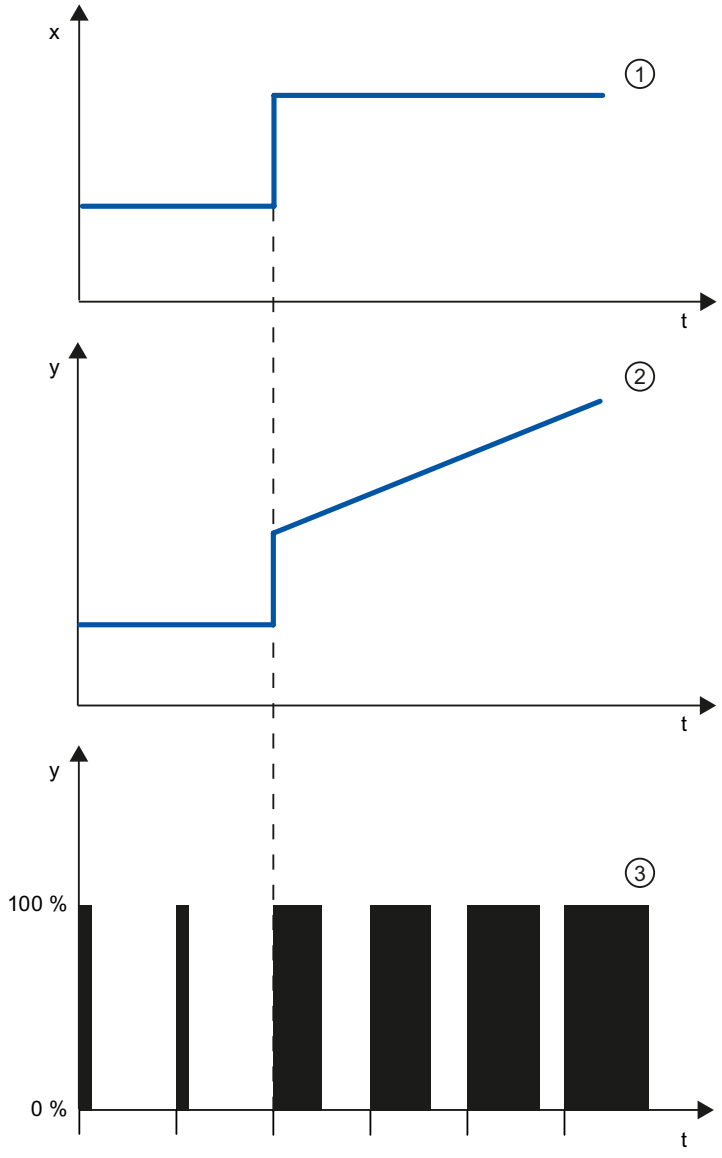
The derivative action generates a output value as a function of the rate of change of the process value. A derivative action by itself is not suitable for controlling because the output value only

follows a step of the process value. As long as the process value remains constant, the output value will no longer change.

The response to disturbances of the derivative action is improved in combination with a proportional action. Disturbances are not corrected completely. The good dynamic response is advantageous. A well attenuated, non-oscillating response is achieved during approach and setpoint change.

A controller with derivative action is not appropriate if a controlled system has pulsing measured quantities, for example, in the case of pressure or flow control systems.

Step response of a PI-action controller



① Control deviation
② Output value of a continuous controller
③ Output value of a pulse controller

An integral action in the controller adds the control deviation as a function of the time. This means that the controller corrects the system until the control deviation is eliminated. A sustained control deviation is generated at controllers with proportional action only. This effect can be eliminated by means of an integral action in the controller.

In practical experience, a combination of the proportional, integral and derivative actions is ideal, depending on the requirements placed on the control response. The time response of the individual components can be described by the controller parameters proportional gain GAIN, integral action time TI (integral action), and derivative action time TD (derivative action).

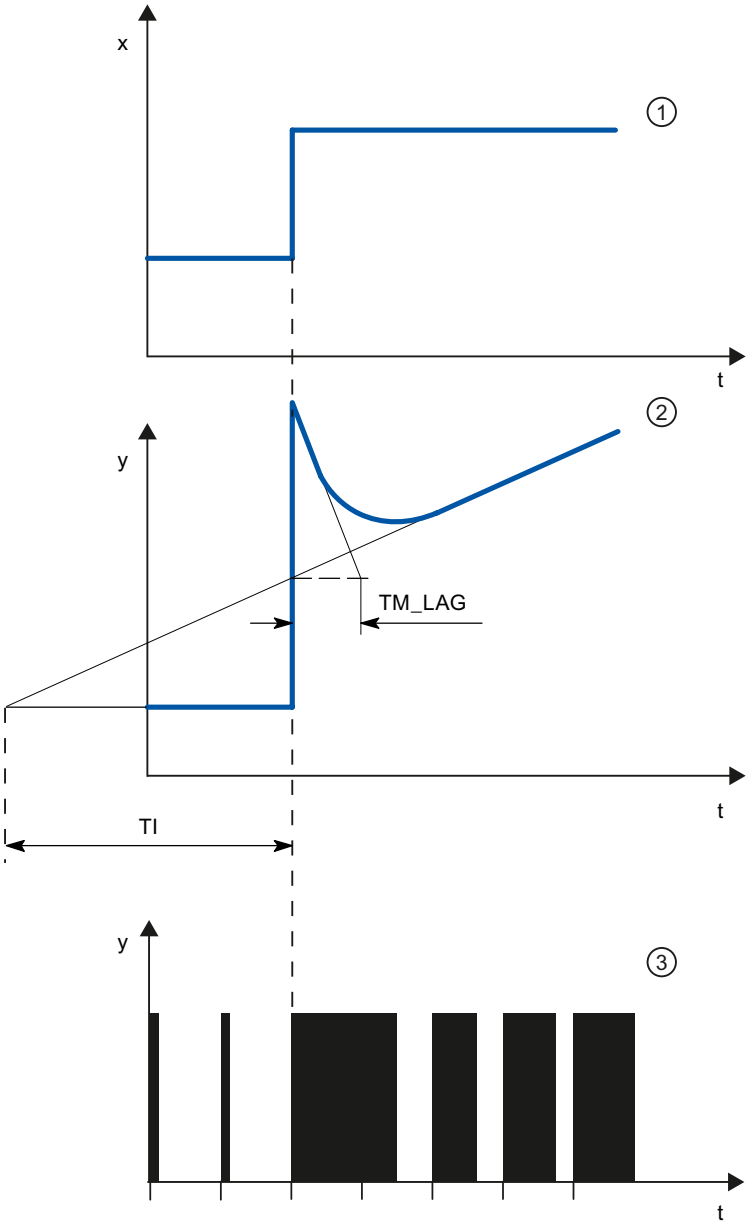
Equation for PI-action controller

The following applies for the step response of the PI-action controller in the time range:

$$y = GAIN \cdot X_W \cdot \left(1 + \frac{1}{TI \cdot t} \right)$$

t = time interval since the step of the control deviation

Step response of a PID controller



①	Control deviation
②	Output value of a continuous controller
③	Output value of a pulse controller
TM_LAG	Delay of the Derivative action
T_i	Integral action time

Equation for PID controller

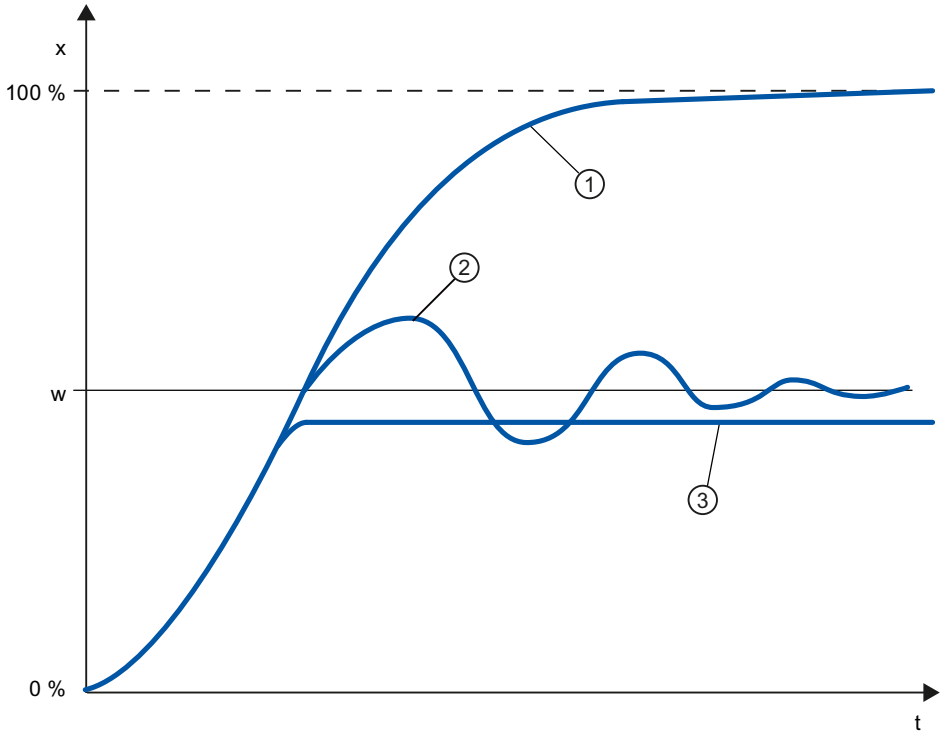
The following applies for the step response of the PID controller in the time range:

$$y = GAIN \cdot X_w \cdot \left(1 + \frac{1}{TI \cdot t} + \frac{TD}{TM_LAG} \cdot e^{\frac{-t}{TM_LAG}} \right)$$

t = time interval since the step of the control deviation

Response of a controlled system with different controller structures

Most of the controller systems occurring in process engineering can be controlled by means of a controller with PI-action response. In the case of slow controlled system with a large dead time, for example temperature control systems, the control result can be improved by means of a controller with PID action.



①	No controller
②	PID controller
③	PD-action controller
w	Setpoint
x	Process value

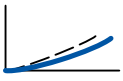
Controllers with PI and PID action have the advantage that the process value does not have any deviation from the setpoint value after settling. The process value oscillates over the setpoint during approach.

13.1.1.7 Selection of the controller structure for specified controlled systems

Selection of the Suitable Controller Structures

To achieve optimum control results, select a controller structure that is suitable for the controlled system and that you can adapt to the controlled system within specific limits.

The table below provides an overview of suitable combinations of a controller structure and controlled system.

Controlled system		Controller structure			
		P	PD	PI	PID
	With dead time only	Unsuitable	Unsuitable	Suitable	Unsuitable
	PT1 with dead time	Unsuitable	Unsuitable	Well suited	Well suited
	PT2 with dead time	Unsuitable	Suited conditionally	Well suited	Well suited
	Higher order	Unsuitable	Unsuitable	Suited conditionally	Well suited
	Not self-regulating	Well suited	Well suited	Well suited	Well suited

The table below provides an overview of suitable combinations of a controller structure and physical quantity.

Physical quantity	Controller structure			
	P	PD	PI	PID
	Sustained control deviation		No sustained control deviation	
Temperature	For low performance requirements and proportional action controlled systems with $T_u/Tg < 0,1$	Well suited	The most suitable controller structures for high performance requirements (except for specially adapted special controllers)	
Pressure	Suitable, if the delay time is inconsiderable	Unsuitable	The most suitable controller structures for high performance requirements (except for specially adapted special controllers)	
Flow rate	Unsuitable, because required GAIN range is usually too large	Unsuitable	Suitable, but integral action controller alone often better	Hardly required

13.1.1.8 PID parameter settings

Rule of Thumb for the Parameter Setting

Controller structure	Setting
P	GAIN ≈ v_{max} × T_u [° C]
PI	GAIN ≈ 1.2 × v_{max} × T_u [° C]
PD	GAIN ≈ 0.83 × v_{max} × T_u [° C] TD ≈ 0.25 × v_{max} × T_u [min] TM_LAG ≈ 0.5 × TD[min]
PID	GAIN ≈ 0.83 × v_{max} × T_u [° C] TI ≈ 2 × T_u [min] TD ≈ 0.4 × T_u [min] TM_LAG ≈ 0.5 × TD[min]
PD/PID	GAIN ≈ 0.4 × v_{max} × T_u [° C] TI ≈ 2 × T_u [min] TD ≈ 0.4 × T_u [min] TM_LAG ≈ 0.5 × TD[min]

Instead of $v_{max} = \Delta_x / \Delta_t$, you can use X_{max} / T_g.

In the case of controllers with PID structure the setting of the integral action time and differential-action time is usually coupled with each other.

The ratio TI / TD lies between 4 and 5 and is optimal for most controlled systems.

Non-observance of the differential-action time TD is uncritical at PD controllers.

In the case of PI and PID controllers, control oscillations occur if the integral action time TI has been select by more than half too small.

An integral action time that is too large slows down the settling times of disturbances. One cannot expect that the control loops operate "optimally" after the first parameter settings. Experience shows that adjusting is always necessary, when a system exists that is "difficult to control" with $T_u / T_g > 0.3$.

13.1.2 Configuring a software controller

13.1.2.1 Overview of software controller

For the configuration of a software controller, you need an instruction with the control algorithm and a technology object. The technology object for a software controller corresponds with the instance DB of the instruction. The configuration of the controller is saved in the technology object. In contrast to the instance DBs of other instructions, technology objects are not stored for the program resources, but rather under CPU > Technology objects.

Technology objects and instructions

CPU	Library	Instruction	Technology object	Description
S7-1200	Compact PID	PID_Compact V1.X	PID_Compact V1.X	Universal PID controller with integrated tuning
S7-1200		PID_3Step V1.X	PID_3Step V1.X	PID controller with integrated tuning for valves
S7-1500 S7-1200 V4.x		PID_Compact V2.X	PID_Compact V2.X	Universal PID controller with integrated tuning
S7-1500 S7-1200 V4.x		PID_3Step V2.X	PID_3Step V2.X	PID controller with integrated tuning for valves
S7-1500 ≥ V1.7 S7-1200 ≥ V4.1		PID_Temp V1.0	PID_Temp V1.0	Universal PID temperature controller with integrated tuning
S7-1500/300/400	PID basic functions	CONT_C	CONT_C	Continuous controller
S7-1500/300/400		CONT_S	CONT_S	Step controller for actuators with integrating behavior
S7-1500/300/400		PULSEGEN	-	Pulse generator for actuators with proportional behavior
S7-1500/300/400		TCONT_CP	TCONT_CP	Continuous temperature controller with pulse generator
S7-1500/300/400		TCONT_S	TCONT_S	Temperature controller for actuators with integrating behavior
S7-300/400	PID Self Tuner	TUN_EC	TUN_EC	Optimization of a continuous controller
S7-300/400		TUN_ES	TUN_ES	Optimization of a step controller
S7-300/400	Standard PID Control (PID Professional optional package)	PID_CP	PID_CP	Continuous controller with pulse generator
S7-300/400		PID_ES	PID_ES	Step controller for actuators with integrating behavior
S7-300/400		LP_SCHED	-	Distribute controller calls

CPU	Library	Instruction	Technology object	Description
S7-300/400	Modular PID Control (PID Professional optional package)	A_DEAD_B	-	Filter interfering signal from control deviation
S7-300/400		CRP_IN	-	Scale analog input signal
S7-300/400		CRP_OUT	-	Scale analog output signal
S7-300/400		DEAD_T	-	Delay output of input signal
S7-300/400		DEADBAND	-	Suppress small fluctuations to the process value
S7-300/400		DIF	-	Differentiate input signals over time
S7-300/400		ERR_MON		Monitor control deviation
S7-300/400		INTEG	-	Integrate input signals over time
S7-300/400		LAG1ST	-	First-order delay element
S7-300/400		LAG2ND	-	Second-order delay element
S7-300/400		LIMALARM	-	Report limit values
S7-300/400		LIMITER	-	Limiting the manipulated variable
S7-300/400		LMNGEN_C	-	Determine manipulated variable for continuous controller
S7-300/400		LMNGEN_S	-	Determine manipulated variable for step controller
S7-300/400		NONLIN	-	Linearize encoder signal
S7-300/400		NORM	-	Scale process value physically
S7-300/400		OVERRIDE	-	Switch manipulated variable from 2 PID controllers to 1 actuator
S7-300/400		PARA_CTL	-	Switch parameter sets
S7-300/400		PID	-	PID algorithm
S7-300/400		PUSLEGEN_M	-	Generate pulse for proportional actuators
S7-300/400		RMP_SOAK	-	Specify setpoint according to ramp / soak
S7-300/400		ROC_LIM	-	Limit rate of change
S7-300/400		SCALE_M	-	Scale process value
S7-300/400		SP_GEN	-	Specify setpoint manually
S7-300/400		SPLT_RAN	-	Split manipulated variable range
S7-300/400		SWITCH	-	Switch analog values
S7-300/400		LP_SCHED_M	-	Distribute controller calls

13.1.2.2 Steps for the configuration of a software controller

All SW-controllers are configured according to the same scheme:

Step	Description
1	Add technology object (Page 7201)
2	Configure technology object (Page 7202)
3	Call instruction in the user program (Page 7203)
4	Download technology object to device (Page 7204)
5	Commission software controller (Page 7205)
6	Save optimized PID parameters in the project (Page 7205)

Step	Description
7	Comparing values (Page 7207)
8	Display instances of a technology object (Page 7226)

13.1.2.3 Add technology objects

Add technology object in the project navigator

When a technology object is added, an instance DB is created for the instruction of this technology object. The configuration of the technology object is stored in this instance DB.

Requirement

A project with a CPU has been created.

Procedure

To add a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click "Add new object".
   The "Add new object" dialog box opens.
4. Click on the "PID" button.
   All available PID-controllers for this CPU are displayed.
5. Select the instruction for the technology object, for example, PID_Compact.
6. Enter an individual name for the technology object in the "Name" input field.
7. Select the "Manual" option if you want to change the suggested data block number of the instance DB.
8. Click "Further information" if you want to add own information to the technology object.
9. Confirm with "OK".

Result

The new technology object has been created and stored in the project tree in the "Technology objects" folder. The technology object is used if the instruction for this technology object is called in a cyclic interrupt OB.

Note

You can select the "Add new and open" check box at the bottom of the dialog box. This opens the configuration of the technology object after adding has been completed.

13.1.2.4 Configure technology objects

The properties of a technology object on a S7-1200 CPU can be configured in two ways.

- In the Inspector window of the programming editor
- In the configuration editor

The properties of a technology object on a S7-300/400 CPU can only be configured in the configuration editor.

Inspector window of the programming editor

In the Inspector window of the programming editor you can only configure the parameters required for operation.

The offline values of the parameters are also shown in online mode. You can only change the online values in the commissioning window.

To open the Inspector window of the technology object, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Double click the block (cyclic interrupt OB) in which you open the instruction of the SW-controller.
   The block is opened in the work area.
3. Click on the instruction of the SW-controller.
4. In the Inspector window, select the "Properties" and "Configuration" tabs consecutively.

Configuration window

For each technology object, there is a specific configuration window in which you can configure all properties.

To open the configuration window of the technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.

Symbols

Icons in the area navigation of the configuration and in the Inspector window show additional details about the completeness of the configuration:

	The configuration contains default values and is complete. The configuration exclusively contains default values. With these default values the use of the technology object is possible without further changes.
	The configuration contains values defined by the user and is complete All input fields of the configuration contain valid values and at least one default setting was changed.
	The configuration is incomplete or faulty At least one input field or a collapsible list contains no or one invalid value. The corresponding field or the drop-down list box has a red background. When clicked the roll-out error message indicates the cause of the error.

The properties of a technology object are described in detail in the chapter for the technology object.

13.1.2.5 Call instruction in the user program

The instruction of the software controller must be called in a cyclic interrupt OB. The sampling time of the software controller is determined by the interval between the calls in the cyclic interrupt OB.

Requirement

The cyclic interrupt OB is created and the cycle time of the cyclic interrupt OB is correctly configured.

Procedure

Proceed as follows to call the instruction in the user program:

1. Open the CPU folder in the project tree.
2. Open the "Program blocks" folder.
3. Double-click the cyclic interrupt OB.
   The block is opened in the work area.
4. Open the "Technology" group in the "Instructions" window and the "PID Control" folder.
   The folder contains all instructions for software controllers that can be configured on the CPU.
5. Select the instruction and drag it to your cyclic interrupt OB.
   The "Call options" dialog box opens.
6. Select a technology object or type the name for a new technology object from the "Name" list.

Result

If the technology object does not exist yet, it is added. The instruction is added in the cyclic interrupt OB. The technology object is assigned to this call of the instruction.

13.1.2.6 Downloading technology objects to device

A new or modified configuration of the technology object must be downloaded to the CPU for the online mode. The following characteristics apply when downloading retentive data:

- **Software (changes only)**
  - S7-1200, S7-1500:
    Retentive data is retained.
  - S7-300/400:
    Retentive data is updated immediately. CPU does not change to Stop.
- **Download PLC program to device and reset**
  - S7-1200, S7-1500:
    Retentive data is updated at the next change from Stop to RUN. The PLC program can only be downloaded completely.
  - S7-300/400:
    Retentive data is updated at the next change from Stop to RUN.

Downloading retentive data to an S7-1200 or S7-1500 CPU

Note

The download and reset of the PLC program during ongoing system operation can result in serious damages or injuries in the case of malfunctions or program errors.

Make sure that dangerous states cannot occur before you download and reset the PLC program.

Proceed as follows to download the retentive data:

1. Select the entry of the CPU in the project tree.
2. Select the command "Download and reset PLC program" from the "Online" menu.
   - If you have not established an online connection yet, the "Extended download" dialog opens. In this case, set all required parameters for the connection and click "Download".
   - If the online connection has been defined, the project data is compiled, if necessary, and the dialog "Load preview" opens. This dialog displays messages and recommends actions necessary for download.
3. Check the messages.
   As soon as download is possible, the "Download" button becomes active.
4. Click on "Download".
   The complete PLC program is downloaded and the "Load results" dialog opens. This dialog displays the status and the actions after the download.
5. If the modules are to restart immediately after the download, select the check box "Start all".
6. Close the dialog "Download results" with "Finish".

Result

The complete PLC program is downloaded to the device. Blocks that only exist online in the device are deleted. By downloading all affected blocks and by deleting any blocks in the device that are not required, you avoid inconsistencies between the blocks in the user program.

The messages under "Info > General" in the Inspector window indicate whether the download was successful.

13.1.2.7 Commissioning software controller

Procedure

To open the "Commissioning" work area of the technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Commissioning" object.

The commissioning functions are specific for each controller and are described there.

13.1.2.8 Save optimized PID parameter in the project

The software controller is optimized in the CPU. Through this, the values in the instance-DB on the CPU no longer agree with those in the project.

To update the PID parameter in the project with the optimized PID parameters, proceed as follows:

Requirement

- An online connection to the CPU is established and the CPU is in "RUN" mode.
- The functions of the commissioning window have been enabled by means of the "Start" button.

Procedure

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Open a technology object.
4. Double click on "Commissioning".
5. Click on the icon "Upload PID parameters".
6. Save the project.

Result

The currently active PID parameters are stored in the project data. When reloading the project data in the CPU, the optimized parameters are used.

13.1.2.9 Comparing values

Comparison display and boundary conditions

The "Compare values" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of actual values in the project
- Transfer of start values of the project to the CPU as actual values

Icons and operator controls

The following icons and operator controls are available:

Icon	Function
	Start value PLC matches the configured Start value project
	Start value PLC does not match the configured Start value project
	The comparison of the Start value PLC with the configured Start value project cannot be performed
	At least one of the two comparison values has a process-related or syntax error.
	Transfers actual values to the offline project
	Transfers updated start values in the project to the CPU (initialize setting values)
	Opens the "Compare values" dialog

Boundary conditions

The "Compare values" function is available for S7-1200 and S7-1500 without limitations.

The following limitation applies to S7-300 and S7-400:

In monitoring mode, an S7-300/S7-400 cannot transfer the start values to the CPU. These values cannot be displayed online with "Compare values".

The actual values of the technology object are displayed and can be changed directly.

Comparing values

The procedure is shown in the following using "PID Parameters" as an example.

Requirements

- A project with a software controller is configured.
- The project is downloaded to the CPU.
- The configuration dialog is open in the project navigator.

Procedure

1. Open the desired software controller in the project navigation.
2. Double-click the "Configuration" object.
3. Navigate within the configuration window to the "PID Parameters" dialog.
4. Click the icon to activate monitoring mode.
   The icons and operator controls (Page 7204) of the "Compare values" function are shown behind the parameters.
5. Click the desired parameter in the input box and change the parameter values manually by entering them directly.
   - If the background of the input box is gray, this value is a read-only value and cannot be changed.
   - To change the values in the "PID Parameters" dialog, enable manual entry by selecting the "Enable manual entry" check box beforehand.
6. Click the icon to open the dialog for the start values.
   This dialog indicates two values of the parameter:
   - Start value in CPU: The start value in the CPU is shown in the top part.
   - Start value in the project: The configured start value in the project is shown in the bottom part.
7. Enter the desired value in the input box for the project.

Error detection

The input of incorrect values is detected. Corrections are suggested in this case.

If you enter a value with incorrect syntax, a rollout containing the corresponding error message opens below the parameter. The incorrect value is not applied.

If you enter a value that is incorrect for the process, a dialog opens containing the error message and a suggested correction:

- Click "No" to accept this suggested correction and correct your input.
- Click "OK" to apply the incorrect value.

Notice

Malfunctions of the controller

Values incorrect for the process can result in controller malfunctions.

Backing up actual values

Click the icon to transfer the actual controller values to the start values of your configured project.

Transferring project values to the CPU

Click the icon to transfer the configured values of your project to the CPU.

Caution

Prevent personal injury and property damage!

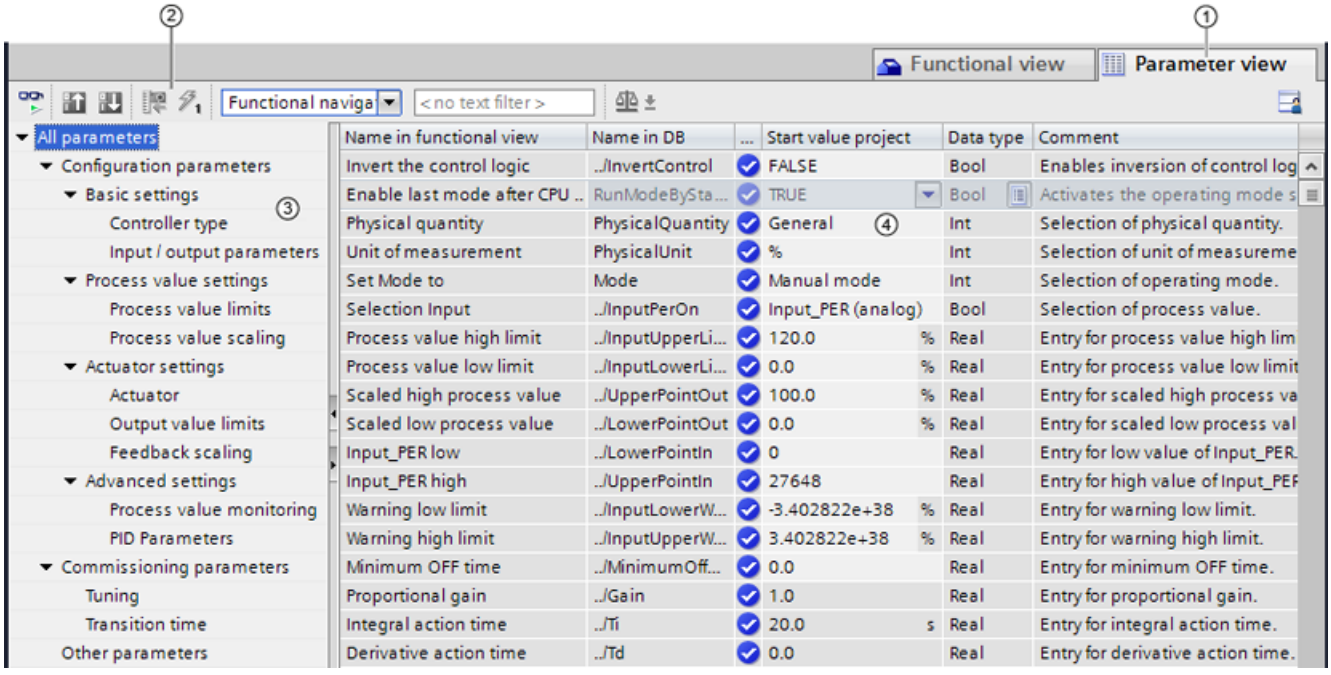
Downloading and resetting of the user program while the plant is operating may result in significant property damage and severe personal injuries in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you download and reset the user program.

13.1.2.10 Parameter view

Introduction to the parameter view

The Parameter view provides you with a general overview of all relevant parameters of a technology object. You obtain an overview of the parameter settings and can easily change them in offline and online mode.



① "Parameter view" tab

② Toolbar (Page 7211)

③ Navigation (Page 7211)

④ Parameter table (Page 7212)

Function scope

The following functions are available for analyzing the parameters of the technology objects and for enabling targeted monitoring and modification.

Display functions:

- Display of parameter values in offline and online mode
- Display of status information of the parameters
- Display of value deviations and option for direct correction
- Display of configuration errors
- Display of value changes as a result of parameter dependencies

- Display of all memory values of a parameter: Start value PLC, Start value project, Monitor value
- Display of the parameter comparison of the memory values of a parameter

Operator control functions:

- Navigation for quickly changing between the parameters and parameter structures.
- Text filter for faster searches for particular parameters.
- Sorting function for customizing the order of parameters and parameter groups to requirements.
- Memory function for backing up structural settings of the Parameter view.
- Monitoring and modifying of parameter values online.
- Function for saving a snapshot of parameter values of the CPU in order to capture momentary situations and to respond to them.
- Function for applying a snapshot of parameter values as start values.
- Download of modified start values to the CPU.
- Comparison functions for comparing parameter values with one another.

Validity

The Parameter view described here is available for the following technology objects:

- PID_Compact
- PID_3Step
- PID_Temp
- CONT_C (S7-1500 only)
- CONT_S (S7-1500 only)
- TCONT_CP (S7-1500 only)
- TCONT_S (S7-1500 only)
- TO_Axis_PTO (S7-1200 Motion Control)
- TO_Positioning_Axis (S7-1200 Motion Control)
- TO_CommandTable_PTO (S7-1200 Motion Control)
- TO_CommandTable (S7-1200 Motion Control)

Structure of the parameter view

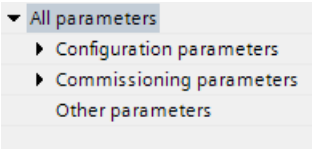
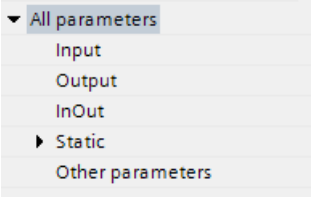
Toolbar

The following functions can be selected in the toolbar of the parameter view.

Icon	Function	Explanation
	Monitor all	Starts the monitoring of visible parameters in the active Parameter view (online mode).
	Create snapshot of monitor values and accept setpoints of this snapshot as start values	Applies the current monitor values to the “Snapshot” column and updates the start values in the project. Only in online mode for PID_Compact and PID_3Step.
	Initialize setpoints	Transfers the start values updated in the project to the CPU. Only in online mode for PID_Compact and PID_3Step.
	Create snapshot of monitor values	Applies the current monitor values to the “Snapshot” column. Only in online mode.
	Modify all selected parameters immediately and once	This command is executed once and as quickly as possible without reference to any particular point in the user program. Only in online mode.
	Select navigation structure	Toggles between functional navigation and data navigation.
	Text filter...	After entry of a character string: Display of all parameters containing the specified string in one of the currently visible columns.
	Selection of compare values	Selection of parameter values that are to be compared with one another in online mode (Start value project, Start value PLC, Snapshot) Only in online mode.
	Save window settings	Saves your display settings for the Parameter view (e.g., selected navigation structure, activated table columns, etc.)

Navigation

Within the "Parameter view" tab, the following alternative navigation structures can be selected.

Navigation		Explanation
Functional navigation		In the functional navigation, the structure of the parameters is based on the structure in the configuration dialog ("Functional view" tab), commissioning dialog, and diagnostics dialog. The last group "Other parameters" contains all other parameters of the technology object.
Data navigation		In the data navigation, the structure of the parameters is based on the structure in the instance DB / technology DB. The last group "Other parameters" contains the parameters that are not contained in the instance DB / technology DB.

You can use the "Select navigation structure" drop-down list to toggle the navigation structure.

Parameter table

The table below shows the meaning of the individual columns of the parameter table. You can show or hide the columns as required.

- Column "Offline" = X: Column is visible in offline mode.
- Column "Online" = X: Column is visible in online mode (online connection to the CPU).

Column	Explanation	Offline	Online
Name in functional view	Name of the parameter in the functional view. The display field is empty for parameters that are not configured via the technology object.	X	X
Full name in DB	Complete path of the parameter in the instance DB / technology DB. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Name in DB	Name of the parameter in the instance DB / technology DB. If the parameter is part of a structure or UDT, the prefix ". ./" is added. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Status of configuration	Display of the completeness of the configuration using status symbols. see Status of configuration (offline) (Page 7220)	X	
Compare result	Result of the "Compare values" function. This column is shown if there is an online connection and the "Monitor all" button is selected.		X
Start value project	Configured start value in the project. Error indication if entered values have a syntax or process-related error.	X	X
Default value	Value that is pre-assigned to the parameter. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Snapshot	Snapshot of the current values in the CPU (monitor values). Error indication if values have a process-related error.	X	X
Start value PLC	Start value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Monitor value	Current value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Modify value	Value that is to be used to change the monitor valuet. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if entered values have a syntax or process-related error.		X
Selection for transmission	Selection of the Modify values that are to be transmitted using the "Modify all selected parameters immediately and once" button. This column is displayed together with the "Modify value" column.		X

Column	Explanation	Offline	Online
Minimum value	Minimum process-related value of the parameter. If the minimum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Maximum value	Maximum process-related value of the parameter. If the maximum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Setpoint	Designates the parameter as a setpoint. These parameters can be initialized online.	X	X
Data type	Data type of the parameter. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Retain	Designates the value as a retentive value. The values of retentive parameters are retained even after the voltage supply is switched off.	X	X
Accessible from HMI	Indicates whether the HMI can access this parameter during runtime.	X	X
Visible in HMI	Indicates whether the parameter is visible in the selection list of the HMI by default.	X	X
Comment	Brief description of the parameter.	X	X

See also

Comparing values (Page 7204)

Opening the parameter view

Requirement

The technology object has been added in the project tree, i.e., the associated instance DB / technology DB of the instruction has been created.

Procedure

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.
4. Select the "Parameter view" tab in the top right corner.

Result

The Parameter view opens. Each displayed parameter is represented by one row in the parameter table.

The displayable parameter properties (table columns) vary depending on whether you are working with the Parameter view in offline or online mode.

In addition, you can selectively display and hide individual table columns.

See also

Default setting of the parameter view (Page 7214)

Default setting of the parameter view

Default settings

To enable you to work efficiently with the Parameter view, you can customize the parameter display and save your settings.

The following customizations are possible and can be saved:

- Show and hide columns
- Change column width
- Change order of the columns
- Toggle navigation
- Select parameter group in the navigation
- Selection of compare values

Show and hide columns

To show or hide columns in the parameter table, follow these steps:

1. Position the cursor in the header of the parameter table.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the check box for the column.
4. To hide a column, clear the check box for the column.

or

1. Position the cursor in the header of the parameter table.
2. Select the "Show all columns" command in the shortcut menu if all columns of the offline or online mode are to be displayed.

Some columns can only be displayed in online mode: see Parameter table (Page 7210).

Change column width

To customize the width of a column so that all texts in the rows can be read, follow these steps:

1. Position the cursor in the header of the parameter table to the right of the column to be customized until the shape of the cursor changes to a cross.
2. Then double-click this location.

or

1. Open the shortcut menu on the header of the parameter table.
2. Click
   - "Optimize column width" or
   - "Optimize width of all columns".

If the column width setting is too narrow, the complete content of individual fields are shown if you hover the cursor briefly over the relevant field.

Change order of the columns

The columns of the parameter table can be arranged in any way.

To change the order of the columns, follow these steps:

1. Click on the column header and use a drag-and-drop operation to move it to the desired location.
   When you release the mouse button, the column is anchored to the new position.

Toggle navigation

To toggle the display form of the parameters, follow these steps:

1. Select the desired navigation in the “Select navigation structure” drop-down list.
   - Data navigation
   - Functional navigation

See also Navigation (Page 7209).

Select parameter group in the navigation

Within the selected navigation, you choose between the “All parameters” display or the display of a subordinate parameter group of your choice.

1. Click the desired parameter group in the navigation.
   The parameter table only displays the parameters of the parameter group.

Selection of compare values (online)

To set the compare values for the “Compare values” function, follow these steps:

1. Select the desired compare values in the “Selection of compare values” drop-down list.
   - Start value project / Start value PLC
   - Start value project / Snapshot
   - Start value PLC / Snapshot

The “Start value project / Start value PLC” option is set by default.

Saving the default setting of the Parameter view

To save the above customizations of the Parameter view, follow these steps:

1. Customize the Parameter view according to your requirements.
2. Click the “Save window settings” button at the top right of the Parameter view.

Working with the parameter view

Overview

The following table provides an overview of the functions of the Parameter view in online and offline mode described in the following.

- Column "Offline" = X: This function is possible in offline mode.
- Column "Online" = X: This function is possible in online mode.

Function/action	Offline	Online
Filtering the parameter table (Page 7217)	X	X
Sorting the parameter table (Page 7217)	X	X
Transferring parameter data to other editors (Page 7218)	X	X
Indicating errors (Page 7218)	X	X
Editing start values in the project (Page 7219)	X	X
Status of configuration (offline) (Page 7220)	X	
Monitoring values online in the parameter view (Page 7220)		X
Create snapshot of monitor values (Page 7221)		X
Modifying values (Page 7222)		X
Comparing values (Page 7223)		X
Applying values from the online program as start values (Page 7224)		X
Initializing setpoints in the online program (Page 7225)		X

Filtering the parameter table

You can filter the parameters in the parameter table in the following ways:

- With the text filter
- With the subgroups of the navigation

Both filter methods can be used simultaneously.

With the text filter

Texts that are visible in the parameter table can be filtered. This means only texts in displayed parameter rows and columns can be filtered.

1. Enter the desired character string for filtering in the “Text filter...” input box.
   The parameter table displays only the parameters containing the character string.

The text filtering is reset.

- When another parameter group is selected in the navigation.
- When navigation is changed from data navigation to functional navigation, or vice versa.

With the subgroups of the navigation

1. Click the desired parameter group in the navigation, e.g., "Static".
   The parameter table only shows the static parameters. You can select further subgroups for some groups of the navigation.
2. Click “All parameters” in the navigation if all parameters are to be shown again.

Sorting the parameter table

The values of the parameters are arranged in rows. The parameter table can be sorted by any displayed column.

- In columns containing numerical values, sorting is based on the magnitude of the numerical value.
- In text columns, sorting is alphabetical.

Sorting by column

1. Position the cursor in the header cell of the desired column.
   The background of this cell turns blue.
2. Click the column header.

Result

The entire parameter table is sorted by the selected column. A triangle with tip facing up appears in the column header.

Clicking the column header again changes the sorting as follows:

- Symbol "▲": Parameter table is sorted in ascending order.
- Symbol "▼": Parameter table is sorted in descending order.
- No symbol: The sorting is removed again. The parameter table assumes the default display.

The "../" prefix in the "Name in DB" column is ignored when sorting.

Transferring parameter data to other editors

After selecting an entire parameter row of the parameter table, you can use the following:

- Drag-and-drop
- <Ctrl+C>/<Ctrl+V>
- Copy/Paste via shortcut menu

Transfer parameters to the following editors of the TIA Portal:

- Program editor
- Watch table
- Signal table for trace function

The parameter is inserted with its full name: See information in "Full name in DB" column.

Indicating errors

Error indication

Parameter assignment errors that result in compilation errors (e.g., limit violation) are indicated in the Parameter view.

Every time a value is input in the Parameter view, a check is made for process-related and syntax errors and the result is indicated.

Bad values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Table field with red background
  If you click the bad field, a roll-out error message appears with information of the permissible value range or the required syntax (format)

Compilation error

From the error message of the compiler, you can directly open the Parameter view (functional navigation) containing the parameter causing the error in situations where the parameter is not displayed in the configuration dialog.

Editing start values in the project

With the Parameter view, you can edit the start values in the project in offline mode and online mode.

- You make value changes in the “Start value project” column of the parameter table.
- In the “Status of configuration” column of the parameter table, the progress of the configuration is indicated by the familiar status symbols from the configuration dialog of the technology object.

Boundary conditions

- If other parameters depend on the parameter whose start value was changed, the start value of the dependent parameters are also adapted.
- If a parameter of a technology object is not editable, it is also not editable in the parameter view. The ability to edit a parameter can also depend on the values of other parameters.

Defining new start values

To define start values for parameters in the Parameter view, follow these steps:

1. Open the Parameter view of the technology object.
2. Enter the desired start values in the "Start value project" column. The value must match the data type of the parameter and must not exceed the value range of the parameter. The limits of the value range can be seen in the “Maximum value” and “Minimum value” columns.

The "Status of configuration" column indicates the progress of the configuration with colored symbols.

See also Status of configuration (offline) (Page 7220)

Following adaptation of the start values and downloading of the technology object to the CPU, the parameters take the defined value at startup if they are not declared as retentive (“Retain” column).

Error indication

When a start value is input, a check is made for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Red background in the “Start value project” field
  If you click on the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Correcting bad start values

1. Correct bad start values using information from the roll-out error message.
   Red error symbol, red field background, and roll-out error message are no longer displayed.

The project cannot be successfully compiled unless the start values are error-free.

Status of configuration (offline)

The status of the configuration is indicated by icons:

- In the “Status of configuration” column in the parameter table
- In the navigation structure of the functional navigation and data navigation

Symbol in “Status of configuration” column

Symbol	Meaning
	The start value of the parameter corresponds to the default value and is valid. A start value has not yet been defined by the user.
	The start value of the parameter contains a value defined by the user. The start value is different than the default value. The start value is error-free and valid.
	The start value of the parameter is invalid (syntax or process-related error). The input box has a red background. When clicked, the roll-out error message indicates the cause of the error.
	Only for S7-1200 Motion Control: The start value of the parameter is valid but contains warnings. The input box has a yellow background.

Symbol in the navigation

The symbols in the navigation indicate the progress of the configuration in the same way as in the configuration dialog of the technology object.

See also

Configure technology objects (Page 7200)

Monitoring values online in the parameter view

You can monitor the values currently taken by the parameters of the technology object in the CPU (monitor values) directly in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.

- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

Procedure

1. Start the monitoring by clicking .
   As soon as the Parameter view is online, the following columns are additionally displayed:
   - Compare result
   - Start value PLC
   - Monitor value
   - Modify value
   - Selection for transmission

   The "Monitor value" column shows the current parameter values on the CPU.
   Meaning of the additional columns: see Parameter table (Page 7210)
2. Stop the monitoring by clicking again.

Display

All columns that are only available online have an orange background:

- Values in light-orange cells can be changed.
- Values in cells with a dark orange background cannot be changed.

Create snapshot of monitor values

You can back up the current values of the technology object on the CPU (monitor values) and display them in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To show the current parameter values, follow these steps:

1. In the Parameter view, click the “Create snapshot of monitor values" icon .

Result

The current monitor values are transferred once to the "Snapshot" column of the parameter table.

You can analyze the values "frozen" in this way while the monitor values continue to be updated in the "Monitor values" column.

Modifying values

With the Parameter view, you can modify values of the technology object in the CPU.

You can assign values to the parameter once (Modify value) and modify them immediately. The modify request is executed as quickly as possible without reference to any particular point in the user program.

Danger

Danger when modifying:

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you use the "Modify" function.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.
- The parameter can be modified (associated field in the "Modify value" column has a light-orange background).

Procedure

To modify parameters immediately, follow these steps:

1. Enter the desired modify values in the “Modify values” column of the parameter table.
2. Check whether the check box for modifying is selected in the "Select for transmission" column.
   The modify values and associated check boxes of dependent parameters are automatically adapted at the same time.
3. Click the “Modify all selected parameters immediately and once” icon .

The selected parameters are modified once and immediately with the specified values and can be monitored in the "Modify values" column. The check boxes for modifying in the

"Selection for transmission" column are automatically cleared after the modify request is complete.

Error indication

When a start value is input, a check is made immediately for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red background in the “Modify value” field

and

- If you click the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Bad modify values

- Modify values with process-related errors can be transmitted.
- Modify values with syntax errors **cannot** be transmitted.

Comparing values

You can use comparison functions to compare the following memory values of a parameter:

- Start value project
- Start value PLC
- Snapshot

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To compare the start values on the various target systems, follow these steps:

1. Click the "Selection of compare values" icon .
   A selection list containing the comparison options opens:
   - Start value project - Start value PLC (default setting)
   - Start value project - Snapshot
   - Start value PLC - Snapshot
2. Select the desired comparison option.
   The selected comparison option is executed as follows:
   - A scales symbol appears in the header cells of the two columns selected for comparison.
   - Symbols are used in the "Compare result" column to indicate the result of the comparison of the selected columns.

Symbol in "Compare result" column

Symbol	Meaning
	The compare values are equal and error-free.
	The compare values are not equal and error-free.
	At least one of the two compare values has a process-related or syntax error.
	The comparison cannot be performed. At least one of the two compare values is not available (e.g., snapshot).

Symbol in the navigation

The symbols are shown in the same way in the navigation if the comparison result applies to at least one of the parameters below the displayed navigation structure.

Applying values from the online program as start values

In order to apply optimized values from the CPU to the project as start values, you create a snapshot of the monitor values. Values of the snapshot marked as a "Setpoint" are then applied to the project as start values.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To apply optimized values from the CPU, follow these steps:

1. Click the "Create snapshot of monitor values and accept setpoints of this snapshot as start values" icon .

Result

The current monitor values are applied to the "Snapshot" column and their setpoints are copied to the "Start value project" column as new start values.

Note

Applying values of individual parameters

You can also apply the values of individual parameters that are not marked as a setpoint from the "Snapshot" column to the "Start values project" column. To do so, copy the values and insert them into the "Start value project" column using the "Copy" and "Paste" commands in the shortcut menu.

Initializing setpoints in the online program

You can initialize all parameters that are marked as a "Setpoint" in the Parameter view with new values in the CPU in one step. In so doing, the start values are downloaded from the project to the CPU. The CPU remains in "RUN" mode.

To avoid data loss on the CPU during a cold restart or warm restart, you must also download the technology object to the CPU.

Danger

Danger when changing parameter values

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you reinitialize the setpoints.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

- The “Monitor all” button is selected.
- The parameters marked as a "Setpoint" have a "Start value project" that is free of process-related and syntax errors

Procedure

To initialize all setpoints, follow these steps:

1. Enter the desired values in the "Start value project" column.
   Ensure that the start values are free of process-related and syntax errors.
2. Click the "Initialize setpoints" icon .

Result

The setpoints in the CPU are initialized with the start values from the project.

13.1.2.11 Display instance DB of a technology object.

An instance DB, in which the parameter and static variables are saved, is created for each technology object.

Procedure

To display the instance DB of a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Highlight a technology object.
4. Select the command "Open DB editor" in the shortcut menu.

13.1.3 Using PID_Compact

13.1.3.1 Technology object PID_Compact

The technology object PID_Compact provides a continuous PID controller with integrated optimization. You can alternatively configure a pulse controller. Both manual and automatic mode are possible.

PID-Compact continuously acquires the measured process value within a control loop and compares it with the required setpoint. From the resulting control deviation, the instruction PID_Compact calculates an output value by which the process value is adapted as quickly

and stable as possible to the setpoint. The output value for the PID controller consists of three actions:

- **P** action
  The proportional action of the output value increases in proportion to the control deviation.
- **I** action
  The integral action of the output value increases until the control deviation has been balanced.
- **D** action
  The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID_Compact calculates the proportional, integral and derivative parameters for your controlled system during pretuning. Fine tuning can be used to tune the parameters further. You do not need to manually determine the parameters.

Additional information

- Overview of software controller (Page 7196)
- Add technology objects (Page 7199)
- Configure technology objects (Page 7200)
- Configuring PID_Compact V2 (Page 7227)
- Configuring PID_Compact V1 (Page 7243)

13.1.3.2 PID_Compact V2

Configuring PID_Compact V2

Basic settings

Introduction

Configure the following properties of the "PID_Compact" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)

Setpoint, process value and output value

You can only configure the setpoint, process value and output value in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Controller type

Physical quantity

Select the physical quantity and unit of measurement for setpoint, process value, and disturbance variable in the "Controller type" group. Setpoint, process value, and disturbance variable is displayed in this unit of measurement.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_Compact does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Startup characteristics

1. To switch to "Inactive" mode after CPU restart, clear the "Activate Mode after CPU restart" check box.
   To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.
2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.
   After a complete download to the device, PID_Compact starts in the selected operating mode. With each additional restart, PID_Compact starts in the mode that was last saved in Mode.

Example

You have selected the "Activate Mode after CPU restart" check box and the entry "Pretuning" in the "Set Mode to" list. After a complete download to the device, PID_Compact starts in the

"Pretuning" mode. If pretuning is still active, PID_Compact starts in "Pretuning" mode again after restart of the CPU. If pretuning was successfully completed and automatic mode is active, PID_Compact starts in "Automatic mode" after restart of the CPU.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value

PID_Compact will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Output value

PID_Compact offers three output values. Your actuator will determine which output value you use.

- Output_PER
  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output
  The output value needs to be processed by the user program, for example because of nonlinear actuator response.
- Output_PWM
  The actuator is controlled via a digital output. Pulse width modulation creates minimum ON and minimum OFF times.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output_PER (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to process the output value using the user program:

1. Select the entry "Output" in the drop-down list "Output".
2. Select "Instance DB".
   The calculated output value is saved in the instance data block.
3. For the preparation of the output value, use the output parameter Output.
4. Transfer the processed output value to the actuator via a digital or analog CPU output.

Proceed as follows to use the digital output value:

1. Select the entry "Output_PWM" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the digital output.

Process value settings

Scaling the process value

If you have configured the use of Input_PER in the basic setting, you must convert the value of the analog input to the physical quantity of the process value. The current configuration is displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are stored in the hardware configuration. To use the value pairs from the hardware configuration, follow these steps:

1. Select the PID_Compact instruction in the programming editor.
2. Interconnect Input_PER with an analog input in the basic settings.
3. Click the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

Process value limits

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can configure how PID_Compact reacts to an error in automatic mode in the output value settings.

Advanced settings

Monitoring process value

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_Compact instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98 °C; warning high limit = 90 °C

Warning low limit = 10 °C; process value low limit = 0 °C

PID_Compact will respond as follows:

Process value	InputWarning_H	InputWarning_L	Error-Bits	Operating mode
> 98 °C	TRUE	FALSE	0001h	Inactive or Substitute output value with error monitoring
≤ 98 °C and > 90 °C	TRUE	FALSE	0000h	Automatic mode
≤ 90 °C and ≥ 10 °C	FALSE	FALSE	0000h	Automatic mode
< 10 °C and ≥ 0 °C	FALSE	TRUE	0000h	Automatic mode
< 0 °C	FALSE	TRUE	0001h	Inactive or Substitute output value with error monitoring

In the output value settings, you can specify the reaction of PID_Compact when the process value high limit or low limit is violated.

See also

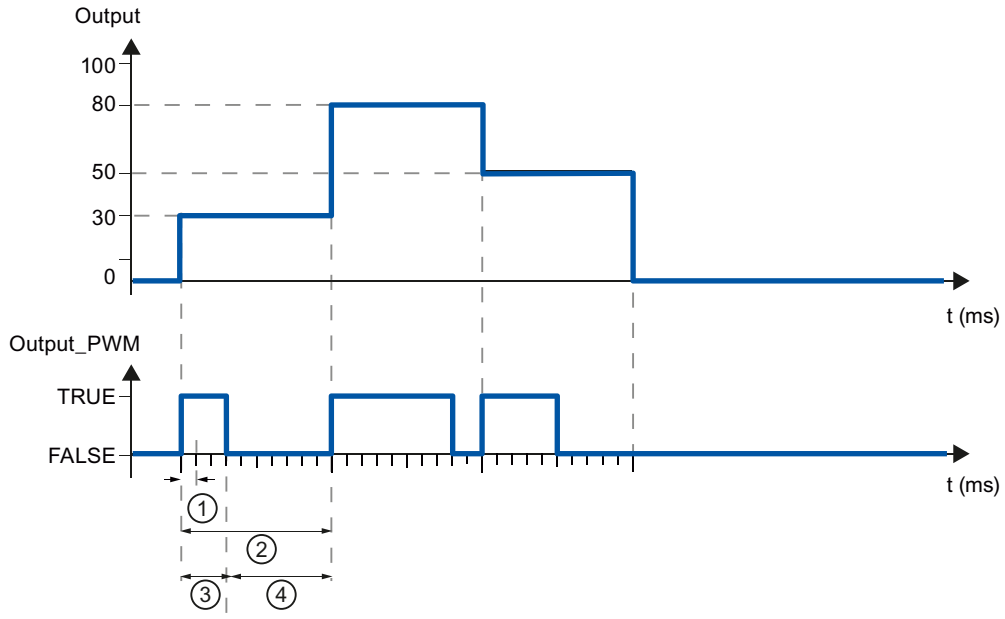
Parameters State and Mode V2 (Page 3462)

PWM limits

The value at the output parameter Output is transformed into a pulse sequence that is output at output parameter Output_PWM by means of a pulse width modulation. Output is calculated in the PID algorithm sampling time, Output_PWM is output in the PID_Compact sampling time.

The PID algorithm sampling time is determined during pretuning or fine tuning. If manually setting the PID parameters, you will also need to configure the PID algorithm sampling time. The PID_Compact sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the value at Output and is always an integer multiple of the PID_Compact sampling time.



① PID_Compact sampling time
② PID algorithm sampling time
③ Pulse duration
④ Break time

The "Minimum ON time" and the "Minimum OFF time" are rounded to an integer multiple of the PID_Compact sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

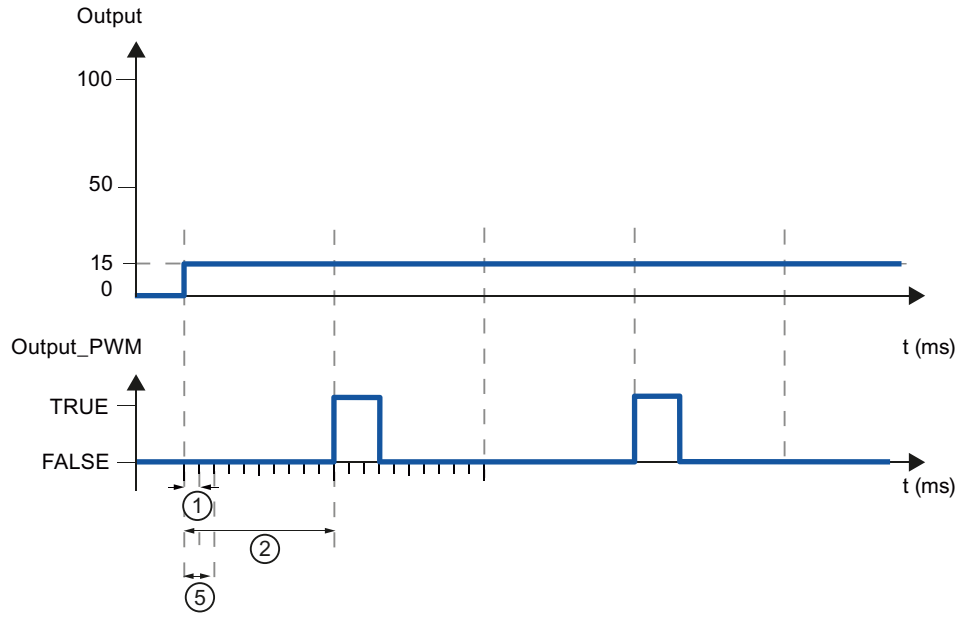
Example

PID_Compact sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

Output is a constant 15%. The smallest pulse that PID_Compact can output is 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.



① PID_Compact sampling time
② PID algorithm sampling time
⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you are using "Output" or "Output_PER", you must configure the value 0.0 for the minimum ON and OFF times.

Note

The minimum ON and OFF times only affect the output parameter Output_PWM and are not used for any pulse generators integrated in the CPU.

Output value

Output value limits

In the "Output value limits" configuration window, configure the absolute limits of your output value in percent. Absolute output value limits are not violated in neither manual mode nor automatic mode. If an output value outside the limits is specified in manual mode, the effective value is limited in the CPU to the configured limits.

The output value limits must match the control logic.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0%
Output_PER	-100.0 to 100.0%
Output_PWM	0.0 to 100.0%

Reaction to error

Notice

Your system may be damaged.

If you output "Current value while error pending " or "Substitute output value while error pending" in the event of an error, PID_Compact remains in automatic mode. This may cause a violation of the process value limits and damage your system.

It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID_Compact is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

PID_Compact generates a programmable output value in response to an error:

- Zero (inactive)
  PID_Compact outputs 0.0 as output value for all errors and switches to "Inactive" mode.
  The controller is only reactivated by a falling edge at Reset or a rising edge at ModeActivate.
- Current value while error is pending
  If the following errors occur in **automatic mode**, PID_Compact returns to automatic mode as soon as the errors are no longer pending.
  If one or more of the following errors occur, PID_Compact stays in automatic mode:
  - 0001h: The "Input" parameter is outside the process value limits.
  - 0800h: Sampling time error
  - 40000h: Invalid value at Disturbance parameter.

  If one or more of the following errors occur in **automatic mode**, PID_Compact switches to "Substitute output value with error monitoring" mode and outputs the last valid output value:
  - 0002h: Invalid value at Input_PER parameter.
  - 0200h: Invalid value at Input parameter.
  - 0400h: Calculation of output value failed.
  - 1000h: Invalid value at Setpoint parameter.

  If an error occurs in **manual mode**, PID_Compact continues using the manual value as the output value. If the manual value is invalid, the substitute output value is used. If the manual value and substitute output value are invalid, the output value low limit is used.
  If the following error occurs during a **pretuning or fine tuning**, PID_Compact remains in active mode:
  - 0020h: Pretuning is not permitted during fine tuning.

  When any other error occurs, PID_Compact cancels the tuning and switches to the mode from which tuning was started.
  As soon as no errors are pending, PID_Compact returns to automatic mode.
- Substitute output value while error is pending
  PID_Compact outputs the substitute output value.
  If the following error occurs, PID_Compact stays in "Substitute output value with error monitoring" mode and outputs the output value low limit:
  - 20000h: Invalid value at SubstituteOutput tag.

  For all other errors, PID_Compact reacts as described for "Current value while error is pending".

See also

Parameters State and Mode V2 (Page 3462)

PID parameters

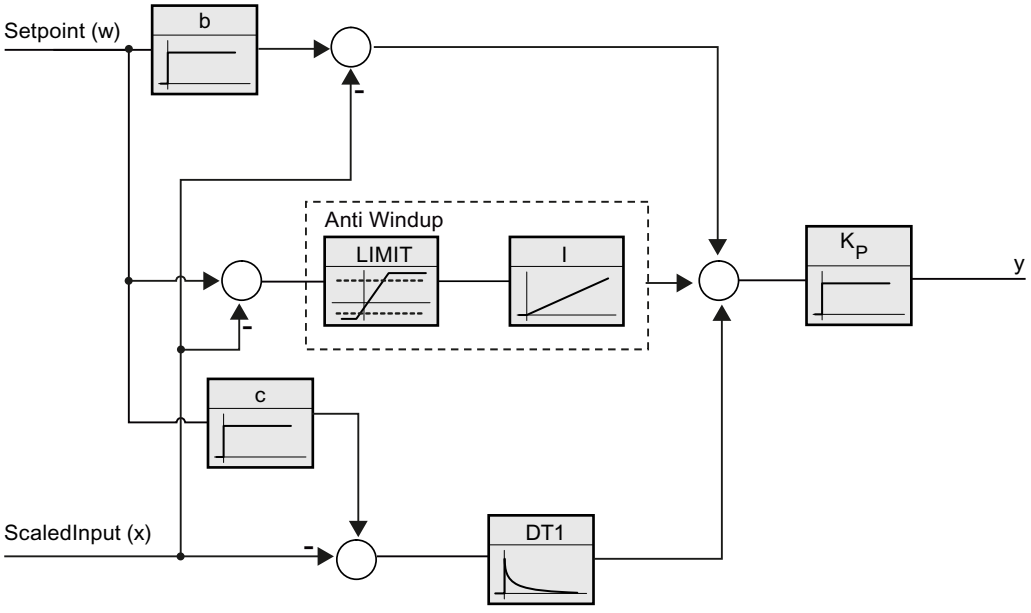
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_Compact.

Downloading technology objects to device (Page 7202)

Proportional gain

The value specifies the proportional gain of the controller. PID_Compact does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with one dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

If you use Output_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The PID algorithm sampling time corresponds to the time period of the pulse width modulation. The cycle time should be at least 10 times the PID algorithm sampling time.

Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list.

- **PID**
  Calculates PID parameters during pretuning and fine tuning.
- **PI**
  Calculates PI parameters during pretuning and fine tuning.
- **User-defined**
  The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program.

Commissioning PID_Compact V2

Pretuning

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The PID parameters are backed up before being recalculated.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- PID_Compact is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".

- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is > 50% of the setpoint.

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_Compact > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list.
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it can be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_Compact responds with the configured reaction to errors.

See also

Parameters State and Mode V2 (Page 3462)

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

Requirement

- The PID_Compact instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The setpoint and the process value lie within the configured limits.
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID_Compact is in one of the following operating modes: Inactive, automatic mode, or manual mode.

Process depends on initial situation

Fine tuning can be started from the following operating modes: "Inactive", "automatic mode", or "manual mode". Fine tuning proceeds as follows when started from:

- Automatic mode
  Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.
  PID_Compact controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact responds with the configured reaction to errors.
  An attempt is made to reach the setpoint with the minimum or maximum output value if the process value for pretuning is already too near the setpoint. This can produce increased overshoot.

Procedure

To perform fine tuning, follow these steps:

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed that tuning is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If no errors occurred during fine tuning, the PID parameters have been tuned. PID_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during "fine tuning", PID_Compact responds with the configured response to errors.

See also

Parameters State and Mode V2 (Page 3462)

"Manual" mode

The following section describes how you can use the "manual mode" operating mode in the commissioning window of the "PID_Compact" technology object. Manual mode is also possible when an error is pending.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established and the CPU is in the "RUN" mode.

Procedure

Use "Manual mode" in the commissioning window if you want to test the controlled system by specifying a manual value. To define a manual value, follow these steps:

1. Click the "Start" icon.
2. Select the "Manual mode" check box in the "Online status of controller" area.
   PID_Compact operates in manual mode. The most recent current output value remains in effect.
3. Enter the manual value in the "Output" field as a % value.
4. Click the icon.

Result

The manual value is written to the CPU and immediately goes into effect.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller. The switchover to automatic mode is bumpless.

See also

Parameters State and Mode V2 (Page 3462)

13.1.3.3 PID_Compact V1

Configuring PID_Compact V1

Basic settings

Introduction

Configure the following properties of the "PID_Compact" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)

Setpoint, process value and output value

You can only configure the setpoint, process value and output value in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Controller type

Physical quantity

Select the unit of measurement and physical quantity for the setpoint and process value in the "Controller type" group. The setpoint and process value will be displayed in this unit.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_Compact does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Start-up behavior after reset

To change straight to the last active mode after restarting the CPU, select the "Enable last mode after CPU restart" check box.

PID_Compact will remain in "Inactive" mode if the check box is cleared.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value

PID_Compact will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Output value

PID_Compact offers three output values. Your actuator will determine which output value you use.

- Output_PER
  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output
  The output value needs to be processed by the user program, for example because of nonlinear actuator response.
- Output_PWM
  The actuator is controlled via a digital output. Pulse width modulation creates minimum ON and minimum OFF times.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output_PER (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to process the output value using the user program:

1. Select the entry "Output" in the drop-down list "Output".
2. Select "Instance DB".
   The calculated output value is saved in the instance data block.
3. For the preparation of the output value, use the output parameter Output.
4. Transfer the processed output value to the actuator via a digital or analog CPU output.

Proceed as follows to use the digital output value:

1. Select the entry "Output_PWM" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the digital output.

Process value settings

Configure the scaling of your process value and specify the process value absolute limits In the "Process value settings" configuration window.

Scaling the process value

If you have configured the use of Input_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration will be displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID_Compact in the programming editor.
2. Connect Input_PER with an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

Monitoring process value

Specify the absolute high and low limit of the process value. As soon as these limits are violated during operation, the controller switches off and the output value is set to 0%. You must enter reasonable limits for your controlled system. Reasonable limits are important during optimization to obtain optimal PID parameters.

The default for the "High limit process value" is 120 %. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "High limit process value". Only a wire-break and a short-circuit are recognized and the PID_Compact switches to "Inactive" mode.

Warning

If you set very high process value limits (for example -3.4*10^{38}...+3.4*10^{38}), process value monitoring will be disabled. Your system may then be damaged if an error occurs.

See also

Monitoring process value (Page 7248)

PWM limits (Page 7248)

Output value limits (Page 7250)

PID parameters (Page 7251)

Advanced settings

Monitoring process value

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_Compact instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_Compact will respond as follows:

Process value	InputWarning_H	InputWarning_L	Operating mode
> 98° C	TRUE	FALSE	Inactive
≤ 98° C and > 90° C	TRUE	FALSE	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	Automatic mode
< 0° C	FALSE	TRUE	Inactive

See also

Process value settings (Page 7244)

PWM limits (Page 7248)

Output value limits (Page 7250)

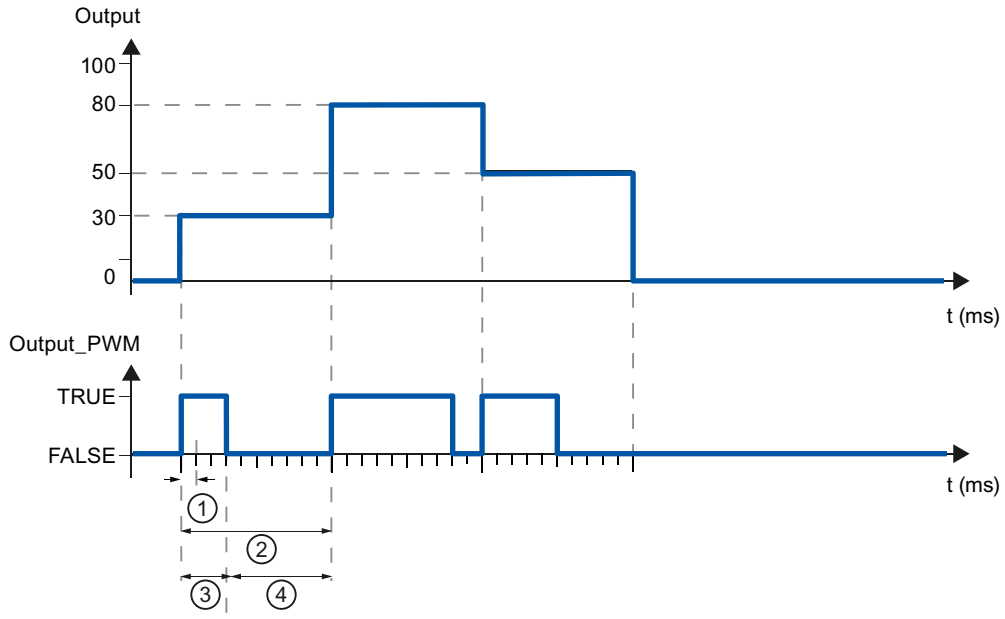
PID parameters (Page 7251)

PWM limits

The value at the output parameter Output is transformed into a pulse sequence that is output at output parameter Output_PWM by means of a pulse width modulation. Output is calculated in the PID algorithm sampling time, Output_PWM is output in the PID_Compact sampling time.

The PID algorithm sampling time is determined during pretuning or fine tuning. If manually setting the PID parameters, you will also need to configure the PID algorithm sampling time. The PID_Compact sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the value at Output and is always an integer multiple of the PID_Compact sampling time.



① PID_Compact sampling time
② PID algorithm sampling time
③ Pulse duration
④ Break time

The "Minimum ON time" and the "Minimum OFF time" are rounded to an integer multiple of the PID_Compact sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

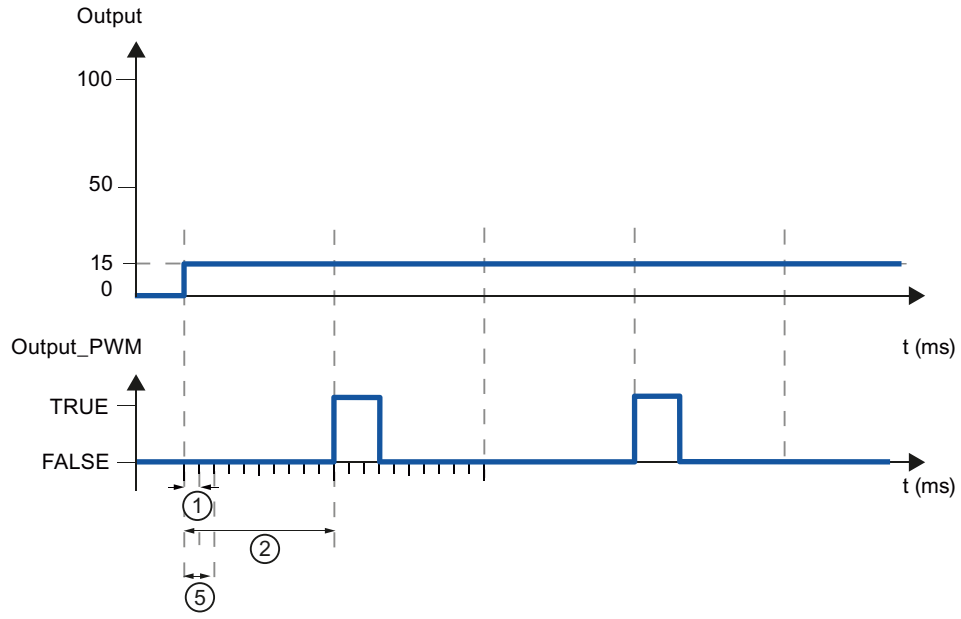
Example

PID_Compact sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

Output is a constant 15%. The smallest pulse that PID_Compact can output is 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.



① PID_Compact sampling time
② PID algorithm sampling time
⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you are using "Output" or "Output_PER", you must configure the value 0.0 for the minimum ON and OFF times.

Note

The minimum ON and OFF times only affect the output parameter Output_PWM and are not used for any pulse generators integrated in the CPU.

See also

Process value settings (Page 7244)

Monitoring process value (Page 7246)

Output value limits (Page 7250)

PID parameters (Page 7251)

Output value limits

In the "Output value limits" configuration window, configure the absolute limits of your output value in percent. Absolute output value limits are not violated in neither manual mode nor in automatic mode. If a output value outside the limits is specified in manual mode, the effective value is limited in the CPU to the configured limits.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0
Output_PER	-100.0 to 100.0
Output_PWM	0.0 to 100.0

PID_Compact sets the output value to 0.0 if an error occurs. 0.0 must therefore always be within the output value limits. You will need to add an offset to Output and Output_PER in the user program if you want an output value low limit of greater than 0.0.

See also

Process value settings (Page 7244)

Monitoring process value (Page 7246)

PWM limits (Page 7246)

PID parameters (Page 7251)

PID parameters

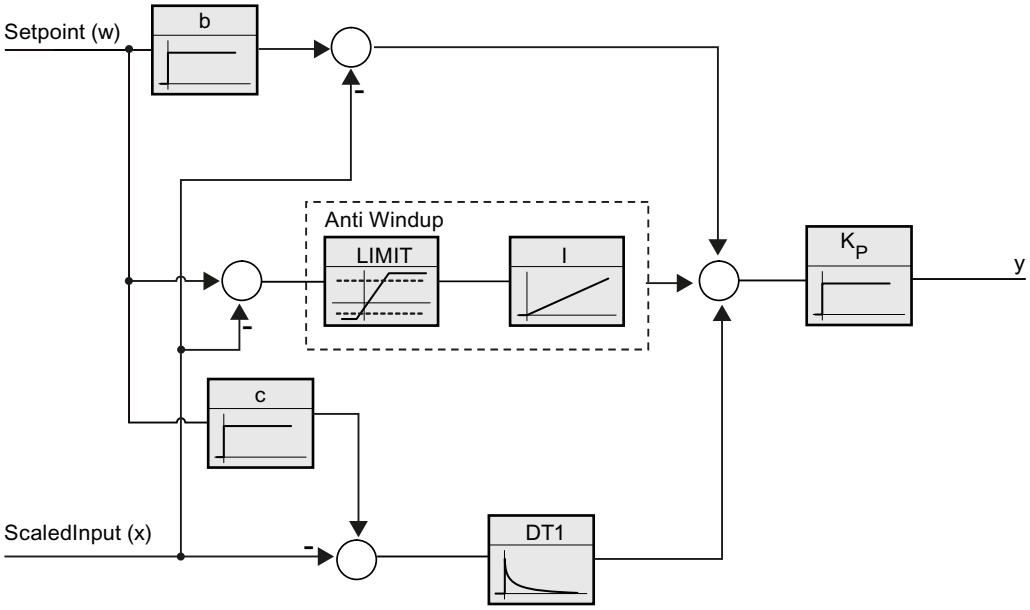
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_Compact.

Auto-Hotspot

Proportional gain

The value specifies the proportional gain of the controller. PID_Compact does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

If you use Output_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The PID algorithm sampling time corresponds to the time period of the pulse width modulation. The cycle time should be at least 10 times the PID algorithm sampling time.

Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list.

- **PID**
  Calculates PID parameters during pretuning and fine tuning.
- **PI**
  Calculates PI parameters during pretuning and fine tuning.
- **User-defined**
  The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program.

See also

Downloading technology objects to device (Page 7202)

Commissioning PID_Compact V1

Commissioning

The commissioning window helps you commission the PID controller. You can monitor the values for the setpoint, process value and output value along the time axis in the trend view. The following functions are supported in the commissioning window:

- Controller pretuning
- Controller fine tuning
  Use fine tuning for fine adjustments to the PID parameters.
- Monitoring the current closed-loop control in the trend view
- Testing the controlled system by specifying a manual output value

All functions require an online connection to the CPU to have been established.

Basic handling

- Select the desired sampling time in the "Sampling time" drop-down list.
  All values in the commissioning window are updated in the selected update time.
- Click the "Start" icon in the measuring group if you want to use the commissioning functions.
  Value recording is started. The current values for the setpoint, process value and output value are entered in the trend view. Operation of the commissioning window is enabled.
- Click the "Stop" icon if you want to end the commissioning functions.
  The values recorded in the trend view can continue to be analyzed.

Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

See also

Pretuning (Page 7254)

Fine tuning (Page 7256)

"Manual" mode (Page 7257)

Pretuning

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The PID parameters are backed up before being recalculated.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- PID_Compact is in "inactive" or "manual" mode.
- The setpoint may not be changed during controller tuning. PID_Compact will otherwise be deactivated.
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is > 50% of the setpoint.

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_Compact > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list.
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_Compact will change to "Inactive" mode.

See also

Parameters State and sRet.i_Mode V1 (Page 3479)

Commissioning (Page 7252)

Fine tuning (Page 7256)

"Manual" mode (Page 7257)

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.

PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

Requirement

- The PID_Compact instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- The setpoint may not be changed during controller tuning.
- PID_Compact is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning can be started in "inactive", "automatic" or "manual" mode. Fine tuning proceeds as follows when started in:

- Automatic mode
  Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.
  PID_Comact will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  If the requirements for pretuning are met, pretuning is started. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode.
  An attempt is made to reach the setpoint with a minimum or maximum output value if the process value for pretuning is already too near the setpoint. This can produce increased overshoot.

Procedure

Proceed as follows to carry out "fine tuning":

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

The PID parameters will have been optimized if fine tuning has been executed without errors. PID_Compact changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during "fine tuning", PID_Compact will change to "inactive" mode.

See also

Parameters State and sRet.i_Mode V1 (Page 3479)

Commissioning (Page 7252)

Pretuning (Page 7252)

"Manual" mode (Page 7257)

"Manual" mode

The following section describes how you can use the "Manual" operating mode in the commissioning window of the "PID Compact" technology object.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established and the CPU is in the "RUN" mode.
- The functions of the commissioning window have been enabled via the "Start" icon.

Procedure

Use "Manual mode" in the commissioning window if you want to test the process by specifying a manual value. To define a manual value, proceed as follows:

1. Select the check box "Manual mode" in the "Online status of the controller" area. PID_Compact operates in manual mode. The most recent current output value remains in effect.
2. Enter the manual value in the "Output" field as a % value.
3. Click the control icon .

Result

The manual value is written to the CPU and immediately goes into effect.

Note

PID_Compact continues to monitor the process value. If the process value limits are exceeded, PID_Compact is deactivated.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller. The change to automatic mode is bumpless.

See also

Parameters State and sRet.i_Mode V1 (Page 3479)

Commissioning (Page 7252)

Pretuning (Page 7252)

Fine tuning (Page 7254)

13.1.4 Using PID_3Step

13.1.4.1 Technology object PID_3Step

The technology object PID_3Step provides a PID controller with tuning for valves or actuators with integral response.

You can configure the following controllers:

- Three-point step controller with position feedback
- Three-point step controller without position feedback
- Valve controller with analog output value

PID_3Step continuously acquires the measured process value within a control loop and compares it with the setpoint. From the resulting control deviation, PID_3Step calculates an

output value through which the process value reaches the setpoint as quickly and steadily as possible. The output value for the PID controller consists of three actions:

- **P** action
  The proportional action of the output value increases in proportion to the control deviation.
- **I** action
  The integral action of the output value increases until the control deviation has been balanced.
- **D** action
  The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID_3Step calculates the proportional, integral and derivative parameters for your controlled system during pretuning. Fine tuning can be used to tune the parameters further. You do not need to manually determine the parameters.

Additional information

- Overview of software controller (Page 7196)
- Add technology objects (Page 7199)
- Configure technology objects (Page 7200)
- Configuring PID_3Step V2 (Page 7259)
- Configuring PID_3Step V1 (Page 7276)

13.1.4.2 PID_3Step V2

Configuring PID_3Step V2

Basic settings

Introduction

Configure the following properties of the "PID_3Step" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)
- Position feedback (only in the Inspector window)

Setpoint, process value, output value and position feedback

You can only configure the setpoint, process value, output value and position feedback in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Controller type

Physical quantity

Select the physical quantity and unit of measurement for setpoint, process value, and disturbance variable in the "Controller type" group. Setpoint, process value, and disturbance variable is displayed in this unit of measurement.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_3Step does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Startup characteristics

1. To switch to "Inactive" mode after CPU restart, clear the "Activate Mode after CPU restart" check box.
   To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.
2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.
   After a complete download to the device, PID_3Step starts in the selected operating mode. With each additional restart, PID_3Step starts in the mode that was last saved in Mode.

Example

You have selected the "Activate Mode after CPU restart" check box and the entry "Pretuning" in the "Set Mode to" list. After a complete download to the device, PID_3Step starts in the "Pretuning" mode. If pretuning is still active, PID_3Step starts in "Pretuning" mode again after

restart of the CPU. If pretuning was successfully completed and automatic mode is active, PID_3Step starts in "Automatic mode" after restart of the CPU.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value

PID_3Step will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Position feedback

Position feedback configuration depends upon the actuator used.

- Actuator without position feedback
- Actuator with digital endstop signals

- Actuator with analog position feedback
- Actuator with analog position feedback and endstop signals

Actuator without position feedback

Proceed as follows to configure PID_3Step for an actuator without position feedback:

1. Select the entry "No Feedback" in the drop-down list "Feedback".

Actuator with digital endstop signals

Proceed as follows to configure PID_3Step for an actuator with endstop signals:

1. Select the entry "No Feedback" in the drop-down list "Feedback".
2. Activate the "Actuator endstop signals" check box.
3. Select "Instruction" as source for Actuator_H and Actuator_L.
4. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Actuator with analog position feedback

Proceed as follows to configure PID_3Step for an actuator with analog position feedback:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
   - Use the analog input value for Feedback_PER. Configure Feedback_PER scaling in the actuator settings.
   - Process the analog input value for Feedback using your user program.
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.

Actuator with analog position feedback and endstop signals

Proceed as follows to configure PID_3Step for an actuator with analog position feedback and endstop signals:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.
4. Activate the "Actuator endstop signals" check box.
5. Select "Instruction" as source for Actuator_H and Actuator_L.
6. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Output value

PID_3Step offers an analog output value (Output_PER) and digital output values (Output_UP, Output_DN). Your actuator will determine which output value you use.

- Output_PER
  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output_UP, Output_DN
  The actuator is controlled via two digital outputs.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the digital output value:

1. Select the entry "Output (digital)" in the drop-down list "Output".
2. Select "Instruction" for Output_UP and Output_DN.
3. Enter the addresses of the digital outputs.

Proceed as follows to process the output value using the user program:

1. Select the entry corresponding to the actuator in the drop-down list "Output".
2. Select "Instruction".
3. Enter the name of the variable you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.

Process value settings

Scaling the process value

If you have configured the use of Input_PER in the basic setting, you must convert the value of the analog input to the physical quantity of the process value. The current configuration is displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" text boxs.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are stored in the hardware configuration. To use the value pairs from the hardware configuration, follow these steps:

1. Select the PID_3Step instruction in the programming editor.
2. Interconnect Input_PER with an analog input in the basic settings.
3. Click the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

Process value limits

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can specify how PID_3Step responds to errors in automatic mode in the actuator settings.

Actuator settings

Actuator

Actuator-specific times

Configure the motor transition time and the minimum ON and OFF times to prevent damage to the actuator. You can find the specifications in the actuator data sheet.

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. You can measure the motor transition time during commissioning.

The motor transition time is retentive. If you enter the motor transition time manually, you must completely download PID_3Step.

Downloading technology objects to device (Page 7202)

If you are using "Output_UP" or "Output_DN", you can reduce the switching frequency with the minimum on and minimum OFF time.

The on or off times calculated are totaled in automatic mode and only become effective when the sum is greater than or equal to the minimum on or OFF time.

Manual_UP = TRUE or Manual_DN = TRUE in manual mode operates the actuator for at least the minimum ON or OFF time.

Reaction to error

PID_3Step is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

Notice

Your system may be damaged.

If you output "Current value while error pending" or "Substitute output value while error pending" in the event of an error, PID_3Step remains in automatic mode even if the process value limits are violated. This may damage your system.

It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID_3Step generates a programmable output value in response to an error:

- Current value
  PID_3Step is switched off and no longer modifies the actuator position.
- Current value for error while error is pending
  The controller functions of PID_3Step are switched off and the position of the actuator is no longer changed.
  If the following errors occur in automatic mode, PID_3Step returns to automatic mode as soon as the errors are no longer pending.
  - 0002h: Invalid value at Input_PER parameter.
  - 0200h: Invalid value at Input parameter.
  - 0400h: Calculation of output value failed.
  - 1000h: Invalid value at Setpoint parameter.
  - 2000h: Invalid value at Feedback_PER parameter.
  - 4000h: Invalid value at Feedback parameter.
  - 8000h: Error during digital position feedback.
  - 20000h: Invalid value at SavePosition tag.

  If one or more of the following errors occur, PID_3Step stays in automatic mode:
  - 0001h: The Input parameter is outside the process value limits.
  - 0800h: Sampling time error
  - 40000h: Invalid value at Disturbance parameter.

  PID_3Step remains in manual mode if an error occurs in manual mode.
  If an error occurs during tuning or transition time measurement, PID_3Step switches to the mode in which tuning or transition time measurement was started. Only in the event of the following error is tuning not aborted:
  - 0020h: Pretuning is not permitted during fine tuning.

- Substitute output value
  PID_3Step moves the actuator to the substitute output value and then switches off.
- Substitute output value while error is pending
  PID_3Step moves the actuator to the substitute output value. When the substitute output value is reached, PID_3Step reacts as it does with "Current value for while error is pending".

Enter the substitute output value in "%".

Only substitute output values 0% and 100% can be approached precisely in the case of actuators without analog position feedback. A substitute output value not equal to 0% or 100% is approached via an internally simulated position feedback. This procedure does not, however, allow the exact approach of substitute output value.

All substitute output values can be approached precisely with actuators with analog position feedback.

Scaling position feedback

Scaling position feedback

If you have configured the use of Feedback_PER in the basic settings, you will need to convert the value of the analog input into %. The current configuration will be displayed in the "Feedback" display.

Feedback_PER is scaled using a low and high value pair.

1. Enter the low pair of values in the "Low endstop" and "Low" input boxes.
2. Enter the high pair of values in the "High endstop" and "High" input boxes.

"Low endstop" must be less than "High endstop"; "Low" must be less than "High".

The valid values for "High endstop" and "Low endstop" depend upon:

- No Feedback, Feedback, Feedback_PER
- Output (analog), Output (digital)

Output	Feedback	Low endstop	High endstop
Output (digital)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (digital)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (digital)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (analog)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%

Output value limits

Limiting the output value

You can exceed or undershoot the output value limits during the transition time measurement and with mode = 10. The output value is limited to these values in all other modes.

Enter the absolute output value limits in the "Output value high limit" and "Output value low limit" input boxes. The output value limits must be within "Low endstop" and "High endstop".

If no Feedback is available and Output (digital) is set, you cannot limit the output value. Output_UP and Output_DN are then reset upon Actuator_H = TRUE or Actuator_L = TRUE. If no endstop signals are available, Output_UP and Output_DN are reset after a travel time of 150% of the motor actuating time.

Advanced settings

Monitoring process value

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_3Step instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_3Step will respond as follows:

Process value	InputWarning_H	InputWarning_L	Error-Bits	Operating mode
> 98° C	TRUE	FALSE	0001h	As configured
≤ 98° C and > 90° C	TRUE	FALSE	0000h	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	0000h	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	0000h	Automatic mode
< 0° C	FALSE	TRUE	0001h	As configured

In the actuator settings, you can configure the response of PID_3Step when the process value high limit or low limit is violated.

PID parameters

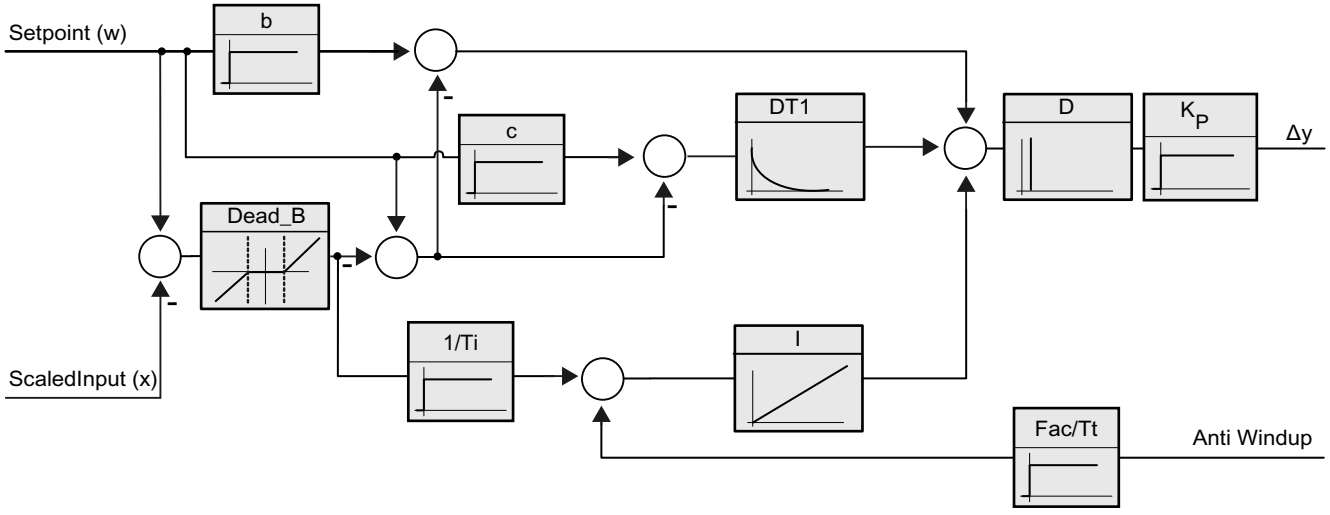
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
Δy	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_3Step.

Downloading technology objects to device (Page 7202)

Proportional gain

The value specifies the proportional gain of the controller. PID_3Step does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the PID_3Step sampling time. All other functions of PID_3Step are executed at every call.

Deadband width

The deadband suppresses the noise component in the steady controller state. The deadband width specifies the size of the deadband. The deadband is off if the deadband width is 0.0.

Commissioning PID_3Step V2

Pretuning

The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The PID parameters are backed up before being recalculated.

The setpoint is frozen during pretuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.
- PID_3Step is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_3Step > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list in the working area "Tuning".
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_3Step responds with the configured reaction to errors.

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

The setpoint is frozen during fine tuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.

- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID_3Step is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning proceeds as follows when started from:

- Automatic mode
  Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.
  PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  Pretuning is always started first. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

Procedure

To perform fine tuning, follow these steps:

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   – An online connection will be established.
   – Value recording is started.
   – The process of fine tuning is started.
   – The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If no errors occurred during fine tuning, the PID parameters have been tuned. PID_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID_3Step responds with the configured response to errors.

Commissioning with manual PID parameters

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.
- PID_3Step is in "inactive" mode.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

Procedure

Proceed as follows to commission PID_3Step with manual PID parameters:

1. Double-click on "PID_3Step > Configuration" in the project tree.
2. Click on "Advanced settings > PID Parameters" in the configuration window.
3. Select the check box "Enable direct input".
4. Enter the PID parameters.
5. Double-click the "PID_3Step > Commissioning" entry in the project tree.
6. Establish an online connection to the CPU.
7. Load the PID parameters to the CPU.
8. Click the "Start PID_3Step" icon.

Result

PID_3Step changes to automatic mode and controls using the current PID parameters.

See also

PID parameters (Page 7265)

Measuring the motor transition time

Introduction

PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

Requirement

- Feedback or Feedback_PER has been selected in the basic settings and the signal has been connected.
- An online connection to the CPU has been established.

1. Select the "Use position feedback" check box.
2. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. The difference between "Target position" and "Position feedback" must be at least 50% of the valid output value range.
3. Click the "Start" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. When the transition time measurement is ended and ActivateRecoverMode = TRUE, PID_3Step switches to the operating mode from which the

transition time measurement was started. If the transition time measurement is ended and ActivateRecoverMode = FALSE, PID_3Step changes to "Inactive" mode.

Note

Click on the icon "Upload measured transition time" to load the motor transition time measured to the project.

Actuators with endstop signals

Proceed as follows to measure the transition time of actuators with endstop signals:

Requirement

- The "Endstop signals" check box in the basic settings has been selected and Actuator_H and Actuator_L are connected.
- An online connection to the CPU has been established.

Proceed as follows to measure motor transition time with endstop signals:

1. Select the "Use actuator endstop signals" check box.
2. Select the direction in which the actuator is to be moved.
   - Open - Close - Open
     The actuator is moved first to the high endstop, then to the low endstop and then back to the high endstop.
   - Close - Open - Close
     The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop.
3. Click the "Start" icon.

Result

The actuator is moved in the selected direction. Time measurement will start once the actuator has reached the first endstop and will end when the actuator reaches this endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. When the transition time measurement is ended and ActivateRecoverMode = TRUE, PID_3Step switches to the operating mode from which the transition time measurement was started. If the transition time measurement is ended and ActivateRecoverMode = FALSE, PID_3Step changes to "Inactive" mode.

Cancelling transition time measurement

PID_3Step switches to "Inactive" mode if you cancel transition time measurement by pressing the Stop button.

13.1.4.3 PID_3Step V1

Configuring PID_3Step V1

Basic settings

Introduction

Configure the following properties of the "PID_3Step" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)
- Position feedback (only in the Inspector window)

Setpoint, process value, output value and position feedback

You can only configure the setpoint, process value, output value and position feedback in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Controller type

Physical quantity

Select the unit of measurement and physical quantity for the setpoint and process value in the "Controller type" group. The setpoint and process value will be displayed in this unit.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_3Step does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Start-up behavior after reset

To change straight to the last active mode after restarting the CPU, select the "Enable last mode after CPU restart" check box.

PID_3Step will remain in "Inactive" mode if the check box is cleared.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value

PID_3Step will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Position feedback

Position feedback configuration depends upon the actuator used.

- Actuator without position feedback
- Actuator with digital endstop signals
- Actuator with analog position feedback
- Actuator with analog position feedback and endstop signals

Actuator without position feedback

Proceed as follows to configure PID_3Step for an actuator without position feedback:

1. Select the entry "No Feedback" in the drop-down list "Feedback".

Actuator with digital endstop signals

Proceed as follows to configure PID_3Step for an actuator with endstop signals:

1. Select the entry "No Feedback" in the drop-down list "Feedback".
2. Activate the "Actuator endstop signals" check box.
3. Select "Instruction" as source for Actuator_H and Actuator_L.
4. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Actuator with analog position feedback

Proceed as follows to configure PID_3Step for an actuator with analog position feedback:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
   - Use the analog input value for Feedback_PER. Configure Feedback_PER scaling in the actuator settings.
   - Process the analog input value for Feedback using your user program.
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.

Actuator with analog position feedback and endstop signals

Proceed as follows to configure PID_3Step for an actuator with analog position feedback and endstop signals:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.
4. Activate the "Actuator endstop signals" check box.
5. Select "Instruction" as source for Actuator_H and Actuator_L.
6. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Output value

PID_3Step offers an analog output value (Output_PER) and digital output values (Output_UP, Output_DN). Your actuator will determine which output value you use.

- Output_PER
  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output_UP, Output_DN
  The actuator is controlled via two digital outputs.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the digital output value:

1. Select the entry "Output (digital)" in the drop-down list "Output".
2. Select "Instruction" for Output_UP and Output_DN.
3. Enter the addresses of the digital outputs.

Proceed as follows to process the output value using the user program:

1. Select the entry corresponding to the actuator in the drop-down list "Output".
2. Select "Instruction".
3. Enter the name of the variable you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.

Process value settings

Configure the scaling of your process value and specify the process value absolute limits In the "Process value settings" configuration window.

Scaling the process value

If you have configured the use of Input_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration will be displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID_3Step in the programming editor.
2. Connect Input_PER to an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.
   The existing values will be overwritten with the values from the hardware configuration.

Monitoring process value

Specify the absolute high and low limit of the process value. You must enter reasonable limits for your controlled system. Reasonable limits are important during optimization to obtain optimal PID parameters. The default for the "High limit process value" is 120 %. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). This setting ensures that an error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error.

Notice

Your system may be damaged.

If you set very high process value limits (for example $-3.4*10^{38}...+3.4*10^{38}$), process value monitoring will be disabled. Your system may then be damaged if an error occurs. You need to configure useful process value limits for your controlled system.

Actuator settings

Actuator-specific times

Configure the motor transition time and the minimum ON and OFF times to prevent damage to the actuator. You can find the specifications in the actuator data sheet.

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The maximum time that the actuator is moved in one direction is 110% of the motor transition time. You can measure the motor transition time during commissioning.

If you are using "Output_UP" or "Output_DN", you can reduce the switching frequency with the minimum on and minimum OFF time.

The on or off times calculated are totaled in automatic mode and only become effective when the sum is greater than or equal to the minimum on or OFF time.

A rising edge at Manual_UP or Manual_DN in manual mode will operate the actuator for at least the minimum on or OFF time.

Reaction to error

PID_3Step is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

PID_3Step generates a programmable output value in response to an error:

- Current value
  PID_3Step is switched off and no longer modifies the actuator position.
- Current value for error while error is pending
  The controller functions of PID_3Step are switched off and the position of the actuator is no longer changed.
  If the following errors occur in automatic mode, PID_3Step returns to automatic mode as soon as the errors are no longer pending.
  - 0002h: Invalid value at Input_PER parameter.
  - 0200h: Invalid value at Input parameter.
  - 0800h: Sampling time error
  - 1000h: Invalid value at Setpoint parameter.
  - 2000h: Invalid value at Feedback_PER parameter.
  - 4000h: Invalid value at Feedback parameter.
  - 8000h: Error during digital position feedback.

  If one of these error occurs in manual mode, PID_3Step remains in manual mode.
  If an error occurs during the tuning or transition time measurement, PID_3Step is switched off.
- Substitute output value
  PID_3Step moves the actuator to the substitute output value and then switches off.
- Substitute output value while error is pending
  PID_3Step moves the actuator to the substitute output value. When the substitute output value is reached, PID_3Step reacts as it does with "Current value for while error is pending".

Enter the substitute output value in "%".

Only substitute output values 0% and 100% can be approached precisely in the case of actuators without analog position feedback. The actuator is moved in one direction at 110%

of the motor transition time to ensure the high or low endstop is reached. There endstop signals take priority. A substitute output value not equal to 0% or 100% is approached via an internally simulated position feedback. This procedure does not, however, allow the exact approach of substitute output value.

All substitute output values can be approached precisely with actuators with analog position feedback.

Scaling position feedback

If you have configured the use of Feedback_PER in the basic settings, you will need to convert the value of the analog input into %. The current configuration will be displayed in the "Feedback" display.

Feedback_PER is scaled using a low and high value pair.

1. Enter the low pair of values in the "Low endstop" and "Low" input boxes.
2. Enter the high pair of values in the "High endstop" and "High" input boxes.

"Low endstop" must be less than "High endstop"; "Low" must be less than "High".

The valid values for "High endstop" and "Low endstop" depend upon:

- No Feedback, Feedback, Feedback_PER
- Output (analog), Output (digital)

Output	Feedback	Low endstop	High endstop
Output (digital)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (digital)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (digital)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (analog)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%

Limiting the output value

You can only exceed or undershoot the output value limits during the transition time measurement. The output value is limited to these values in all other modes.

Enter the absolute output value limits in the "Output value high limit" and "Output value low limit" input boxes. The output value limits must be within "Low endstop" and "High endstop".

If no Feedback is available and Output (digital) is set, you cannot limit the output value. The digital outputs are reset with Actuator_H = TRUE or Actuator_L = TRUE, or after a travel time amounting to 110% of the motor transition time.

Advanced settings

Monitoring process value

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_3Step instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_3Step will respond as follows:

Process value	InputWarning_H	InputWarning_L	Operating mode
> 98° C	TRUE	FALSE	Inactive
≤ 98° C and > 90° C	TRUE	FALSE	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	Automatic mode
< 0° C	FALSE	TRUE	Inactive

PID parameters

The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

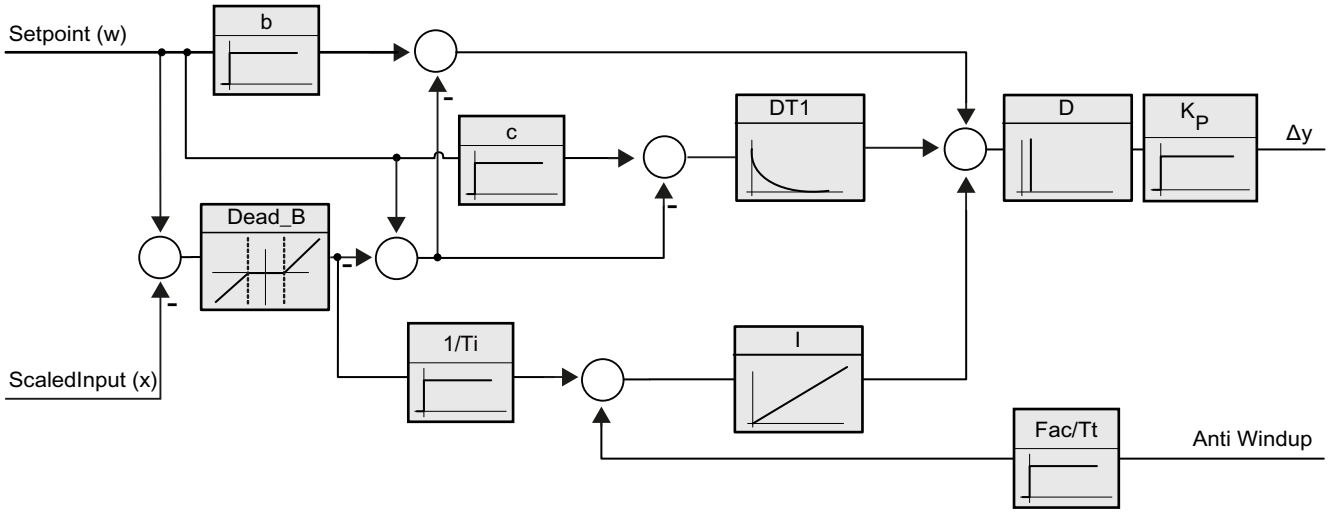
The PID algorithm operates according to the following equation:

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
Δy	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting

w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_3Step.

Auto-Hotspot

Proportional gain

The value specifies the proportional gain of the controller. PID_3Step does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the PID_3Step sampling time. All other functions of PID_3Step are executed at every call.

Deadband width

The deadband suppresses the noise component in the steady controller state. The deadband width specifies the size of the deadband. The deadband is off if the deadband width is 0.0.

See also

Downloading technology objects to device (Page 7202)

Commissioning PID_3Step V1

Commissioning

You can monitor the setpoint, process value and output value over time in the "Tuning" working area. The following commissioning functions are supported in the curve plotter:

- Controller pretuning
- Controller fine tuning
- Monitoring the current closed-loop control in the trend view

All functions require an online connection to the CPU to have been established.

Basic handling

- Select the desired sampling time in the "Sampling time" drop-down list.
  All values in the tuning working area are updated in the selected update time.
- Click the "Start" icon in the measuring group if you want to use the commissioning functions.
  Value recording is started. The current values for the setpoint, process value and output value are entered in the trend view. Operation of the commissioning window is enabled.
- Click the "Stop" icon if you want to end the commissioning functions.
  The values recorded in the trend view can continue to be analyzed.
- Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

Pretuning

The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The PID parameters are backed up before being recalculated.

The setpoint is frozen during pretuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- PID_3Step is in "inactive" or "manual" mode.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_3Step > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list in the working area "Tuning".
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_3Step changes to "Inactive" mode.

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.

PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

The setpoint is frozen during fine tuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- The motor transition time has been configured or measured.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID_3Step is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning proceeds as follows when started in:

- Automatic mode
  Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.
  PID_3Step will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  Pretuning is always started first. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

Procedure

Proceed as follows to carry out "fine tuning":

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

The PID parameters will have been optimized if fine tuning has been executed without errors. PID_3Step changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID_3Step will change to "inactive" mode.

Commissioning with manual PID parameters

Procedure

Proceed as follows to commission PID_3Step with manual PID parameters:

1. Double-click on "PID_3Step > Configuration" in the project tree.
2. Click on "Advanced settings > PID Parameters" in the configuration window.
3. Select the check box "Enable direct input".
4. Enter the PID parameters.
5. Double-click on "PID_3Step > Commissioning" in the project tree.
6. Establish an online connection to the CPU.
7. Load the PID parameters to the CPU.
8. Click on the "Activate controller" icon.

Result

PID_3Step changes to automatic mode and controls using the current PID parameters.

Measuring the motor transition time

Introduction

PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

Requirement

- Feedback or Feedback_PER has been selected in the basic settings and the signal has been connected.
- An online connection to the CPU has been established.

1. Select the "Use position feedback" check box.
2. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. The difference between "Target position" and "Position feedback" must be at least 50% of the valid output value range.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. PID_3Step will change to "Inactive" mode once transition time measurement is complete.

Note

Click on the icon "Upload measured transition time" to load the motor transition time measured to the project.

Actuators with endstop signals

Proceed as follows to measure the transition time of actuators with endstop signals:

Requirement

- The "Endstop signals" check box in the basic settings has been selected and Actuator_H and Actuator_L are connected.
- An online connection to the CPU has been established.

Proceed as follows to measure motor transition time with endstop signals:

1. Select the "Use actuator endstop signals" check box.
2. Select the direction in which the actuator is to be moved.
   - Open - Close - Open
     The actuator is moved first to the high endstop, then to the low endstop and then back to the high endstop.
   - Close - Open - Close
     The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved in the selected direction. Time measurement will start once the actuator has reached the first endstop and will end when the actuator reaches this endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. PID_3Step will change to "Inactive" mode once transition time measurement is complete.

Cancelling transition time measurement

PID_3Step will change to "Inactive" mode immediately if you cancel transition time measurement. The actuator will stop being moved. You can reactive PID-3Step in the curve plotter.

13.1.5 Using PID_Temp

13.1.5.1 Technology object PID_Temp

The PID_Temp technology object provides a continuous PID controller with integrated tuning. PID_Temp is especially designed for temperature control and is suited for heating or heating/cooling applications. Two outputs are available for this purpose, one each for heating and cooling. PID_Temp can also be used for other control tasks. PID_Temp is cascadable and can be used in manual or automatic mode.

PID_Temp continuously acquires the measured process value within a control loop and compares it with the set setpoint. From the resulting control deviations, the instruction PID_Temp calculates the output value for heating and/or cooling which is used to adjust the process value to the setpoint. The output values for the PID controller consist of three actions:

- Proportional action
  The proportional action of the output value increases in proportion to the control deviation.
- Integral action
  The integral action of the output value increases until the control deviation has been balanced.
- Derivative action
  The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID_Temp calculates the proportional, integral and derivative parameters for your controlled system during "pretuning". "Fine tuning" can be used to tune the parameters further. You do not need to manually determine the parameters.

Either a fixed cooling factor or two PID parameter sets can be used for heating-and-cooling applications.

Additional information

- Overview of software controller (Page 7196)
- Add technology objects (Page 7199)
- Configure technology objects (Page 7200)
- Configuring PID_Temp (Page 7292)

13.1.5.2 Configuring PID_Temp

Basic settings

Introduction

Configure the following properties of the "PID_Temp" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Start-up behavior after reset
- Source and input of the setpoint (only in the Inspector window)
- Selection of the process value
- Source and input of the process value (only in the Inspector window)
- Selection of the heating output value
- Source and input of the heating output value (only in the Inspector window)
- Activation and selection of the cooling output value
- Source and input of the cooling output value (only in the Inspector window)
- Activation of PID_Temp as master or slave of a cascade
- Number of slaves
- Selection of the master (only in the Inspector window)

Setpoint, process value, heating output value and cooling output value

You can select the source and enter values or tags for the setpoint, process value, heating output value and cooling output value in the Inspector window of the programming editor.

Select the source for each value:

- Instance DB:
  The value saved in the instance DB is used. The value must be updated by the user program in the instance DB. There should be no value at the instruction. Can be changed using HMI.
- Instruction:
  The value connected to the instruction is used. The value is written to the instance DB each time the instruction is called. Cannot be changed using HMI.

Controller type

Physical quantity

Select the unit of measurement and physical quantity for the setpoint and the process value in the "Controller type" group. The setpoint and the process value are displayed in this unit.

Startup characteristics

1. To switch to "Inactive"mode after CPU restart, clear the "Activate Mode after CPU restart"check box.
   To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.
2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.
   After a complete "Download to device", PID_Temp starts in the selected operating mode. With each additional restart, PID_Temp starts in the mode that was last saved in Mode. When selecting pretuning or fine tuning, you also have to set or reset the Heat.EnableTuning and Cool.EnableTuning tags in order to choose between tuning for heating and tuning for cooling.

Example:

You have selected the "Activate Mode after CPU restart" check box and the "Pretuning" entry in the "Set Mode to" list. After a complete "Download to device", PID_Temp starts in the "Pretuning" mode. If pretuning is still active, PID_Temp starts in "Pretuning" mode again after restart of the CPU (heating/cooling depends on the tags Heat.EnableTuning and Cool.EnableCooling). If pretuning was successfully completed and automatic mode is active, PID_Temp starts in "Automatic mode" after restart of the CPU.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL tag in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL tag is possible, for example for the time-controlled change of the setpoint.

Process value

PID_Temp will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Heating and cooling output value

The PID_Temp instruction provides a PID controller with integrated tuning for temperature processes. PID_Temp is suitable for heating or heating-and-cooling applications.

PID_Temp provides the following output values. Your actuator will determine which output value you use.

- OutputHeat
  Heating output value (floating-point format): The output value for heating needs to be processed by the user program, for example, because of non-linear actuator response.
- OutputHeat_PER
  Analog heating output value: The actuator for heating is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V, 4...20 mA.
- OutputHeat_PWM
  Pulse-width modulated heating output value: The actuator for heating is controlled via a digital output. Pulse width modulation creates variable ON and OFF times.
- OutputCool
  Cooling output value (floating-point format): The output value for cooling needs to be processed by the user program, for example because of non-linear actuator response.
- OutputCool_PER
  Analog cooling output value: The actuator for cooling is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V, 4...20 mA.
- OutputCool_PWM
  Pulse-width modulated cooling output value: The actuator for cooling is controlled via a digital output. Pulse width modulation creates variable ON and OFF times.

The cooling output is only available if it was activated via the "Activate cooling" check box.

- If the check box is cleared, the output value of the PID algorithm (PidOutputSum) is scaled and output at the outputs for heating.
- If the check box is selected, positive output values of the PID algorithm (PidOutputSum) are scaled and output at the outputs for heating. Negative output values of the PID algorithm are scaled and output at the outputs for cooling. You can choose between two methods for output value calculation at the output settings.

Note

Note:

- The OutputHeat_PWM, OutputHeat_PER, OutputCool_PWM, OutputCool_PER outputs are only calculated if you select these correspondingly from the drop-down list.
- The OutputHeat output is always calculated.
- The OutputCool output is calculated if the check box for cooling is selected.
- The "Activate cooling" check box is only available if the controller is not configured as a master in a cascade.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "OutputHeat_PER" or "OutputCool_PER" in the drop-down list "OutputHeat" or "OutputCool".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the pulse-width modulated output value:

1. Select the entry "OutputHeat_PWM" or "OutputCool_PWM" in the drop-down list "OutputHeat" or "OutputCool".
2. Select "Instruction".
3. Enter the address of the digital output.

Proceed as follows to process the output value using the user program:

1. Select the entry "OutputHeat" or "OutputCool" in the drop-down list "OutputHeat" or "OutpuCool".
2. Select "Instruction".
3. Enter the name of the variable you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.

Cascade

If a PID_Temp instance receives its setpoint from a higher-level master controller and outputs its output value in turn to a subordinate slave controller, this PID_Temp instance is both a master controller and a slave controller simultaneously. Both configurations listed below then have to be carried out for such a PID_Temp instance. This is the case, for example, for the middle PID_Temp instance in a cascade control system with three concatenated measured variables and three PID_Temp instances.

Configuring a controller as master in a cascade

A master controller specifies the setpoint of a slave controller through its output.

In order to use PID_Temp as master in a cascade, you have to deactivate the cooling in the basic settings. In order to configure this PID_Temp instance as a master controller in a cascade, activate the "Controller is master" check box. The selection of the output value for heating is set automatically to OutputHeat.

OutputHeat_PWM and OutputHeat_PER cannot be used at a master in a cascade.

Subsequently specify the number of directly subordinate slave controllers that receive their setpoint from this master controller.

If no own scaling function is used when assigning the OutputHeat parameter of the master to the Setpoint parameter of the slave, it may be necessary to adapt the output value limits and the output scaling of the master to the setpoint/process value range of the slave. This can be done in the output settings of the master in the "OutputHeat / OutputCool" section.

Configuring a controller as a slave in a cascade

A slave controller receives its setpoint (Setpoint parameter) from the output of its master controller (OutputHeat parameter).

In order to configure this PID_Temp instance as a slave controller in a cascade, activate the "Controller is slave" check box in the basic settings.

Subsequently select the PID_Temp instance that is to be used as the master controller for this slave controller in the Inspector window of the programming editor. The Master and Setpoint parameters of the slave controller are interconnected with the selected master controller through this selection (the existing interconnections at these parameters are overwritten). This interconnection allows the exchange of information and the setpoint specification between master and slave. If required, the interconnection can be changed subsequently at the Setpoint parameter of the slave controller in order, for example, to insert an additional filter. The interconnection at the parameter master may not be changed subsequently.

The "Controller is master" check box has to be selected and the number of slaves has to be configured correctly at the selected master controller. The master controller has to be called before the slave controller in the same cyclic interrupt OB.

Additional information

Additional information about program creation, configuration and commissioning when PID_Temp is used in cascade control systems is available under Cascade control with PID_Temp (Page 7321).

Process value settings

Process value limits

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can specify how PID_Temp responds to errors in automatic mode in the output settings.

Scale process value

If you have configured the use of Input_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration is displayed in the Input_PER display.

Input_PER is scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input fields.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID_Temp in the programming editor.
2. Interconnect Input_PER with an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.

The existing values are overwritten with the values from the hardware configuration.

Output settings

Basic settings output

Method for heating and cooling

If cooling is activated in the basic settings, two methods are available for calculating the PID output value:

- PID parameter changeover (Config.AdvancedCooling = TRUE):
  Output value calculation for cooling is carried out by means of a separate PID parameter set. The PID algorithm decides on the basis of the calculated output value and the control deviation whether the PID parameters are used for heating or cooling. This method is suitable if the heating and cooling actuators have different time responses and different gains.
  Pretuning and fine tuning for cooling are only available if this method is selected.
- Cooling factor (Config.AdvancedCooling = FALSE):
  Output value calculation for cooling is effected with the PID parameters for heating under consideration of the configurable cooling factor Config.CoolFactor. This method is suitable if the heating and cooling actuators have a similar time response but different gains. If this method is selected, pretuning and fine tuning for cooling as well as the PID parameter set for cooling are not available. Only the tunings for heating can be carried out.

Cooling factor

If the cooling factor is selected as the method for heating/cooling, this factor is used in the calculation of the output value for cooling. This allows different gains of heating and cooling actuators to be used.

The cooling factor is not set automatically or adjusted during tuning. You have to configure the correct cooling factor manually by using the ratio "Heating actuator gain/Cooling actuator gain".

Example: Cooling factor = 2.0 means that the heating actuator gain is twice as high as the cooling actuator gain.

The cooling factor is only effective and can only be changed if "Cooling factor" is selected as the method for heating/cooling.

Reaction to error

Notice

Your system may be damaged.

If you output "Current value while error is pending " or "Substitute output value while error is pending" in the event of an error, PID_Temp remains in automatic mode or in manual mode. This may cause a violation of the process value limits and damage your system.

It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID_Temp is preset so that the controller stays active in most cases in the event of an error.

If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error.

PID_Temp generates a programmable output value in response to an error:

- Zero (inactive)
  At all errors, PID_Temp switches to the "Inactive" operating mode and outputs the following:
  - 0.0 as PID output value (PidOutputSum)
  - 0.0 as output value for heating (OutputHeat) and output value for cooling (OutputCool)
  - 0 as analog output value for heating (OutputHeat_PER) and analog output value for cooling (OutputCool_PER)
  - FALSE as PWM output value for heating (OutputHeat_PWM) and PWM output value for cooling (OutputCool_PWM)

  This is independent of the configured output value limits and the scaling. The controller is only reactivated by a falling edge at Reset or a rising edge at ModeActivate.
- Current value while error is pending
  The error response depends on the error occurring and the operating mode.
  If one or more of the following errors occur in automatic mode, PID_Temp stays in automatic mode:
  - 0000001h: The Input parameter is outside the process value limits.
  - 0000800h: Sampling time error
  - 0040000h: Invalid value at Disturbance parameter.
  - 8000000h: Error during the calculation of the PID parameters.

  If one or more of the following errors occur in automatic mode, PID_Temp switches to "Substitute output value with error monitoring" mode and outputs the last valid PID output value (PidOutputSum):
  - 0000002h: Invalid value at Input_PER parameter.
  - 0000200h: Invalid value at Input parameter.
  - 0000400h: Calculation of output value failed.
  - 0001000h: Invalid value at Setpoint or SubstituteSetpoint parameter.

  The values at the outputs for heating and cooling resulting from the PID output value are produced by the configured output scaling.
  As soon as the errors are no longer pending, PID_Temp switches back to automatic mode.
  If an error occurs during manual mode, PID_Temp remains in manual mode and continues to use the manual value as the PID output value.
  If the manual value is invalid, the configured substitute output value is used.
  If the manual value and substitute output value are invalid, the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit) is used.
  If the following error occurs during pretuning or fine tuning, PID_Temp remains in active mode:
  - 0000020h: Pretuning is not permitted during fine tuning.

  When any other error occurs, PID_Temp cancels the tuning and switches to the mode from which tuning was started.
- Substitute output value while error is pending

PID_Temp behaves as described at "Current value while error is pending", but outputs the configured substitute output value (SubstituteOutput) as a PID output value (PidOutputSum) in "Substitute output value with error monitoring" operating mode.
The values at the outputs for heating and cooling resulting from the PID output value are produced by the configured output scaling.
In the case of controllers with activated cooling output (Config.ActivateCooling = TRUE), enter:

– A positive substitute output value to output the value at the outputs for heating.

– A negative substitute output value to output the value at the outputs for cooling.

If the following error occurs, PID_Temp stays in "Substitute output value with error monitoring" mode and outputs the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit):

– 0020000h: Invalid value at SubstituteOutput tag.

Output value limits and output value scaling

Depending on the operating mode, the PID output value (PidOutputSum) is calculated automatically by the PID algorithm or by the manual value (ManualValue) or the configured substitute output value (SubstituteOutput).

The PID output value is limited depending on the configuration:

- If the cooling is deactivated in the basic settings (Config.ActivateCooling = FALSE), the value is limited to the high limit of the PID output value (heating) (Config.Output.Heat.PidUpperLimit) and the low limit of the PID output value (heating) (Config.Output.Heat.PidLowerLimit).
  You can configure both limits at the horizontal axis of the scaling characteristic line in the "OutputHeat / OutputCool" section. These are displayed in the "OutputHeat_PWM / OutputCool_PWM" and "OutputHeat_PER / OutputCool_PER" sections, but cannot be changed.
- If the cooling is activated in the basic settings (Config.ActivateCooling = TRUE), the value is limited to the high limit of the PID output value (Config.Output.Heat.PidUpperLimit) and the low limit of the PID output value (cooling) (Config.Output.Cool.PidLowerLimit).
  You can configure both limits at the horizontal axis of the scaling characteristic line in the "OutputHeat / OutputCool" section. These are displayed in the "OutputHeat_PWM / OutputCool_PWM" and "OutputHeat_PER / OutputCool_PER" sections, but cannot be changed.
  The low limit of the PID output value (heating) (Config.Output.Heat.PidLowerLimit) and the high limit of the PID output value (cooling) (Config.Output.Cool.PidUpperLimit) cannot be changed and have to be assigned the value 0.0.

The PID output value is scaled and output at the outputs for heating and cooling.
Scaling can be specified separately for each output and is specified across 2 value pairs each, consisting of a limit value of the PID output value and a scaling value:

Output	Value pair	Parameter
OutputHeat	Value pair 1	High limit of PID output value (heating) Config.Output.Heat.PidUpperLimit, Scaled upper output value (heating) Config.Output.Heat.UpperScaling
	Value pair 2	Low limit of PID output value (heating) Config.Output.Heat.PidLowerLimit, Scaled lower output value (heating) Config.Output.Heat.LowerScaling
OutputHeat_PWM	Value pair 1	High limit of PID output value (heating) Config.Output.Heat.PidUpperLimit, Scaled upper PWM output value (heating) Config.Output.Heat.PwmUpperScaling
	Value pair 2	Low limit of PID output value (heating) Config.Output.Heat.PidLowerLimit, Scaled lower PWM output value (heating) Config.Output.Heat.PwmLowerScaling
OutputHeat_PER	Value pair 1	High limit of PID output value (heating) Config.Output.Heat.PidUpperLimit, Scaled upper analog output value (heating) Config.Output.Heat.PerUpperScaling
	Value pair 2	Low limit of PID output value (heating) Config.Output.Heat.PidLowerLimit, Scaled lower analog output value (heating) Config.Output.Heat.PerLowerScaling
OutputCool	Value pair 1	Low limit of PID output value (cooling) Config.Output.Cool.PidLowerLimit, Scaled upper output value (cooling) Config.Output.Cool.UpperScaling
	Value pair 2	High limit of PID output value (cooling) Config.Output.Cool.PidUpperLimit, Scaled lower output value (cooling) Config.Output.Cool.LowerScaling
OutputCool_PWM	Value pair 1	Low limit of PID output value (cooling) Config.Output.Cool.PidLowerLimit, Scaled upper PWM output value (cooling) Config.Output.Cool.PwmUpperScaling
	Value pair 2	High limit of PID output value (cooling) Config.Output.Cool.PidUpperLimit, Scaled lower output value (cooling) Config.Output.Cool.PwmLowerScaling

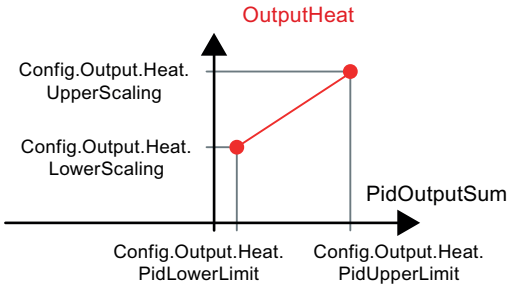
Output	Value pair	Parameter
OutputCool_PER	Value pair 1	Low limit of PID output value (cooling) Config.Output.Cool.PidLowerLimit, Scaled upper analog output value (cooling) Config.Output.Cool.PerUpperScaling
	Value pair 2	High limit of PID output value (cooling) Config.Output.Cool.PidUpperLimit, Scaled low analog output value (cooling) Config.Output.Cool.PerLowerScaling

The low limit of PID output value (heating) (Config.Output.Heat.PidLowerLimit) has to have the value 0.0, if the cooling is activated (Config.ActivateCooling = TRUE).

The high limit of PID output value (cooling) Config.Output.Cool.PidUpperLimit) must always have the value 0.0.

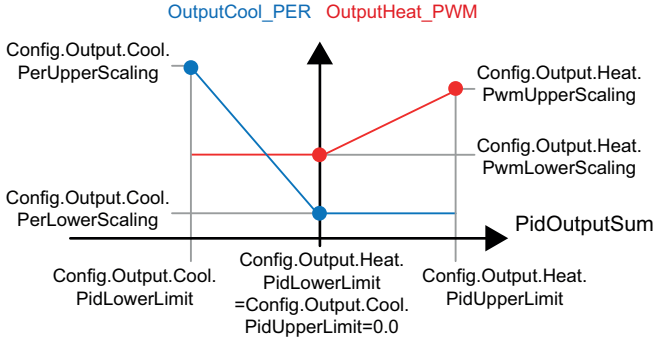
Example:

Output scaling when the OutputHeat output is used (cooling deactivated. The low limit of PID output value (heating) (Config.Output.Heat.PidLowerLimit) may be unequal to 0.0):



Example:

Output scaling when the OutputHeat_PWM and OutputCool_PER outputs are used (cooling activated. The low limit of PID output value (heating) (Config.Output.Heat.PidLowerLimit) must be 0.0):



With the exception of the "Inactive" operating mode, the value at an output always lies between its scaled upper output value and the scaled lower output value, for example for OutputHeat

always between the scaled upper output value (heating) (Config.Output.Heat.UpperScaling) and the scaled lower output value (heating) (Config.Output.Heat.LowerScaling).

If you want to limit the value at the associated output, you therefore have to adapt these scaling values as well.

You can configure the scaling values of an output at the vertical axes of the scaling characteristic line. Each output has two separate scaling values. These can only be changed for OutputHeat_PWM, OutputCool_PWM, OutputHeat_PER and OutputCool_PER if the corresponding output is selected in the basic settings. The cooling has to be activated additionally in the basic settings at all the outputs for cooling.

The trend view in the commissioning dialog box only records the values of OutputHeat and OutputCool, irrespective of the selected output in the basic settings. Therefore, if necessary, adapt the scaling values for OutputHeat/OutputCool if you use OutputHeat_PWM, or OutputHeat_PER or OutputCool_PWM if you use OutputCool_PER and want to use the trend view in the commissioning dialog.

Advanced settings

Process value monitoring

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning is displayed at the PID_Temp instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits are used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_Temp will respond as follows:

Process value	InputWarning_H	InputWarning_L	ErrorBits
> 98 °C	TRUE	FALSE	0001h
≤ 98° C and > 90° C	TRUE	FALSE	0000h
≤ 90° C and ≥ 10° C	FALSE	FALSE	0000h
< 10° C and ≥ 0° C	FALSE	TRUE	0000h
< 0° C	FALSE	TRUE	0001h

You can configure the response of PID_Temp when the process value high limit or low limit is violated in the output settings.

PWM limits

The PID output value PidOutputSum is scaled and transformed via a pulse width modulation into a pulse train that is output at the output parameter OutputHeat_PWM or OutputCool_PWM. The "Sampling time of PID algorithm" represents the time between two calculations of the PID output value. The sampling time is used as time period of the pulse width modulation.

During heating, the PID output value is always calculated in the "Sampling time of PID algorithm for heating".

Calculation of the PID output value during cooling depends on the type of cooling selected in "Basic settings Output":

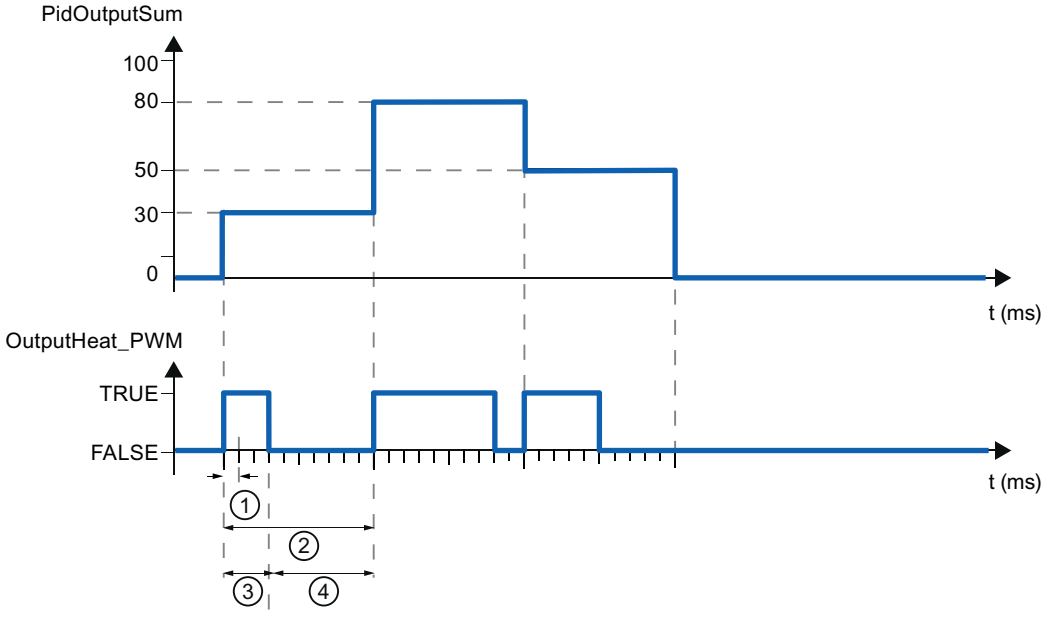
- If the cooling factor is used, the "Sampling time of PID algorithm for heating" applies.
- If the PID parameter changeover is used, the "Sampling time of PID algorithm for cooling" applies.

OutputHeat_PWM and OutputCool_PWM are output in the sampling time PID_Temp (corresponds to the cycle time of the calling OB).

The PID algorithm sampling time for heating or cooling is determined during pretuning or fine tuning. If you set the PID parameters manually, you will also need to configure the PID algorithm sampling time for heating or cooling. The PID_Temp sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the PID output value and is always an integer multiple of the PID_Temp sampling time.

Example for OutputHeat_PWM



① PID_Temp sampling time
② PID algorithm sampling time for heating
③ Pulse duration
④ Break time

The "Minimum ON time" and the "Minimum OFF time" can be set separately for heating and cooling, rounded to an integer multiple of the PID_Temp sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

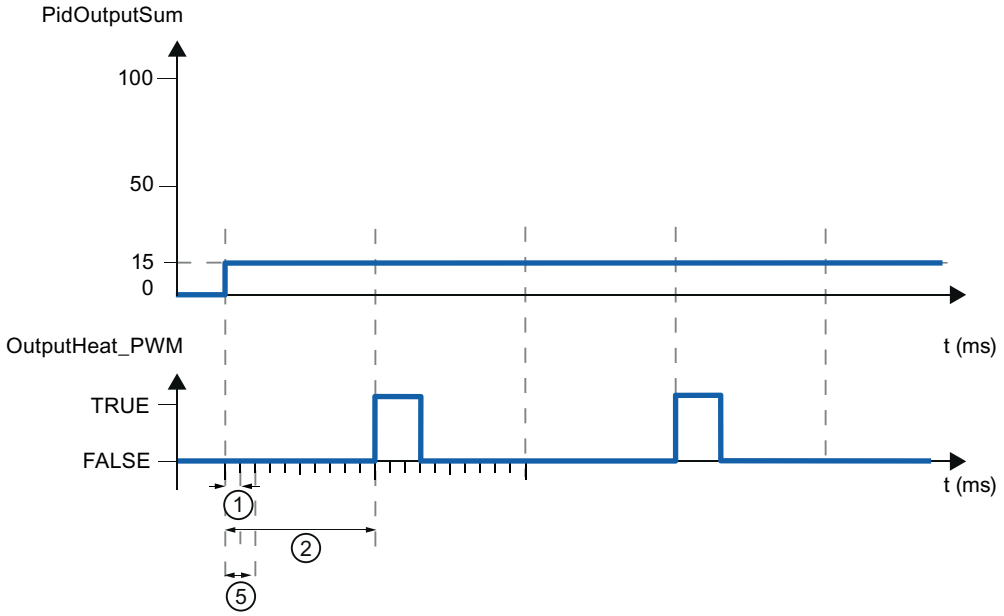
Example for OutputHeat_PWM

PID_Temp sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

The PID output value PidOutputSum amounts to 15% constantly. The smallest pulse that PID_Temp can output corresponds to 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.



① PID_Temp sampling time

② PID algorithm sampling time for heating

⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you have selected OutputHeat/OutputCool or OutputHeat_PER/OutputCool_PER as the output in the basic settings, the minimum ON time and the minimum OFF time are not evaluated and cannot be changed.

If the "Sampling time of PID algorithm" (Retain.CtrlParams.Heat.Cycle or Retain.CtrlParams.Cool.Cycle) and thus the period duration of the pulse width modulation is very high when OutputHeat_PWM or OutputCool_PWM is used, you can specify a deviating shorter period duration at the parameters Config.Output.Heat.PwmPeriode or

Config.Output.Cool.PwmPeriode in order to improve smoothness of the process value (see also PwmPeriode tag (Page 3605)).

Note

The minimum ON and OFF times only affect the output parameters OutputHeat_PWM or OutputCool_PWM and are not used for any pulse generators integrated in the CPU.

PID parameters

The PID parameters are displayed in the "PID Parameters" configuration window.

If cooling is activated in the basic settings and PID parameter changeover is selected as the method for heating/cooling in the output settings, two parameter sets are available: One for heating and one for cooling.

In this case, the PID algorithm decides on the basis of the calculated output value and the control deviation whether the PID parameters for heating or cooling are used.

If cooling is deactivated or the cooling factor is selected as the method for heating/cooling, the parameter set for heating is always used.

During tuning, the PID parameters are adapted to the controlled system with the exception of the deadband width that has to be configured manually.

PID_Temp is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions.

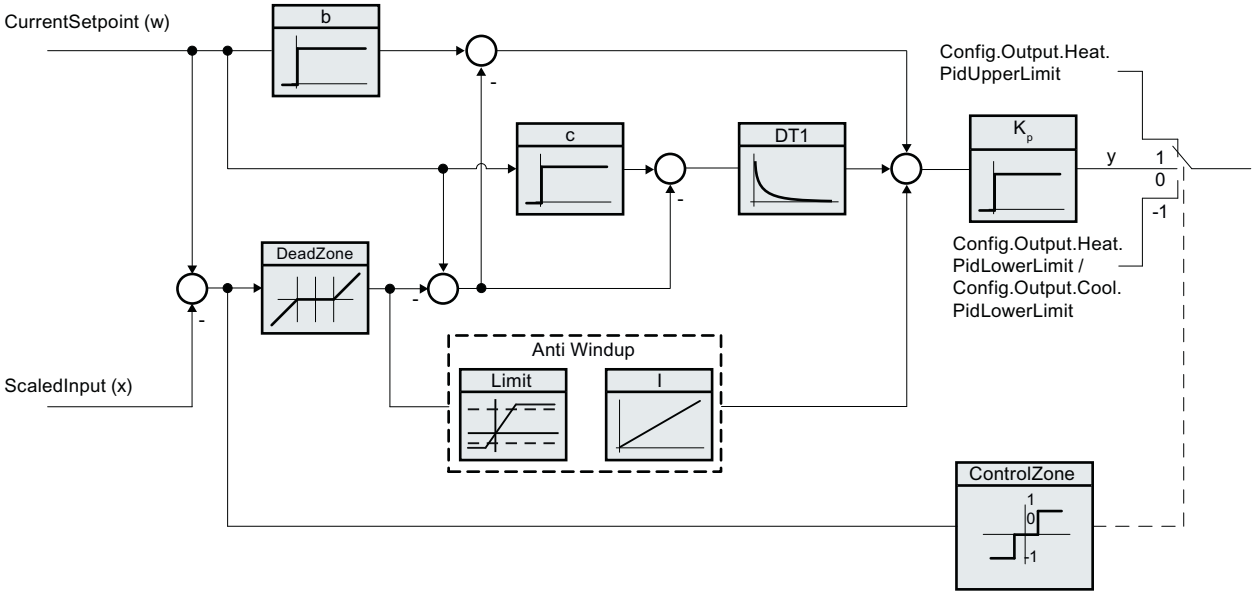
The PID algorithm operates according to the following equation (control zone and deadband deactivated):

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description	Associated parameters of the PID_Temp instruction
y	Output value of the PID algorithm	-
K_p	Proportional gain	Retain.CtrlParams.Heat.Gain Retain.CtrlParams.Cool.Gain CoolFactor
s	Laplace operator	-
b	Proportional action weighting	Retain.CtrlParams.Heat.PWeighting Retain.CtrlParams.Cool.PWeighting
w	Setpoint	CurrentSetpoint
x	Process value	ScaledInput
T_I	Integral action time	Retain.CtrlParams.Heat.Ti Retain.CtrlParams.Cool.Ti
T_D	Derivative action time	Retain.CtrlParams.Heat.Td Retain.CtrlParams.Cool.Td

Symbol	Description	Associated parameters of the PID_Temp instruction
a	Coefficient for derivative-action delay (Derivative delay T1 = a × T_D)	Retain.CtrlParams.Heat.TdFiltRatio Retain.CtrlParams.Cool.TdFiltRatio
c	Derivative action weighting	Retain.CtrlParams.Heat.DWeighting Retain.CtrlParams.Cool.DWeighting
DeadZone	Deadband width	Retain.CtrlParams.Heat.DeadZone Retain.CtrlParams.Cool.DeadZone
ControlZone	Control zone width	Retain.CtrlParams.Heat.ControlZone Retain.CtrlParams.Cool.ControlZone

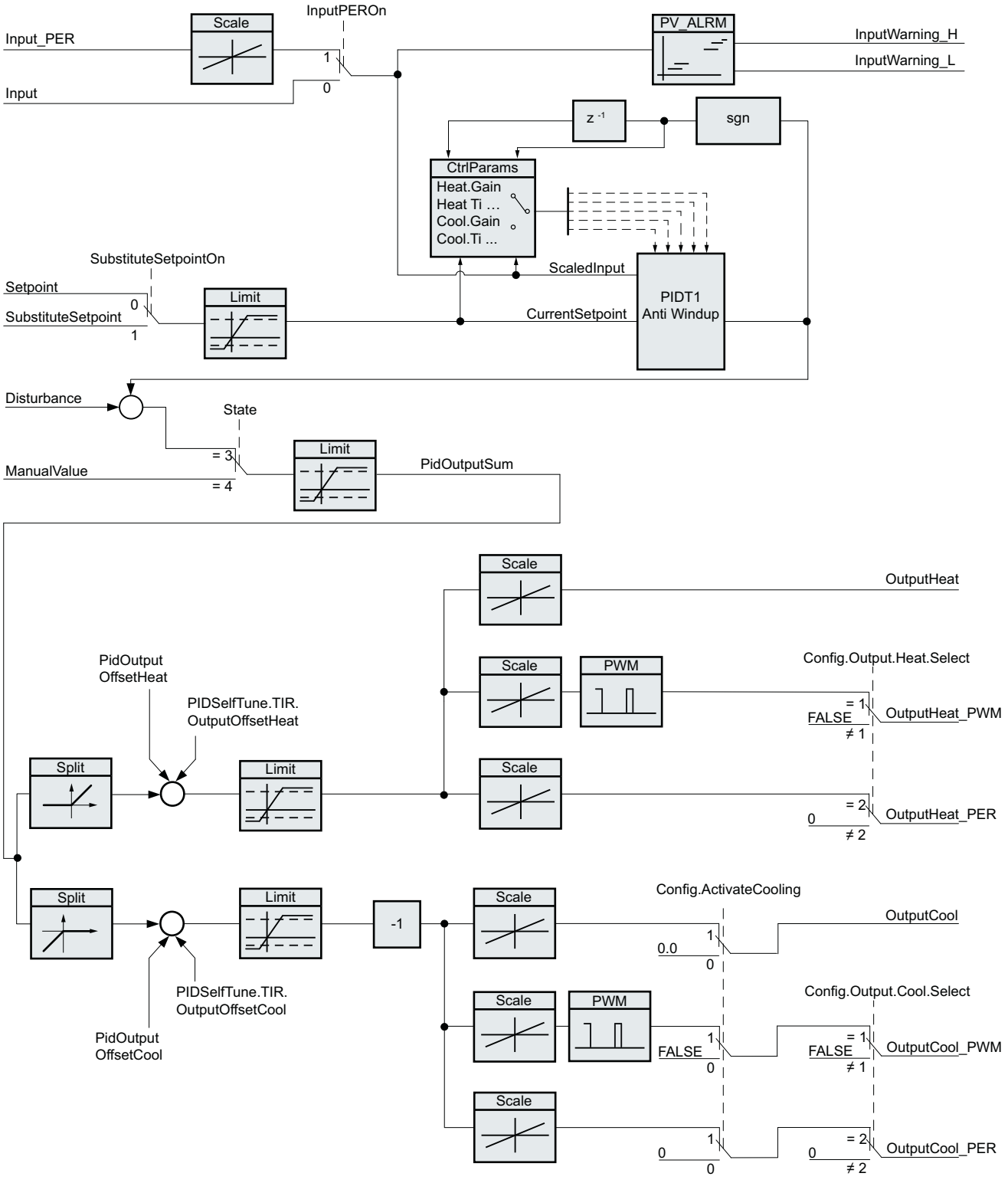
The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_Temp (Downloading technology objects to device (Page 7202)).

PID_Temp block diagram

The following block diagram shows how the PID algorithm is integrated in the PID_Temp.



Proportional gain

The value specifies the proportional gain of the controller. PID_Temp does not operate with a negative proportional gain and only supports the normal control direction, meaning that an increase in the process value is achieved by an increase in the PID output value (PidOutputSum).

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with one dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of "PID algorithm" represents the time between two calculations of the PID output value. It is calculated during tuning and rounded to a multiple of the PID_Temp sampling time (cycle time of the cyclic interrupt OB). All other functions of PID_Temp are executed at every call.

If you use OutputHeat_PWM or OutputCool_PWM, the sampling time of the PID algorithm is used as the period duration of the pulse width modulation. The accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The cycle time should be a maximum of one tenth of the PID algorithm sampling time.

The sampling time of the PID algorithm that is used as the period duration of the pulse width modulation at OutputCool_PWM depends on the method for heating/cooling selected in "Basic settings Output":

- If the cooling factor is used, the "sampling time of the PID algorithm for heating" also applies to OutputCool_PWM.
- If the PID parameter changeover is used, the "sampling time PID algorithm for cooling" applies as the period duration for OutputCool_PWM.

If the sampling time of the PID algorithm and thus the period duration of the pulse width modulation is very high when OutputHeat_PWM or OutputCool_PWM is used, you can specify a deviating shorter period duration at the parameters Config.Output.Heat.PwmPeriode or Config.Output.Cool.PwmPeriode in order to improve smoothness of the process value.

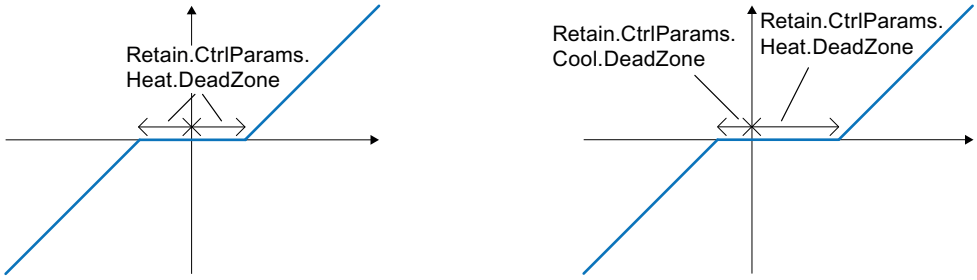
Deadband width

If the process value is affected by noise, the noise can also have an effect on the output value. The output value may fluctuate considerably when controller gain is high and the derivative action is activated. If the process value lies within the deadband around the setpoint, the control deviation is suppressed so that the PID algorithm does not react and unnecessary fluctuations of the output value are reduced.

The deadband width for heating is not set automatically during tuning. You have to correctly configure the deadband width manually. The deadband is deactivated by setting the deadband width = 0.0.

If cooling is activated in the basic settings and PID parameter changeover is selected as the method for heating/cooling in the output settings, the deadband lies between "Setpoint - deadband width (heating)" and "Setpoint + deadband width (cooling)".

If cooling is deactivated in the basic settings or the cooling factor is used, the deadband lies symmetrically between "Setpoint - deadband width (heating)" and "Setpoint + deadband width (heating)".



Deadband with deactivated cooling or cooling factor (left) or activated cooling and PID parameter changeover (right). The x / horizontal axis displays the control deviation = setpoint - process value. The y / vertical axis shows the output signal of the deadband that is passed to the PID algorithm.

Control zone width

If the process value exits the control zone around the setpoint, the minimum or maximum output value is output. This means that the process value reaches the setpoint faster.

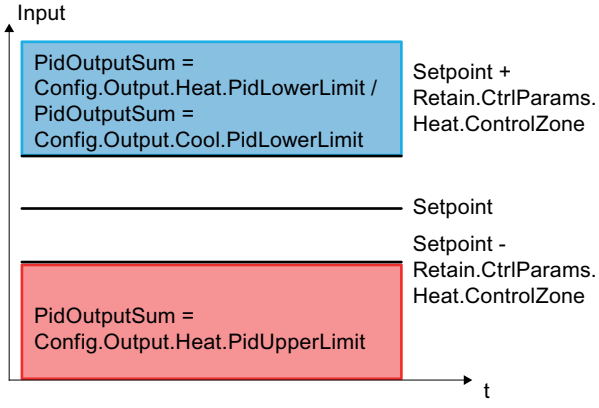
If the process value lies within the control zone around the setpoint, the output value is calculated by the PID algorithm.

The control zone width for heating or cooling is only set automatically during the pretuning, if "PID (temperature)" is selected as the controller structure for cooling or heating.

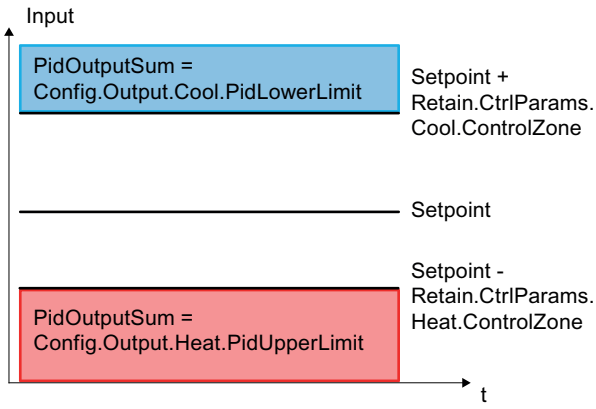
The control zone is deactivated by setting the control zone width = 3.402822e+38.

If cooling is deactivated in the basic settings or the cooling factor is used, the control zone lies symmetrically between "Setpoint - control zone width (heating)" and "Setpoint + control zone width (heating)".

If cooling is activated in the basic settings and PID parameter changeover is selected as the method for heating/cooling in the output settings, the control zone lies between "Setpoint - control zone width (heating)" and "Setpoint + control zone width (cooling)".



Control zone with deactivated cooling or cooling factor.



Control zone with activated cooling and PID parameter changeover.

Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list. You can specify the rules for tuning for heating and for tuning for cooling separately.

- PID (temperature)
  Calculates PID parameters during pretuning and fine tuning.
  Pretuning is designed for temperature processes and results in a slower and rather asymptotic control response with smaller overshoots than with the "PID" option. Fine tuning is identical to the "PID" option.
  The control zone width is determined automatically during pretuning only if this option is selected.
- PID
  Calculates PID parameters during pretuning and fine tuning.
- PI
  Calculates PI parameters during pretuning and fine tuning.
- User-defined
  The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program or the parameter view.

13.1.5.3 Commissioning PID_Temp

Commissioning

The commissioning window helps you commission the PID controller. You can monitor the values for the setpoint, process value and the output values for heating and cooling along the time axis in the trend view. The following functions are supported in the commissioning window:

- Controller pretuning
- Controller fine tuning
  Use fine tuning for fine adjustments to the PID parameters.
- Monitoring the current closed-loop control in the trend view

- Testing the controlled system by specifying a manual PID output value and a substitute setpoint
- Saving the actual values of the PID parameters to an offline project.

All functions require an online connection to the CPU.

The online connection to the CPU is established, if it does not exist already, and operation of the commissioning window is enabled by means of the "Monitor all" or "Start" buttons of the trend view.

Operation of the trend view

- Select the desired sampling time in the "Sampling time" drop-down list.
  All the values of the trend view are updated in the selected sampling time.
- Click the "Start" icon in the Measurement group if you want to use the trend view.
  Value recording is started. The current values for the setpoint, process value and output values for heating and cooling are entered in the trend view.
- Click the "Stop" icon if you want to end the trend view.
  The values recorded in the trend view can continue to be analyzed.

Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

Pretuning

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The PID parameters are backed up before being recalculated.

PID_Temp offers different pretuning types depending on the configuration:

- Pretuning heating
  A jump is output at the output value heating, the PID parameters for heating are calculated and then the setpoint is used as the control variable in automatic mode.
- Pretuning heating and cooling
  A jump is output at the output value heating.
  As soon as the process value is near the setpoint, a jump to the output value cooling is output.
  The PID parameters for heating (Retain.CtrlParams.Heat structure) and cooling (Retain.CtrlParams.Cool structure) are calculated and then the setpoint is used as the control variable in automatic mode.
- Pretuning cooling
  A jump is output at the output value cooling.
  The PID parameters for cooling are calculated and then the setpoint is used as the control variable in automatic mode.

If you want to tune the PID parameters for heating and cooling, you can expect improved control response by carrying out "Pretuning heating" and subsequently "Pretuning cooling" than by carrying out "Pretuning heating and cooling". However, carrying out pretuning in two steps takes more time.

General requirements

- The PID_Temp instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- PID_Temp is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".
- The setpoint and the process value lie within the configured limits (see Process value monitoring (Page 7302) configuration).

Requirements for pretuning heating

- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is greater than 50% of the setpoint.
- The setpoint is greater than the process value.

Requirements for pretuning heating and cooling

- The cooling output in the "Basic settings" is activated (Config.ActivateCooling = TRUE).
- The PID parameter changeover in the "Basic settings of output value" is activated (Config.AdvancedCooling = TRUE).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is greater than 50% of the setpoint.
- The setpoint is greater than the process value.

Requirements for pretuning cooling

- The cooling output in the "Basic settings" is activated (Config.ActivateCooling = TRUE).
- The PID parameter changeover in the "Basic settings of output value" is activated (Config.AdvancedCooling = TRUE).
- "Pretuning heating" or "Pretuning heating and cooling" has been carried out successfully (PIDSelfTune.SUT.ProcParHeatOk = TRUE). The same setpoint should be used for all tunings.
- The difference between setpoint and process value is smaller than 5% of the difference between process value high limit and process value low limit.

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all" button or start the trend view.
   An online connection will be established.
3. Select the desired pretuning entry from the "Tuning mode" drop-down list.
4. Click the "Start" icon.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar ("Progress" tag) has not changed for a long period and it is to be assumed that the tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_Temp switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_Temp responds with the configured reaction to errors.

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. The PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID_Temp automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

PID_Temp offers different fine tuning types depending on the configuration:

- Fine tuning heating:
  PID_Temp generates an oscillation of the process value with periodic changes at the output value heating and calculates the PID parameters for heating.
- Fine tuning cooling:
  PID_Temp generates an oscillation of the process value with periodic changes at the output value cooling and calculates the PID parameters for cooling.

Temporary tuning offset for heating/cooling controllers

If PID_Temp is used as a heating/cooling controller (Config.ActivateCooling = TRUE), the PID output value (PidOutputSum) at the setpoint has to fulfill the following requirements so that process value oscillation can be generated and fine tuning can be carried out successfully:

- Positive PID output value for fine tuning heating
- Negative PID output value for fine tuning cooling

If this condition is not fulfilled, you can specify a temporary offset for fine tuning that is output at the opposing output.

- Offset for cooling output (PIDSelfTune.TIR.OutputOffsetCool) at fine tuning heating. Before starting tuning, enter a negative tuning offset cooling that is smaller than the PID output value (PidOutputSum) at the setpoint in the stationary state.
- Offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) at fine tuning cooling Before starting tuning, enter a positive tuning offset heating that is greater than the PID output value (PidOutputSum) at the setpoint in the stationary state.

The defined offset is balanced by the PID algorithm so that the process value remains at the setpoint. The height of the offset allows the PID output value to be adapted correspondingly so that it fulfills the requirement mentioned above.

In order to avoid greater overshoots of the process value at specification of the offset, this can also be increased in several steps.

If PID_Temp exits the fine tuning mode, the tuning offset is reset.

Example: Specification of an offset for fine tuning cooling

- Without offset
  - Setpoint = Process value (ScaledInput) = 80 °C
  - PID output value (PidOutputSum) = 30.0
  - Output value heating (OutputHeat) = 30.0
  - Output value cooling (OutputCool) = 0.0
    Oscillation of the process value around the setpoint cannot be generated with the cooling output alone. Fine tuning would fail here.
- With offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) = 80.0
  - Setpoint = Process value (ScaledInput) = 80 °C
  - PID output value (PidOutputSum) = -50.0
  - Output value heating (OutputHeat) = 80.0
  - Output value cooling (OutputCool) = -50.0
    Thanks to the specification of an offset for the heating output, the cooling output can now generate oscillation of the process value around the setpoint. Fine tuning can now be carried out successfully.

General requirements

- The PID_Temp instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID_Temp is in inactive mode, automatic mode or manual mode.

Requirements for fine tuning heating

- Heat.EnableTuning = TRUE
- Cool.EnableTuning = FALSE
- If PID_Temp is configured as a heating-and-cooling controller (Config.ActivateCooling = TRUE), the heating output has to be active at the operating point where tuning is to be carried out.
  PidOutputSum > 0.0 (see tuning offset)

Requirements for fine tuning cooling

- Heat.EnableTuning = FALSE
- Cool.EnableTuning = TRUE
- The cooling output is activated (Config.ActivateCooling = TRUE).
- The PID parameter changeover is activated (Config.AdvancedCooling = TRUE).
- The cooling output has to be active at the operating point where tuning is to be carried out.
  PidOutputSum < 0.0 (see tuning offset)

Process depends on initial situation

Fine tuning can be started from the following operating modes: "Inactive", "automatic mode", or "manual mode".

Fine tuning proceeds as follows when started from:

- Automatic mode with PIDSelfTune.TIR.RunIn = FALSE (default)
  Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.
  PID_Temp controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive, manual mode or automatic mode with PIDSelfTune.TIR.RunIn = TRUE
  An attempt is made to reach the setpoint with the minimum or maximum output value (two-point control):
  - With minimum or maximum output value heating at fine tuning heating.
  - With minimum or maximum output value cooling at fine tuning cooling.

  This can produce increased overshoot. When the setpoint is reached, fine tuning is started. If the setpoint cannot be reached, PID_Temp does not abort tuning automatically.

Procedure

To perform fine tuning, follow these steps:

1. Double-click the "PID_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all" button or start the trend view.
   An online connection will be established.
3. Select the desired fine tuning entry from the "Tuning mode" drop-down list.
4. If required (see tuning offset), specify a tuning offset and wait until the stationary state is reached again.
5. Click the "Start" icon.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group if the progress bar ("Progress" tag) has not changed for a long period and it is to be assumed that the tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

In the following phases in particular, tuning is not aborted automatically if the setpoint cannot be reached.

- "Attempting to reach setpoint for heating with two-point control."
- "Attempting to reach setpoint for cooling with two-point control."

Result

If no errors occurred during fine tuning, the PID parameters have been tuned. PID_Temp switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID_Temp responds with the configured response to errors.

"Manual" mode

The following section describes how you can use "Manual mode" in the commissioning window of the "PID_Temp" technology object.

Manual mode is also possible when an error is pending.

Requirement

- The "PID_Temp" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established.
- The CPU is in "RUN" mode.

Procedure

If you want to test the controlled system by specifying a manual value, use "Manual mode" in the commissioning window.

To define a manual value, follow these steps:

1. Double-click the "PID_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all" button or start the trend view.
   An online connection will be established.
3. Select the "Manual mode" check box in the "Online status of controller" area.
   PID_Temp operates in manual mode. The most recent current output value remains in effect.
4. Enter the manual value in the editable field as a % value.
   If cooling is activated in the basic settings, enter the manual value as follows:
   - Enter a positive manual value to output the value at the outputs for heating.
   - Enter a negative manual value to output the value at the outputs for cooling.
5. Click the icon.

Result

The manual value is written to the CPU and immediately goes into effect.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller.

The switchover to automatic mode is bumpless.

Substitute setpoint

The following section describes how you can use the substitute setpoint in the commissioning window of the "PID_Temp" technology object.

Requirement

- The "PID_Temp" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established.
- The CPU is in "RUN" mode.

Procedure

If you want to use a different value as the setpoint than that specified at the "Setpoint" parameter (for example to tune a slave in a cascade), use the substitute setpoint in the commissioning window.

Proceed as follows to specify a substitute setpoint:

1. Double-click the "PID_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all" button or start the trend view.
   An online connection will be established.
3. Select the "Subst.Setpoint" check box in the "Online status of controller" section.
   The substitute setpoint (SubstituteSetpoint tag) is initialized with the most recently updated setpoint and now used.
4. Enter the substitute setpoint in the editable field.
5. Click the icon.

Result

The substitute setpoint is written to the CPU and immediately goes into effect.

Clear the "Subst.Setpoint" check box if the value at the "Setpoint" parameter is to be used again as setpoint.

The switchover is not bumpless.

Cascade commissioning

Information about cascade commissioning with PID_Temp is available under Commissioning (Page 7325).

13.1.5.4 Cascade control with PID_Temp

Introduction

In cascade control, several control loops are nested within each other. In the process, slaves receive their setpoint (Setpoint) from the output value (OutputHeat) of the respective higher-level master.

A prerequisite for establishing a cascade control system is that the controlled system can be divided into subsystems, each with its own measured variable.

Setpoint specification for the controlled variable is carried out at the outmost master.

The output value of the innermost slave is applied to the actuator and thus acts on the controlled system.

The following major advantages result from the use of a cascade control system in comparison with a single-loop control system:

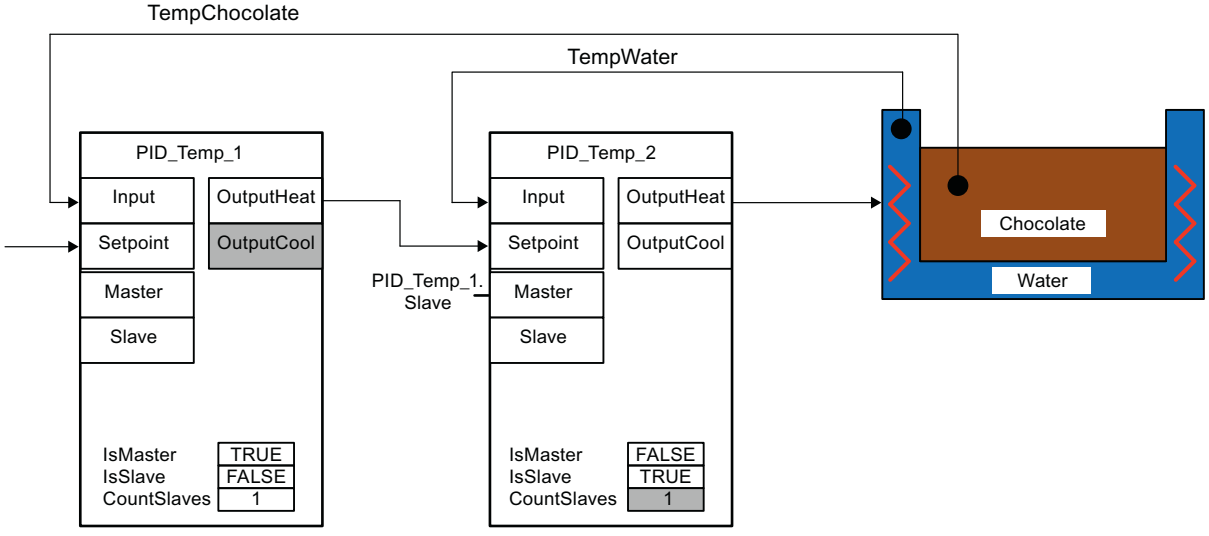
- Thanks to the additional subordinate control loops, disturbances which occur there are corrected quickly. Their influence on the controlled variable is reduced considerably. The disturbance behavior is thus improved.
- The subordinate control loops act in linearizing form. The negative effects of such non-linearities on the controlled variable are thus moderated.

PID_Temp offers the following functionality especially for use in cascade control systems:

- Specification of a substitute setpoint
- Exchange of status information between master and slave (for example, current operating mode)
- Different Anti-Wind-Up modes (response of the master to limitation of its slave)

Example

The following block diagram shows a cascade control system with PID_Temp using the simplified example of a chocolate melting unit:



The PID_Temp_1 master compares the process value of the chocolate temperature (TempChocolate) with the setpoint specification by the user at the Setpoint parameter. Its output value OutputHeat forms the setpoint of the slave PID_Temp_2.

PID_Temp_2 attempts to regulate the process value of the water-bath temperature (TempWater) to this setpoint. The output value of PID_Temp_2 acts directly on the actuator of the controlled system (heating of the water bath) and thus influences the water-bath temperature. The water-bath temperature in turn has an effect on the chocolate temperature.

See also

Program creation (Page 7323)

Program creation

Observe the following points during program creation:

- Number of PID_Temp instances
  The number of different PID_Temp instances called up in a cyclic interrupt OB has to agree with the number of concatenated measured variables in the process.
  There are two concatenated measured variables in the example: TempChocolate and TempWater. Therefore two PID_Temp instances are required.
- Call sequence
  A master has to be called before its slaves in the same cyclic interrupt OB.
  The outermost master at which the user setpoint is specified is called first.
  The slave whose setpoint is specified by the outermost master is called next, etc.
  The innermost slave that acts on the actuator of the process with its output value is called last.
  In the example, PID_Temp_1 is called before PID_Temp_2.
- Interconnection of the measured variables
  The outermost master is interconnected with the outermost measured variable that is to be regulated to the user setpoint.
  The innermost slave is interconnected with the innermost measured variable that is influenced directly by the actuator.
  Interconnection of the measured variables with PID_Temp is carried out with the parameters Input or Input_PER.
  In the example, the outermost measured variable TempChocolate is interconnected with PID_Temp_1 and the innermost measured variable TempWater with PID_Temp_2.
- Interconnection of the output value of the master to the setpoint of the slave
  The output value (OutputHeat) of a master has to be assigned to the setpoint (Setpoint) of its slave.
  This interconnection can be carried out in the programming editor or automatically in the Inspector window of the slave in the basic settings via the selection of the master.
  If required, you can insert your own filter or scaling functions, for example in order to adapt the output value range of the master to the setpoint/process value range of the slave.
  In the example, OutputHeat of PID_Temp_1 is assigned to Setpoint of PID_Temp_2.
- Interconnection of the interface for information exchange between master and slave
  The "Slave" parameter of a master has to be assigned to the "Master" parameter of all its directly subordinate slaves (which receive their setpoint from this master). The assignment should be carried out via the interface of the Slave in order to allow the interconnection of a master with several slaves and the display of the interconnection in the Inspector window of the slave in the basic settings.
  This interconnection can be carried out in the programming editor or automatically in the Inspector window of the slave in the basic settings via the selection of the master.
  The Anti-Wind-Up functionality and the evaluation of the slave operating modes at the master can only function correctly if this interconnection is carried out.
  In the example, the "Slave" parameter of PID_Temp_1 is assigned to the "Master" parameter of PID_Temp_2.

Program code of the example using SCL (without assignment of the output value of the slave to the actuator):

```
"PID_Temp_1"(Input:="TempChocolate");
"PID_Temp_2"(Input:="TempWater", Master := "PID_Temp_1".Slave,
Setpoint := "PID_Temp_1".OutputHeat);
```

See also

PID_Temp ActivateRecoverMode tag (Page 3601)

Configuration

You can carry out the configuration via your user program, the configuration editor or the Inspector window of the PID_Temp call.

When using PID_Temp in a cascade control system, ensure the correct configuration of the settings specified below.

If a PID_Temp instance receives its setpoint from a superior master controller and outputs its output value in turn to a subordinate slave controller, this PID_Temp instance is both a master controller and a slave controller simultaneously. Both configurations listed below have to be carried out for such a PID_Temp instance. This is the case, for example, for the middle PID_Temp instance in a cascade control system with three concatenated measured variables and three PID_Temp instances.

Configuration of a master

Setting in the configuration editor or Inspector window	DB parameter	Explanation
Basic settings → Cascade: Activate "Controller is master" check box	Config.Cascade.IsMaster = TRUE	Activates this controller as a master in a cascade
Basic settings → Cascade: Number of slaves	Config.Cascade.CountSlaves	Number of directly subordinate slaves that receive their setpoint directly from this master
Basic settings → Input/output parameters: Selection of the output value (heating) = OutputHeat	Config.Output.Heat.Select = 0	The master only uses the output parameter OutputHeat. OutputHeat_PWM and OutputHeat_PER are deactivated.
Basic settings → Input/output parameters: Clear "Activate cooling" check box	Config.ActivateCooling = FALSE	The cooling has to be deactivated at a master.

Setting in the configuration editor or Inspector window	DB parameter	Explanation
Output settings → Output limits and scaling → OutputHeat / OutputCool: Low limit of PID output value (heating), High limit of PID output value (heating), Scaled lower output value (heating), Scaled upper output value (heating)	Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PidUpperLimit, Config.Output.Heat.LowerScaling, Config.Output.Heat.UpperScaling	If no own scaling function is used when assigning OutputHeat of the master to Setpoint of the slave, it may be necessary to adapt the output value limits and the output scaling of the master to the setpoint/process value range of the slave.
This tag is not available in the Inspector window or in the function view of the configuration editor. You can change it via the parameter view of the configuration editor.	Config.Cascade.AntiWindUpMode	The Anti-Wind-Up mode determines how the integral action of this master is treated if directly subordinate slaves reach their output value limits. Options are: • AntiWindUpMode = 0: The AntiWindUp functionality is deactivated. The master does not react to the limitation of its slaves. • AntiWindUpMode = 1 (default): The integral action of the master is reduced in the relationship "Slaves in limitation/ Number of slaves". This reduces the effects of the limitation on the control behavior. • AntiWindUpMode = 2: The integral action of the master is held as soon as a slave is in limitation.

Configuration of a slave

Setting in the configuration editor or Inspector window	DB parameter	Explanation
Basic settings → Cascade: Select the "Controller is slave" check box	Config.Cascade.IsSlave = TRUE	Activates this controller as a slave in a cascade

Commissioning

After compiling and loading of the program, you can start commissioning of the cascade control system.

Begin with the innermost slave at commissioning (implementation of tuning or change to automatic mode with existing PID parameters) and continue outwards until the outermost master has been reached.

In the above example, commissioning starts with PID_Temp_2 and is continued with PID_Temp_1.

Tuning the slave

Tuning of PID_Temp requires a constant setpoint. Therefore, activate the substitute setpoint of a slave (SubstituteSetpoint and SubstituteSetpointOn tags) to tune the slave or set the associated master to manual mode by using a corresponding manual value. This ensures that the setpoint of the slave remains constant during tuning.

Tuning the master

In order for a master to influence the process or to carry out tuning, all the downstream slaves have to be in automatic mode and their substitute setpoint has to be deactivated. A master evaluates these conditions through the interface for information exchange between master and slave (Master parameter and Slave parameter) and displays the current state at the AllSlaveAutomaticState and NoSlaveSubstituteSetpoint tags. Corresponding status messages are output in the commissioning editor.

Status message in the commissioning editor of the master	DB parameter of the master	Correction
One or more slaves are not in automatic mode.	AllSlaveAutomaticState = FALSE, NoSlaveSubstituteSetpoint = TRUE	First, carry out commissioning of all downstream slaves. Ensure that the following conditions are fulfilled before carrying out tuning or activating manual mode or automatic mode of the master: • All downstream slaves are in automatic mode (state = 3). • All downstream slaves have deactivated the substitute setpoint (SubstituteSetpointOn = FALSE).
One or more slaves have activated the substitute setpoint.	AllSlaveAutomaticState = TRUE, NoSlaveSubstituteSetpoint = FALSE	
One or more slaves are not in automatic mode and have activated the substitute setpoint.	AllSlaveAutomaticState = FALSE, NoSlaveSubstituteSetpoint = FALSE	

If pretuning or fine tuning is started for a master, PID_Temp aborts tuning in the following cases and displays an error with ErrorBits = DW#16#0200000:

- One or more slaves are not in automatic mode (AllSlaveAutomaticState = FALSE)
- One or more slaves have activated the substitute setpoint (NoSlaveSubstituteSetpoint = FALSE).

The subsequent operating mode changeover depends on ActivateRecoverMode.

Substitute setpoint

In order to specify a setpoint, PID_Temp offers a substitute setpoint at the SubstituteSetpoint tags in addition to the Setpoint parameter. This can be activated by setting SubstituteSetpointOn = TRUE or by selecting the corresponding check box in the commissioning editor.

The substitute setpoint allows you to specify the setpoint temporarily directly at the slave, for example during commissioning or tuning.

In this case, the interconnection of the output value of the master with the setpoint of the slave that is required for normal operation of the cascade control system does not have to be changed in the program

In order for a master to influence the process or to carry out tuning, the substitute setpoint has to be deactivated at all downstream slaves.

You can monitor the currently effective setpoint as it is used by the PID algorithm for calculation at the CurrentSetpoint tags.

Operating modes and fault response

The master or slave of a PID_Temp instance does not change the operating mode of this PID_Temp instance.

If a fault occurs at one of its slaves, the master remains in its current operating mode.

If a fault occurs at its master, the slave remains in its current operating mode. However, further operation of the slave then depends on the fault and the configured fault response of the master since the output value of the master is used as the setpoint of the slave:

- If ActivateRecoverMode = TRUE is configured at the master. and the fault does not prevent the calculation of OutputHeat, the fault does not have any effect on the slave.
- If ActivateRecoverMode = TRUE is configured at the master and the fault prevents the calculation of OutputHeat, the master outputs the last output value or the configured substitute output value SubstituteOutput, depending on SetSubstituteOutput. This is then used by the slave as the setpoint.
  PID_Temp is preconfigured so that the substitute output value 0.0 is output in this case (ActivateRecoverMode = TRUE, SetSubstituteOutput = TRUE, SubstituteOutput = 0.0). Configure a suitable substitute output value for your application or activate the use of the last valid PID output value (SetSubstituteOutput = FALSE).
- If ActivateRecoverMode = FALSE is configured at the master, the master changes to the "Inactive" mode when a fault occurs and outputs OutputHeat = 0.0. The slave then uses 0.0 as the setpoint.

The fault response is located in the output settings in the configuration editor.

13.1.5.5 Multi-zone controlling with PID_Temp

Introduction

In a multi-zone control system, several sections, so-called zones, of a plant are controlled simultaneously to different temperatures. A multi-zone control system is characterized by the mutual influence of the temperature zones through thermal coupling, i.e. the process value of one zone can influence the process value of a different zone through thermal coupling. The strength that this influence has depends on the structure of the plant and the selected operating points of the zones.

Example: Extrusion plant as it is used, for example, in plastics processing.

The substance mixture that passes through the extruder has to be controlled to different temperatures for optimal processing. For example, different temperatures can be required at the filling point of the extruder than at the outlet nozzle. The individual temperature zones mutually influence each other through thermal coupling.

When PID_Temp is used in multi-zone control systems, each temperature zone is controlled by a separate PID_Temp instance.

Observe the following explanations if you want to use the PID_Temp in a multi-zone control system.

Separate pretuning for heating and cooling

Initial commissioning of a plant as a rule begins with the carrying out of pretuning in order to carry out initial setting of the PID parameters and control to the operating point. The pretuning for multi-zone control systems is often carried out simultaneously for all zones.

PID_Temp offers the possibility of carrying out pretuning for heating and cooling in one step (Mode = 1, Heat.EnableTuning = TRUE, Cool.EnableTuning = TRUE) for controllers with activated cooling and PID parameter changeover as the method for heating/cooling (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE).

However, it is advisable not to use this tuning for simultaneous pretuning of several PID_Temp instances in a multi-zone control system. Instead, first carry out the pretuning for heating (Mode = 1, Heat.EnableTuning = TRUE, Cool.EnableTuning = FALSE) and the pretuning for cooling (Mode = 1, Heat.EnableTuning = FALSE, Cool.EnableTuning = TRUE) separately.

Pretuning for cooling should not be started until all zones have completed pretuning for heating and have reached their operating points.

This reduces mutual influencing through thermal coupling between the zones during tuning.

Adapting the delay time

If PID_Temp is used in a multi-zone control system with strong thermal couplings between the zones, you should ensure that the adaption of the delay time is deactivated for pretuning with PIDSelfTune.SUT.AdaptDelayTime = 0. Otherwise, the determination of the delay time can be incorrect if the cooling of a zone is prevented by the thermal influence of other zones during the adapting of the delay time (heating is deactivated in this phase).

Temporary deactivation of cooling

PID_Temp offers the possibility of deactivating cooling temporarily in automatic mode for controllers with active cooling (Config.ActivateCooling = TRUE) by setting DisableCooling = TRUE.

This ensures that this controller does not cool in automatic mode during commissioning while the controllers of other zones have not yet completed tuning of heating. The tuning could otherwise be influenced negatively by the thermal coupling between the zones.

Procedure

You can proceed as follows during the commissioning of multi-zone control systems with relevant thermal couplings:

1. Set DisableCooling = TRUE for all controllers with activated cooling.
2. Set PIDSelfTune.SUT.AdaptDelayTime = 0 for all controllers.
3. Specify the desired setpoints (Setpoint parameter) and start pretuning for heating (Mode = 1, Heat.EnableTuning = TRUE, Cool.EnableTuning = FALSE) simultaneously for all controllers.

4. Wait until all the controllers have completed pretuning for heating.
5. Set DisableCooling = FALSE for all controllers with activated cooling.
6. Wait until the process values of all the zones are steady and close to the respective setpoint. If the setpoint cannot be reached permanently for a zone, the heating or cooling actuator is too weak.
7. Start pretuning for cooling (Mode = 1, Heat.EnableTuning = FALSE, Cool.EnableTuning = TRUE) for all controllers with activated cooling.

Note

Limit violation of the process value

If the cooling is deactivated in automatic mode with DisableCooling = TRUE, this can cause the process value to exceed the setpoint and the process value limits while DisableCooling = TRUE. Observe the process values and intervene, if appropriate, if you use DisableCooling.

Note

Multi-zone control systems

For multi-zone control systems, the thermal couplings between the zones can result in increased overshoots, permanent or temporary violation of limits and permanent or temporary control deviations during commissioning or operation. Observe the process values and be ready to intervene. Depending on the system, it can be necessary to deviate from the procedure described above.

Synchronization of several fine tuning processes

If fine tuning is started from automatic mode with PIDSelfTune.TIR.RunIn = FALSE, PID_Temp tries to reach the setpoint with PID controlling and the current PID parameters. The actual tuning does not start until the setpoint is reached. The time required to reach the setpoint can be different for the individual zones of a multi-zone control system.

If you want to carry out fine tuning for several zones simultaneously, PID_Temp offers the possibility to synchronize these by waiting with the further tuning steps after the setpoint has been reached.

Procedure

This ensures that all the controllers have reached their setpoint when the actual tuning steps start. This reduces mutual influencing through thermal coupling between the zones during tuning.

Proceed as follows for controllers for whose zones you want to carry out fine tuning simultaneously:

1. Set PIDSelfTune.TIR.WaitForControlIn = TRUE for all controllers.
   These controllers have to be in automatic mode with PIDSelfTune.TIR.RunIn = FALSE.
2. Specify the desired setpoints (Setpoint parameters) and start fine tuning for all controllers.

3. Wait until PIDSelfTune.TIR.ControlInReady = TRUE at all controllers.
4. Set PIDSelfTune.TIR.FinishControlIn = TRUE for all controllers.

All controllers then start the actual tuning simultaneously.

13.2 Using S7-1200 Motion Control

13.2.1 Introduction

13.2.1.1 Motion functionality of the CPU S7-1200

The TIA Portal, together with the motion control functionality of the CPU S7-1200, supports you in controlling stepper motors and servo motors:

- You configure the positioning axis and command table technology objects in the TIA Portal. The CPU S7-1200 uses these technology objects to control the outputs that control the drives.
- In the user program you control the axis by means of Motion Control instructions and initiate motion commands of your drive.

You can find a multi-media introduction on the Internet (http://www.automation.siemens.com/mcms/topics/en/simatic/simatic-technology/integrated-functions/simatic-s7-1200/Pages/Default.aspx).

See also

Hardware components for motion control (Page 7331)

Integration of the positioning axis technology object (Page 7357)

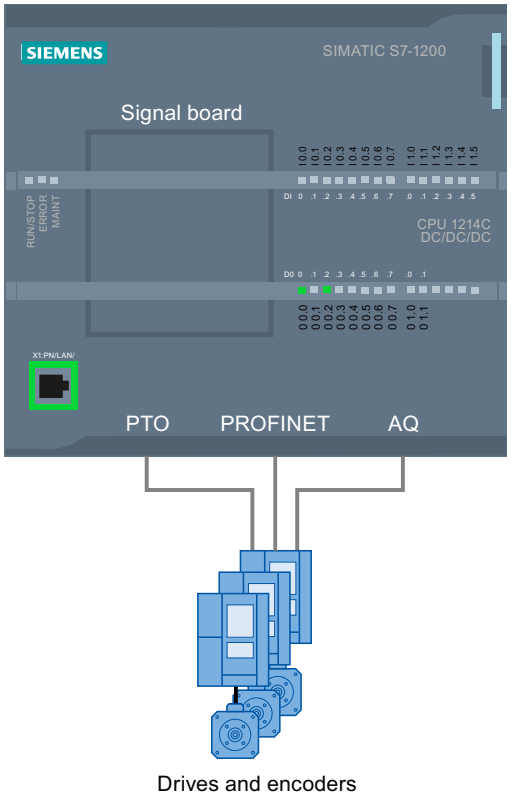
Tools of the positioning axis technology object (Page 7360)

Use of the command table technology object (Page 7415)

Command table technology object tools (Page 7416)

13.2.1.2 Hardware components for motion control

The representation below shows the basic hardware configuration for a motion control application with the CPU S7-1200.



CPU S7-1200

CPU S7-1200 combines the functionality of a programmable logic controller with motion control functionality for operation of drives. The motion control functionality takes over the control and monitoring of the drives.

Signal board

You add further inputs and outputs to the CPU with the signal boards.

You can use the digital outputs as pulse generator outputs for controlling drives as required. In CPUs with relay outputs, the pulse signal cannot be output on the onboard outputs because the relays do not support the necessary switching frequencies. To be able to work with the PTO (Pulse Train Output) on these CPUs, you must use a signal board with digital outputs.

You can use the analog outputs for controlling connected analog drives as required.

PROFINET

Use the PROFINET interface to establish the online connection between the CPU S7-1200 and the programming device. In addition to the online functions of the CPU, additional commissioning and diagnostic functions are available for motion control.

PROFINET still supports the PROFIdrive profile for connecting PROFIdrive-capable drives.

Drives and encoders

Drives permit the movement of the axis. Encoders provide the actual position for the closed loop position control of the axis.

The table below shows the connection possibilities for drives and encoders:

Drive connection	Closed/open loop control of axis	Encoder connection
PTO (Pulse Train Output) (Stepper motors and servo motors with pulse interface)	Speed-controlled	-
Analog output (AQ)	Position-controlled	• Encoder on high-speed counter (HSC) • Encoder on technology module (TM) • PROFIdrive encoder on PROFINET
PROFIdrive	Position-controlled	• Encoder on drive • Encoder on high-speed counter (HSC) • Encoder on technology module (TM) • PROFIdrive encoder on PROFINET

Ordering information for CPU firmware V4.1

The order information listed below applies to the currently installed product phase (without any installed Hardware Support Packages) of the TIA Portal.

Name	MLFB - article number
CPU 1211C DC/DC/DC	6ES7211-1AE40-0XB0
CPU 1211C AC/DC/RLY	6ES7211-1BE40-0XB0
CPU 1211C DC/DC/RLY	6ES7211-1HE40-0XB0
CPU 1212C DC/DC/DC	6ES7212-1AE40-0XB0
CPU 1212C AC/DC/RLY	6ES7212-1BE40-0XB0
CPU 1212C DC/DC/RLY	6ES7212-1HE40-0XB0
CPU 1214C DC/DC/DC	6ES7214-1AG40-0XB0
CPU 1214C AC/DC/RLY	6ES7214-1BG40-0XB0
CPU 1214C DC/DC/RLY	6ES7214-1HG40-0XB0
CPU 1214FC DC/DC/DC	6ES7214-1AF40-0XB0
CPU 1214FC DC/DC/RLY	6ES7214-1HF40-0XB0
CPU 1215C DC/DC/DC	6ES7215-1AG40-0XB0
CPU 1215C AC/DC/RLY	6ES7215-1BG40-0XB0

Name	MLFB - article number
CPU 1215C DC/DC/RLY	6ES7215-1HG40-0XB0
CPU 1215FC DC/DC/DC	6ES7215-1AF40-0XB0
CPU 1215FC DC/DC/RLY	6ES7215-1HF40-0XB0
CPU 1217C DC/DC/DC	6ES7217-1AG40-0XB0
Signal board DI4 x DC 24 V (200 kHz)	6ES7221-3BD30-0XB0
Signal board DI4 x DC 5 V (200 kHz)	6ES7 221-3AD30-0XB0
Signal board DQ4 x DC 24 V (200 kHz)	6ES7222-1BD30-0XB0
Signal board DQ4 x DC 5 V (200 kHz)	6ES7222-1AD30-0XB0
Signal board DI2/DQ2 x DC 24 V (20 kHz)	6ES7223-0BD30-0XB0
Signal board DI2/DQ2 x DC 24 V (200 kHz)	6ES7223-3BD30-0XB0
Signal board DI2/DQ2 x DC 5 V (200 kHz)	6ES7223-3AD30-0XB0
Signal board AQ1 x 12 bit (±10 V, 0 to 20 mA)	6ES7 232-4HA30-0XB0

Use a Hardware Support Package (HSP) to install new hardware components. The hardware component will then be available in the hardware catalog.

See also

Motion functionality of the CPU S7-1200 (Page 7328)

CPU outputs relevant for motion control (Page 7333)

13.2.2 Basics for working with S7-1200 Motion Control

13.2.2.1 Drive connection via PTO

CPU outputs relevant for motion control

The number of usable drives depends on the number of PTOs (pulse train outputs) and the number of available pulse generator outputs.

The following tables provide information about the relevant dependencies:

Maximum number of PTOs

4 PTOs are available for each CPU with technology version V4. This means a maximum of 4 drives can be controlled.

Signal type of the PTO

Depending on the signal type of the PTO, 1-2 pulse generator outputs are required per PTO (drive):

Signal type	Number of pulse generator outputs
Pulse A and direction B (direction output disabled) *)	1
Pulse A and direction B *)	2
Count up A and count down B	2
A/B phase-shifted	2
A/B phase-shifted - fourfold	2

*) The direction output must be on-board or on a signal board.

Usable pulse generator outputs and limit frequencies

The relay variants of the CPUs can only access the pulse generator outputs of a signal board.

Depending on the CPU and signal board, the following pulse signal generator outputs can be used with the following limit frequencies:

On-board	Q0.0	Q0.1	Q0.2	Q0.3	Q0.4	Q0.5	Q0.6	Q0.7	Q1.0	Q1.1
CPU 1211 (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	-	-	-	-	-	-
CPU 1212 (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	30 kHz	30 kHz	-	-	-	-
CPU 1214(F) (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	30 kHz	30 kHz	30 kHz	30 kHz	30 kHz	30 kHz
CPU 1215(F) (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	30 kHz	30 kHz	30 kHz	30 kHz	30 kHz	30 kHz
CPU 1217 (DC/DC/DC)	1 MHz	1 MHz	1 MHz	1 MHz	100 kHz	100 kHz	100 kHz	100 kHz	100 kHz	100 kHz
Signal board	**Qx.0**	**Qx.1**	**Qx.2**	**Qx.3**	-	-	-	-	-	-
Signal board DI2/DO2 x DC24V 20kHz	20 kHz	20 kHz	-	-	-	-	-	-	-	-
Signal board DI2/DO2 x DC24V 200kHz	200 kHz	200 kHz	-	-	-	-	-	-	-	-
Signal board DO4 x DC24V 200kHz	200 kHz	200 kHz	200 kHz	200 kHz	-	-	-	-	-	-
Signal board DI2/DO2 x DC5V 200kHz	200 kHz	200 kHz	-	-	-	-	-	-	-	-
Signal board DO4 x DC5V 200kHz	200 kHz	200 kHz	200 kHz	200 kHz	-	-	-	-	-	-

The low limit frequency is always 1Hz.

The pulse generator outputs can be freely assigned to the PTOs.

Note

If pulse generator outputs with different limit frequencies are used in accordance with the signal type, the low limit frequency is used in each case.

Signal type "Pulse A and direction B" is an exception. With this type of signal, the limit frequency of the pulse generator output is always used.

Note

Access to pulse generator outputs via the process image

The firmware takes control via the corresponding pulse generator and direction outputs if the PTO (Pulse Train Output) is selected and assigned to an axis.

With this takeover of the control function, the connection between the process image and I/O output is also disconnected. Although the user has the option of writing the process image of pulse generator and direction outputs via the user program or watch table, this is not transferred to the I/O output. Accordingly, it is also not possible to monitor the I/O output via the user program or watch table. The information read reflects the value of the process image and does not match the real status of the I/O output.

For all other CPU outputs that are not used permanently by the CPU firmware, the status of the I/O output can be controlled or monitored via the process image, as usual.

Outputs for drive signals

For motion control, you can optionally parameterize a drive interface for "Drive enabled" and "Drive ready".

When using the drive interface the digital output for the drive enable and the digital input for "drive ready" can be freely selected.

Acceleration/deceleration limits

The following limits apply to acceleration and deceleration:

Acceleration / deceleration	Value
Minimum acceleration/deceleration	5.0E-3 pulses/s^2
Maximum acceleration/deceleration	9.5E+9 pulses/s^2

Jerk limits

The following limits apply to the jerk:

Jerk	Value
Minimum jerk	4.0E-3 pulses/s^3
Maximum jerk	1.0E+10 pulses/s^3

See also

CPU outputs relevant for motion control (technology version V1...3) (Page 7533)

How the pulse interface works (Page 7336)

Relationship between the signal type and the direction of travel (Page 7336)

Hardware and software limit switches (Page 7346)

Jerk limit (Page 7347)

Homing (Page 7348)

Hardware components for motion control (Page 7329)

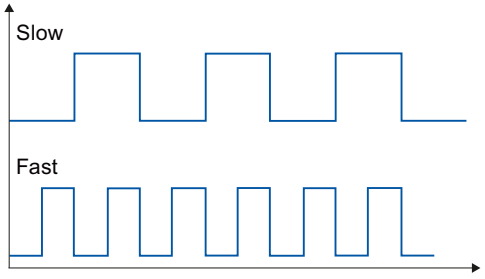
Integration of the positioning axis technology object (Page 7357)

Tools of the positioning axis technology object (Page 7360)

How the pulse interface works

Depending on the settings of the stepper motor, each pulse affects the movement of the stepper motor by a specific angle. If the stepper motor is set to 1000 pulses per revolution, for example, it moves 0.36° per pulse.

The speed of the stepper motor is determined by the number of pulses per time unit.



(The statements made here also apply to servo motors with pulse interface.)

See also

CPU outputs relevant for motion control (Page 7331)

Relationship between the signal type and the direction of travel (Page 7336)

Hardware and software limit switches (Page 7346)

Jerk limit (Page 7347)

Homing (Page 7348)

Integration of the positioning axis technology object (Page 7357)

Tools of the positioning axis technology object (Page 7360)

Relationship between the signal type and the direction of travel

The CPU outputs the velocity and direction of travel via two outputs.

The relationships between the configuration and direction of travel differ depending on the selected signal type. You can configure the following signal types in the axis configuration under "Basic parameters > General":

- "PTO (pulse A and direction B)"
- "PTO (count up A, count down B)" (as of V4)
- "PTO (A/B phase-shifted)" (as of V4)
- "PTO (A/B phase-shifted - fourfold)" (as of V4)

You configure the direction under "Extended Parameters > Mechanics" in the axis configuration. If you select the "Invert direction" option, the direction logic described below for the respective signal type is inverted.

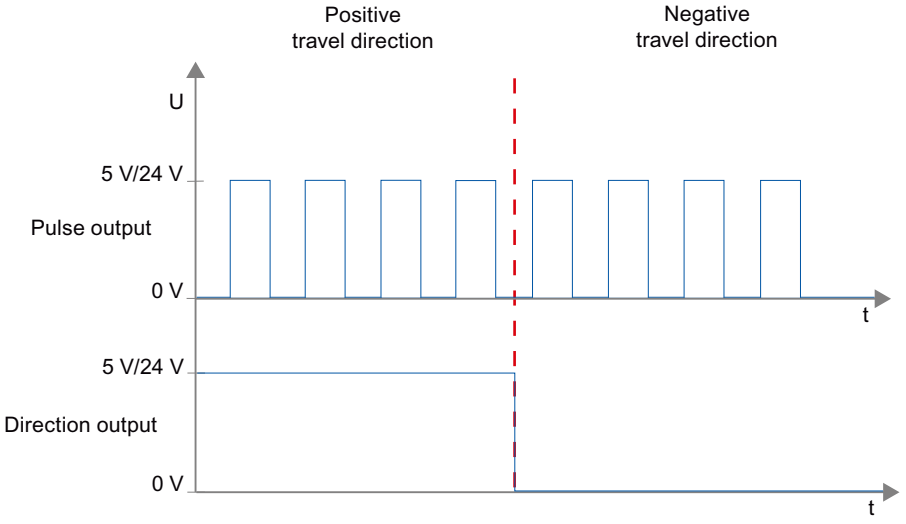
PTO (pulse A and direction B)

The pulse output pulses and the direction output level are evaluated for this signal type.

The pulses are output via the pulse output of the CPU. The direction output of the CPU specifies the direction of rotation of the drive:

- 5 V/24 V at direction output ⇒ positive direction of rotation
- 0 V at direction output ⇒ negative direction of rotation

The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.

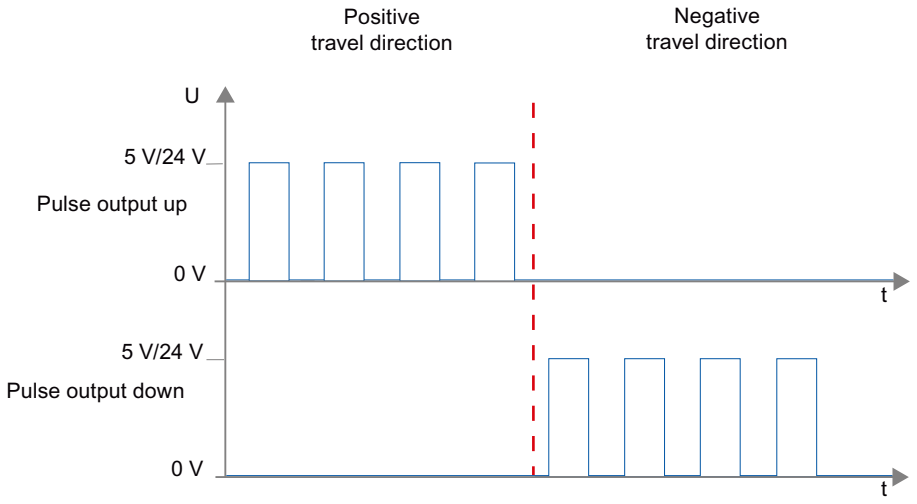


PTO (count up A, count down B) (as of V4)

The pulses of one output are evaluated for this signal type.

The pulse for the positive direction is output via the "Pulse output up" The pulse for the negative direction is output via the "Pulse output down"

The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.



PTO (A/B phase-shifted) (as of V4)

The positive edges of one output in each case are evaluated for this signal type.

The pulse is output via the "Signal A" output and phase-shifted via the "Signal B" output. The phase shifting between the outputs defines the direction of rotation:

- Signal A leads signal B by 90° ⇒ positive direction of rotation
- Signal B leads signal A by 90° ⇒ negative direction of rotation

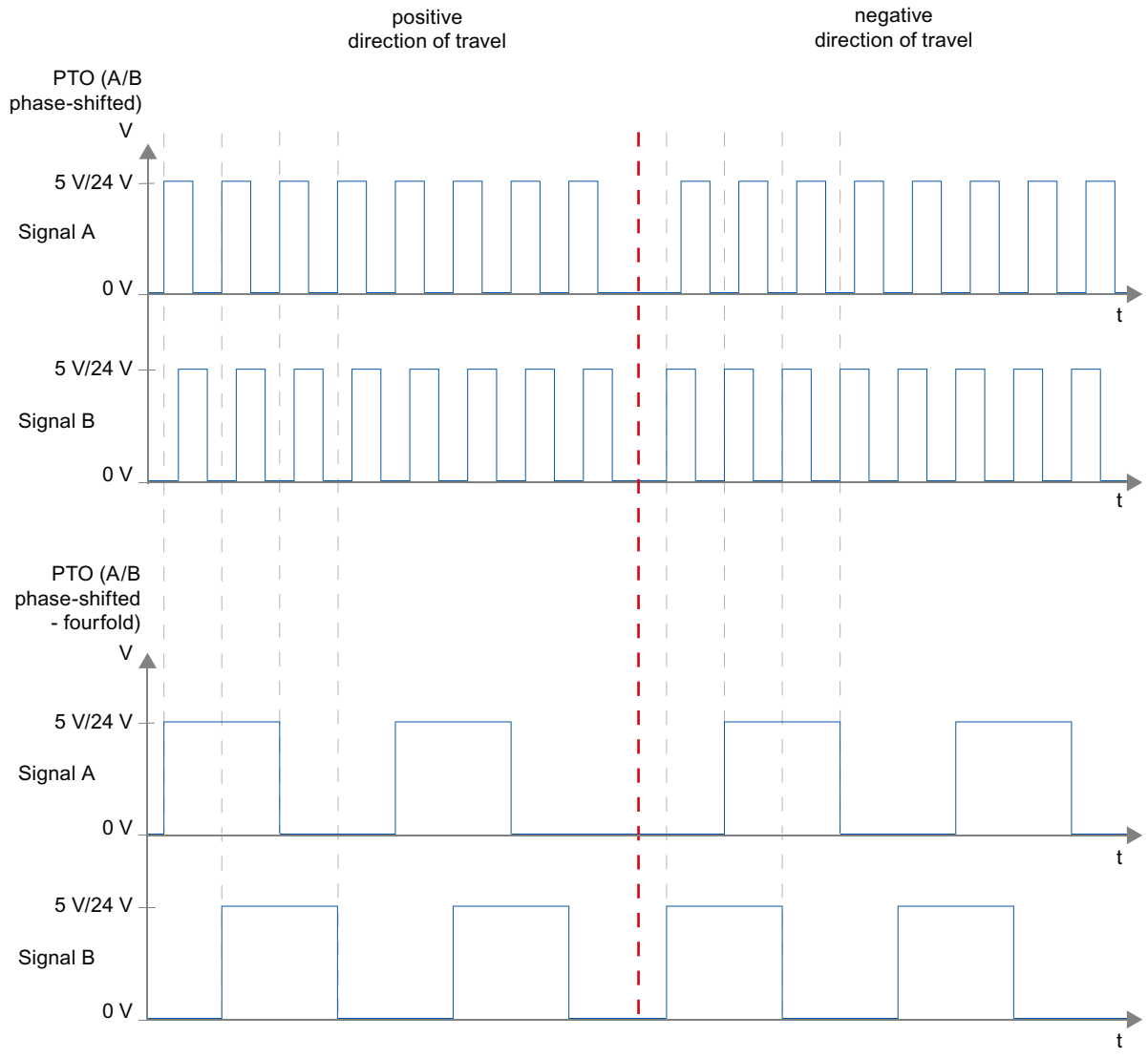
PTO (A/B phase-shifted - fourfold) (as of V4)

The positive and negative edges of both outputs are evaluated for this signal type. A pulse period has four edges with two phases (A and B). The pulse frequency at the output is therefore reduced to a quarter.

The pulse is output via the "Signal A" output and phase-shifted via the "Signal B" output. The phase shifting between the outputs defines the direction of rotation:

- Signal A leads signal B by 90° ⇒ positive direction of rotation
- Signal B leads signal A by 90° ⇒ negative direction of rotation

The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.



Invert direction

If you select the "Invert rotation signal" option, the direction logic is inverted:

- **PTO (pulse A and direction B)**
  - 0 V at direction output (low level) ⇒ positive direction of rotation
  - 5 V/24 V at direction output (high level) ⇒ negative direction of rotation

  The specified voltage depends on the hardware used. The voltages indicated do not apply to the differential outputs of CPU 1217.
- **PTO (count up A, count down B)**
  The outputs "Pulse output down" and "Pulse output up" are swapped.

- **PTO (A/B phase-shifted)**
  The "Signal A" and "Signal B" outputs are swapped.
- **"PTO (A/B phase-shifted - fourfold)**
  The "Signal A" and "Signal B" outputs are swapped.

See also

CPU outputs relevant for motion control (Page 7331)

How the pulse interface works (Page 7334)

Hardware and software limit switches (Page 7346)

Jerk limit (Page 7347)

Homing (Page 7348)

Integration of the positioning axis technology object (Page 7357)

Tools of the positioning axis technology object (Page 7360)

13.2.2.2 PROFIdrive/analog drive connection

Drive and encoder connection

A drive and an encoder are assigned to a positioning axis with drive connection via PROFIdrive/ analog drive connection.

Drives with PROFIdrive capability are connected by means of PROFIdrive telegrams. Drives with analog setpoint interfaces are connected using an analog output and an optional enable signal. The setpoint value at the drive is specified either with PROFIdrive telegrams, or with an analog output.

The encoder value is transmitted via PROFIdrive telegrams or a HSC interface.

Options for connecting

Drives with PROFIdrive capability are connected via the PROFINET interface of the CPU.

Drives with analog setpoint interface are connected with the CPU via one of the following connections:

- Analog output via signal board
- Analog output via analog output module

The following connection options are available for an encoder:

- Encoder on drive
- Encoder on high speed counter (HSC)
- Encoder on technology module
- PROFIdrive encoder directly on PROFINET IO

Maximum number of axes

Depending on the controller, you can control up to a maximum of eight drives via PROFIdrive or the analog drive connection.

PROFIdrive

PROFIdrive is the standardized standard profile for drive technology in the connection of drives and encoders via PROFINET IO. Drives and encoders that support the PROFIdrive profile are connected according to the PROFIdrive standard.

Communication between controller and drive/encoder is by means of various PROFIdrive telegrams. Each of the telegrams has a standardized structure. Depending on the application, you can select the applicable telegram. Control words and status words as well as setpoints and actual values are transmitted in the PROFIdrive telegrams.

Telegrams for PROFIdrive

The setpoint of a positioning axis is transmitted to a drive via PROFIdrive telegram 1, 2 or 3. The encoder value is transmitted either in a telegram together with the setpoint (telegram 3), or in a separate encoder telegram (telegram 81 or telegram 83).

The following table shows the supported PROFIdrive telegrams for the assignment of drives and encoders:

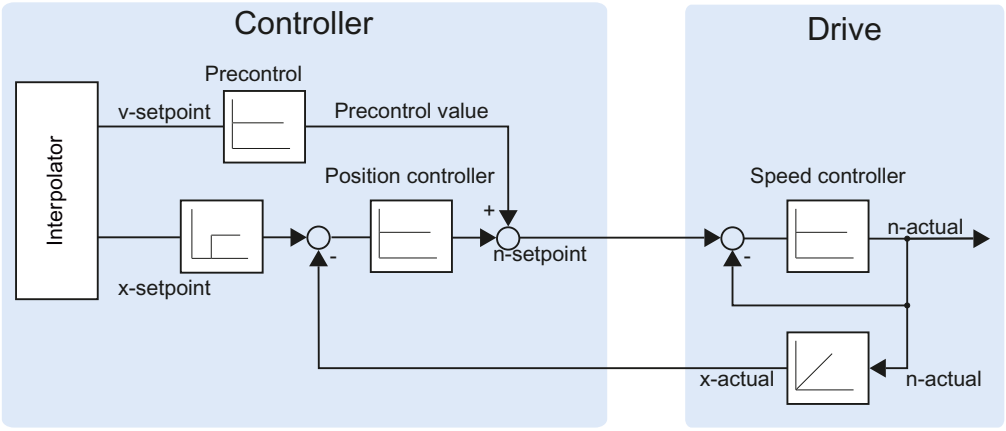
Telegram	Brief description
Standard telegrams	
1	• 16 bit speed setpoint (NSET) • 16 bit actual speed (NACT)
2	• 32 bit speed setpoint (NSET) • 32 bit actual speed (NACT) • Signs of life
3	• 32 bit speed setpoint (NSET) • 32 bit actual speed (NACT) • Encoder value • Signs of life
Standard telegrams encoder	
81	• Encoder value • Signs of life
83	• 32 bit actual speed (NACT) • Encoder value • Signs of life

Closed loop control

If you connect a drive via PROFIdrive or an analog setpoint interface, all motions of the axis are position-controlled. The position controller is a P controller with precontrol of velocity.

Controller structure

The following figure shows the controller structure of an S7-1200 Motion Control:



The MC-Interpolator [OB92] calculates the setpoint position for the axis. The difference between the setpoint and actual position is multiplied by the value of the position controller. The resulting value is added to the precontrol value and output as setpoint speed at the drive.

The encoder records the actual position of the axis and returns it to the controller via a PROFIdrive telegram or a HSC (high speed counter) interface.

Process response

Organization Blocks for Motion Control

Description

When you create a technology object, organization blocks are automatically created for processing the technology objects. The Motion Control functionality of the technology objects creates its own execution level, and is called according to the Motion Control application cycle.

The following blocks are created:

- **MC-Servo [OB91]**
  Calculation of the Position Controller
- **MC-Interpolator [OB92]**
  Evaluation of the motion control instructions, generation of setpoints and monitoring functionality

The organization blocks are protected (know-how protection). The program code cannot be viewed or changed.

The frequency relationship of the two organization blocks to one another is always 1:1. MC-Servo [OB91] is always executed before MC-Interpolator [OB92].

You can set the application cycle and the priority of the organization blocks according to your requirements for control quality and system load.

Application cycle

You can set the application cycle in which the MC-Servo [OB91] is called in the properties of the organization block under "General > Cycle time".

The MC-Servo [OB91] is called cyclically with the specified application cycle.

The selected application cycle must be long enough to be able to process the technology objects in one cycle. If the processing time of the technology objects is longer than the application cycle, overflows (Page 7343) will occur.

You can check the runtime of MC-Servo [OB91] and MC-Interpolator [OB92] with the extended instruction "RT_INFO".

Priority

You can configure the priority of the organization blocks as needed in their properties under "General > Properties > Priority":

- **MC-Servo [OB91]**
  Priority 17 to 26 (default value 25)
- **MC-Interpolator [OB92]**
  Priority 17 to 26 (default value 24)

The priority of MC-Servo [OB91] must be at least one higher than the priority of the MC-Interpolator [OB92].

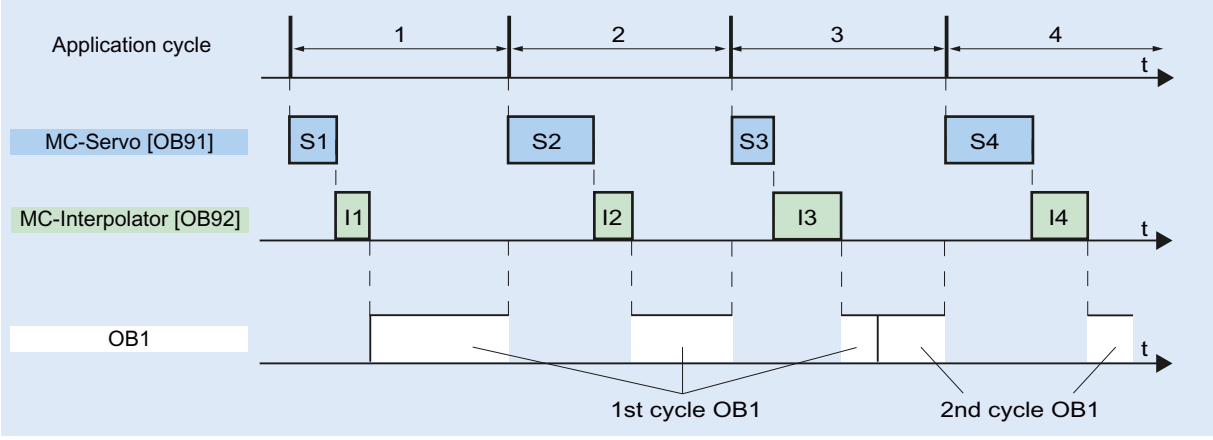
Operational Sequence and Timeouts

When processing the motion control functionality, the organization blocks MC-Servo [OB91] and MC-Interpolator [OB92] are called and processed in each application cycle. The remaining cycle time is available for the processing of your user program.

For error-free program execution, keep to the following rules:

- In each application cycle, MC-Servo [OB91] must be started and executed completely.
- In every application cycle, the relevant MC-Interpolator [OB92] must at least be started.

The following figure shows an example of the error-free operational sequence for the processing of organization block OB1:

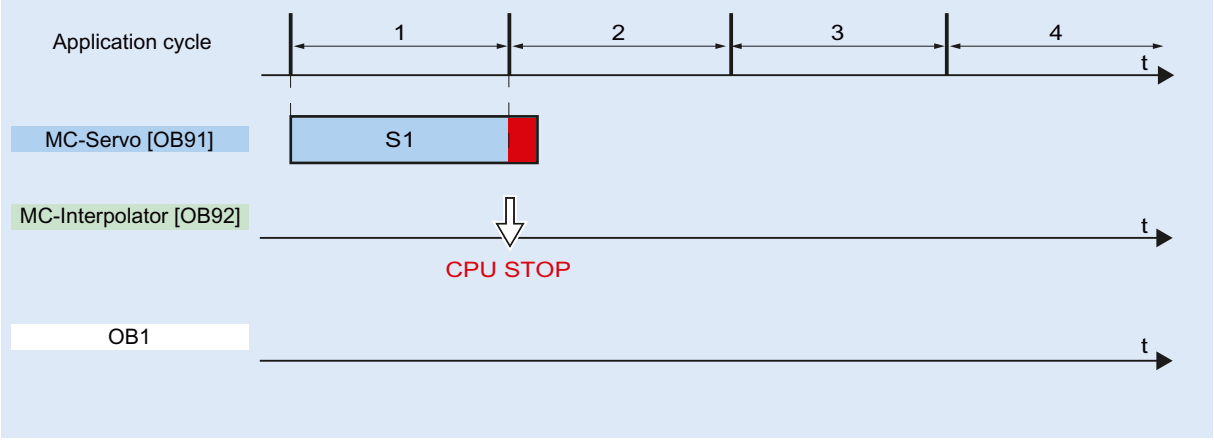


Overflows

If the set application cycle is not adhered to, for example because the application cycle is too short, overflows can occur.

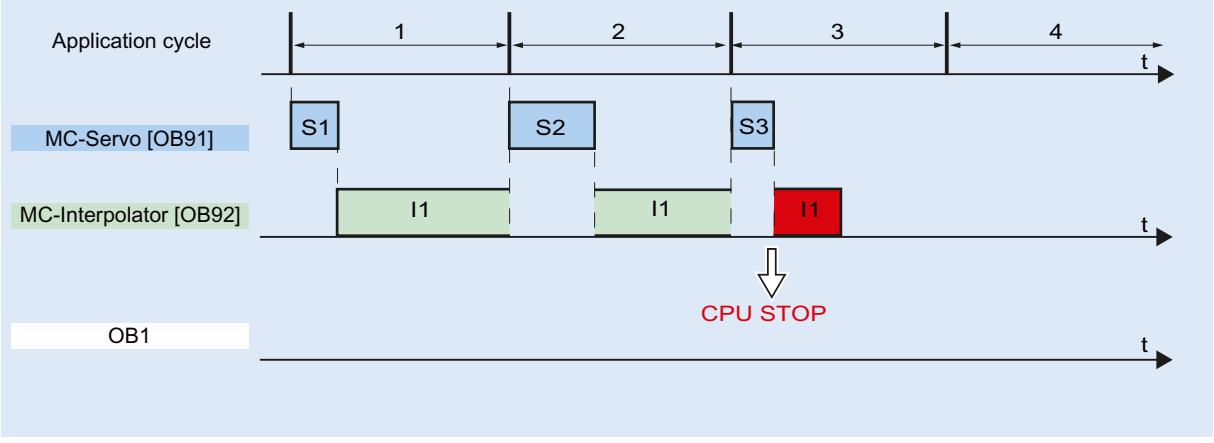
The CPU will not tolerate overflow of MC-Servo [OB91]. An overflow will cause the CPU to change to STOP mode.

The following figure shows the operational sequence if there is an overflow of MC-Servo [OB91]:



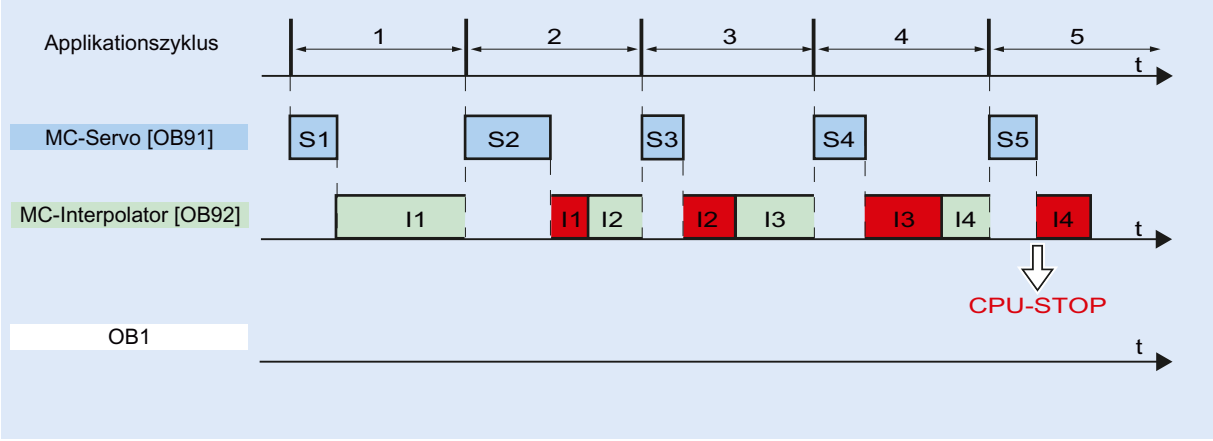
The execution of an MC-Interpolator [OB92] may only be interrupted by one MC-Servo [OB91] call. If more interruptions occur, the CPU switches to STOP mode.

The following figure shows the sequence when an MC-Interpolator [OB92] is interrupted over two time slices:



The CPU tolerates a maximum of three consecutive overflows of MC-Interpolator [OB92]. If more overflows occur, the CPU switches to STOP mode.

The following figure shows the sequence if there are four consecutive individual overflows of MC-Interpolator [OB92]:

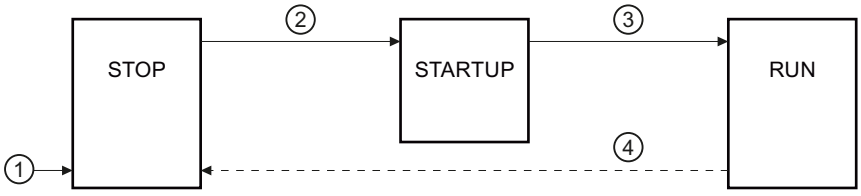


Operating modes

This section examines the behavior of Motion Control in each operating mode, and in the transitions between operating modes. A general description of the operating modes can be found in system manual S7-1200.

Operating modes and transitions

The CPU has three operating modes: STOP, STARTUP and RUN. The following figure shows the operating modes and the operating mode transitions:



STOP mode

In STOP mode the user program is not processed and all process outputs are disabled. Thus no Motion Control jobs are executed.

The technology data blocks are updated.

STARTUP mode

Before the CPU starts processing of the cyclical user program, the startup OBs are processed one time.

In STARTUP mode, the process outputs are disabled. Motion Control jobs are rejected.

The technology data blocks are updated.

RUN mode

The user program is processed in RUN mode.

In RUN mode, the programmed Motion Control jobs are cyclically called and processed.

The technology data blocks are updated.

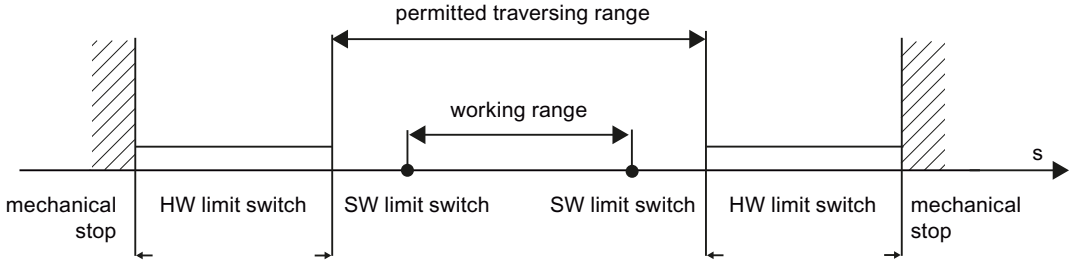
Operating mode transitions

The following table shows the behavior of Motion Control in the transitions between the operating modes:

No.	Operating mode transition	Behavior
①	POWER ON → STOP	The CPU performs a restart of the technology objects. The technology objects are reinitialized with the values from the load memory.
②	STOP → STARTUP	Not relevant to Motion Control.
③	STARTUP → RUN	The process outputs are enabled.
④	RUN → STOP	When the CPU changes to RUN mode after STOP mode, all technology objects are disabled in accordance with the error response "remove enablement". Running Motion Control jobs are terminated.

13.2.2.3 Hardware and software limit switches

Use the hardware and software limit switches to limit the "permitted traversing range" and the "working range" of your positioning axis technology object. The relationships are shown in the following diagram:



Hardware limit switches are limit switches that limit the maximum "permitted traversing range" of the axis. Hardware limit switches are physical switching elements that must be connected to interrupt-capable inputs of the CPU.

Software limit switches limit the "working range" of the axis. They should fall inside the hardware limit switches relative to the traversing range. Since the positions of the software limit switches can be flexibly set, the working range of the axis can be adapted on an individual basis, depending on the current traversing profile. In contrast to hardware limit switches, software limit switches are implemented exclusively via the software and do not require their own switching elements.

Hardware and software limit switches must be activated prior to use in the configuration or in the user program. Software limit switches are only active after homing the axis.

See also

CPU outputs relevant for motion control (Page 7331)

How the pulse interface works (Page 7334)

Relationship between the signal type and the direction of travel (Page 7334)

Jerk limit (Page 7347)

Homing (Page 7348)

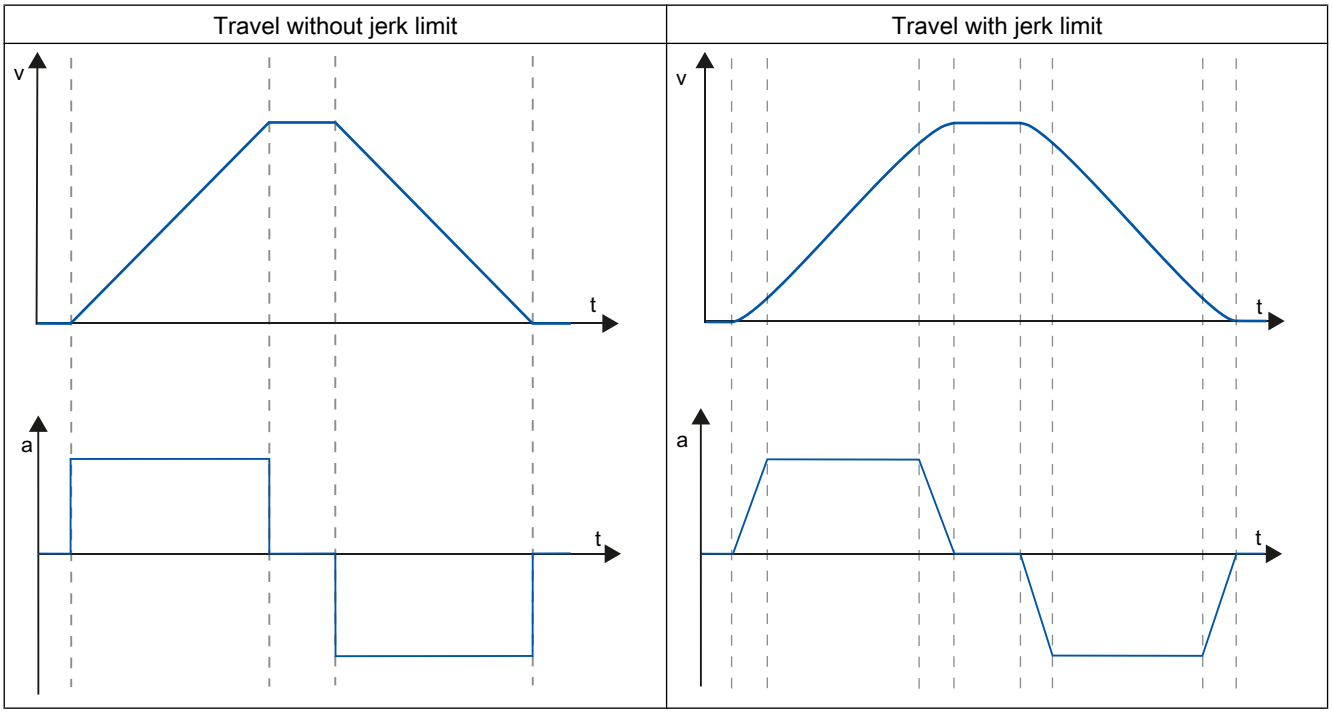
Integration of the positioning axis technology object (Page 7357)

Tools of the positioning axis technology object (Page 7360)

Position limits (Page 7377)

13.2.2.4 Jerk limit

With the jerk limit you can reduce the stresses on your mechanics during an acceleration and deceleration ramp. The acceleration and deceleration value is not changed abruptly when the jerk limiter is active; it is gradually increased and decreased. The figure below shows the velocity and acceleration curve without and with jerk limit.



The jerk limit gives a "smoothed" velocity profile of the axis motion. This ensures, for example, soft starting and braking of a conveyor belt.

See also

Behavior of the axis when using the jerk limit (Page 7386)

CPU outputs relevant for motion control (Page 7331)

How the pulse interface works (Page 7334)

Relationship between the signal type and the direction of travel (Page 7334)

Hardware and software limit switches (Page 7344)

Homing (Page 7348)

Integration of the positioning axis technology object (Page 7357)

Tools of the positioning axis technology object (Page 7360)

13.2.2.5 Homing

Homing means matching the axis coordinates of the technology object to the real, physical location of the drive. For position-controlled axes the entries and displays for the position refer exactly to these axis coordinates. Therefore, agreement between the axis coordinates and the real situation is extremely important. This step is necessary to ensure that the absolute target position of the axis is also achieved exactly with the drive.

In the S7-1200 CPU, axis homing is implemented with the motion control instruction, "MC_Home". The following homing modes exist:

Homing modes

- **Active homing**
  In active homing mode, the motion control instruction "MC_Home" performs the required reference point approach. When the homing switch is detected, the axis is homed according to the configuration. Active traversing motions are aborted.
- **Passive homing**
  During passive homing, the "MC_Home" Motion Control instruction does not carry out any homing motion. The travel required for this step must be implemented by the user via other Motion Control instructions. When the homing switch is detected, the axis is homed according to the configuration. Active traversing motions are not aborted upon start of passive homing.
- **Direct homing absolute**
  The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The value of input parameter "Position" of motion control instruction "MC_Home" is set immediately as the reference point of the axis.
- **Direct homing relative**
  The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The following statement applies to the axis position after homing:
  New axis position = current axis position + value of parameter "Position" of instruction "MC_Home".

See also

CPU outputs relevant for motion control (Page 7331)

How the pulse interface works (Page 7334)

Relationship between the signal type and the direction of travel (Page 7334)

Hardware and software limit switches (Page 7344)

Jerk limit (Page 7345)

Integration of the positioning axis technology object (Page 7357)

Tools of the positioning axis technology object (Page 7360)

Homing (positioning axis technology object as of V2) (Page 7388)

13.2.3 Guidelines on use of motion control

The guidelines described here present the basic procedure for using motion control with the CPU S7-1200.

Requirements

To use the positioning axis technology object, a project with a CPU S7-1200 must be created.

Procedure

Follow the steps below in the order given to use motion control with the CPU S7-1200. Use the following links for this purpose:

1. Adding a positioning axis technology object (Page 7362)
2. Working with the configuration dialog (Page 7363)
3. Download to CPU (Page 7434)
4. Function test of the axis in the commissioning window (Page 7435)
5. Programming (Page 7440)
6. Diagnostics of the axis control (Page 7459)

13.2.4 Using versions

13.2.4.1 Overview of versions

The relationship between the relevant versions for S7-1200 Motion Control can be found in the following table:

Technology version

You can check the currently selected technology version in the "Instructions" task card in folder "Technology > Motion Control > S7-1200 Motion Control" and in the "Add new object" dialog.

Select the technology version in the "Instructions" task card in folder "Technology > Motion Control > S7-1200 Motion Control".

If a technology object with an alternative version is added in the "Add new object" dialog, the technology version will also be changed.

Note

The selection of an alternative technology version will also affect the Motion Control Instructions version (task card).

The technology objects and Motion Control instructions will only be converted to the selected version upon compilation or "Download to device".

Version of the technology object

The version of a technology object can be checked in the inspector window under "Properties > General > Information" in the "Version" box.

Motion Control instruction version

The Motion Control instruction version can be checked in the inspector window under "Properties > General > Information" in the "Version" box.

If the Motion Control instruction version used is not in line with the following compatibility list, the relevant Motion Control instructions will be highlighted in the program editor.

Compatibility list

Technology		CPU	Technology object	Motion Control Instruction
V1.0		V1.0, V2.0, V2.1, V2.2, V3.0	Axis V1.0	MC_Power V1.0 MC_Reset V1.0 MC_Home V1.0 MC_Halt V1.0 MC_MoveAbsolute V1.0 MC_MoveRelative V1.0 MC_MoveVelocity V1.0 MC_MoveJog V1.0
V2.0	Innovations: • Jerk limit • Command table • MC_ChangeDynamic	V2.1, V2.2, V3.0	Axis V2.0, Command table V2.0	MC_Power V2.0 MC_Reset V2.0 MC_Home V2.0 MC_Halt V2.0 MC_MoveAbsolute V2.0 MC_MoveRelative V2.0 MC_MoveVelocity V2.0 MC_MoveJog V2.0 MC_CommandTable V2.0 MC_ChangeDynamic V2.0

Technology		CPU	Technology object	Motion Control Instruction
V3.0	Innovation: Load in RUN mode	V2.2, V3.0, V4.0	Axis V3.0, Command table V3.0	MC_Power V3.0 MC_Reset V3.0 MC_Home V3.0 MC_Stop V3.0 MC_MoveAbsolute V3.0 MC_MoveRelative V3.0 MC_MoveVelocity V3.0 MC_MoveJog V3.0 MC_CommandTable V3.0 MC_ChangeDynamic V3.0
V4.0	Innovations: • MC_ReadParam • MC_WriteParam • Standardization of S7-1200 and S7-1500 Motion Control technology data blocks.	V4.0	Positioning axis V4.0, command table V4.0	MC_Power V4.0 MC_Reset V4.0 MC_Home V4.0 MC_Halt V4.0 MC_MoveAbsolute V4.0 MC_MoveRelative V4.0 MC_MoveVelocity V4.0 MC_MoveJog V4.0 MC_CommandTable V4.0 MC_ChangeDynamic V4.0 MC_ReadParam V4.0 MC_WriteParam V4.0
V5.0	Innovations: • Drive connection by means of PROFIdrive • Analog drive connection • Position control for PROFIdrive/analog drive connection • Position monitoring for PROFIdrive/analog drive connection • MC-Servo [OB91] • MC-Interpolator [OB92]	V4.1	Positioning axis V5.0, command table V5.0	MC_Power V5.0 MC_Reset V5.0 MC_Home V5.0 MC_Halt V5.0 MC_MoveAbsolute V5.0 MC_MoveRelative V5.0 MC_MoveVelocity V5.0 MC_MoveJog V5.0 MC_CommandTable V5.0 MC_ChangeDynamic V5.0 MC_ReadParam V5.0 MC_WriteParam V5.0

See also

Changing a technology version (Page 7352)

Compatibility list of tags (Page 7352)

Status of limit switch (Page 7355)

13.2.4.2 Changing a technology version

Before you can access all the benefits of a new technology version, you may need to set / modify the technology version for existing projects.

Note

Compatibility of the technology object tags

When switching between V1...3 and ≥ V4, please see the compatibility list (Page 7352) if you are using tags of the technology object in your user program, watch tables, etc.

Setting/changing a technology version

To set or change the technology version, follow these steps:

1. Open the program editor (e.g., by opening the OB1).
2. In the "Instructions" task card, open the desired technology version in the "Technology > Motion Control > S7-1200 Motion Control" folder.
3. Save and compile the project. Pay attention to any error information that is displayed during compilation. Deal with the causes of the errors indicated.
4. Check the configuration of the technology objects.
5. If necessary, adapt the tag names in the following objects in line with the compatibility list.

- User program
- Watch tables
- Force tables
- HMI configuration
- Trace configuration

See also

Overview of versions (Page 7347)

Status of limit switch (Page 7355)

13.2.4.3 Compatibility list of tags

The technology data blocks for S7-1200 Motion Control and S7-1500 Motion Control have been standardized within the framework of the V4 technology. As of V4, this has resulted in new tags and tag names for the positioning axis and command table technology objects.

Observe the information in the following tables if you have used tags of the technology objects in the user program and you want to convert the project from V1...3 to V4 or higher (or vice versa).

The tags listed in the "Automatic conversion V1...3 to V4" column are converted automatically when the project is compiled. Tag names in monitoring and force tables or the HMI or trace configuration are not converted.

The following tags are new or have been adapted and may have to be corrected in the user program, watch tables, etc.:

Config tags (positioning axis)

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Axis name>.Config.DynamicDefaults.Acceleration	<Axis name>.DynamicDefaults.Acceleration	Yes
<Axis name>.Config.DynamicDefaults.Deceleration	<Axis name>.DynamicDefaults.Deceleration	Yes
<Axis name>.Config.DynamicDefaults.EmergencyDeceleration	<Axis name>.DynamicDefaults.EmergencyDeceleration	Yes
<Axis name>.Config.DynamicDefaults.Jerk	<Axis name>.DynamicDefaults.Jerk	Yes
<Axis name>.Config.DynamicDefaults.JerkActive	Not available The jerk is activated if the configured jerk is > 0.004 pulse/s^3.	No
<Axis name>.Config.DynamicLimits.MaxVelocity	<Axis name>.DynamicLimits.MaxVelocity	Yes
<Axis name>.Config.DynamicLimits.MinVelocity	<Axis name>.DynamicLimits.MinVelocity	Yes
<Axis name>.Config.General.LengthUnit	<Axis name>.Units.LengthUnit	Yes
<Axis name>.Config.Homing.AutoReversal	<Axis name>.Homing.AutoReversal	Yes
<Axis name>.Config.Homing.Direction	<Axis name>.Homing.ApproachDirection	Yes
<Axis name>.Config.Homing.FastVelocity	<Axis name>.Homing.ApproachVelocity	Yes
<Axis name>.Config.Homing.Offset	<Axis name>.Sensor[1].ActiveHoming.HomePositionOffset	Yes
<Axis name>.Config.Homing.SideActiveHoming	<Axis name>.Sensor[1].ActiveHoming.SideInput	Yes
<Axis name>.Config.Homing.SidePassiveHoming	<Axis name>.Sensor[1].PassiveHoming.SideInput	Yes
<Axis name>.Config.Homing.SlowVelocity	<Axis name>.Homing.ReferencingVelocity	Yes
<Axis name>.Config.Homing.SwitchedLevel	<Axis name>.Sensor[1].ActiveHoming.SwitchLevel <Axis name>.Sensor[1].PassiveHoming.SwitchLevel	No
<Axis name>.Config.Mechanics.InverseDirection	<Axis name>.Actor.InverseDirection	Yes
<Axis name>.Config.Mechanics.LeadScrew	<Axis name>.Mechanics.LeadScrew	Yes
<Axis name>.Config.Mechanics.PulsesPerDriveRevolution	<Axis name>.Actor.DriveParameter.PulsesPerDriveRevolution	Yes
<Axis name>.Config.PositionLimits_HW.Active	<Axis name>.PositionLimitsHW.Active	Yes
<Axis name>.Config.PositionLimits_HW.MaxSwitchedLevel	<Axis name>.PositionLimitsHW.MaxSwitchLevel	Yes
<Axis name>.Config.PositionLimits_HW.MinSwitchedLevel	<Axis name>.PositionLimitsHW.MinSwitchLevel	Yes
<Axis name>.Config.PositionLimits_SW.Active	<Axis name>.PositionLimitsSW.Active	Yes

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Axis name>.Config.PositionLimits_SW.MaxPosition	<Axis name>.PositionLimitsSW.MaxPosition	Yes
<Axis name>.Config.PositionLimits_SW.MinPosition	<Axis name>.PositionLimitsSW.MinPosition	Yes
Not available	<Axis name>.Actor.DirectionMode	No
Not available	<Axis name>.Actor.Type	No
Not available	<Axis name>.Sensor[1].ActiveHoming.Mode	No
Not available	<Axis name>.Sensor[1].PassiveHoming.Mode	No

ErrorBits tags (positioning axis)

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Axis name>.ErrorBits.HwLimitMax	<Axis name>.ErrorBits.HWLimit (Note the new status bits and the section Status of the limit switch (Page 7355).)	No
<Axis name>.ErrorBits.HwLimitMin		
<Axis name>.ErrorBits.SwLimitMaxExceeded	<Axis name>.ErrorBits.SWLimit (Note the new status bits and the section Status of the limit switch (Page 7355).)	No
<Axis name>.ErrorBits.SwLimitMaxReached		
<Axis name>.ErrorBits.SwLimitMinExceeded		
<Axis name>.ErrorBits.SwLimitMinReached		
Not available	<Axis name>.ErrorBits.DirectionFault	No

MotionStatus tags (positioning axis)

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Axis name>.MotionStatus.Distance	<Axis name>.StatusPositioning.Distance	Yes
<Axis name>.MotionStatus.Position	<Axis name>.Position	Yes
<Axis name>.MotionStatus.TargetPosition	<Axis name>.StatusPositioning.TargetPosition	Yes
<Axis name>.MotionStatus.Velocity	<Axis name>.Velocity	Yes

StatusBits tags (positioning axis)

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Axis name>.StatusBits.Homing	<Axis name>.StatusBits.HomingCommand	Yes
<Axis name>.StatusBits.SpeedCommand	<Axis name>.StatusBits.VelocityCommand	Yes
Not available	<Axis name>.StatusBits.HWLimitMaxActive	No

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
Not available	<Axis name>.StatusBits.HWLimitMinActive	No
Not available	<Axis name>.StatusBits.SWLimitMaxActive	No
Not available	<Axis name>.StatusBits.SWLimitMinActive	No

Tags (command table)

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Command table>.Config.Command[n].Position	<Command table>.Command[n].Position	Yes
<Command table>.Config.Command[n].Velocity	<Command table>.Command[n].Velocity	Yes
<Command table>.Config.Command[n].Duration	<Command table>.Command[n].Duration	Yes
<Command table>.Config.Command[n].NextStep	<Command table>.Command[n].NextStep	Yes
<Command table>.Config.Command[n].StepCode	<Command table>.Command[n].StepCode	Yes

See also

Overview of versions (Page 7347)

Changing a technology version (Page 7350)

Tags of the positioning axis technology object as of V4 (Page 7498)

13.2.4.4 Status of limit switch

The status and error bits for the display of the reached limit switch have been adapted in version V4.

In order to replicate the behavior of the error bits of versions V1...3, use the following logical operators:

V1...3	V4 or higher
<Axis name>.ErrorBits.HwLimitMin	<Axis name>.ErrorBits.HWLimit AND <Axis name>.StatusBits.HWLimitMinActive
<Axis name>.ErrorBits.HwLimitMax	<Axis name>.ErrorBits.HWLimit AND <Axis name>.StatusBits.HWLimitMaxActive
<Axis name>.ErrorBits.SwLimitMinReached	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position = <Axis name>.PositioningLimits_SW.MinPosition)
<Axis name>.ErrorBits.SwLimitMinExceeded	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position < <Axis name>.PositioningLimits_SW.MinPosition)

V1...3	V4 or higher
<Axis name>.ErrorBits.SwLimitMaxReached	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position = <Axis name>.PositioningLimits_SW.MaxPosition)
<Axis name>.ErrorBits.SwLimitMaxExceeded	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position > <Axis name>.PositioningLimits_SW.MaxPosition)

See also

Overview of versions (Page 7347)

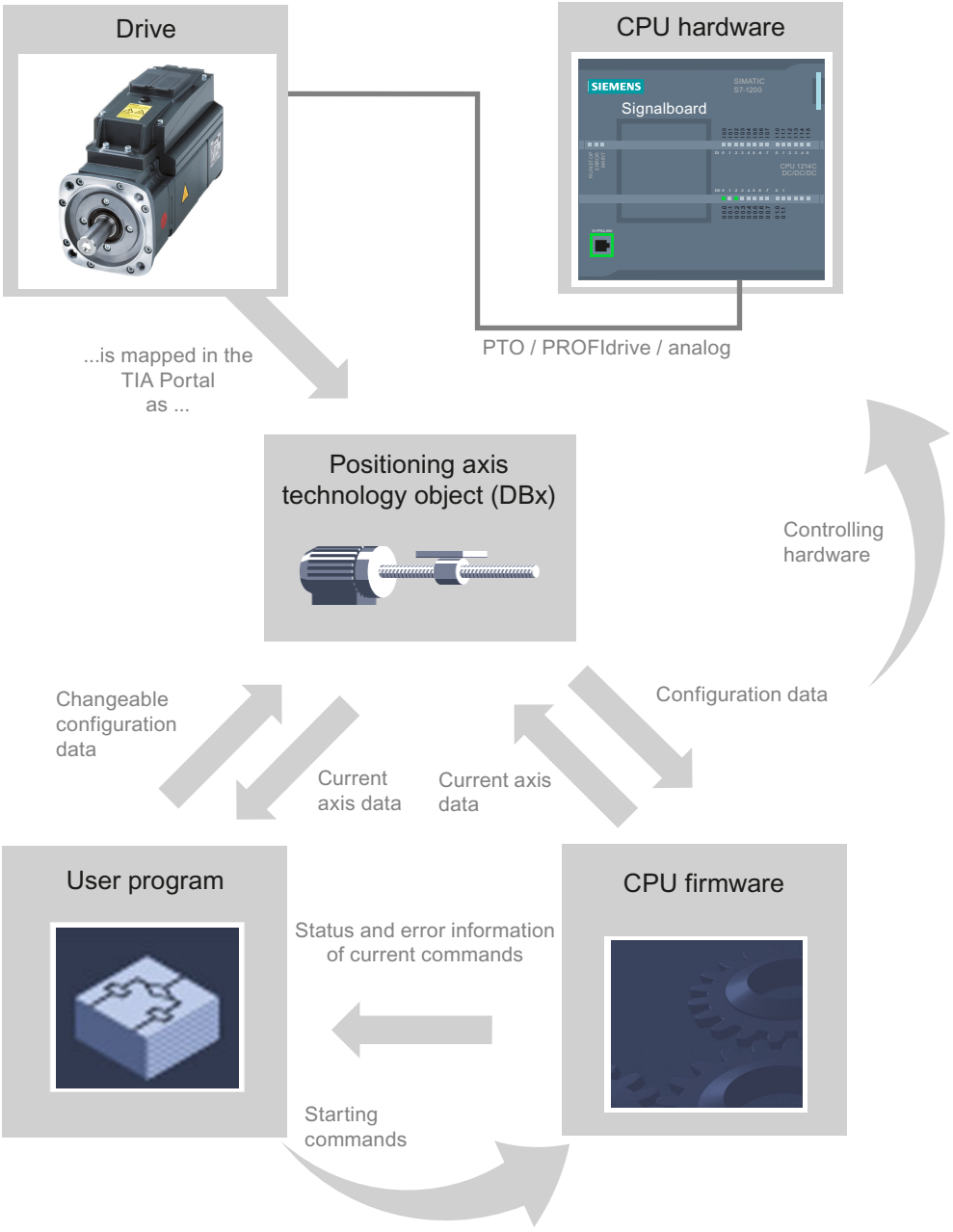
Changing a technology version (Page 7350)

Compatibility list of tags (Page 7350)

13.2.5 Positioning axis technology object

13.2.5.1 Integration of the positioning axis technology object

The following representation shows the relationships between the hardware and software components that are implemented when using the positioning axis technology object:



CPU hardware

The physical drive is controlled and monitored by the CPU hardware.

Drive

The drive represents the unit of power unit and motor. You can use stepper motors and servo motors with pulse, PROFIdrive or analog interfaces.

Positioning axis technology object

The physical drive including mechanics is mapped in the TIA Portal as a positioning axis technology object. To do this, configure the positioning axis technology object with the following parameters:

- Selection of the PTOs (Pulse Train Output)/PROFIdrive drives/analog outputs to be used and configuration of the drive interface
- Parameter for mechanics and gear transmission of the drive (or the machine or system)
- Parameters for position limits and position monitoring
- Parameters for dynamics and homing
- Parameters for the control loop

The configuration of the positioning axis technology object is saved in the technology object (data block). This data block also forms the interface between the user program and the CPU firmware. The current axis data is saved in the data block of the technology object at the runtime of the user program.

User program

You start Motion Control instructions jobs in the CPU firmware with the user program. The following jobs for controlling the axis are possible:

- Enable and disable axis
- Position axis absolutely
- Position axis relatively
- Move axis with velocity set point
- Run axis commands as movement sequence (technology as of V2, PTO only)
- Moving axes in jog mode
- Stop axis
- Reference axis; set reference point
- Change dynamic settings of axis
- Continuously read motion data of the axis
- Write tag of axis
- Acknowledge error

You determine the command parameters with the input parameters of the Motion Control instructions and the axis configuration. The output parameters of the instruction give you up to date information about the status and any errors of the command.

Before starting a command for the axis, you must enable the axis with the Motion Control instruction "MC_Power".

You can read out configuration data and current axis data with the tags of the technology object. You can change individual, changeable tags of the technology object (e.g. the current acceleration) from the user program.

You can also change the dynamic settings of the axis with the Motion Control instruction "MC_ChangeDynamic" and write additional configuration data with "MC_WriteParam". You can read the current motion status of the axis with the Motion Control instruction "MC_ReadParam".

CPU firmware

The motion control jobs started in the user program are processed in the CPU firmware. When using the axis control panel, Motion Control jobs are triggered by operating the axis control panel. The CPU firmware performs the following jobs depending on the configuration:

- Calculate the exact motion profile for motion jobs and emergency stop situations
- Position control for drive connection via PROFIdrive/analog drive connection
- Control of the pulse and direction signal for drive connection via PTO
- Control of the drive enable
- Monitoring of the drive and the hardware and software limit switches
- Up to date feedback of status and error information to the Motion Control instructions in the user program
- Writing of current axis data into the data block of the technology object

See also

Tags of the positioning axis technology object as of V4 (Page 7498)

CPU outputs relevant for motion control (Page 7331)

Relationship between the signal type and the direction of travel (Page 7334)

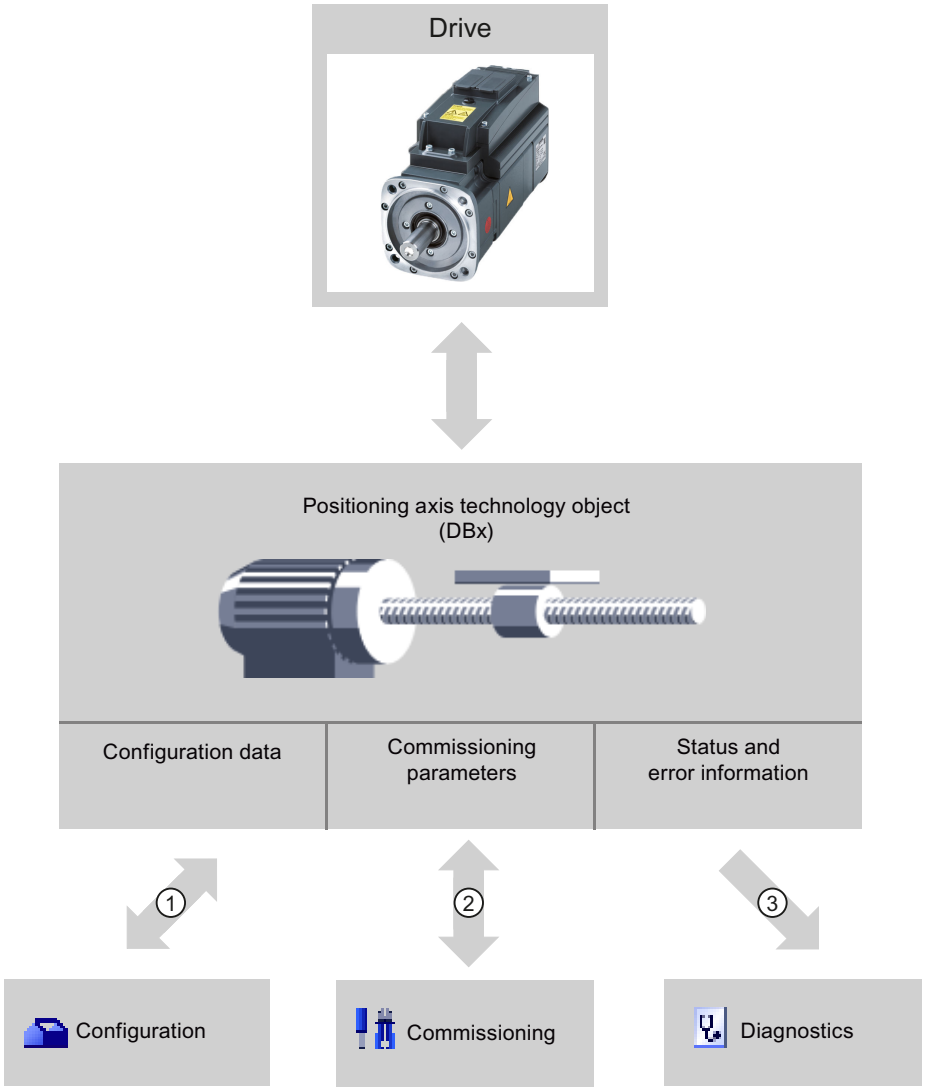
Tools of the positioning axis technology object (Page 7360)

Hardware and software limit switches (Page 7344)

Homing (Page 7346)

13.2.5.2 Tools of the positioning axis technology object

The TIA Portal provides the "Configuration", "Commissioning", and "Diagnostics" tools for the positioning axis technology object. The following representation shows the interaction of the three tools with the technology object and the drive:



①	Reading and writing of configuration data of the technology object
②	Drive control via the technology object. Reading axis status for display on the control panel
③	Readout of the current status and error information of the technology object

Configuration

Use the "Configuration" tool to configure the following properties of the positioning axis technology object:

- Selection of the PTOs (Pulse Train Output)/PROFIdrive drives/analog outputs to be used and configuration of the drive interface
- Properties of the mechanics and the transmission ratio of the drive (or machine/plant)
- Properties of the position limits and the position monitoring
- Properties of the dynamics and the homing
- Parameters of the control loop

Save the configuration in the data block of the technology object.

Commissioning

Use the "Commissioning" tool to test the function of your axis without having to create a user program. When the tool is started, the axis control panel will be displayed. The following commands are available on the axis control panel:

- Enabling and disabling the axis
- Move axis in jog mode
- Position axis in absolute and relative terms
- Home axis
- Acknowledge errors

The dynamic values can be adjusted accordingly for the motion commands. The axis control panel also shows the current axis status.

With drive connection via PROFIdrive / analog output, tuning supports you in determining the optimal gain for the control loop.

Diagnostics

Use the "Diagnostics" tool to keep track of the current status and error information for the axis and drive.

See also

CPU outputs relevant for motion control (Page 7331)

Relationship between the signal type and the direction of travel (Page 7334)

Integration of the positioning axis technology object (Page 7355)

Hardware and software limit switches (Page 7344)

Homing (Page 7346)

Configuring the positioning axis technology object (Page 7363)

Axis control panel (Page 7435)

Axis - Diagnostics (Page 7459)

13.2.5.3 Adding a positioning axis technology object

Requirements

A project with a CPU S7-1200 has been created.

Procedure

To add a positioning axis technology object in the project tree, follow these steps:

1. Open the "CPU > Technology objects" folder in the project tree.
2. Double-click the "Add new object" command.
   The "Add new object" dialog opens.
3. Select the "Motion Control" technology.
4. Open the "Motion Contro > S7-1200 Motion Control" folder.
5. Select the desired technology version in the "Version" column.
6. Select the "TO_PositioningAxis" object.
7. Enter the name of the axis in the "Name" input box.
8. To change the automatically assigned data block number, select the "Manual" option.
9. To display additional information about the technology object, click "Additional information".
10. Confirm your entry with "OK".

Result

The new technology object is created and saved to the "Technology objects" folder in the project tree.

The organization blocks MC-Servo [OB91] and MC-Interpolator [OB92] are automatically created in the "Program blocks" folder. The technology objects are processed in these organization blocks. The position controller is calculated in the MC-Servo [OB91]. The MC-Interpolator [OB92] takes over the evaluation of the Motion Control instructions, the setpoint generation and the monitoring functionality.

See also

Guidelines on use of motion control (Page 7347)

13.2.5.4 Configuring the positioning axis technology object

Working with the configuration dialog

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the group of the required technology object in the project tree.
2. Double-click the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters contain all the parameters which must be configured for a functioning axis.
- **Extended parameters**
  The advanced parameters include parameters to adapt to your drive or your plant.

Configuration window icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values you can use the technology object without additional changes.
	The configuration contains values set by the user and is complete All input fields of the configuration contain valid values and at least one preset value has changed.
	The configuration is incomplete or incorrect At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to display the cause of the error.
	The configuration is valid but contains warnings For example, only one hardware limit switch is configured. Depending on the plant, the lacking configuration of a hardware limit switch may result in a hazard. The corresponding field or the drop-down list is displayed on a yellow background.

See also

Guidelines on use of motion control (Page 7347)

Basic parameters (Page 7364)

Extended parameters (Page 7375)

Compare values

If there is an online connection to the CPU, the "Compare values" function appears in the configuration of the technology object.

The "Compare values" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of actual values in the project
- Transfer of start values of the project to the CPU as actual values

Icons and operator controls

If there is an online connection to the CPU, the actual values are displayed at the parameters.

In addition to the actual values of the parameters, the following symbols appear:

Icon	Description
	Start value in CPU matches the configured Start value in the project
	Start value in CPU does not match the configured Start value in the project
	The comparison of the Start value in CPU with the configured Start value in the project cannot be performed
	Use the button to show the start value of the CPU and the start value of the project for the respective parameter.

The actual value and the start value in the project can be changed directly and then downloaded to the CPU. The change of the actual value is transferred directly to the CPU for directly modifiable parameters.

Basic parameters

Configuration - General

Configure the basic properties of the positioning axis technology object in the "General" configuration window.

Axis name

Define the name of the axis or the name of the positioning axis technology object in this field. The technology object is listed under this name in the project tree.

Drive

Select the type of drive connection:

- **PTO (Pulse Train Output)**
  The drive is connected via a pulse generator output, an optional enable output and an optional ready input.
- **Analog drive connection**
  The drive is connected via an analog output, an optional enable output and an optional ready input.
  All movements of the axis are position-controlled.
- **PROFIdrive**
  The drive is connected via PROFINET. Communication between controller and drive is by means of PROFIdrive telegrams.
  All movements of the axis are position-controlled.

If you select the "Analog drive connection" or "PROFIdrive", additional elements are added to the navigation of the configuration:

- Encoder
- Modulo
- Position monitoring
- Control loop

In the additional configuration windows, you configure the encoders that are to be connected and the resulting options for position control and position monitoring.

Unit of measurement

Select the desired unit for the dimension system of the axis in the drop-down list. The selected unit is used for further configuration of the positioning axis technology object and for displaying the current axis data.

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this unit.

Note

Later changes to the dimension system may not be converted correctly in all the configuration windows of the technology object. In this case check the configuration of all axis parameters.

You may have to adapt the values of the input parameters of the Motion Control instructions to the new unit of measurement in the user program.

See also

CPU outputs relevant for motion control (Page 7331)

Relationship between the signal type and the direction of travel (Page 7334)

Configuration - General ("Axis" technology object V1...3) (Page 7537)

Configuration - Drive

Configuration - Drive - PTO (Pulse Train Output)

In the "Drive" configuration window, configure the pulse generator and the enable and feedback of the drive.

Hardware interface

The pulses are output to the power unit of the drive by fixed assigned digital outputs.

In CPUs with relay outputs, the pulse signal cannot be output at these outputs because the relays do not support the necessary switching frequencies. To be able to work with the PTO (Pulse Train Output) on these CPUs, you must use a signal board with digital outputs.

Note

The PTO requires the functionality of a high-speed counter (HSC). An internal HSC is used for this, the count of which cannot be evaluated.

Selecting the pulse generator

In the drop-down list, select the PTO (Pulse Train Output) to control the stepper motor or servo motor by means of pulse interface. If you have not used the pulse generators and high-speed counters elsewhere in the device configuration, the hardware interface can be configured automatically. In this case, the PTO selected in the drop-down list is displayed with a white background.

The "Device configuration" button takes you to the parameter assignment of the pulse options in the device configuration of the CPU.

Signal type

Select the signal type in the drop-down list. The following signal types are available:

- **PTO (pulse A and direction B)**
  A pulse output and a direction output are used for controlling the stepper motor.
- **PTO (count up A, count down B)**
  One pulse output each for motion in positive direction and negative direction is used for controlling the stepper motor.
- **PTO (A/B phase-shifted)**
  Both pulse outputs for Phase A and for Phase B run at the same frequency.
  The period of the pulse outputs is evaluated at the drive end as a step.
  The phase offset between Phase A and Phase B determines the direction of the motion.
- **PTO (A/B phase offset - quadruple)**
  Both pulse outputs for Phase A and for Phase B run at the same frequency.
  All positive edges and all negative edges of Phase A and Phase B are evaluated as a step at the drive end.
  The phase offset between Phase A and Phase B determines the direction of the motion.

The following table shows the parameters to be configured depending on the signal type:

Signal type/parameter		Description
PTO (pulse A and direction B)		
	Pulse output	Select the pulse output for motions in positive direction in this field. You can select the output using a symbolic address or assign it to an absolute address.
	Activate direction output	With this option, you enable or disable the direction output. The motion direction is restricted when you disable the direction output.
	Direction output	Select the output for the direction output in this field. You can select the output using a symbolic address or assign it to an absolute address.
PTO (count up A, count down B)		
	Pulse output forward	Select the pulse output for motions in positive direction in this field. You can select the output using a symbolic address or assign it to an absolute address.
	Pulse output backward	Select the pulse output for motions in negative direction in this field. You can select the output using a symbolic address or assign it to an absolute address.
PTO (A/B phase offset)/PTO (A/B phase offset - quadruple)		
	Signal A	Select the pulse output for Phase A signals in this field. You can select the output using a symbolic address or assign it to an absolute address.
	Signal B	Select the pulse output for Phase B signals in this field. You can select the output using a symbolic address or assign it to an absolute address.

Drive enable and feedback

In this area, you configure the output for drive enable and the input for the "Drive ready" feedback of the drive:

- **Selection of enable output**
  Select the enable output for the drive enable in this field.
- **Selection of ready input**
  Select the ready input for the "Drive ready" feedback of the drive in this field

Drive enable is controlled by Motion Control instruction "MC_Power" and enables power to the drive. The drive signals "Drive ready" to the CPU if it is ready to start executing movement after receiving the drive enable.

If the drive does not have any interfaces of this type, you do not have to configure the parameters. In this case, select the value TRUE for the ready input.

Configuration - Drive - Analog drive connection

In the "Drive" configuration window, configure the analog output and the enable and feedback of the drive.

Hardware interface

The reference speed is output to the power unit of the drive by means of permanently assigned analog outputs.

Configure the inputs and outputs for the control of the drive in this area:

- **Analog output**
  In this field, select the PLC tag of the analog output via which the drive is controlled. When you open the autocompletion, all output addresses are displayed with 16 bits (WORD, INT, UINT).
  You can also enter an address, for example QW20. If the address is valid, the name "Axis_1_AnalogOutput" for this address is generated and inserted in the tag table.
- **Selection of enable output**
  Select the enable output for the drive enable in this field.
- **Selection of ready input**
  Select the ready input for the "Drive ready" feedback of the drive in this field

Drive enable is controlled by Motion Control instruction "MC_Power" and enables power to the drive. The drive signals "Drive ready" to the CPU if it is ready to start executing movement after receiving the drive enable. If the drive does not have any interfaces of this type, you do not have to configure the parameters. In this case, select the value TRUE for the ready input.

Data exchange with the drive

In this area, you can configure the scaling of the setpoint speed:

- **Reference speed**
  The reference speed of the drive is the speed, with which the drive spins when there is an output of 100% at the analog output. The reference speed must be configured in the drive, and transferred into the configuration of the technology object.
  The analog value that is output at 100% depends on the type of the analog output. As an example, for an analog output with ±10 V, the value 10 V is output at 100%.
  Analog outputs can be overloaded by approximately 17%. If the drive permits the overloading, you can use this to operate an analog output in the -117% to 117% range.
- **Maximum speed**
  In this field, specify the maximum speed of the drive.
- **Invert drive direction**
  To invert the rotation direction of the drive, select the check box.

Configuration - Drive - PROFIdrive

In the "Drive" configuration window, select the PROFIdrive drive and configure the data exchange between the drive and controller.

Selection of PROFIdrive drive

In the "Drive" field, select an already configured PROFIdrive drive.

Data exchange with the drive

In this area, you can configure the data exchange between the drive and controller:

- **Telegram**
  In the drop-down list, select the telegram of the drive. The specification must match the device configuration of the drive.
- **Input/output address**
  The fields show the symbolic and absolute input and output address of the telegram.
- **Reference speed**
  In this field, configure the reference speed of the drive in accordance with the manufacturer's specifications. The drive speed is output in percent in the -200% to 200% range.
- **Maximum speed**
  In this field, specify the maximum speed of the drive.
- **Invert drive direction**
  To invert the rotation direction of the drive, select the check box.

Configuration - Encoder

Encoder connection

Depending on the selection of the encoder connection, you configure various parameters in the "Encoder" configuration window. The following encoder connections are possible:

- Encoder on drive (Page 7369)
- Encoder on high-speed counter (HSC) (Page 7371)
- Encoder on technology module (TM) (Page 7372)
- PROFIdrive encoder on PROFINET (Page 7373)

Configuration - Encoder - Encoder on drive

Data exchange with encoder

In this area, you can configure the data exchange between the encoder and controller:

- **Telegram**
  In the drop-down list, select the telegram of the encoder. The specification must match the device configuration.
- **Input/output address**
  The fields show the symbolic and absolute input and output address of the telegram.

Encoder type

Select the encoder type in the "Encoder type" box. The following encoder types can be selected:

- **Linear incremental**
- **Linear absolute**
- **Rotary incremental**
- **Rotary absolute**

Configure the various parameters depending on the selected encoder type. Depending on the selected encoder type, configure the following parameters:

Encoder type/parameter		Description
Linear incremental		
	Distance between two increments	In this field, you configure the distance between two steps of the encoder.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.
Linear absolute		
	Distance between two increments	In this field, you configure the distance between two steps of the encoder.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Fine resolution - Bits in abs. actual value (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.
Rotary incremental		
	Steps per revolution	In this field, configure the number of steps that the encoder resolves per revolution.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.
Rotary absolute		
	Steps per revolution	In this field, configure the number of steps that the encoder resolves per revolution.
	Number of revolutions	In this field, configure the number of revolutions that the absolute value encoder can detect.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Fine resolution - Bits in abs. actual value (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.

Configuration - Encoder - Encoder on high-speed counter (HSC)

Selection of high-speed counter (HSC)

In the "Select high-speed counter" box, select the high-speed counter to which the encoder transfers the actual value.

Check the filter times of the two high-speed counter digital inputs that are used. The filter times should be as short as possible so that the pulses can be reliably recorded.

HSC interface

Select the operating mode of the high-speed counter in the "Operating mode" box.

Depending on the operating mode, configure the various inputs:

Operating mode/parameter		Description
Two-phase		
	Clock generator forward	Select the pulse output for counting up in this field. You can select the output using a symbolic address or assign it to an absolute address. The frequency and the location (on-board, signal board) of the input are displayed next to the address box.
	Clock generator backward	Select the pulse output for counting down in this field. You can select the output using a symbolic address or assign it to an absolute address. The frequency and the location (on-board, signal board) of the input are displayed next to the address box.
A/B counter / A/B counter quadruple		
	Clock generator A	Select the pulse output for Phase A signals in this field. You can select the output using a symbolic address or assign it to an absolute address. The frequency and the location (on-board, signal board) of the input are displayed next to the address box.
	Clock generator B	Select the pulse output for Phase B signals in this field. You can select the output using a symbolic address or assign it to an absolute address. The frequency and the location (on-board, signal board) of the input are displayed next to the address box.

Encoder type

Select the encoder type in the "Encoder type" box. The following encoder types can be selected:

- **Linear incremental**
- **Rotary incremental**

Configure the various parameters depending on the selected encoder type. Depending on the selected encoder type, configure the following parameters:

Encoder type/parameter		Description
Linear incremental		
	Distance between two increments	In this field, you configure the distance between two steps of the encoder.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.
Rotary incremental		
	Steps per revolution	In this field, configure the number of steps that the encoder resolves per revolution.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.

Configuration - Encoder - Encoder on technology module (TM)

Selection of technology module (TM)

In the "Technology module (TM)" box, select the technology module to which the encoder is connected.

Data exchange with encoder

In this area, you can configure the data exchange between the encoder and controller:

- **Telegram**
  In the drop-down list, select the telegram of the encoder. The specification must match the device configuration.
- **Input/output address**
  The fields show the symbolic and absolute input and output address of the telegram.

Encoder type

Select the encoder type in the "Encoder type" box. The following encoder types can be selected:

- **Linear incremental**
- **Linear absolute**
- **Rotary incremental**
- **Rotary absolute**

Configure the various parameters depending on the selected encoder type. Depending on the selected encoder type, configure the following parameters:

Encoder type/parameter		Description
Linear incremental		
	Distance between two increments	In this field, you configure the distance between two steps of the encoder.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.
Linear absolute		
	Distance between two increments	In this field, you configure the distance between two steps of the encoder.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Fine resolution - Bits in abs. actual value (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.
Rotary incremental		
	Steps per revolution	In this field, configure the number of steps that the encoder resolves per revolution.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.
Rotary absolute		
	Steps per revolution	In this field, configure the number of steps that the encoder resolves per revolution.
	Number of revolutions	In this field, configure the number of revolutions that the absolute value encoder can detect.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Fine resolution - Bits in abs. actual value (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.

Configuration - Encoder - PROFIdrive encoder on PROFINET

Encoder selection

In the "PROFIdrive encoder" box, select a PROFIdrive encoder on PROFINET.

Data exchange with encoder

In this area, you can configure the data exchange between the encoder and controller:

- **Telegram**
  In the drop-down list, select the telegram of the encoder. The specification must match the device configuration.
- **Input/output address**
  The fields show the symbolic and absolute input and output address of the telegram.

Encoder type

Select the encoder type in the "Encoder type" box. The following encoder types can be selected:

- **Linear incremental**
- **Linear absolute**
- **Rotary incremental**
- **Rotary absolute**

Configure the various parameters depending on the selected encoder type. Depending on the selected encoder type, configure the following parameters:

Encoder type/parameter		Description
Linear incremental		
	Distance between two increments	In this field, you configure the distance between two steps of the encoder.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.
Linear absolute		
	Distance between two increments	In this field, you configure the distance between two steps of the encoder.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Fine resolution - Bits in abs. actual value (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.
Rotary incremental		
	Steps per revolution	In this field, configure the number of steps that the encoder resolves per revolution.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.
Rotary absolute		

Encoder type/parameter		Description
	Steps per revolution	In this field, configure the number of steps that the encoder resolves per revolution.
	Number of revolutions	In this field, configure the number of revolutions that the absolute value encoder can detect.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Fine resolution - Bits in abs. actual value (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).
	Invert encoder direction	To invert the actual value of the encoder, select the check box.

Extended parameters

Mechanics

Configuration - Mechanics - PTO (Pulse Train Output)

Configure the mechanical properties of the drive in the "Mechanics" configuration window.

Pulses per motor revolution

Configure the number of pulses required for one revolution of the motor in this box.

Limits (independent of the selected unit of measurement):

- 0 < Pulse per motor revolution ≤ 2147483647

Distance per motor revolution

In this box, configure the load distance per motor revolution covered by the mechanical system of your unit.

Limits (independent of the selected unit of measurement):

- 0.0 < Distance per revolution ≤ 1.0e12

Permitted direction of rotation (technology version as of V4)

Configure this box to determine whether the mechanics of your system are to move in both directions or only in the positive or negative direction.

If you have not activated the direction output of the pulse generator in the "PTO (pulse A and direction B)" mode, the selection is limited to the positive or negative direction.

Invert direction

You can use the "Invert direction" check box to adapt the control system to the direction logic of the drive.

The direction logic is inverted according to the selected mode of the pulse generator:

- **PTO (pulse A and direction B)**
  - 0 V at direction output ⇒ positive direction of rotation
  - 5 V/24 V at direction output ⇒ negative direction of rotation

  The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.
- **PTO (count up A, count down B)**
  The outputs "Pulse output down" and "Pulse output up" are swapped.
- **PTO (A/B phase-shifted)**
  The "Phase A" and "Phase B" outputs are swapped.
- **"PTO (A/B phase-shifted - fourfold)**
  The "Phase A" and "Phase B" outputs are swapped.

Configuration - Mechanics - PROFIdrive/analog drive connection

Configure the mechanical properties of the drive and its encoder in the "Mechanics" configuration window.

Encoder mounting type

In the drop-down list, select how the encoder is mounted on the mechanism. The following encoder installation types are possible:

- **On motor shaft**
- **External measuring system**

Position parameters

Depending on the selected encoder installation type, configure the following position parameters:

Encoder installation type/position parameter		Description
On the motor shaft		
	Load motion per motor revolution	In this field, configure the load distance for one motor revolution.
External measuring system		
	Load motion per motor revolution	In this field, configure the load distance for one motor revolution.
	Distance per encoder revolution	In this field, configure the distance recorded by the external measuring system per encoder revolution.

Configuration - Modulo (PROFIdrive/analog drive connection only)

If an axis is moved in only one direction, the position value continually increases. You can use the "modulo" setting to limit the position value to a recurring reference system.

When "modulo" is enabled, the position value of the technology object is represented by means of a recurring modulo range. The modulo range is defined by the start value and the length.

For example, to limit the position value of an axis to one full rotation, the modulo range can be defined with start value = 0° and length = 360°. With an encoder resolution of 0.1°/encoder step, the position value is represented in the modulo range 0.0° to 359.9°.

Enable modulo

Select the "Enable modulo" check box to use a recurring reference system for the axis (for example, 0.0° to 359.9°).

Modulo start value

In this field, define the position at which the modulo range should begin (for example, 0°).

Modulo length

In this field, define the length of the modulo range (for example, 360°).

Position limits

Requirements for hardware limit switches

Use only hardware limit switches that remain permanently switched after being approached. This switching status may only be revoked after a return to the valid travel range.

See also

Configuration - Position limits (Page 7377)

Behavior of axis when position limits is tripped (Page 7379)

Changing the position limits configuration in the user program (Page 7381)

Configuration - Position limits

Configure the hardware and software limit switches of the axis in the "Position limits" configuration window.

Enable HW limit switches

Activate the function of the low and high hardware limit switch with this check box. The hardware limit switches can be used for purposes of direction reversal during a homing procedure. For details, refer to the configuration description for homing.

Input low / high HW limit switch

Select the digital input for the low or high hardware limit switch from the drop-down list. The input must be interrupt-capable. The digital onboard CPU inputs and the digital inputs of a plugged signal board can be selected as inputs for the HW limit switches.

Note

The digital inputs are set to a filter time of 6.4 ms by default. If these are used as hardware limit switches, undesired decelerations may occur. If this occurs, reduce the filter time for the relevant digital inputs.

The filter time can be set under "Input filter" in the device configuration of the digital inputs.

Select level

In the drop-down list, select the signal level available at the CPU when the hardware limit switch is approached.

- Selection of "Low level" (normally closed contact)
  0 V (FALSE) at CPU input corresponds to hardware limit switch approached
- Selection of "High level" (normally open contact)
  5 V / 24 V (TRUE) at the CPU input = hardware limit switch approached (the actual voltage depends on the hardware used)

Enable SW limit switches

Activate the function of the low and high software limit switch with this check box.

Note

The enabled software limit switch only affects a homed axis.

High and low software limit switch

Enter the position value of the low and high software limit switch in these boxes.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ low software limit switch ≤ 1.0e12
- -1.0e12 ≤ high software limit switch ≤ 1.0e12

The value of the high software limit switch must be greater than or equal to the value of the low software limit switch.

See also

Requirements for hardware limit switches (Page 7375)

Behavior of axis when position limits is tripped (Page 7379)

Changing the position limits configuration in the user program (Page 7381)

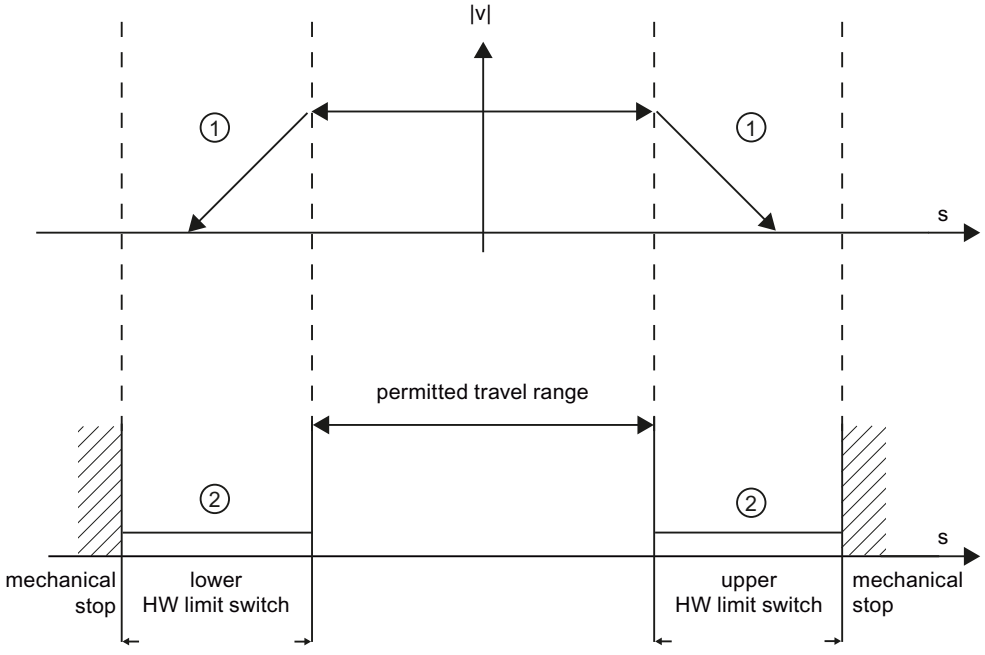
Configuration - Homing - Active (Page 7388)

Behavior of axis when position limits is tripped

Behavior of axis when hardware limit switches are approached

When a hardware limit switch is approached, the axis behaves differently depending on the drive connection:

- **Drive connection via PROFIdrive / analog output**
  When a hardware limit switch is approached, the axis is disabled and, depending on the configuration, braked at the drive and brought to a standstill. You must select the deceleration sufficiently large in the drive so that the axis stops reliably before the mechanical stop.
- **Drive connection by means of PTO (Pulse Train Output)**
  When the hardware limit switches are approached, the axis brakes to a standstill at the configured emergency deceleration. You must select the emergency deceleration sufficiently large so that the axis stops reliably before the mechanical stop. The following diagram presents the behavior of the axis after it approaches the hardware limit switches:



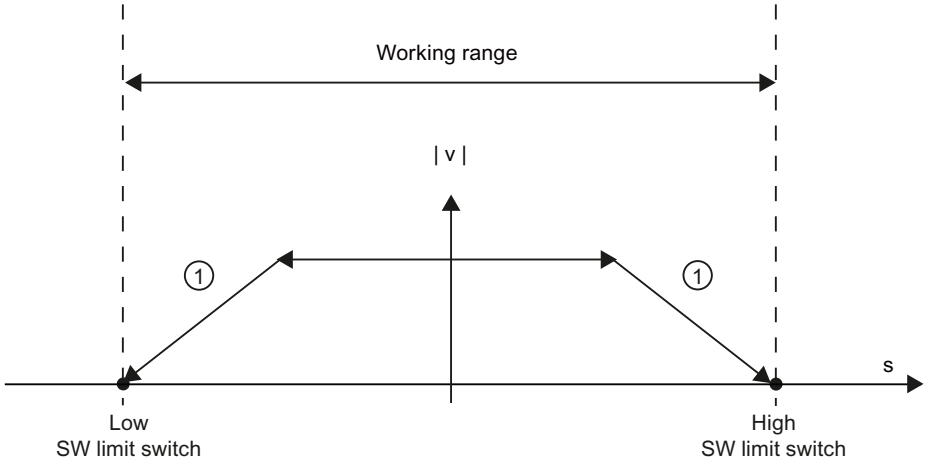
①	The axis brakes to a standstill at the configured emergency stop deceleration.
②	Range in which the HW limit switches signal the status "approached".

The "HW limit switch approached" error is displayed in the motion control instruction to be initiated, in "MC_Power", and in the technology object tags. Instructions for eliminating errors can be found in the Appendix under "List of ErrorIDs and ErrorInfos".

Behavior of axis when software limit switches are reached

If software limit switches are activated, an active motion is stopped at the position of the software limit switch. The axis is braked at the configured deceleration.

The following diagram presents the behavior of the axis until it reaches the software limit switches:



①	The axis brakes to a standstill at the configured deceleration.

The "SW limit switch approached" error is displayed in the initiating Motion Control instruction, at "MC_Power", and in the technology object tags. Instructions for eliminating errors can be found in the Appendix under "List of ErrorIDs and ErrorInfos".

When a software limit switch is overtraveled, the axis behaves differently depending on the drive connection:

- **Drive connection via PROFIdrive / analog output**
  When a software limit switch is overtraveled, the axis is disabled and, depending on the configuration, braked at the drive and brought to a standstill.
- **Drive connection by means of PTO (Pulse Train Output)**
  You can learn about the behavior of the axis when a software limit switch is overtraveled in the sections "Software limit switches in conjunction with a homing operation (Page 7470)" and "Software limit switches in conjunction with dynamic changes (Page 7475)".

Use additional hardware limit switches if a mechanical endstop is located after the software limit switches and there is a risk of mechanical damage.

See also

Requirements for hardware limit switches (Page 7375)

Configuration - Position limits (Page 7375)

Changing the position limits configuration in the user program (Page 7381)

Changing the position limits configuration in the user program

You can change the following configuration parameters during runtime of the user program in the CPU:

Hardware limit switches

You can also activate and deactivate the hardware limit switches during runtime of the user program. Use the following technology object tag for this purpose:

- <Axis name>.PositionLimitsHW.Active

Refer to the description of the technology object tags (Page 7498) in the appendix for information on when changes to the configuration parameter take effect.

Software limit switches

You can also activate and deactivate the software limit switches and change their position values during runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.PositionLimitsSW.Active
  for activating and deactivating the software limit switches
- <Axis name>.PositionLimitsSW.MinPosition
  for changing the position of the low software limit switch
- <Axis name>.PositionLimitsSW.MaxPosition
  for changing the position of the high software limit switch

Refer to the description of technology object tags in the appendix for information on when changes to the configuration parameters take effect.

See also

Compatibility list of tags (Page 7350)

MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)

Requirements for hardware limit switches (Page 7375)

Configuration - Position limits (Page 7375)

Behavior of axis when position limits is tripped (Page 7377)

Dynamics

Configuration - Dynamics - General

Configure the maximum velocity, the start/stop velocity, the acceleration and deceleration, and the jerk limit (positioning axis technology object as of V2) of the axis in the "General dynamics" configuration window.

Unit of velocity limits

Select the unit of measurement with which you want to set the velocity limits in the drop-down list. The unit set here depends on the unit of measurement set under "Configuration - General" and serves only for easier input.

Maximum velocity / Start/stop velocity

Define the maximum permissible velocity and the start/stop velocity of the axis in these boxes. The start/stop velocity is the minimum permissible velocity of the axis and can only be configured for drive connection via PTO (Pulse Train Output). For drive connection via PROFIdrive or analog output, the start/stop velocity is fixed at zero.

Limit values:

The limits indicated below refer to the "Pulses/s" unit of measurement:

- **Positioning axis technology object V4 or higher**
  - 1 ≤ start/stop velocity ≤ 20000 (signal board 20 kHz)
    1 ≤ start/stop velocity ≤ 200000 (signal board 200 kHz)
    1 ≤ start/stop velocity ≤ 100000 (onboard CPU outputs 100 kHz)
    1 ≤ start/stop velocity ≤ 30000 (onboard CPU outputs 30 kHz)
    1 ≤ start/stop velocity ≤ 1000000 (onboard CPU outputs 1 MHz CPU 1217)
  - 1 ≤ maximum velocity ≤ 20000 (signal board 20 kHz)
    1 ≤ maximum velocity ≤ 200000 (signal board 200 kHz)
    1 ≤ maximum velocity ≤ 100000 (onboard CPU outputs 100 kHz)
    1 ≤ maximum velocity ≤ 30000 (onboard CPU outputs 30 kHz)
    1 ≤ maximum velocity ≤ 1000000 (onboard CPU outputs 1 MHz CPU 1217)

You can learn about the limits for the technology object positioning axis < V4 in the appendix Outputs of the CPU relevant for Motion Control (technology version V1...3) (Page 7533).

The value of the maximum velocity must be greater or equal to the value of the start/stop velocity.

The limits for other units of measurement must be converted by the user to conform to the given mechanics.

Acceleration / Deceleration - Ramp-up time / Ramp-down time

Set the desired acceleration in the "Ramp-up time" or "Acceleration" boxes. The desired deceleration can be set in the "Ramp-down time" or "Deceleration" boxes.

The relation between the ramp-up time and acceleration and the ramp-down time and deceleration is shown in the following equations:

$$\textit{Ramp-up time} = \frac{\textit{Maximum velocity} - \textit{Start/stop velocity}}{\textit{Acceleration}}$$

$$\textit{Ramp-down time} = \frac{\textit{Maximum velocity - Start/stop velocity}}{\textit{Deceleration}}$$

Motion jobs started in the user program are performed with the selected acceleration / deceleration.

The limits for acceleration and deceleration with drive connection via PTO (Pulse Train Output) can be found in section CPU outputs relevant for motion control (Page 7331).

Note

Changes to the velocity limits ("start/stop velocity" and "maximum velocity") influence the acceleration and deceleration values of the axis. The ramp-up and ramp-down times are retained.

Enable jerk limit (positioning axis technology object as of V2)

Enable the jerk limit with this check box.

Note

If an error occurs, the axis decelerates with the configured emergency stop deceleration. An enabled jerk limit is not considered here.

Smoothing time / jerk (positioning axis technology object as of V2)

You can input the parameters of the jerk limit in the "Smoothing time" box or alternatively in the "Jerk" box.

- Set the desired jerk for acceleration and deceleration ramp in the "Jerk" box.
- Set the desired smoothing time for the acceleration ramp in the "Smoothing time" box.

Note

Smoothing time V2...3

The set smoothing time visible in the configuration only applies to the acceleration ramp.

If the values for acceleration and deceleration differ, the smoothing time of the deceleration ramp is calculated according to the jerk of the acceleration ramp and used. (See also Behavior of the axis when using the jerk limit (Page 7386)

The smoothing time of the deceleration is adapted as follows:

- **Acceleration > deceleration**
  The smoothing time used for the deceleration ramp is shorter than that for the acceleration ramp.
- **Acceleration < deceleration**
  The smoothing time used for the deceleration ramp is greater than that for the acceleration ramp.
- **Acceleration = deceleration**
  The smoothing times of the acceleration and deceleration ramp are equal.

The relation between smoothing times and jerk is shown in the following equation:

$$\text{Smoothing time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Smoothing time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Motion jobs started in the user program are performed with the selected jerk.

The limits for jerk with drive connection via PTO (Pulse Train Output) can be found in section CPU outputs relevant for motion control (Page 7331).

See also

Behavior of the axis when using the jerk limit (Page 7386)

Hardware components for motion control (Page 7329)

CPU outputs relevant for motion control (Page 7331)

Configuration - Dynamics - Emergency stop (Page 7385)

Changing the configuration of dynamics in the user program (Page 7387)

Configuration - Dynamics - Emergency stop

Configure the emergency stop deceleration of the axis in the "Dynamics emergency stop" configuration window. In the event of an error, and when disabling the axis, the axis is brought to a standstill with this deceleration using the Motion Control instruction "MC_Power" (input parameter StopMode = 0 or 2).

Velocity

The velocity values configured in the "General dynamics" configuration window are once again displayed in this information area.

Deceleration

Set the deceleration value for emergency stop in the "Emergency deceleration" or "Emergency stop ramp-down time" field.

The relation between emergency stop ramp-down time and emergency deceleration is shown in the following equation:

$$\textit{Emergency stop ramp-down} = \frac{\textit{Maximum velocity - Start/stop velocity}}{\textit{Emergency stop deceleration}}$$

The specified emergency deceleration must be sufficient to bring the axis to a standstill in a timely manner in the event of an emergency (for example, when the hardware limit switch is approached prior to reaching the mechanical endstop).

The configured maximum velocity of the axis must be used as a basis for selecting the emergency deceleration.

Limit values:

The limits indicated below refer to the "Pulses/s^{2}" unit of measurement.

- As of CPU firmware V3
  0.005 ≤ emergency stop deceleration ≤ 9.5E9
- CPU Firmware V1...2
  0.28 ≤ emergency stop deceleration ≤ 9.5E9

The limits for other units of measurement must be converted to conform to the given mechanics.

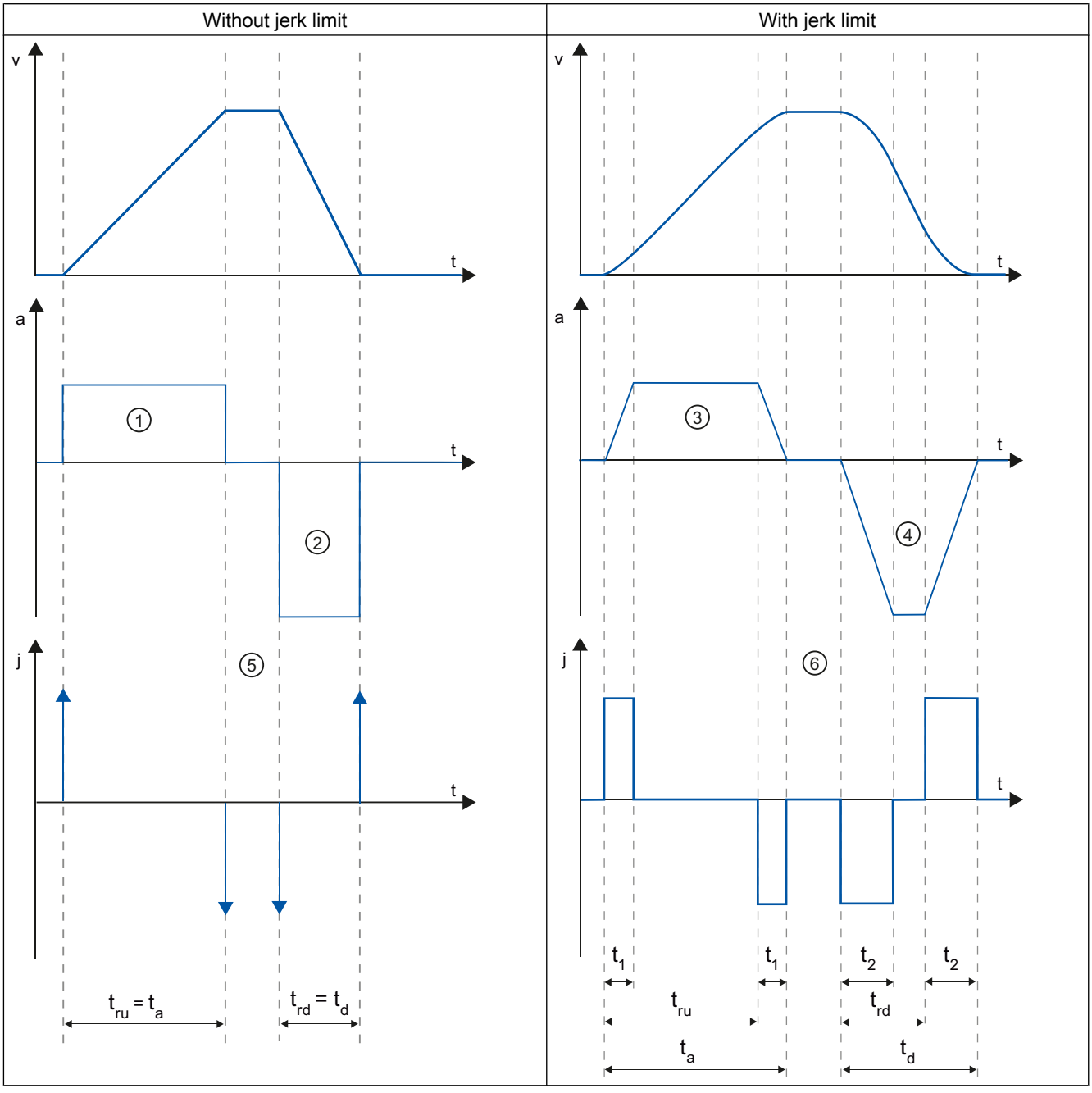
See also

Configuration - Dynamics - General (Page 7379)

Changing the configuration of dynamics in the user program (Page 7387)

Behavior of the axis when using the jerk limit

Axis acceleration and deceleration is not stopped abruptly when the jerk limit is activated; it is adjusted gently according to the set step or smoothing time. The diagram below details the behavior of the axis with and without activated jerk limit:



t	Time axis
v	Velocity
a	Acceleration

j	Jerk
t_{ru}	Rampup time
t_a	Time taken for the axis to accelerate
t_{rd}	Deceleration time
t_d	Time taken for the axis to decelerate
t_1	Smoothing time of the acceleration ramp
t_2	Smoothing time of the deceleration ramp

The example shows travel in which the deceleration value ② is twice the acceleration value ①. The resulting ramp-down time t_{rd} is therefore only half the length of the ramp-up time t_{ru}.

Acceleration ① and deceleration ② change abruptly without a jerk limit. Acceleration ① and deceleration ② change gradually with activated jerk limiter. As the jerk applies to entire motion, the rate is the same for the increase and decrease in acceleration and deceleration.

The step value j becomes infinitely high ⑤ as soon as the change is made without jerk limit. The step is limited to the configured value ⑥ when the jerk limit is activated.

The smoothing time t_1 given in the configuration applies to the acceleration ramp. The deceleration ramp smoothing time t_2 is calculated from the configured jerk value and the configured deceleration.

See also

Configuration - Dynamics - General (Page 7379)

Changing the configuration of dynamics in the user program

You can change the following configuration parameters during runtime of the user program in the CPU:

Acceleration and deceleration

You can also change the values for acceleration and deceleration during runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.DynamicDefaults.Acceleration
  for changing acceleration
- <Axis name>.DynamicDefaults.Deceleration
  for changing deceleration

Refer to the description of the technology object tags (Page 7498) in the appendix for information on when changes to the configuration parameters take effect.

Emergency stop deceleration

You can also change the value for the emergency stop deceleration during runtime of the user program. Use the following technology object tag for this purpose:

- <Axis name>.DynamicDefaults.EmergencyDeceleration

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

Note

After changes to this parameter, it may be necessary to adapt the positions of the hardware limit switches and other safety-relevant settings.

Jerk limit

You can also activate and deactivate the jerk limit at runtime of the user program and change the value for the jerk. To do this, use the technology object tag <Axis name>.DynamicDefaults.Jerk.

If you enter a value > 0.004 pulse/s^3 for the jerk, the jerk limit is activated with the specified value.

If you enter a value = 0.0 for the jerk, the jerk limit is deactivated.

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

See also

Changing configuration of dynamic values in user program ("Axis" technology object V1...3) (Page 7544)

Compatibility list of tags (Page 7350)

MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)

Configuration - Dynamics - General (Page 7379)

Configuration - Dynamics - Emergency stop (Page 7383)

Homing (positioning axis technology object as of V2)

Configuration - Homing - Active

Configure the necessary parameters for active homing in the "Active homing" configuration window. Active homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 3.

Select homing mode (drive connection via PROFIdrive V5 or higher only)

Select one of the following homing modes:

- Use zero mark via PROFIdrive telegram and proximity switch
- Use zero mark via PROFIdrive telegram
- Use homing mark via digital input

If you have selected drive connection via PTO (Pulse Train Output) or analog output, a homing mark via a digital input is used by default.

Digital inputs

In this area, you configure the homing switch:

- **Input homing switch**
  Select the digital input for the homing switch in this field.

  Note

  The digital inputs are set to a filter time of 6.4 ms by default.

  When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

  The specified filter time must be less than the duration of the input signal at the homing switch.

  For drive connection via PTO (Pulse Train Output):
  The input must be interrupt-capable. The onboard CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

- **Select level**
  In the drop-down list, select the level of the homing switch that is to be used for homing.

- **Permit auto reverse at HW limit switch**
  Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be enabled for the reversal of direction (at least the hardware limit switch in the direction of approach must be configured).
  If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The homing switch is then sensed in reverse direction.
  If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

  Note

  If possible, use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

  - Keep the approach velocity low.
  - Increase the configured acceleration/deceleration.
  - Increase the distance between the hardware limit switch and the mechanical endstop.

Approach/homing direction

With the direction selection, you determine the approach direction used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the homing switch to carry out the homing operation.

Side of homing switch

This is where you select whether the axis is to be homed on the top or bottom side of the homing switch.

Approach velocity

In this field, specify the velocity at which the homing switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity

In this field, specify the velocity at which the homing switch is to be approached for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset

If the desired home position deviates from the position of the homing switch, the home position offset can be specified in this field.

If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Configuration - Homing - Passive

Configure the necessary parameters for passive homing in the "Homing - Passive" configuration window.

The movement for passive homing must be triggered by the user (e.g. using an axis motion command). Passive homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 2.

Select homing mode (drive connection via PROFIdrive V5 or higher only)

Select one of the following homing modes:

- **Use zero mark via PROFIdrive telegram and proximity switch**
  The system checks for when the proximity switch is reached. After the proximity switch is reached and is left again in the assigned homing direction, zero mark detection is enabled via the PROFIdrive telegram. When the zero mark is reached in the pre-selected direction, then the actual position of the technology object is set to the homing mark position.
- **Use zero mark via PROFIdrive telegram**
  The system enables zero mark detection as soon as the actual value of the technology object moves in the assigned homing direction. When the zero mark is reached in the specified homing direction, the actual position of the technology object is set to the homing mark position.
- **Use homing mark via digital input**
  The system checks the state of the digital input as soon as the actual value of the axis or encoder moves in the assigned homing direction. When the homing mark is reached (setting of the digital input) in the specified homing direction, the actual position of the technology object is set to the homing mark position.

If you have selected drive connection via PTO (Pulse Train Output), a homing mark via a digital input is used by default.

Digital inputs

In this area, you configure the homing switch:

- **Input homing switch**
  Select the digital input for the homing switch in this field. The input must be interrupt-capable. The onboard CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

  Note

  The digital inputs are set to a filter time of 6.4 ms by default.

  When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

  The specified filter time must be less than the duration of the input signal at the homing switch.

- **Select level**
  In the drop-down list, select the level of the homing switch that is to be used for homing.

Side of homing switch

This is where you select whether the axis is to be homed on the top or bottom side of the homing switch.

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Note

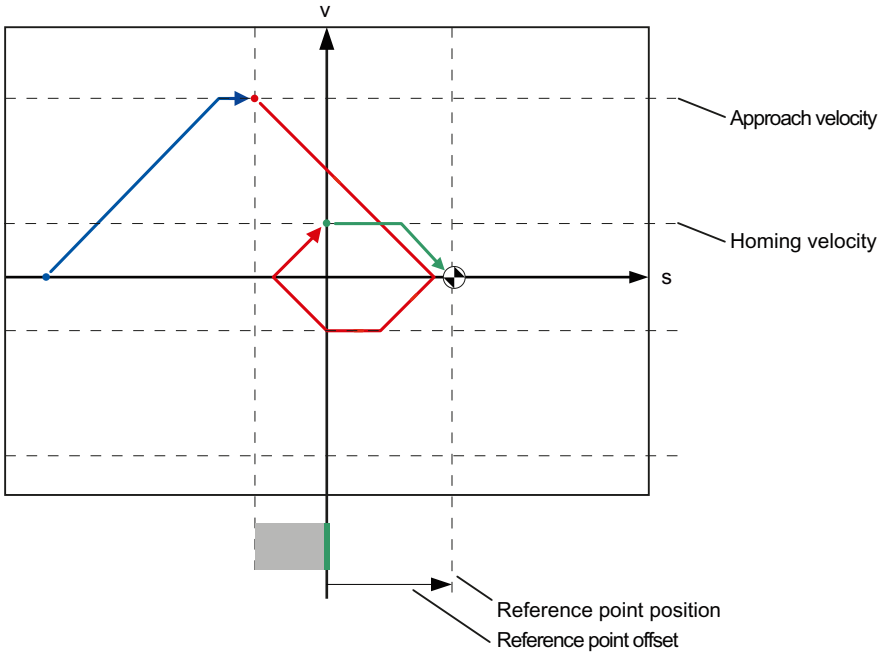
If passive homing is carried out without an axis motion command (axis at a standstill), homing will be executed upon the next rising or falling edge at the homing switch.

Sequence - Active homing

You start active homing with motion control instruction "MC_Home" (input parameter Mode = 3). The "Position" input parameter specifies the absolute home position. Alternatively, you can start active homing on the axis control panel for test purposes.

The diagram below shows an example of a characteristic curve for an active home position approach with the following configuration parameters:

- "Homing mode" = "Use homing mark via digital input"
- "Approach/homing direction" = "Positive direction"
- "Side of homing switch" = "Top side"
- Value of "home position offset" > 0



Search for homing switch (blue curve section)

When active homing starts, the axis accelerates to the configured "approach velocity" and searches at this velocity for the homing switch. The tag <axis name>.StatusBits.HomingDone is set to FALSE.

Reference point approach (red curve section)

When the homing switch is detected, the axis in this example brakes and reverses, to be homed to the configured side of the homing switch at the configured homing velocity. Homing causes the tag <axis name>.StatusBits.HomingDone to change to TRUE.

Travel to home position offset (green curve segment)

After homing, the axis moves at the homing velocity along the path to the home position offset. There the axis is at the homing point position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

See also

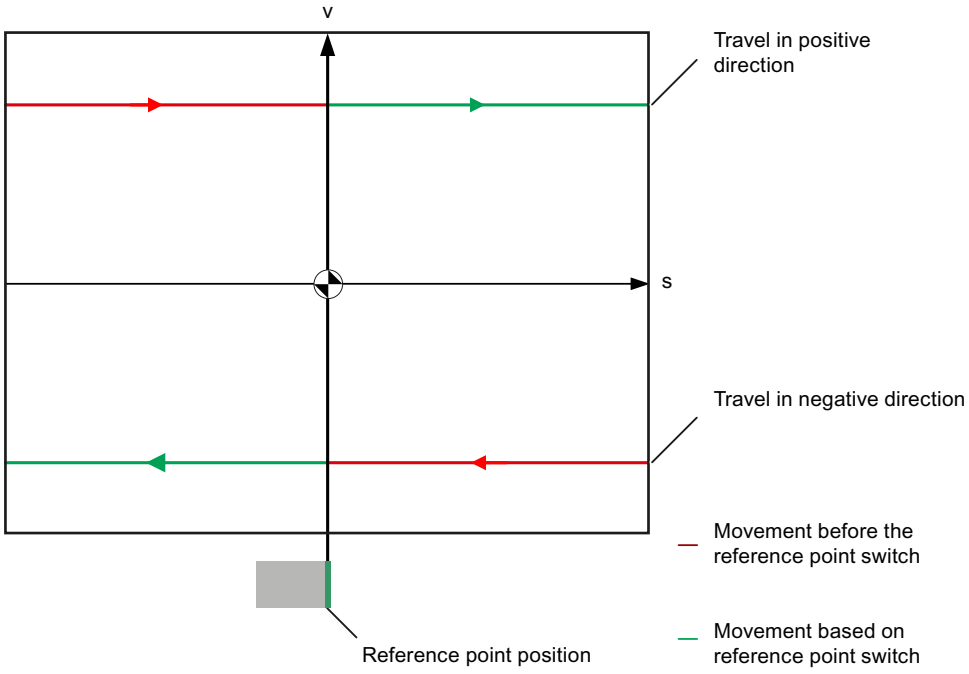
Configuration - Homing - General (Axis technology object V2...3) (Page 7542)

Sequence - Passive homing

Passive homing is started with Motion Control instruction "MC_Home" (input parameter Mode = 2). Input parameter "Position" specifies the absolute reference point position.

The diagram below shows an example of a characteristic curve for passive homing with the following configuration parameters:

- "Side of homing switch" = "Top side"
- "Homing mode" = "Use homing mark via digital input"



Movement towards homing switch (red section of curve)

The Motion Control instruction "MC_Home" does not itself carry out any homing motion when passive homing is started. The travel required for reaching the homing switch must be implemented by the user via other motion control instructions such as "MC_MoveRelative". The tag <axis name>.StatusBits.HomingDone remains TRUE during passive homing if the axis has already been homed.

Axis homing (transition from red to green section of curve)

The axis is homed when the configured side of the homing switch is reached. The current position of the axis is set to the home position. This is specified at the "Position" parameter of the "MC_Home" Motion Control instruction. The tag <axis name>.StatusBits.HomingDone will be set to "TRUE" if the axis has not been homed before. The travel previously started is not canceled.

Movement beyond homing switch (green section of curve)

Following homing at the homing switch, the axis continues and completes the previously started travel with the corrected axis position.

Changing the homing configuration in the user program

With positioning axis technology object as of V2, you can change the following configuration parameters during runtime of the user program in the CPU:

Passive homing

You can change the end of the homing switch for passive homing during the user program runtime. Use the following technology object tag for this purpose:

- <Axis name>.Sensor[1].PassiveHoming.SideInput
  for changing the side of the homing switch
- <Axis name>.Sensor[1].PassiveHoming.Mode
  for changing the homing mode

Refer to the description of the technology object tags (Page 7498) in the appendix for information on when changes to the configuration parameter take effect.

Active homing

You can change the direction of approach, the side of the homing switch, the approach velocity, the homing velocity, and the home position offset for active homing during the program runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.Homing.AutoReversal
  for changing the auto reverse at the HW limit switch
- <Axis name>.Homing.ApproachDirection
  for changing the approach/homing direction

- <Axis name>.Sensor[1].ActiveHoming.SideInput
  for changing the side of the homing switch
- <Axis name>.Homing.ApproachVelocity
  for changing the approach velocity
- <Axis name>.Homing.ReferencingVelocity
  for changing the homing velocity
- <Axis name>.Sensor[1].ActiveHoming.HomePositionOffset
  for changing the home position offset
- <Axis name>.Sensor[1].ActiveHoming.Mode
  for changing the homing mode

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

See also

Compatibility list of tags (Page 7350)

MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)

Position monitoring

Configuration - Position monitoring (PROFIdrive/analog drive connection only)

In the "Position monitoring" configuration window, configure the criteria for monitoring the target position.

Position monitoring monitors the behavior of the actual position at the end of the setpoint calculation. As soon as the setpoint velocity reaches the value zero, the actual position value must be located within a tolerance time in the positioning window. The actual value must not exit the positioning window during the minimum dwell time.

If the actual position reaches the positioning window within the tolerance time and remains in the positioning window for the minimum dwell time, the status bit <axis name>.StatusBits.Done is set. This completes a motion command.

Position monitoring does not make any distinction between how the setpoint interpolation was completed. The end of setpoint interpolation can, for example, be reached as follows:

- By the setpoint reaching the target position
- By position-controlled stopping during the motion through the Motion Control instruction "MC_Halt"

In the following cases, the axis is stopped by the position monitoring and a positioning error (ErrorID 16#800F) is displayed at the Motion Control instruction:

- The actual value does not reach the positioning window within the tolerance time.
- The actual value exits the positioning window during the minimum dwell time.

Positioning window

In this field, configure the size of the positioning window.

Tolerance time

In this field, configure the tolerance time within which the position value must reach the positioning window.

Minimum dwell time in positioning window

In this field, configure the minimum dwell time for which the actual position value must be located in the positioning window.

Configuration - Following error (PROFIdrive/analog drive connection only)

In the "Following error" configuration window, you configure the permissible deviation of the actual position of the axis from the setpoint position.

The following error is the difference between the setpoint position and the actual position value of the axis. The transmission times of the setpoint to the drive and of the actual value to the controller are taken into account in the calculation of the following error.

The following error is monitored based on a velocity-dependent following error limit. The permissible following error depends on the setpoint velocity.

A constant permissible following error can be specified for velocities lower than an adjustable velocity low limit. Above this low velocity limit, the permissible following error increases in proportion to the setpoint velocity. The maximum following error is permitted at the maximum velocity.

If the permitted following error is exceeded, the axis is stopped and an error (ErrorID 16#800D) is displayed at the Motion Control instruction.

Enable following error monitoring

Select the check box to enable following error monitoring.

When following error monitoring is enabled, the axis is stopped in the error range (orange).

Maximum following error

In this field, configure the following error that is permissible at maximum velocity.

Following error

In this field, configure the permissible following error for low velocities (without dynamic adaptation).

Start dynamic adjustment

In this field, configure the velocity above which the following error should be dynamically adapted. Above this velocity, the following error up to the maximum velocity will be adapted to the maximum following error.

Maximum velocity

This box shows the maximum velocity configured under "Dynamics > General".

Configuration - Standstill signal (PROFIdrive/analog drive connection only)

In the "Standstill signal" configuration window, configure the criteria for standstill detection.

To display the standstill (<Axis name>.StatusBits.StandStill), the velocity of the axis must remain in the standstill window for the minimum dwell time.

Standstill window

In this field, configure the size of the standstill window.

Minimum dwell time in standstill window

In this field, configure the minimum dwell time in the standstill window.

Configuration - Control loop (PROFIdrive/analog drive connection only)

In the "Control loop" configuration window, configure the precontrol and the gain Kv of the position control loop.

The Kv factor affects the following parameters:

- Positioning accuracy and stop control
- Uniformity of motion
- Positioning time

The better the mechanical conditions of the axis are (high stiffness), the higher you can configure the Kv factor. This reduces the following error, and a higher dynamic response is achieved.

The "Tuning (Page 7438)" function supports you in determining the optimum gain for the position control of the axis.

Precontrol

In this field, configure the velocity precontrol of the position control loop as a percentage.

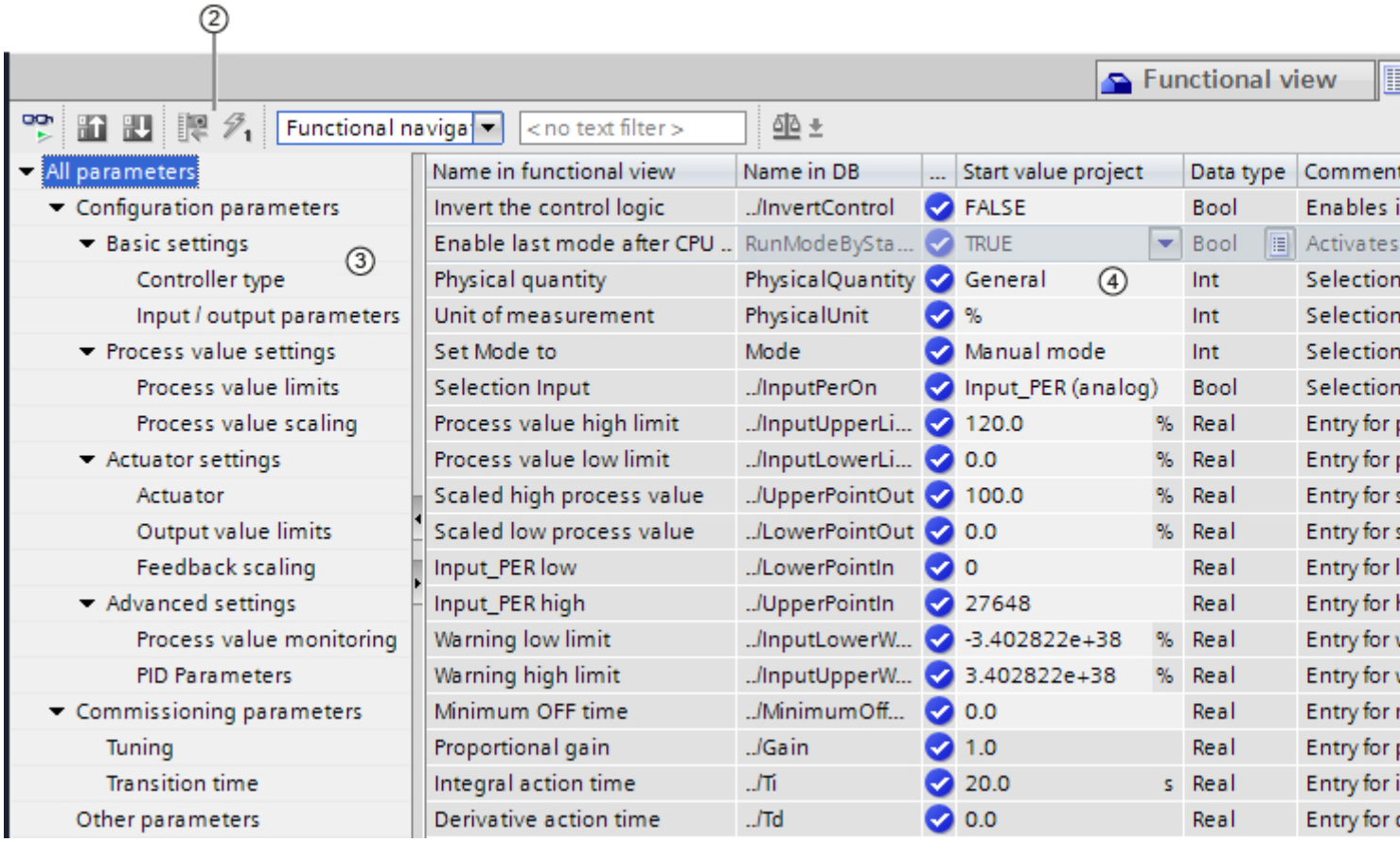
Gain (Kv factor)

In this field, you configure the gain Kv of the position control loop.

Parameter view

Introduction to the parameter view

The Parameter view provides you with a general overview of all relevant parameters of a technology object. You obtain an overview of the parameter settings and can easily change them in offline and online mode.



① "Parameter view" tab

② Toolbar (Page 7209)

③ Navigation (Page 7209)

④ Parameter table (Page 7210)

Function scope

The following functions are available for analyzing the parameters of the technology objects and for enabling targeted monitoring and modification.

Display functions:

- Display of parameter values in offline and online mode
- Display of status information of the parameters

- Display of value deviations and option for direct correction
- Display of configuration errors
- Display of value changes as a result of parameter dependencies
- Display of all memory values of a parameter: Start value PLC, Start value project, Monitor value
- Display of the parameter comparison of the memory values of a parameter

Operator control functions:

- Navigation for quickly changing between the parameters and parameter structures.
- Text filter for faster searches for particular parameters.
- Sorting function for customizing the order of parameters and parameter groups to requirements.
- Memory function for backing up structural settings of the Parameter view.
- Monitoring and modifying of parameter values online.
- Function for saving a snapshot of parameter values of the CPU in order to capture momentary situations and to respond to them.
- Function for applying a snapshot of parameter values as start values.
- Download of modified start values to the CPU.
- Comparison functions for comparing parameter values with one another.

Validity

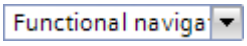
The Parameter view is available for the following technology objects:

- PID_Compact
- PID_3Step
- CONT_C (S7-1500 only)
- CONT_S (S7-1500 only)
- TCONT_CP (S7-1500 only)
- TCONT_S (S7-1500 only)
- TO_Axis_PTO (S7-1200 Motion Control)
- TO_Positioning_Axis (S7-1200 Motion Control)
- TO_CommandTable_PTO (S7-1200 Motion Control)
- TO_CommandTable (S7-1200 Motion Control)

Structure of the parameter view

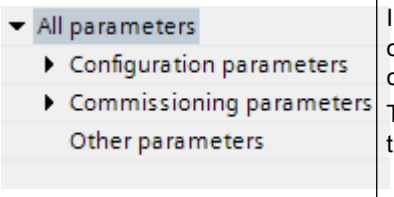
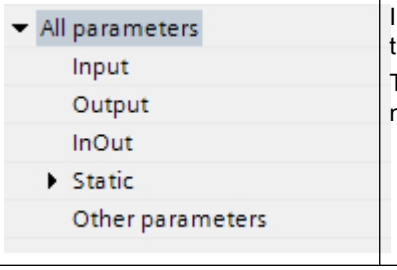
Toolbar

The following functions can be selected in the toolbar of the parameter view.

Icon	Function	Explanation
	Monitor all	Starts the monitoring of visible parameters in the active Parameter view (online mode).
	Create snapshot of monitor values and accept setpoints of this snapshot as start values	Applies the current monitor values to the “Snapshot” column and updates the start values in the project. Only in online mode for PID_Compact and PID_3Step.
	Initialize setpoints	Transfers the start values updated in the project to the CPU. Only in online mode for PID_Compact and PID_3Step.
	Create snapshot of monitor values	Applies the current monitor values to the “Snapshot” column. Only in online mode.
	Modify all selected parameters immediately and once	This command is executed once and as quickly as possible without reference to any particular point in the user program. Only in online mode.
	Select navigation structure	Toggles between functional navigation and data navigation.
	Text filter...	After entry of a character string: Display of all parameters containing the specified string in one of the currently visible columns.
	Selection of compare values	Selection of parameter values that are to be compared with one another in online mode (Start value project, Start value PLC, Snapshot) Only in online mode.
	Save window settings	Saves your display settings for the Parameter view (e.g., selected navigation structure, activated table columns, etc.)

Navigation

Within the "Parameter view" tab, the following alternative navigation structures can be selected.

Navigation		Explanation
Functional navigation		In the functional navigation, the structure of the parameters is based on the structure in the configuration dialog ("Functional view" tab), commissioning dialog, and diagnostics dialog. The last group "Other parameters" contains all other parameters of the technology object.
Data navigation		In the data navigation, the structure of the parameters is based on the structure in the instance DB. The last group "Other parameters" contains the parameters that are not contained in the instance DB.

You can use the "Select navigation structure" drop-down list to toggle the navigation structure.

Parameter table

The table below shows the meaning of the individual columns of the parameter table. You can show or hide the columns as required.

- Column "Offline" = X: Column is visible in offline mode.
- Column "Online" = X: Column is visible in online mode (online connection to the CPU).

Column	Explanation	Offline	Online
Name in functional view	Name of the parameter in the functional view. The display field is empty for parameters that are not configured via the technology object.	X	X
Full name in DB	Complete path of the parameter in the instance DB. The display field is empty for parameters that are not contained in the instance DB.	X	X
Name in DB	Name of the parameter in the instance DB. If the parameter is part of a structure or UDT, the prefix ". ./" is added. The display field is empty for parameters that are not contained in the instance DB.	X	X
Status of configuration	Display of the completeness of the configuration using status symbols. see Status of configuration (offline) (Page 7218)	X	
Compare result	Result of the "Compare values" function. This column is shown if there is an online connection and the "Monitor all" button is selected. see Comparing values (Page 7204)		X
Start value project	Configured start value in the project. Error indication if entered values have a syntax or process-related error.	X	X

Column	Explanation	Offline	Online
Default value	Value that is pre-assigned to the parameter. The display field is empty for parameters that are not contained in the instance DB.	X	X
Snapshot	Snapshot of the current values in the CPU (monitor values). Error indication if values have a process-related error.	X	X
Start value PLC	Start value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Monitor value	Current value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Modify value	Value that is to be used to change the monitor valuet. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if entered values have a syntax or process-related error.		X
Selection for transmission	Selection of the Modify values that are to be transmitted using the "Modify all selected parameters immediately and once" button. This column is displayed together with the "Modify value" column.		X
Minimum value	Minimum process-related value of the parameter. If the minimum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Maximum value	Maximum process-related value of the parameter. If the maximum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Setpoint	Designates the parameter as a setpoint. These parameters can be initialized online.	X	X
Data type	Data type of the parameter. The display field is empty for parameters that are not contained in the instance DB.	X	X
Retain	Designates the value as a retentive value. The values of retentive parameters are retained even after the voltage supply is switched off.	X	X
Accessible from HMI	Indicates whether the HMI can access this parameter during runtime.	X	X
Visible in HMI	Indicates whether the parameter is visible in the selection list of the HMI by default.	X	X
Comment	Brief description of the parameter.	X	X

Opening the parameter view

Requirement

The technology object has been added in the project tree, i.e., the associated instance DB of the instruction has been created.

Procedure

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.
4. Select the "Parameter view" tab in the top right corner.

Result

The Parameter view opens. Each displayed parameter is represented by one row in the parameter table.

The displayable parameter properties (table columns) vary depending on whether you are working with the Parameter view in offline or online mode.

In addition, you can selectively display and hide individual table columns.

See also

Default setting of the parameter view (Page 7212)

Default setting of the parameter view

Default settings

To enable you to work efficiently with the Parameter view, you can customize the parameter display and save your settings.

The following customizations are possible and can be saved:

- Show and hide columns
- Change column width
- Change order of the columns
- Toggle navigation
- Select parameter group in the navigation
- Selection of compare values

Show and hide columns

To show or hide columns in the parameter table, follow these steps:

1. Position the cursor in the header of the parameter table.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the check box for the column.
4. To hide a column, clear the check box for the column.

or

1. Position the cursor in the header of the parameter table.
2. Select the "Show all columns" command in the shortcut menu if all columns of the offline or online mode are to be displayed.

Some columns can only be displayed in online mode: see Parameter table (Page 7210).

Change column width

To customize the width of a column so that all texts in the rows can be read, follow these steps:

1. Position the cursor in the header of the parameter table to the right of the column to be customized until the shape of the cursor changes to a cross.
2. Then double-click this location.

or

1. Open the shortcut menu on the header of the parameter table.
2. Click
   - "Optimize column width" or
   - "Optimize width of all columns".

If the column width setting is too narrow, the complete content of individual fields are shown if you hover the cursor briefly over the relevant field.

Change order of the columns

The columns of the parameter table can be arranged in any way.

To change the order of the columns, follow these steps:

1. Click on the column header and use a drag-and-drop operation to move it to the desired location.
   When you release the mouse button, the column is anchored to the new position.

Toggle navigation

To toggle the display form of the parameters, follow these steps:

1. Select the desired navigation in the “Select navigation structure” drop-down list.
   - Data navigation
   - Functional navigation

See also Navigation (Page 7209).

Select parameter group in the navigation

Within the selected navigation, you choose between the “All parameters” display or the display of a subordinate parameter group of your choice.

1. Click the desired parameter group in the navigation.
   The parameter table only displays the parameters of the parameter group.

Selection of compare values (online)

To set the compare values for the “Compare values” function, follow these steps:

1. Select the desired compare values in the “Selection of compare values” drop-down list.
   - Start value project / Start value PLC
   - Start value project / Snapshot
   - Start value PLC / Snapshot

The “Start value project / Start value PLC” option is set by default.

Save default setting of the Parameter view

To save the above customizations of the Parameter view, follow these steps:

1. Customize the Parameter view according to your requirements.
2. Click the “Save window settings” button at the top right of the Parameter view.

Working with the parameter view

Overview

The following table provides an overview of the functions of the Parameter view in online and offline mode described in the following.

- Column "Offline" = X: This function is possible in offline mode.
- Column "Online" = X: This function is possible in online mode.

Function/action	Offline	Online
Filtering the parameter table (Page 7215)	X	X
Sorting the parameter table (Page 7215)	X	X

Function/action	Offline	Online
Transferring parameter data to other editors (Page 7216)	X	X
Indicating errors (Page 7216)	X	X
Editing start values in the project (Page 7217)	X	X
Status of configuration (offline) (Page 7218)	X	
Monitoring values online in the parameter view (Page 7218)		X
Create snapshot of monitor values (Page 7219)		X
Modifying values (Page 7220)		X
Comparing values (Page 7221)		X
Applying values from the online program as start values (Page 7222)		X
Initializing setpoints in the online program (Page 7223)		X

Filtering the parameter table

You can filter the parameters in the parameter table in the following ways:

- With the text filter
- With the subgroups of the navigation

Both filter methods can be used simultaneously.

With the text filter

Texts that are visible in the parameter table can be filtered. This means only texts in displayed parameter rows and columns can be filtered.

1. Enter the desired character string for filtering in the “Text filter...” input box.
   The parameter table displays only the parameters containing the character string.

The text filtering is reset.

- When another parameter group is selected in the navigation.
- When navigation is changed from data navigation to functional navigation, or vice versa.

With the subgroups of the navigation

1. Click the desired parameter group in the navigation, e.g., "Static".
   The parameter table only shows the static parameters. You can select further subgroups for some groups of the navigation.
2. Click “All parameters” in the navigation if all parameters are to be shown again.

Sorting the parameter table

The values of the parameters are arranged in rows. The parameter table can be sorted by any displayed column.

- In columns containing numerical values, sorting is based on the magnitude of the numerical value.
- In text columns, sorting is alphabetical.

Sorting by column

1. Position the cursor in the header cell of the desired column.
   The background of this cell turns blue.
2. Click the column header.

Result

The entire parameter table is sorted by the selected column. A triangle with tip facing up appears in the column header.

Clicking the column header again changes the sorting as follows:

- Symbol "▲": Parameter table is sorted in ascending order.
- Symbol "▼": Parameter table is sorted in descending order.
- No symbol: The sorting is removed again. The parameter table assumes the default display.

The "../" prefix in the "Name in DB" column is ignored when sorting.

Transferring parameter data to other editors

After selecting an entire parameter row of the parameter table, you can use the following:

- Drag-and-drop
- <Ctrl+C>/<Ctrl+V>
- Copy/Paste via shortcut menu

Transfer parameters to the following editors of the TIA Portal:

- Program editor
- Watch table
- Signal table for trace function

The parameter is inserted with its full name: See information in "Full name in DB" column.

Indicating errors

Error indication

Parameter assignment errors that result in compilation errors (e.g., limit violation) are indicated in the Parameter view.

Every time a value is input in the Parameter view, a check is made for process-related and syntax errors and the result is indicated.

Bad values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Table field with red background
  If you click the bad field, a roll-out error message appears with information of the permissible value range or the required syntax (format)

Compilation error

From the error message of the compiler, you can directly open the Parameter view (functional navigation) containing the parameter causing the error in situations where the parameter is not displayed in the configuration dialog.

Editing start values in the project

With the Parameter view, you can edit the start values in the project in offline mode and online mode.

- You make value changes in the “Start value project” column of the parameter table.
- In the “Status of configuration” column of the parameter table, the progress of the configuration is indicated by the familiar status symbols from the configuration dialog of the technology object.

Boundary conditions

- If other parameters depend on the parameter whose start value was changed, the start value of the dependent parameters are also adapted.
- If a parameter of a technology object is not editable, it is also not editable in the parameter view. The ability to edit a parameter can also depend on the values of other parameters.

Defining new start values

To define start values for parameters in the Parameter view, follow these steps:

1. Open the Parameter view of the technology object.
2. Enter the desired start values in the "Start value project" column. The value must match the data type of the parameter and must not exceed the value range of the parameter. The limits of the value range can be seen in the “Maximum value” and “Minimum value” columns.

The "Status of configuration" column indicates the progress of the configuration with colored symbols.

See also Status of configuration (offline) (Page 7218)

Following adaptation of the start values and downloading of the technology object to the CPU, the parameters take the defined value at startup if they are not declared as retentive (“Retain” column).

Error indication

When a start value is input, a check is made for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Red background in the “Start value project” field
  If you click on the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Correcting bad start values

1. Correct bad start values using information from the roll-out error message.
   Red error symbol, red field background, and roll-out error message are no longer displayed.

The project cannot be successfully compiled unless the start values are error-free.

Status of configuration (offline)

The status of the configuration is indicated by icons:

- In the “Status of configuration” column in the parameter table
- In the navigation structure of the functional navigation and data navigation

Symbol in “Status of configuration” column

Symbol	Meaning
	The start value of the parameter corresponds to the default value and is valid. A start value has not yet been defined by the user.
	The start value of the parameter contains a value defined by the user. The start value is different than the default value. The start value is error-free and valid.
	The start value of the parameter is invalid (syntax or process-related error). The input box has a red background. When clicked, the roll-out error message indicates the cause of the error.
	Only for S7-1200 Motion Control: The start value of the parameter is valid but contains warnings. The input box has a yellow background.

Symbol in the navigation

The symbols in the navigation indicate the progress of the configuration in the same way as in the configuration dialog of the technology object.

See Configure technology objects (Page 7200)

Monitoring values online in the parameter view

You can monitor the values currently taken by the parameters of the technology object in the CPU (monitor values) directly in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

Procedure

1. Start the monitoring by clicking .
   As soon as the Parameter view is online, the following columns are additionally displayed:
   - Compare result
   - Start value PLC
   - Monitor value
   - Modify value
   - Selection for transmission

   The "Monitor value" column shows the current parameter values on the CPU.
   Meaning of the additional columns: see Parameter table (Page 7210)
2. Stop the monitoring by clicking again.

Display

All columns that are only available online have an orange background:

- Values in light-orange cells can be changed.
- Values in cells with a dark orange background cannot be changed.

Create snapshot of monitor values

You can back up the current values of the technology object on the CPU (monitor values) and display them in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").

- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To show the current parameter values, follow these steps:

1. In the Parameter view, click the “Create snapshot of monitor values" icon.

Result

The current monitor values are transferred once to the "Snapshot" column of the parameter table.

You can analyze the values "frozen" in this way while the monitor values continue to be updated in the "Monitor values" column.

Modifying values

With the Parameter view, you can modify values of the technology object in the CPU.

You can assign values to the parameter once (Modify value) and modify them immediately. The modify request is executed as quickly as possible without reference to any particular point in the user program.

Danger

Danger when modifying:

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you use the "Modify" function.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.
- The parameter can be modified (associated field in the "Modify value" column has a light-orange background).

Procedure

To modify parameters immediately, follow these steps:

1. Enter the desired modify values in the “Modify values” column of the parameter table.
2. Check whether the check box for modifying is selected in the "Select for transmission" column.
   The modify values and associated check boxes of dependent parameters are automatically adapted at the same time.
3. Click the “Modify all selected parameters immediately and once” icon .

The selected parameters are modified once and immediately with the specified values and can be monitored in the "Modify values" column. The check boxes for modifying in the "Selection for transmission" column are automatically cleared after the modify request is complete.

Error indication

When a start value is input, a check is made immediately for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red background in the “Modify value” field

and

- If you click the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Bad modify values

- Modify values with process-related errors can be transmitted.
- Modify values with syntax errors **cannot** be transmitted.

Comparing values

You can use comparison functions to compare the following memory values of a parameter:

- Start value project
- Start value PLC
- Snapshot

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To compare the start values on the various target systems, follow these steps:

1. Click the "Selection of compare values" icon .
   A selection list containing the comparison options opens:
   - Start value project - Start value PLC (default setting)
   - Start value project - Snapshot
   - Start value PLC - Snapshot
2. Select the desired comparison option.
   The selected comparison option is executed as follows:
   - A scales symbol appears in the header cells of the two columns selected for comparison.
   - Symbols are used in the "Compare result" column to indicate the result of the comparison of the selected columns.

Symbol in "Compare result" column

Symbol	Meaning
	The compare values are equal and error-free.
	The compare values are not equal and error-free.
	At least one of the two compare values has a process-related or syntax error.
	The comparison cannot be performed. At least one of the two compare values is not available (e.g., snapshot).

Symbol in the navigation

The symbols are shown in the same way in the navigation if the comparison result applies to at least one of the parameters below the displayed navigation structure.

Applying values from the online program as start values

In order to apply optimized values from the CPU to the project as start values, you create a snapshot of the monitor values. Values of the snapshot marked as a "Setpoint" are then applied to the project as start values.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To apply optimized values from the CPU, follow these steps:

1. Click the "Create snapshot of monitor values and accept setpoints of this snapshot as start values" icon .

Result

The current monitor values are applied to the "Snapshot" column and their setpoints are copied to the "Start value project" column as new start values.

Note

Applying values of individual parameters

You can also apply the values of individual parameters that are not marked as a setpoint from the "Snapshot" column to the "Start values project" column. To do so, copy the values and insert them into the "Start value project" column using the "Copy" and "Paste" commands in the shortcut menu.

Initializing setpoints in the online program

You can initialize all parameters that are marked as a "Setpoint" in the Parameter view with new values in the CPU in one step. In so doing, the start values are downloaded from the project to the CPU. The CPU remains in "RUN" mode.

To avoid data loss on the CPU during a cold restart or warm restart, you must also download the technology object to the CPU.

Danger

Danger when changing parameter values

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you reinitialize the setpoints.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

- The “Monitor all” button is selected.
- The parameters marked as a "Setpoint" have a "Start value project" that is free of process-related and syntax errors

Procedure

To initialize all setpoints, follow these steps:

1. Enter the desired values in the "Start value project" column.
   Ensure that the start values are free of process-related and syntax errors.
2. Click the "Initialize setpoints" icon .

Result

The setpoints in the CPU are initialized with the start values from the project.

13.2.6 Technology object command table

13.2.6.1 Use of the command table technology object

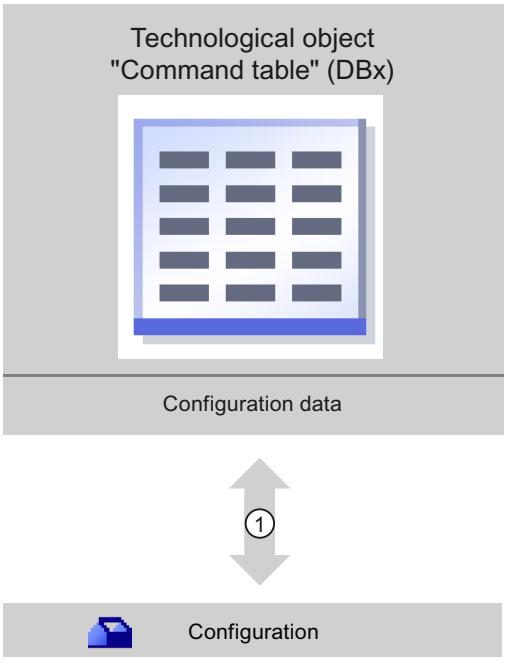
The technology object "Command table" allows you to combine multiple individual axis control commands in one movement sequence. The technology object can be used as of technology version V2 for axes with drive connection via PTO (Pulse Train Output).

You configure the movement sequence as a table in a configuration dialog.

The motion profile of the movement sequences can be checked on a graph before the project is loaded to the CPU. The command tables created are then linked to an axis and used in the user program with the "MC_CommandTable" Motion Control instruction. You can process part or all of the command table.

13.2.6.2 Command table technology object tools

The "Configuration" tool is provided in the TIA Portal for the "Command Table" technology object. The representation below shows the interaction of the tool with the technology object:



①	Writing and reading the configuration of the technology object

Configuration

Configure the following properties of the "Command Table" technology object with the "Configuration" tool:

- You can create one or more movement sequences by configuring individual jobs.
- You can configure the graphic display to check your movement sequence using an axis already configured or a configurable default axis.

The movement sequence data are saved in the data block of the technology object.

13.2.6.3 Adding the technological object command table

Requirements

- A project with a CPU S7-1200 has been created.
- The CPU firmware version is V2.1 or higher

Procedure

Proceed as follows to add a "Command table" technology object in the project tree:

1. Open the "CPU > Technology objects" folder in the project tree.
2. Double-click the "Add new object" command.
   The "Add new object" dialog opens.
3. Select the "Motion Control" technology.
4. Open the "Motion Contro > S7-1200 Motion Control" folder.
5. Select the desired technology version in the "Version" column.
6. Select the "TO_CommandTable" object.
7. Enter the name of the command table in the "Name" input box.
8. To change the automatically assigned data block number, select the "Manual" option.
9. To display additional information about the technology object, click "Additional information".
10. Confirm your entry with "OK".

Result

The new technology object is created and saved to the "Technology objects" folder in the project tree.

13.2.6.4 Configuring the command table technology object

Working with the configuration dialog

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the group of the required technology object in the project tree.
2. Double-click the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters contain all parameters which must be configured for a functional command table.
- **Extended parameters**
  The extended parameters contain the parameters of the default axis or display the parameter values of the axis selected.

Configuration window icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values you can use the technology object without additional changes.
	The configuration contains values set by the user and is complete All input fields of the configuration contain valid values and at least one preset value has changed.
	The configuration is incomplete or incorrect At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to display the cause of the error.
	The configuration contains mutually incompatible parameter values The configuration contains parameter values that contradict each other either in size or logic. The corresponding field or the drop-down list is displayed on a yellow background.

See also

Guidelines on use of motion control (Page 7347)

Basic parameters (Page 7419)

Extended parameters (Page 7431)

Compare values

If there is an online connection to the CPU, the "Compare values" function appears in the configuration of the technology object.

The "Compare values" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of actual values in the project
- Transfer of start values of the project to the CPU as actual values

Icons and operator controls

If there is an online connection to the CPU, the actual values are displayed at the parameters.

In addition to the actual values of the parameters, the following symbols appear:

Icon	Description
	Start value in CPU matches the configured Start value in the project
	Start value in CPU does not match the configured Start value in the project

Icon	Description
	The comparison of the Start value in CPU with the configured Start value in the project cannot be performed
	Use the button to show the start value of the CPU and the start value of the project for the respective parameter.

The actual value and the start value in the project can be changed directly and then downloaded to the CPU. The change of the actual value is transferred directly to the CPU for directly modifiable parameters.

Basic parameters

Configuration - General

Configure the name of the technology object in the "General" configuration window.

Name

Define the name of the command table or the name of the "Command table" technology object in this field. The technology object is listed under this name in the project tree.

See also

Configuration - Command table (Page 7419)

Shortcut menu commands - Command table (Page 7423)

Working with the trend diagram (Page 7424)

Shortcut menu commands - Curve chart (Page 7427)

Transition from "Complete command" to "Blend motion" (Page 7428)

Changing the command table configuration in the user program (Page 7430)

Configuration - Command table

Create the desired movement sequence in the "Command Table" configuration window and check the result against the graphic view in the trend diagram.

Note

Small deviations are possible between the time behavior and position in the trend shown and the real movement of the axis. Movements in response to software limit switches being reached are not shown.

Enable warnings

Activate the display of warnings in the command table with this check box.

Use axis parameters of

From the drop-down list, select which axis parameters are to be used for selecting the graphic view of and checking the movement sequence. Select "Default axis" if you have yet to add an axis to the "Technology object" folder or wish to use values which have not been configured in any of the available axes. You configure the properties of the default axis under "Advanced parameters".

The axis parameters of the axis selected at the "Axis" parameter are used to process the command table in the user program.

Column: Step

Shows the step number of the command.

Column: Command type

In this column, select the command types which are to be used for processing the command table. Up to 32 commands can be entered. The commands will be processed in sequence. You can choose between the following entries and command types:

- **Empty**
  The entry serves as a placeholder for any commands to be added. The empty entry is ignored when the command table is processed.
- **Halt**
  Stop axis
  (the command only takes effect after a "Velocity set point" command)
- **Positioning Relative**
  Position axis relatively
- **Positioning Absolute**
  Position axis absolutely
- **Velocity set point**
  Move axis at set velocity
- **Wait**
  Waits until the given period is over. Wait does not stop active travel.
- **Separator**
  Adds a Separator line above the selected line. The Separator line acts as a range limit for the graphic display of the trend view.
  Use the Separator lines if you wish to process parts of the command table.

Column: Position/travel path

Enter the position or travel path for the selected command in this column:

- **Command "Positioning Relative"**
  The command will move the axis by the given travel path.
- **Command "Positioning Absolute"**
  The command will move the axis by the given position.
- **Separator**
  The value given specifies the start position for the graphic display.

Limit values (independent of the selected user unit):

- -1.0e12 ≤ position / distance ≤ -1.0e-12
- 1.0e-12 ≤ position / distance ≤ 1.0e12
- Position / travel path = 0.0

Column: Velocity

In this column, you enter the velocity for the selected command:

- **Command "Positioning Relative"**
  The command will move the axis at the given velocity.
  The given velocity will not be reached if the travel path selected is not large enough.
- **Command "Positioning Absolute"**
  The command will move the axis at the given velocity.
  The given velocity will not be reached if the target position is too close to the starting position.
- **Command " Velocity set point"**
  The command will move the axis at the given velocity.
  The given velocity will not be reached during the command if too short a runtime is selected.

Limit values (independent of the selected user unit):

- For the commands: "Positioning Relative" and "Positioning Absolute"
  - 1.0e-12 ≤ velocity ≤ 1.0e12
- For the command: "Velocity set point"
  - -1.0e12 ≤ velocity ≤ -1.0e-12
  - 1.0e-12 ≤ velocity ≤ 1.0e12
  - Velocity = 0.0

Column: Duration

Enter the duration of the selected command in this column:

- **Command " Velocity set point"**
  The command will move the axis for the specified duration. The duration includes both the acceleration phase and the constant travel phase. The next command will be processed once the duration is over.
- **Command "Wait"**
  Waits until the given duration is over.

Limit values (independent of the selected user unit):

- 0.001s ≤ duration ≤ 64800s

Column: Next step

Select the mode of transition to the next step from the drop-down list:

- **Complete command**
  The command will be completed. The next command will be processed immediately.
- **Blend motion**
  The motion of the current command will be blended with the motion of the following command. The transition mode "Blend motion" is available with command types "Positioning Relative" and "Positioning Absolute".
  Motion will be blended with motions of the following command types:
  - Positioning Relative
  - Positioning Absolute
  - Velocity set point

No blending occurs with other command types.

For the exact behavior of the axis when a command is appended or overlapped, see: Transition from "Complete command" to "Blend motion" (Page 7428)

Column: Step code

Enter a numerical value / bit pattern in this column which is to be output at the "StepCode" output parameter of the "MC_CommandTable" Motion Control instruction while the command is being processed.

Limit values:

- 0 ≤ code number ≤ 65535

See also

Configuration - General (Page 7417)

Shortcut menu commands - Command table (Page 7423)

Working with the trend diagram (Page 7424)

Shortcut menu commands - Curve chart (Page 7427)

Transition from "Complete command" to "Blend motion" (Page 7428)

Changing the command table configuration in the user program (Page 7430)

Shortcut menu commands - Command table

The following shortcut menu commands are available in the command table:

Insert empty line

Adds an empty line above the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Add empty line

Adds an empty line below the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Insert separator line

Adds a separator line above the selected line.

You cannot have two consecutive separator lines.

Add separator line

Adds a separator line below the selected line.

You cannot have two consecutive separator lines, nor can you add a separator line at the end of the command table.

Cut

Removes the selected lines or content of the selected cell and saves them/it in the clipboard.

Selected lines will be deleted and the subsequent lines of the command table shifted up.

Copy

Copies the selected lines or content of the selected cell and saves them/it in the clipboard.

Paste

- Selected lines:
  Pastes the lines from the clipboard into the table above the selected line.
- Selected cell:
  Pastes the content of the clipboard into the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Replace

Replaces the selected lines with the lines in the clipboard.

Delete

Deletes the selected lines. The lines below in the command table shift up.

See also

Configuration - General (Page 7417)

Configuration - Command table (Page 7417)

Working with the trend diagram (Page 7424)

Shortcut menu commands - Curve chart (Page 7427)

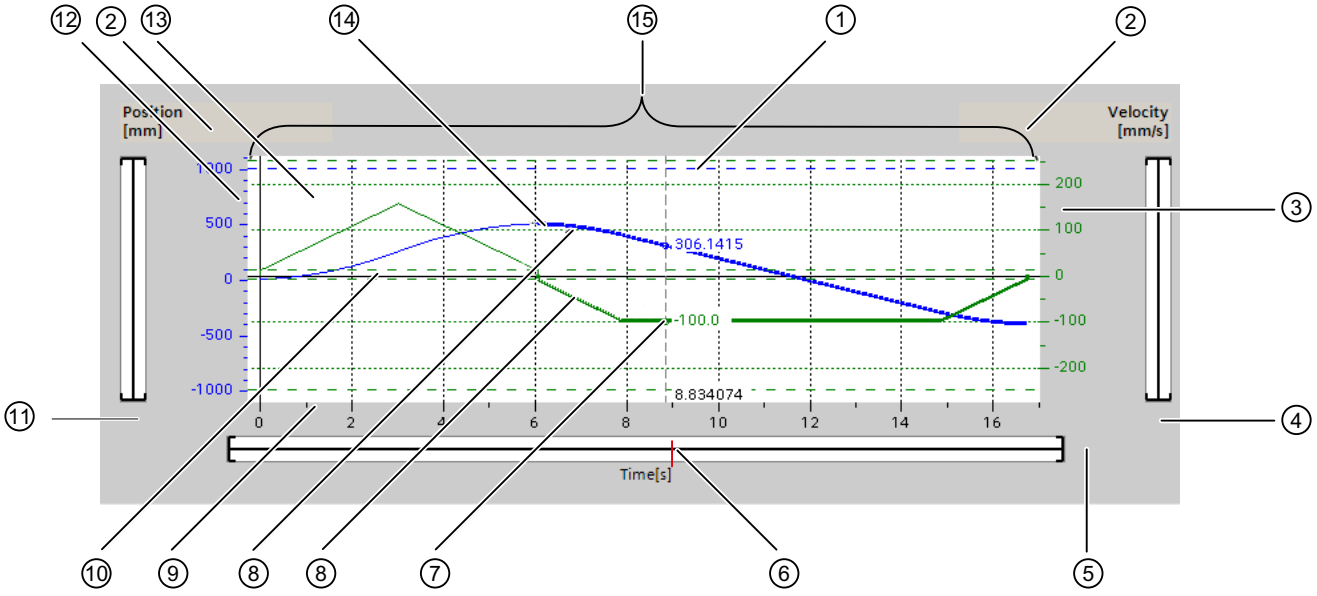
Transition from "Complete command" to "Blend motion" (Page 7428)

Changing the command table configuration in the user program (Page 7430)

Working with the trend diagram

The following tools and information are available in the trend view:

Trend view and components



①	Ruler
②	Selecting the grid
③	Velocity axis scale range
④	Scroll bar, velocity axis
⑤	Scroll bar time axis
⑥	Ruler position marking
⑦	Velocity curve
⑧	Curve section of a selected command
⑨	Time axis scale range
⑩	Start/stop velocity
⑪	Scroll bar, position axis
⑫	Position axis scale range
⑬	Software limit switch position
⑭	Position curve
⑮	Trend view

Selecting separator sections

If the command table consists of multiple sections separated by separators, you can select these sections in the trend view by selecting a command in the section.

Selecting commands

Commands can be selected in the trend view and in the command table:

- Click on a point on the velocity or position curve in the trend view. The corresponding command will be highlighted in the command table.
- Select a command in the command table.
  The corresponding section of curve will be highlighted.

Selecting the visible range of the trend view

Follow the steps below to adjust the section of the trend view to be displayed:

Select the scaling in the shortcut menu:

- Scale to curves:
  Scales the axes so the position and velocity curves are visible.
- Scale to curves and limits:
  Scales the axes so the position and velocity curves, the positions of the activated software limit switches and the minimum and maximum velocity limits are visible.

The view selected will be marked in the shortcut menu with a tick.

Selecting the section to be shown within the range:

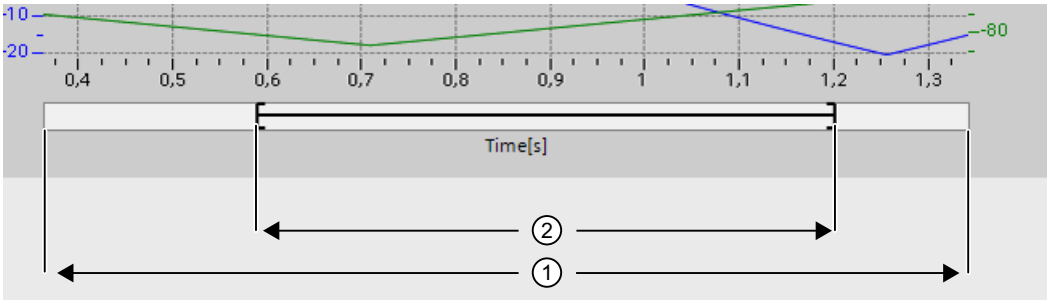
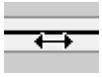
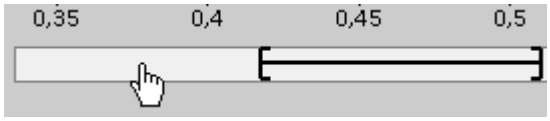
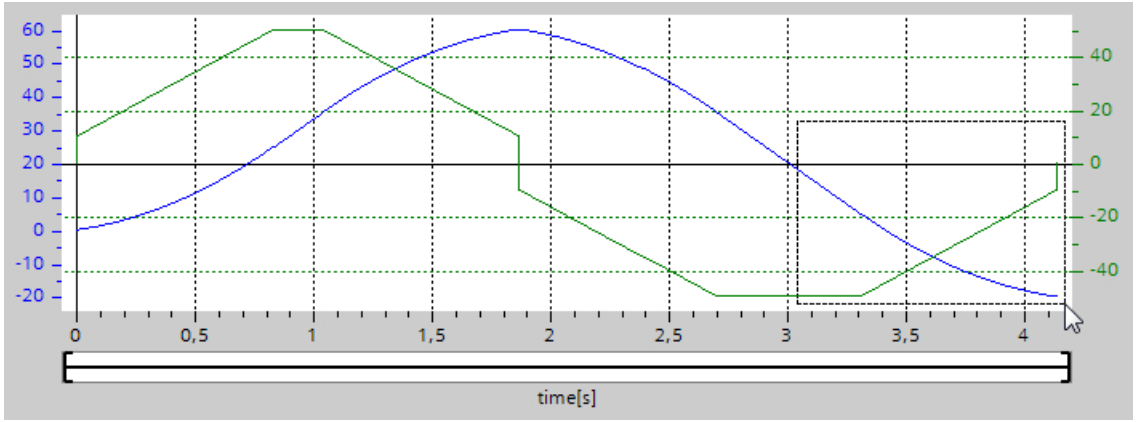


Figure 13-1 CmdTable_Scle01_new

①	Range which the curve values and / or limits are within. (see Selecting in the shortcut menu)
②	Selected range to be shown in the trend window. You set the range with the margin cursor at the right-hand and left-hand margin. You set the position within range ① with the drag cursor.  You can also define the position by clicking in range ①. 

Selecting the section to be shown with the mouse:

Drag a section of the trend view by clicking and dragging with the mouse. The section of curve selected will be enlarged once you release the mouse.



Undoing the last change to the section:

Select the shortcut command "Undo zoom" to undo the last change to the section.

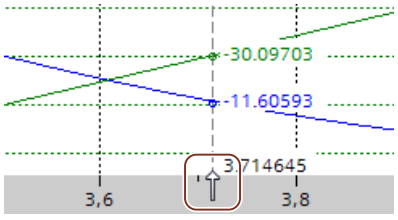
Synchronizing the grid

Click on the axis scales to select whether the grid is to be synchronized with the position axis or velocity axis.

Reading off curve values from the ruler

Activate the ruler using the shortcut menu command "Show ruler".

You can move the ruler to any point on the curves using the ruler cursor.



See also

Configuration - General (Page 7417)

Configuration - Command table (Page 7417)

Shortcut menu commands - Command table (Page 7421)

Shortcut menu commands - Curve chart (Page 7427)

Transition from "Complete command" to "Blend motion" (Page 7428)

Changing the command table configuration in the user program (Page 7430)

Shortcut menu commands - Curve chart

The following shortcut menu commands are available in the curve window:

Zoom 100%

Selects a zoom factor which will show 100% of the curve values and / or limits.

Undo zoom

Undoes the last zoom change.

Scaling on trends

Scales the axes so the position and velocity trends are visible.

Scaling on trends and limits

Scales the axes so the position and velocity trends, the positions of the activated software limit switches and the minimum and maximum velocity limits are visible.

Show velocity limits

Shows the lines of the velocity limits.

Show software limit switches

Shows the lines of the software limit switches.

Show measuring ruler

Fades the measuring ruler in / out

Use the measuring ruler when you want to see the individual values of the trends.

See also

Configuration - General (Page 7417)

Configuration - Command table (Page 7417)

Shortcut menu commands - Command table (Page 7421)

Working with the trend diagram (Page 7422)

Transition from "Complete command" to "Blend motion" (Page 7428)

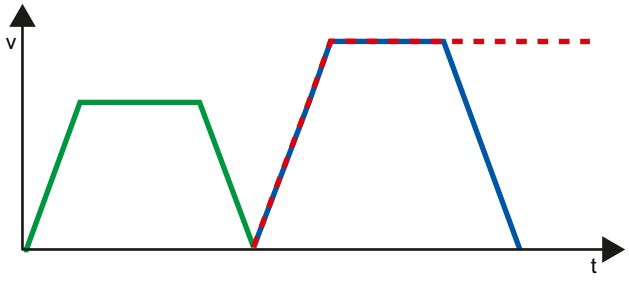
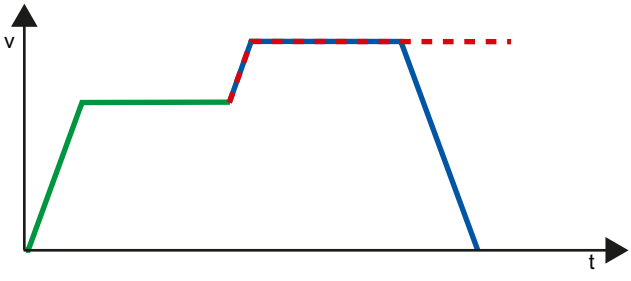
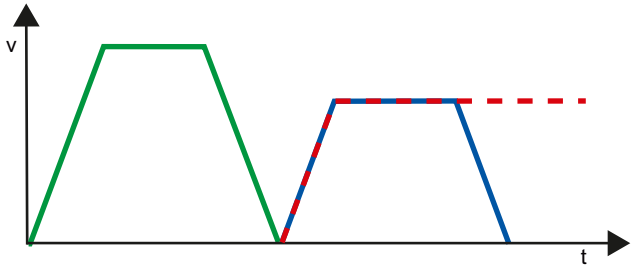
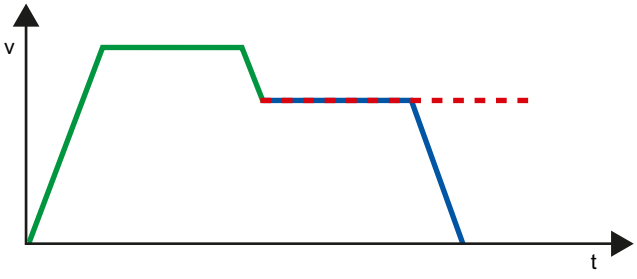
Changing the command table configuration in the user program (Page 7430)

Transition from "Complete command" to "Blend motion"

The charts below show the transition between movements in various different transition modes in the "Next step" column:

Motion transition with preceding positioning jobs

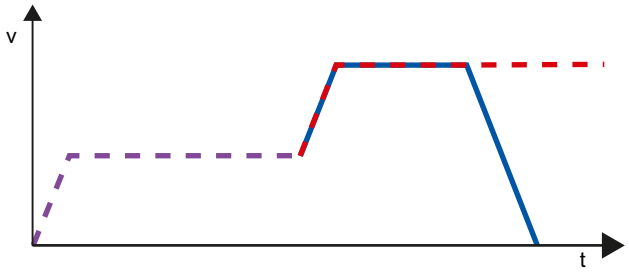
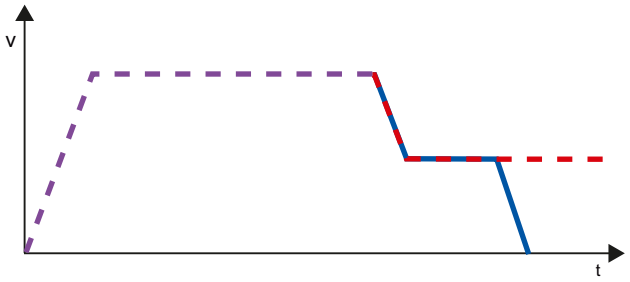
The following diagrams show a command sequence with two motion tasks. The first command is for positioning (green). The second command is for velocity (red) or positioning (blue):

Complete job	Blend motion
Transition from lower to higher velocity  A job with high velocity is appended to a previous positioning job. The positioning job terminates at its target position at velocity "0". The second job starts from standstill.	**Transition from lower to higher velocity**  A job with high velocity is overlapped with a previous positioning job. The first positioning job terminates without standstill at its target position. The second job starts with the new velocity.
Transition from higher to lower velocity  A job with low velocity is appended to a previous positioning job. The positioning job terminates at its target position at velocity "0". The second job starts from standstill.	**Transition from higher to lower velocity**  A job with low velocity is overlapped with a previous positioning job. The first positioning job terminates without standstill at its target position. The first job starts with the new velocity.

—	1. Job "Positioning Relative" or "Positioning Absolute"
- -	2. Job "Velocity set point"
—	2. Job "Positioning Relative" or "Positioning Absolute"

Motion transition with preceding velocity jobs

The following diagrams show a command sequence with two motion tasks. The first command is for velocity (violet). The second command is for velocity (red) or positioning (blue):

Transition from lower to higher velocity	Transition from higher to lower velocity
	
A job with a high velocity is appended to a previous velocity job. The first velocity job ends after the defined runtime. The second job starts with the new velocity.	A command with lower velocity is blended with a previous velocity command. The second job starts with the new velocity.

	1. Job "Velocity set point"
	2. Job "Velocity set point"
	2. Job "Positioning Relative" or "Positioning Absolute"

See also

Configuration - General (Page 7417)

Configuration - Command table (Page 7417)

Shortcut menu commands - Command table (Page 7421)

Working with the trend diagram (Page 7422)

Shortcut menu commands - Curve chart (Page 7425)

Changing the command table configuration in the user program (Page 7430)

Changing the command table configuration in the user program

You can change the following configuration parameters during runtime of the user program in the CPU:

Commands and corresponding values

You can also change the parameters of the command table during the runtime of the user program. Use the following technology object tags for this purpose:

- <Table name>.Command[1..32].Type
  for changing the command type
- <Table name>.Command[1..32].Position
  for changing the position/travel distance

- <Table name>.Command[1..32].Velocity
  for changing the velocity
- <Table name>.Command[1..32].Duration
  for changing the duration
- <Table name>.Command[1..32].NextStep
  for changing the parameter "Next step"
- <Table name>.Command[1..32].StepCode
  for changing the step code

Refer to the description of the technology object tags (Page 7532) in the appendix for information on when changes to the configuration parameters take effect.

See also

Compatibility list of tags (Page 7350)

Configuration - General (Page 7417)

Configuration - Command table (Page 7417)

Shortcut menu commands - Command table (Page 7421)

Working with the trend diagram (Page 7422)

Shortcut menu commands - Curve chart (Page 7425)

Transition from "Complete command" to "Blend motion" (Page 7426)

Extended parameters

Configuration - Extended parameters

Configure the basic properties of the chart view of the "Command table" technology object in the "Extended parameters" configuration window.

Note

If the default axis has been selected under "Use axis parameters of", the unit of measurement can be edited. If a configured axis has been selected, the unit of measurement for this axis will be displayed.

Use axis parameters of

From the drop-down list, select which axis parameters are to be used for selecting the graphic view of and checking the movement sequence. Select "Default axis" if you have yet to add an axis to the "Technology object" folder or wish to use values which have not been configured in any of the available axes.

The axis parameters of the axis selected at the "Axis" parameter will be used to process the command table in the user program.

Unit of measurement position

Enter the unit of measurement for the default axis in this field. If a preconfigured axis has been selected under "Use axis parameters of", the unit of measurement configured in these parameter will be displayed.

Copy axis parameters

Select the direction of copy and the axis for copying the axis parameters. You can copy the axis parameters of the default axis to the selected axis or accept the axis parameters of the selected axis for the default axis. Use the "Apply configuration" button to copy the axis parameters according to your configuration.

Configuration - Dynamics

Configure the acceleration and deceleration and the jerk limit for the default axis in the "Dynamics" configuration window.

Note

If the default axis has been selected under "Use axis parameters of", the following fields can be edited. If a configured axis has been selected, the values of this axis will be displayed.

Acceleration / deceleration

Set the desired acceleration of the default axis in the "Acceleration" field. The desired deceleration can be set in the "Deceleration" field.

Motion jobs configured in the command table will be calculated with the selected acceleration / deceleration.

Limit values:

- 1.0e-12 ≤ acceleration ≤ 1.0e12
- 1.0e-12 ≤ deceleration ≤ 1.0e12

Activate jerk limit

Enable the jerk limit with this check box.

Jerk

Set the desired jerk for ramping up and ramping down in the "Jerk" field.

Motion jobs configured in the command table will be calculated with the selected jerk.

Limit values:

- 1.0e-12 ≤ jerk ≤ 1.0e12

Configuration - Limit values

Configure the maximum velocity, the start/stop velocity and the software limit switches of the default axis in the "Limits" configuration window.

Note

If the default axis has been selected under "Use axis parameters of", the following fields can be edited. If a configured axis has been selected, the values of this axis will be displayed.

Maximum velocity / Start/stop velocity

Define the maximum permissible velocity and the start/stop velocity of the default axis in these boxes. The start/stop velocity is the minimum permissible velocity of the default axis.

Limit values:

- 1.0e-12 ≤ start/stop velocity ≤ 1.0e12
  Start/stop velocity = 0.0
- 1.0e-12 ≤ maximum velocity ≤ 1.0e12
  Maximum velocity = 0.0

The value of the maximum velocity must be greater or equal to the value of the start/stop velocity.

Enable software limit switches

Activate the function of the low and high software limit switch with this check box. Movements in response to software limit switches being reached are not shown in the trend view.

Low / high software limit switch

Enter the position value of the low and high software limit switch in these boxes.

Limit values:

- -1.0e12 ≤ low software limit switch ≤ -1.0e-12
  1.0e-12 ≤ low software limit switch ≤ 1.0e12
  Low software limit switch = 0.0
- -1.0e12 ≤ high software limit switch ≤ -1.0e-12
  1.0e-12 ≤ high software limit switch ≤ 1.0e12
  High software limit switch = 0.0

The value of the high software limit switch must be greater than or equal to the value of the low software limit switch.

13.2.7 Download to CPU

The data of the Motion Control technology objects are saved in the data blocks. The conditions for downloading of "blocks" therefore apply when loading a new or modified technology object.

Caution

Possible malfunctions of the axis when loading without hardware configuration

The hardware configuration is modified when the following modifications are made to the axis configuration:

- Modification of the pulse generator (PTO)
- Modification of the HW limit switch address
- Modification of the homing switch address
- Modification of the address of the PROFIdrive telegram
- Modification of the address of the analog output
- Modification of address of enable output or ready input

If the modified configuration of the axis is loaded with the context menu commands "Software" or "Software (all blocks)" without downloading the hardware configuration, this can lead to malfunctions of the axis.

Ensure that the current hardware configuration is downloaded to the CPU under the listed conditions.

Download in CPU S7-1200 RUN mode (from firmware version V2.2)

For CPU S7-1200 from firmware version V2.2, when loading in CPU RUN mode it is checked whether it is possible to load without stopping the CPU.

The following conditions apply when loading data blocks in RUN mode:

	Download to load memory	Download to work memory
Data block modified values	Yes	No
Data block modified structure	Yes (as of firmware V4)	Yes (as of firmware V4) • When downloading with reinitialization • For tags in system reserve for downloading without reinitialization
	No (firmware V2.2...3)	No (firmware V2.2...3)
New data block	Yes	Yes
Data block deleted	Yes	Yes

Also note the following when deleting data blocks and downloading data blocks with reinitialization:

- The axis must be disabled when downloading a positioning axis technology object.
- When downloading a command table technology object, no MC_CommandTable command with this command table must be active (parameter "Busy" = FALSE).
- When downloading an MC_Power instance data block, no MC_Power instruction must be active (parameter "Busy" = FALSE).

From technology version V3.0, Motion Control technology objects (data blocks) can also be downloaded in CPU RUN mode.

Technology objects lower than V3.0 cannot be downloaded in CPU RUN mode.

Select one of the actions described below to download the modified version of a Motion Control technology object (from version V3.0) to the work memory:

- **Technology object positioning axis and command table**
  Change the CPU operating mode from STOP to RUN.
- **Technology object positioning axis**
  Disable the axis and execute a "Restart" using the Motion Control instruction "MC_Reset".
- **Technology object command table**
  Ensure that the command table is not being used. Download the data block of the command table to the work memory using the extended instruction "READ_DBL".

See also

Guidelines on use of motion control (Page 7347)

MC_Reset: Acknowledge fault as of V4 (Page 3379)

13.2.8 Commissioning

13.2.8.1 Axis control panel

Use the axis control panel to move the axis in manual mode, to optimize the axis settings, and to test your system.

The axis control panel can only be used if an online connection to the CPU is established.

Note

Response times of the axis control panel

The response time during axis control panel operation depends on the communication load of the CPU. Close all other online windows of the TIA Portal to minimize the response time.

"Manual control" button

Click "Manual control" to move the axis in manual control mode. Start by disabling the axis in the user program using Motion Control instruction "MC_Power". In "Manual control" mode, the axis control panel takes over master control for the axis functions. The user program has no influence on the axis functions until manual control is ended.

Caution

Additional axes in automatic mode

The Manual control is active for one axis only. If additional axes are in automatic mode, dangerous situations may arise as a result.

In this happens, disable all other axes.

"Automatic mode" button

Click "Automatic mode" to end the "Manual control" mode. The axis control panel passes back the master control and the axis can be controlled by the user program again. The axis must be re-enabled in the user program and homed, if required.

Complete all active traversing motions before switching to automatic control; otherwise, the axis will be braked with the emergency stop deceleration.

"Enable" button

Click "Enable" to enable the axis in "Manual control" mode. When the axis is enabled, the axis control panel functions can be used.

If the axis cannot be enabled because certain conditions are not met, note the error message in the "Error message" box. Information on eliminating errors is available in the appendix under "List of ErrorIDs and ErrorInfos". After the error has been corrected, enable the axis again.

"Disable" button

Click "Disable" if you want to temporarily disable the axis in "Manual control" mode.

"Command" area

Operation in the "Command" area is only possible if the axis is enabled. You can select one of the following command inputs:

- **Jog**
  This command is equivalent to Motion Control command "MC_MoveJog" in the user program.
- **Positioning**
  This command is equivalent to the Motion Control commands "MC_MoveAbsolute" and "MC_MoveRelative" in the user program. The axis must be homed for absolute positioning.
- **Homing**
  This command is equivalent to Motion Control command "MC_Home" in the user program.
  - The "Set reference point" button corresponds to Mode = 0 (direct homing absolute)
  - The "Active homing" button corresponds to Mode = 3 (active homing)

  For active homing, the homing switch must be configured in the axis configuration. The values for approach velocity, homing velocity, and reference position offset are taken from the axis configuration unchanged.

Depending on the selection, the relevant boxes for entry of setpoints and the buttons for starting the command are displayed.

Jerk limitation can be activated and deactivated using the "Activate jerk limit" button. By default, the jerk is applied with 10% of the configured value. This value can be changed as required.

"Axis status" area

If "Manual control" mode is activated, the current axis status and drive status are shown in the "Axis status" area. The current position and velocity of the axis are displayed at "Process values".

Click "Acknowledge" to acknowledge all cleared errors.

The "Info message" box displays advanced information about the status of the axis.

"Current values" area

This area displays the current position and velocity of the axis.

Error message

The "Error message" box shows the current error. In "Manual control" mode, the error entry can be deleted by pressing the "Acknowledge" button once the error is eliminated.

Note

Initial values for velocity, acceleration / deceleration and jerk

For safety reasons, the "Velocity", "Acceleration/deceleration" and "Jerk" parameters are initialized with values equivalent to only 10% of the configured values when the axis control panel is activated. The "Jerk" parameter is only used for technology object "Axis" V2.0 and higher.

The values in the configuration view displayed when you select "Extended parameters > Dynamics > General" are used for initialization.

The "Velocity" parameter on the control panel is derived from the "Maximum velocity" and the "Acceleration/deceleration" parameters from "Acceleration" in the configuration.

The "Velocity", "Acceleration/deceleration" and "Jerk" parameters can be changed in the axis control panel; this does not affect the values in the configuration.

See also

Guidelines on use of motion control (Page 7347)

13.2.8.2 Tuning

The movement of axes with drive connection via PROFIdrive/analog output is position-controlled.

The "Tuning" function supports you in determining the optimal gain (Kv factor) for the control loop (Page 7395) of the axis. The axis velocity profile is recorded by means of the Trace function for this purpose for the duration of a configurable positioning movement. Then you can evaluate the recording, and adapt the gain accordingly.

The "Tuning" function for the positioning axis technology object can be found in the project tree under "Technology object > Commissioning".

The "Tuning" dialog is divided into the following areas:

- Master control
- Axis
- Optimize gain setting
- Trace

Note

No transfer of the parameters

The configured parameter values are discarded after master control is returned. Transfer the values as needed into your configuration.

Master control

In this area, you can take over master control of the technology object, or return it to your user program:

- **"Activate" button**
  With the "Activate" button, you set up an online connection to the CPU and take over master control for the selected technology object. Note the following when taking over master control:
  - To take over master control, the technology object must be disabled in the user program.
  - Until master control is returned, the user program has no influence on the functions of the technology object. Motion Control jobs from the user program to the technology object are rejected with the error ("ErrorID" = 16#8203).

Caution

Additional axes in automatic mode

The master control is only applied for the selected technology object. If additional axes are in automatic mode, dangerous situations may arise as a result.

In this happens, disable all other axes.

- **"Deactivate" button**
  With the "Deactivate" button, you return master control to your user program.

Axis

In this area, enable or disable the technology object for operation with the axis control panel/ tuning:

- **"Enable" button**
  With the "Enable" button, you enable the selected technology object.
- **"Disable" button**
  With the "Disable" button, you disable the selected technology object.

Optimize gain setting

You make the settings for optimization of the gain in this area:

- **"Forward" button**
  With the "Forward" button, you start a test step for optimization in the positive direction.
- **"Backward" button**
  With the "Backward" button, you start a test step for optimization in the negative direction.
- **"Customize dynamics" check box**
  Select this option to adapt the acceleration and the maximum acceleration for the optimization.
- **Acceleration**
  In this field, you configure the acceleration for a test step.
- **Maximum velocity**
  In this field, you configure the maximum velocity for a test step.

- **Measurement duration**
  In this field, you configure the time duration for a test step.
- **Gain**
  In this field, you configure the actual gain of the position controller (Kv).

Trace

The Trace function is displayed in the lower area of the "Tuning" dialog.

With each test step, a Trace recording of the required parameters is automatically started and displayed after completion of the test step. After master control has been returned, the Trace recording is deleted.

You will find a full description of the Trace function in the section on using the trace and logic analyzer function in the TIA Portal help.

13.2.9 Programming

13.2.9.1 Overview of the Motion Control statements

You control the axis with the user program using Motion Control instructions. The instructions start Motion Control commands that execute the desired functions.

The status of the Motion Control commands and any errors that occur during their execution can be obtained from the output parameters of the Motion Control instructions. The following Motion Control instructions are available:

- MC_Power: Enable, disable axes as of V4 (Page 3374)
- MC_Reset: Acknowledge fault as of V4 (Page 3379)
- MC_Home: Home axes, set reference point as of V4 (Page 3381)
- MC_Halt: Stop axes as of V4 (Page 3385)
- MC_MoveAbsolute: Absolute positioning of axes as of V4 (Page 3388)
- MC_MoveRelative: Relative positioning of axes as of V4 (Page 3391)
- MC_MoveVelocity: Move axes at preset rotational speed as of V4 (Page 3395)
- MC_MoveJog: Move axes in jog mode as of V4 (Page 3399)
- MC_CommandTable: Run axis commands as motion sequence as of V4 (Page 3402)
- MC_ChangeDynamic: Change dynamic settings of axis as of V4 (Page 3404)
- MC_ReadParam: Continuously read motion data of a positioning axis as of V4 (Page 3406)
- MC_WriteParam: Write tag of positioning axis as of V4 (Page 3408)

See also

Creating a user program (Page 7441)

Programming notes (Page 7444)

Behavior of the Motion Control commands after POWER OFF and restart (Page 7446)

Monitoring active commands (Page 7446)

Error displays of the Motion Control statements (Page 7457)

13.2.9.2 Creating a user program

In the section below you learn how to create a user program with the basic configuration for controlling your axis. All available axis functions are controlled using the Motion Control instructions to be inserted.

Requirement

- The technology object has been created and configured without errors.

Before creating and testing the user program, it is advisable to test the function of the axis and the corresponding parts of the system with the axis command table.

Procedure

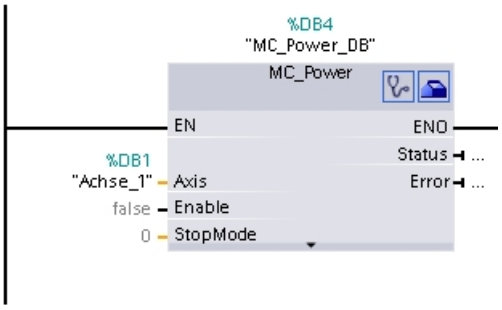
Proceed as follows to create the user program in accordance with the principles described below:

1. In the project tree, double-click your code block (the code block must be called in the cyclic program).
   The code block is opened in the programming editor and all available instructions are displayed.
2. Open the "Technology" category and the "Motion Control" and "S7-1200 Motion Control" folders.
3. Use a drag-and-drop operation to move the "MC_Power" instruction to the desired network of the code block.
   The dialog box for defining the instance DB opens.
4. In the next dialog box, select from the following alternatives:
   Single instance
   Click "Single instance" and select whether you want to define the name and number of the instance DB automatically or manually.
   Multi-instance
   Click "Multi-instance" and select whether you want to define the name of the multi-instance automatically or manually.

5. Click "OK".
   The Motion Control instruction "MC_Power" is inserted into the network.

Parameters marked with "<???>" must be initialized; default values are assigned to all other parameters.
Parameters displayed in black are required for use of the Motion Control instruction.

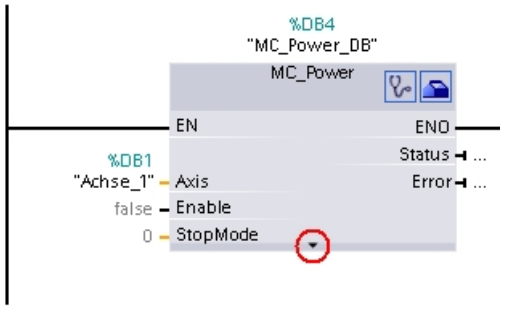
6. Select the technology object in the project tree and drag-and-drop it on <???>.



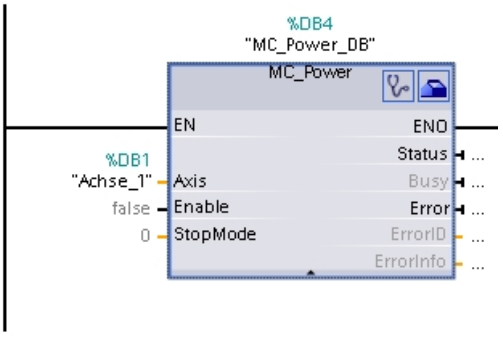
After the selection of the technology object data block, the following buttons are available:

Click the stethoscope icon if you want to open the diagnostics dialog for the technology object.

Click the toolbox icon if you want to open the configuration view of the technology object.



Click the arrow down icon to view additional parameters of the Motion Control instruction.



The grayed-out parameters now visible can be used optionally.

7. Add your choice of Motion Control instructions in accordance with steps 3 to 6.

Result

You have created the basic configuration for axis control in the user program.

Initialize the input parameters of Motion Control instructions in other parts of the user program to initiate the desired jobs for the "Axis" technology object.

Evaluate the output parameters of the Motion Control instructions and the tags of the data block to track the initiated jobs and the status of the axis.

Refer to the detailed description for details on the parameters of Motion Control instructions.

See also

Overview of the Motion Control statements (Page 7438)

Programming notes (Page 7444)

Behavior of the Motion Control commands after POWER OFF and restart (Page 7446)

Monitoring active commands (Page 7446)

Error displays of the Motion Control statements (Page 7457)

13.2.9.3 Programming notes

When creating your user program, note the following information:

- Cyclic call of utilized motion control instructions
  The current status of command execution is available via the output parameters of the motion control instruction. The status is updated with every call of the motion control instruction. Therefore, make sure that the utilized motion control instructions are called cyclically.
- Transfer of parameter values of a motion control instruction
  The parameter values pending for the input parameters are transferred with a positive edge at input parameter "Execute" when the block is called.
  The motion control command is started with these parameter values. Parameter values that are subsequently changed for the motion control instruction are not transferred until the next start of the motion control command.
  Exceptions to this are input parameters "StopMode" of motion control instruction "MC_Power" and "Velocity" of motion control instruction "MC_MoveJog". A change in the input parameter is also applied when "Enable" = TRUE, or "JogForward" and "JogBackward". .

- **Programming under consideration of the status information**
  In a stepwise execution of motion control jobs, make sure to wait for the active command to finish before starting a new command. Use the status messages of the motion control instruction and the "StatusBits" tag of the technology object to check for completion of the active command.
  In the examples below, observe the indicated sequence. Failure to observe the sequence will display an axis or command error.
  - **Axis enable with motion control instruction "MC_Power"**
    You must enable the axis before it can take on motion jobs. Use an AND operation of tag <Axis name>.StatusBits.Enable = TRUE with output parameter Status = TRUE of motion control instruction "MC_Power" to verify that the axis is enabled.
  - **Acknowledge error with motion control instruction "MC_Reset"**
    Prior to starting a motion control command, errors requiring acknowledgement must be acknowledged with "MC_Reset". Eliminate the cause of the error and acknowledge the error with motion control instruction "MC_Reset". Verify that the error has been successfully acknowledged before initiating a new command. For this purpose, use an AND operation of tag <Axis name>.StatusBits.Error = FALSE with output parameter Done = TRUE of motion control instruction "MC_Reset".
  - **Home axis with motion control instruction "MC_Home"**
    Before you can start an MC_MoveAbsolute command, the axis must be homed. Use an AND operation of tag <Axis name>.StatusBits.HomingDone = TRUE with output parameter Done = TRUE of motion control instruction "MC_Home" to verify that the axis has been homed.
- **Override of motion control command processing**
  Motion control jobs for moving an axis can also be executed as overriding jobs.
  If a new motion control command is started for an axis while another motion control command is active, the active command is overridden by the new command before the existing command is completely executed. The overridden command signals this using CommandAborted = TRUE in the motion control instruction. It is possible to override an active MC_MoveRelative command with a MC_MoveAbsolute command.
- **Avoiding multiple use of the same instance**
  All relevant information of a motion control command is stored in its instance.
  Do not start a new command using this instance, if you want to track the status of the current command. Use different instances if you want to track the commands separately. If the same instance is used for multiple motion control commands, the status and error information of the individual commands will overwrite each other.
- **Call of motion control instructions in different priority classes (run levels)**
  Motion Control instructions with the same instance may not be called in different priority classes without interlocking. To learn how to call locked motion control instructions, refer to "Tracking commands from higher priority classes (run levels) (Page 7468)".

See also

Overview of the Motion Control statements (Page 7438)

Creating a user program (Page 7439)

Behavior of the Motion Control commands after POWER OFF and restart (Page 7446)

Monitoring active commands (Page 7446)

Error displays of the Motion Control statements (Page 7457)

Tracking jobs from higher priority classes (execution levels) (Page 7468)

13.2.9.4 Behavior of the Motion Control commands after POWER OFF and restart

A POWER OFF or CPU-STOP aborts all active motion control jobs. All CPU outputs, including pulse and direction outputs, are reset.

After a subsequent POWER ON or CPU restart (CPU RUN), the technology objects and the motion control jobs will be reinitialized.

All actual data of the technology objects as well as all status and error information of the previously active motion control jobs are reset to their initial values.

Before the axis can be reused, it must be enabled again using the Motion Control instruction "MC_Power". If homing is required, the axis must be homed again with Motion Control instruction "MC_Home". When an absolute encoder is used, homing is retained after POWER OFF.

See also

Overview of the Motion Control statements (Page 7438)

Creating a user program (Page 7439)

Programming notes (Page 7442)

Monitoring active commands (Page 7446)

Error displays of the Motion Control statements (Page 7457)

13.2.9.5 Monitoring active commands

Monitoring active commands

There are three typical groups for tracking active Motion Control commands:

- **Motion control instructions with output parameter "Done"**
- **Motion control instruction "MC_MoveVelocity"**
- **Motion control instruction "MC_MoveJog"**

Motion control instructions with "Done" output parameter

Motion control instructions with the output parameter "Done" are started via input parameter "Execute" and have a defined conclusion (for example, with Motion Control instruction "MC_Home": Homing was successful). The command is complete and the axis is at a standstill.

The commands of the following Motion Control instructions have a defined conclusion:

- MC_Reset
- MC_Home

- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_CommandTable (technology object as of V2)
- MC_ChangeDynamic (technology object as of V2)
- MC_WriteParam (as of technology object V4)
- MC_ReadParam (as of technology object V4)

The output parameter "Done" indicates the value TRUE, if the command has been successfully completed.

The output parameters "Busy", "CommandAborted", and "Error" signal that the command is still being processed, has been aborted or an error is pending. The Motion Control instruction "MC_Reset" cannot be aborted and thus has no "CommandAborted" output parameter. The Motion Control instruction "MC_ChangeDynamic" is completed immediately and therefore has no "Busy" or "CommandAborted" output parameters.

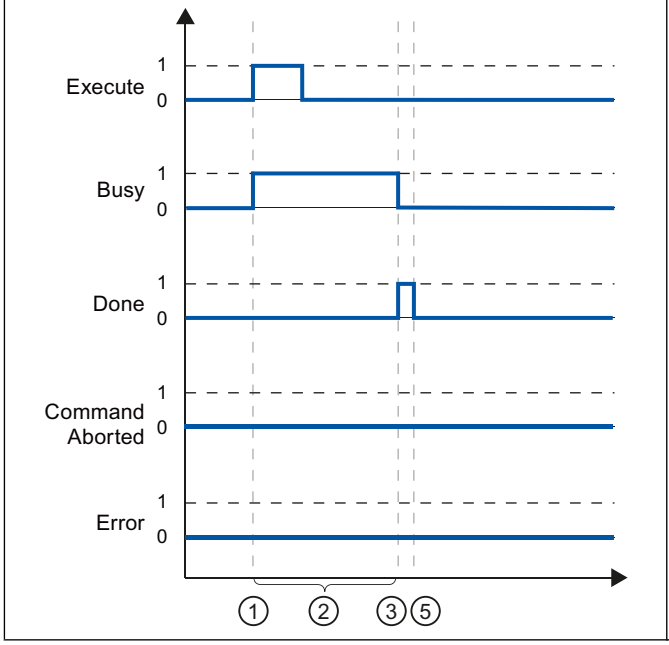
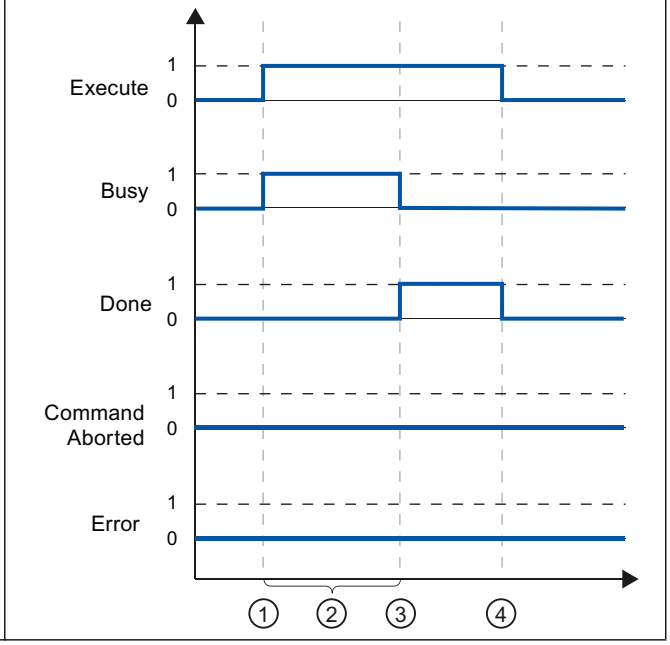
During execution of the Motion Control command, the output parameter "Busy" indicates the value TRUE. If the command has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE. This change occurs regardless of the signal at input parameter "Execute".

Output parameters "Done", "CommandAborted", and "Error" indicate the value TRUE for at least one cycle. These status messages are latched while input parameter "Execute" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

Complete execution of command

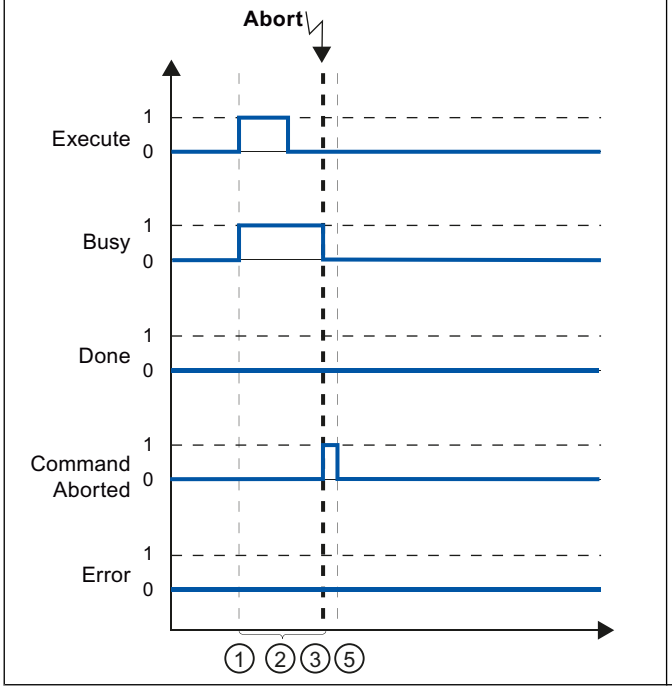
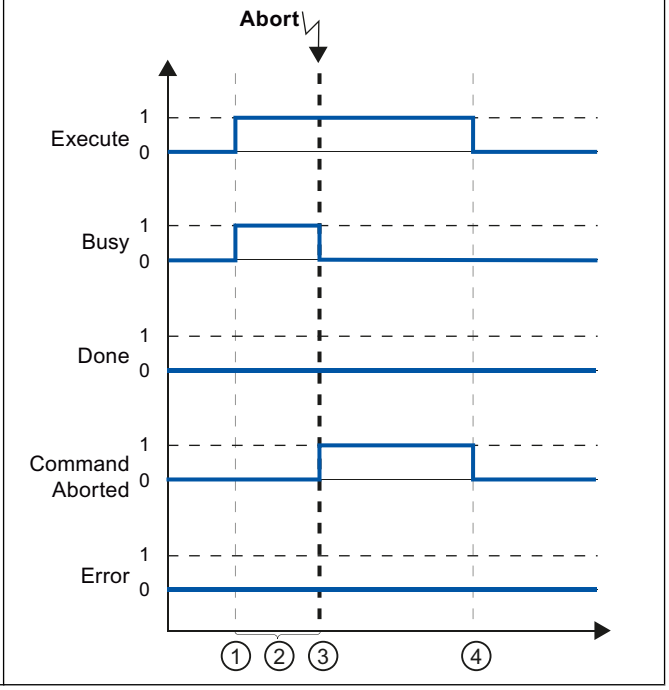
If the Motion Control command has been completely executed by the time of its conclusion, this is indicated by the value TRUE in output parameter "Done". The signal status of the input parameter "Execute" influences the display duration at the output parameter "Done":

"Execute" changes its value to FALSE **during processing** of the command	"Execute" changes its value to FALSE **after completion** of the command
	

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after completion of the command.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	With conclusion of the command (for example, for Motion Control instruction "MC_Home": Homing was successful), output parameter "Busy" changes to FALSE and "Done" to TRUE.
④	If "Execute" retains the value TRUE until after completion of the command, then "Done" also remains TRUE and changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the command is complete, "Done" indicates the value TRUE for only one execution cycle.

Abort command

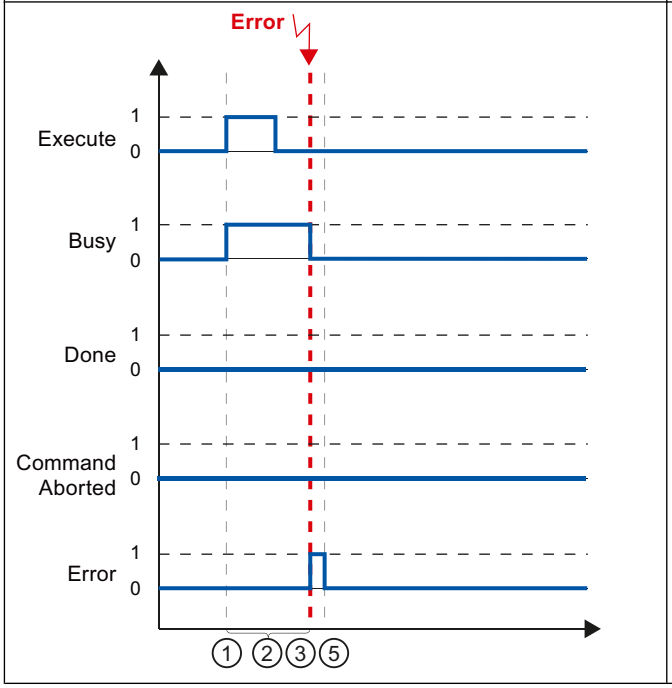
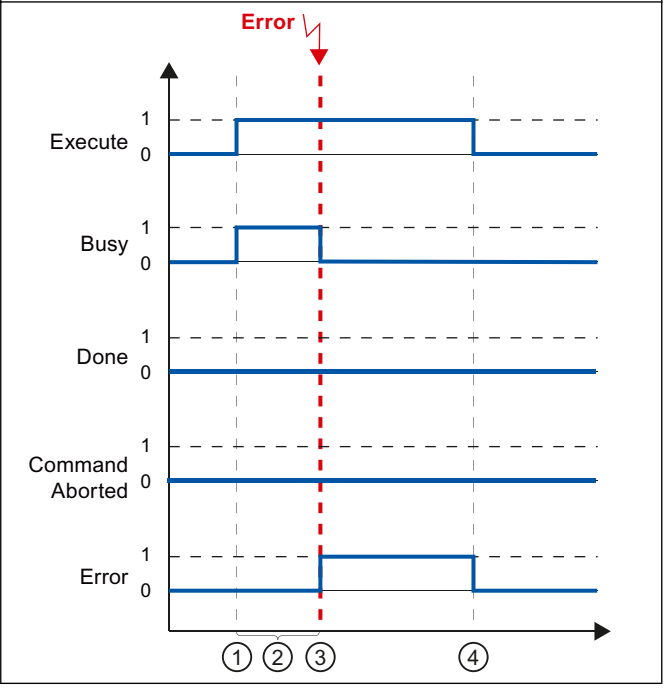
If the Motion Control command is aborted during execution, this is indicated by the value TRUE in output parameter "CommandAborted". The signal status of the input parameter "Execute" influences the display duration at the output parameter "CommandAborted":

"Execute" changes its value to FALSE **before** the command is aborted.	"Execute" changes its value to FALSE **after** the command is aborted.
	

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after completion of the command.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	During command execution, the command is aborted by another Motion Control command. If the command is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	If "Execute" retains the value TRUE until after the command is aborted, then "CommandAborted" also remains TRUE and changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the command is aborted, "CommandAborted" indicates the value TRUE for only one execution cycle.

Error during command execution

If an error occurs during execution of the Motion Control command, this is indicated by the value TRUE in the output parameter "Error". The signal status of the input parameter "Execute" influences the display duration at the output parameter "Error":

"Execute" changes its value to FALSE **before** the error occurs	"Execute" changes its value to FALSE **after** the error occurs
	

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after completion of the command.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during command execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	If "Execute" retains the value TRUE until after the error occurs, then "Error" also remains TRUE and only changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the error occurs, "Error" indicates the value TRUE for only one execution cycle.

Motion control instruction MC_MoveVelocity

A "MC_MoveVelocity" command is started with a positive edge at the "Execute" parameter. The command objective is fulfilled when the assigned velocity is reached and the axis travels at constant velocity. When the assigned velocity is reached and maintained, this is indicated in the "InVelocity" parameter with the value TRUE.

The motion of the axis can, for example, be stopped with an "MC_Halt" command.

The output parameters "Busy", "CommandAborted", and "Error" signal that the command is still being processed, has been aborted or an error is pending.

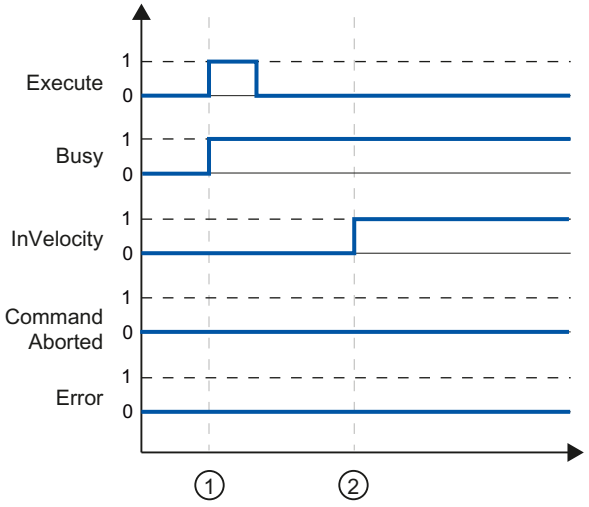
During execution of the Motion Control command, the output parameter "Busy" indicates the value TRUE. If the command has been stopped by another command or by an error, the output parameter "Busy" changes its value to FALSE. This change occurs regardless of the signal at the input parameter "Execute".

The output parameters "CommandAborted" and "Error" show the value TRUE for at least one cycle. These status messages are latched while input parameter "Execute" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

The parameterized velocity is reached

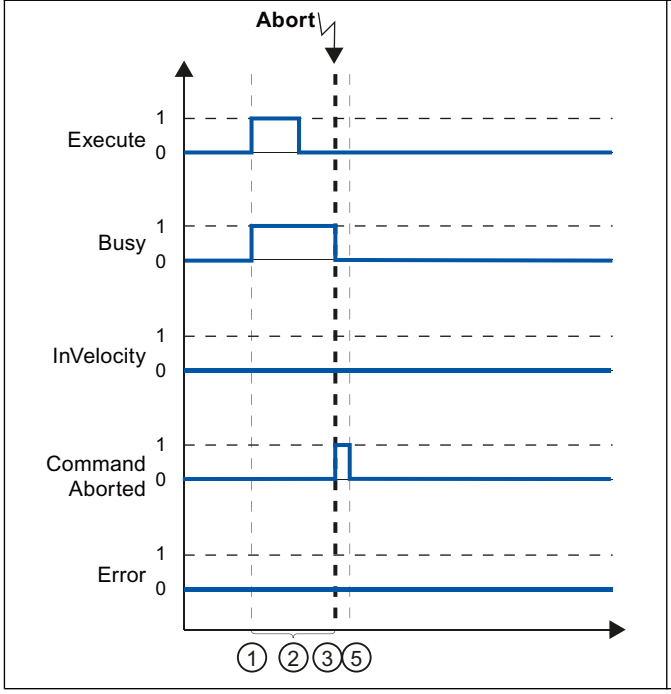
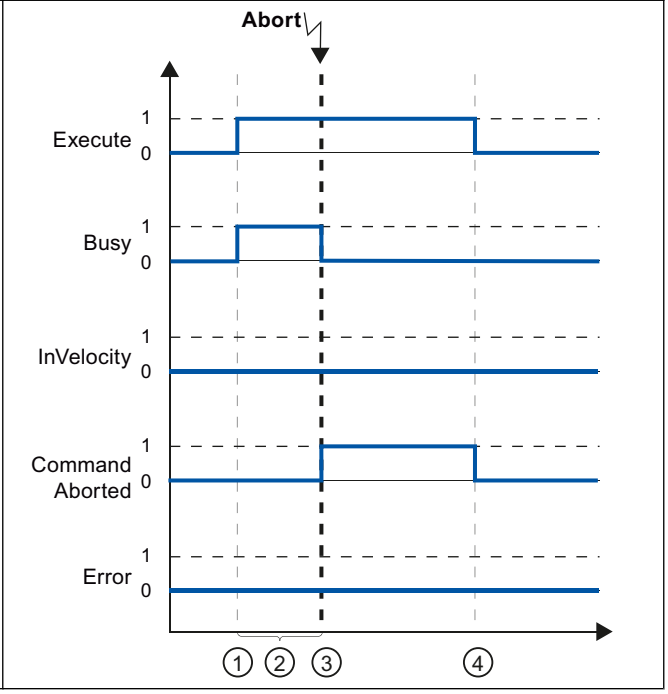
If the Motion Control command has been executed by the time the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity". The parameter "Execute" has no effect on the indication duration in the "InVelocity" parameter.



①	The job is started with a positive edge at the "Execute" parameter. Depending on the programming, "Execute" can be reset to the FALSE value before or after the parameterized velocity has been reached. While the job is active, the parameter "Busy" shows the value TRUE.
②	When the assigned velocity is reached, the "InVelocity" parameter changes to TRUE. The "Busy" and "InVelocity" parameters retain the TRUE value until the "MC_MoveVelocity" command is overridden by another Motion Control command or stopped by an error.

The command is aborted prior to reaching the parameterized velocity

If the Motion Control command is aborted before the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "CommandAborted". The signal status of input parameter "Execute" influences the display duration at the output parameter "CommandAborted".

"Execute" changes its value to FALSE **before** the command is aborted.	"Execute" changes its value to FALSE **after** the command is aborted.
	

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after the command is aborted.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	During command execution, the command is aborted by another Motion Control command. If the command is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	If "Execute" retains the value TRUE until after the command is aborted, then "CommandAborted" also remains TRUE and changes its status to FALSE together with "Execute".
⑤	If "Execute" has been reset to FALSE before the command is aborted, "CommandAborted" indicates the value TRUE for only one execution cycle.

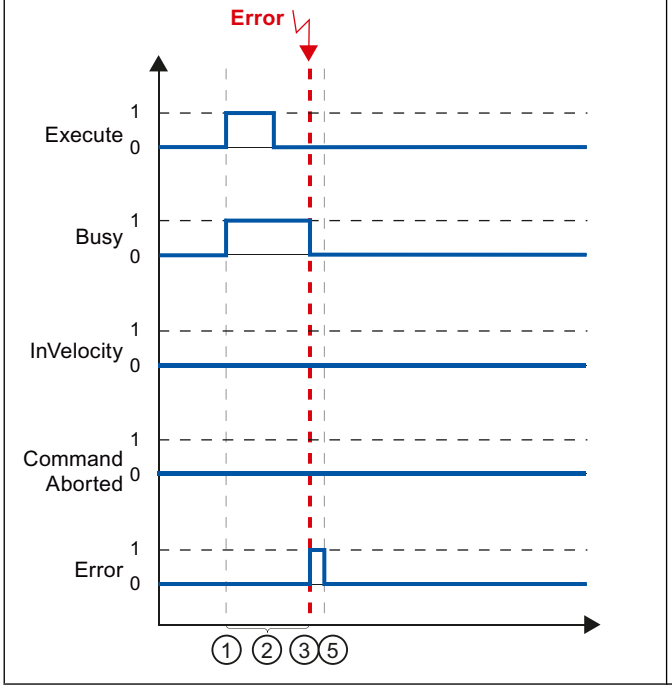
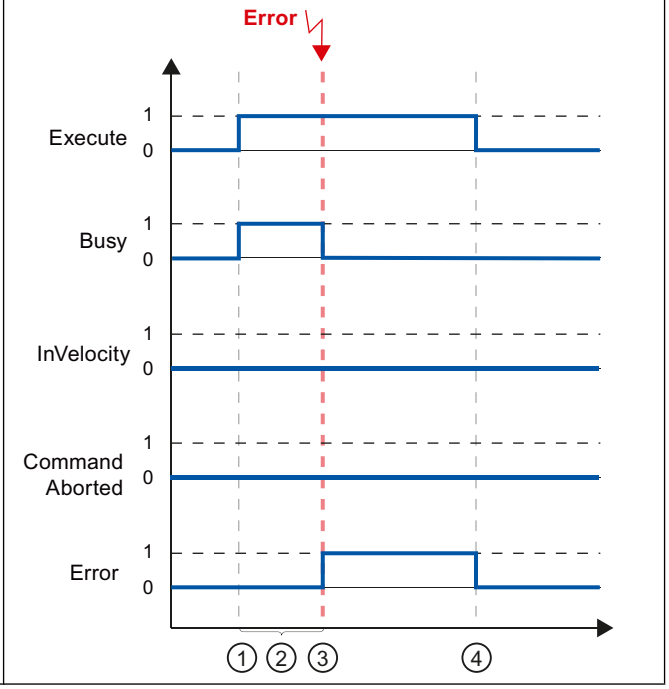
Note

Under the following conditions, an abort is not indicated in output parameter "CommandAborted":

The parameterized velocity has been reached, input parameter "Execute" has the value FALSE, and a new Motion Control command is initiated.

An error has occurred prior to reaching the parameterized velocity

If an error occurs during execution of the Motion Control command before the parameterized velocity has been reached, this is indicated by the value TRUE in the output parameter "Error". The signal status of the input parameter "Execute" influences the display duration at the output parameter "Error":

"Execute" changes its value to FALSE **before** the error occurs	"Execute" changes its value to FALSE **after** the error occurs
	

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after the error has occurred.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during command execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	If "Execute" retains the value TRUE until after the error has occurred, then "Error" also remains TRUE and only changes its status to FALSE together with "Execute".
⑤	If "Execute" has been reset to FALSE before the error occurs, "Error" indicates the value TRUE for only one execution cycle.

Note

Under the following conditions, an error is not indicated in output parameter "Error":

The parameterized velocity has been reached, input parameter "Execute" has the value FALSE, and an axis error occurs (software limit switch is approached, for example).

The error of the axis is only indicated in the "MC_Power" Motion Control instruction.

Motion control instruction MC_MoveJog

The commands of Motion Control instruction "MC_MoveJog" implement a jog operation.

The motion control commands "MC_MoveJog" do not have a defined end. The command objective is fulfilled when the parameterized velocity is reached for the first time and the axis travels at constant velocity. When the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".

The order is complete when input parameter "JogForward" or "JogBackward" has been set to the value FALSE and the axis has come to a standstill.

The output parameters "Busy", "CommandAborted", and "Error" signal that the command is still being processed, has been aborted or an error is pending.

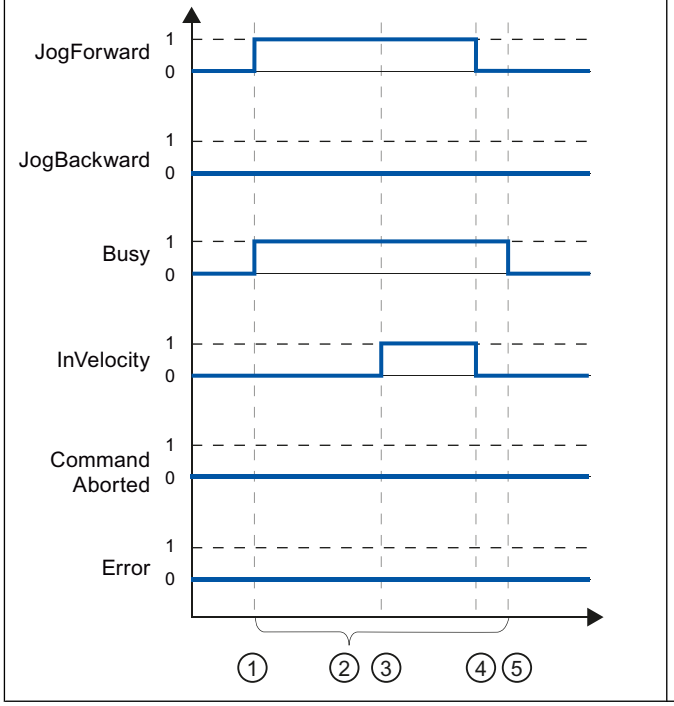
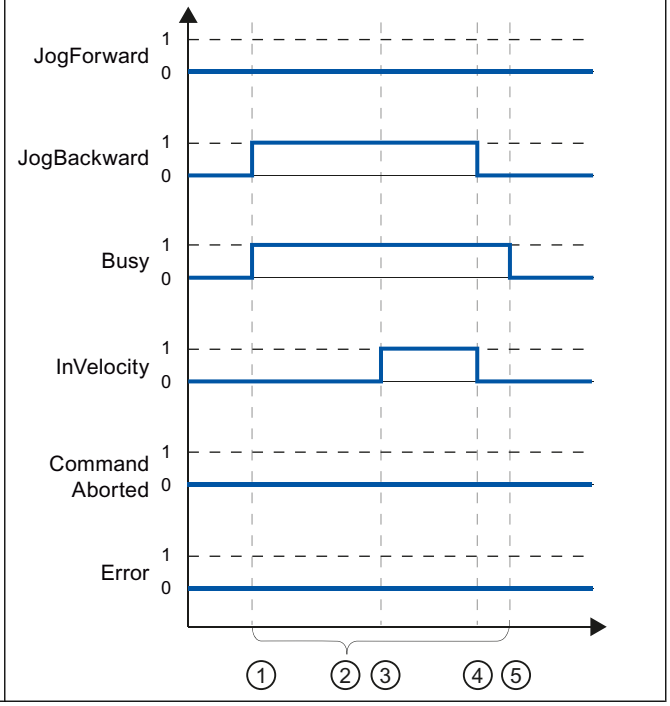
During processing of the motion control command, the output parameter "Busy" indicates the value TRUE. If the command has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE.

The output parameter "InVelocity" indicates the status TRUE, as long as the axis is moving at the parameterized velocity. The output parameters "CommandAborted" and "Error" indicate the status for at least one cycle. These status messages are latched as long as either input parameter "JogForward" or "JogBackward" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

The parameterized velocity is reached and maintained

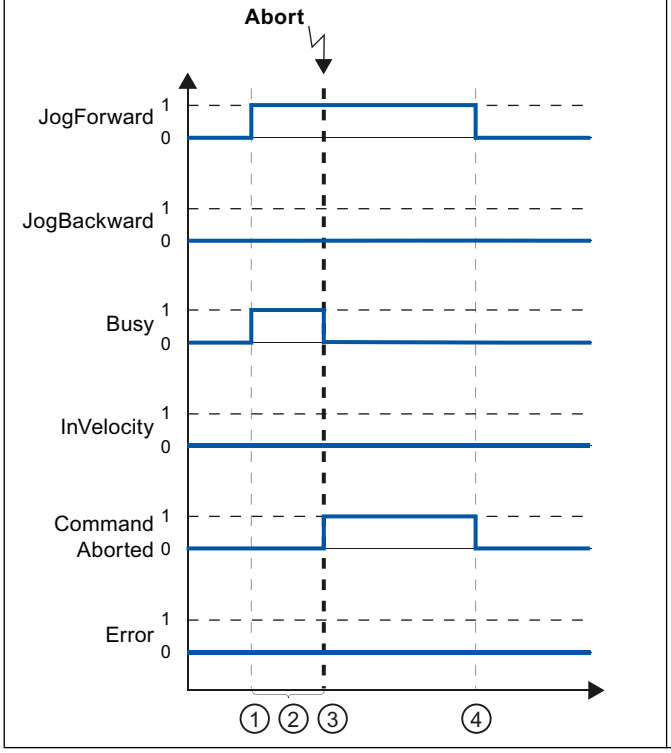
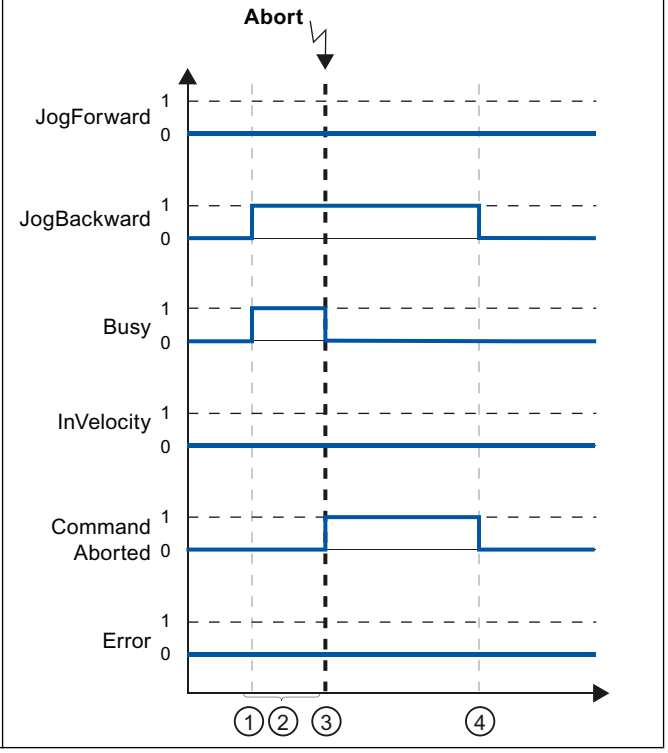
If the motion control command has been executed by the time the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".

Jog mode is controlled by input parameter "JogForward"	Jog mode is controlled by input parameter "JogBackward".
	

①	The command is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	When the parameterized velocity is reached, the output parameter "InVelocity" changes to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the axis motion ends. The axis starts to decelerate. As a result, the axis no longer moves at constant velocity and the output parameter "InVelocity" changes its status to FALSE.
⑤	If the axis has come to a standstill, the motion control command is complete and the output parameter "Busy" changes its value to FALSE.

The command is aborted during execution

If the motion control command is aborted during execution, this is indicated by the value TRUE in output parameter "CommandAborted". The behavior is independent of whether or not the parameterized velocity has been reached.

Jog mode is controlled by input parameter "JogForward".	Jog mode is controlled by input parameter "JogBackward".
	

①	The command is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	During command execution, the command is aborted by another motion control command. If the command is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the output parameter "CommandAborted" changes its value to FALSE.

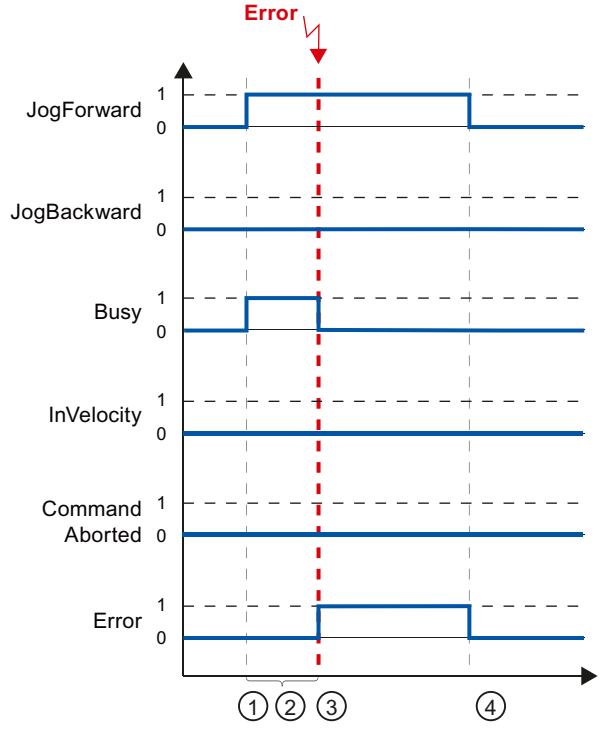
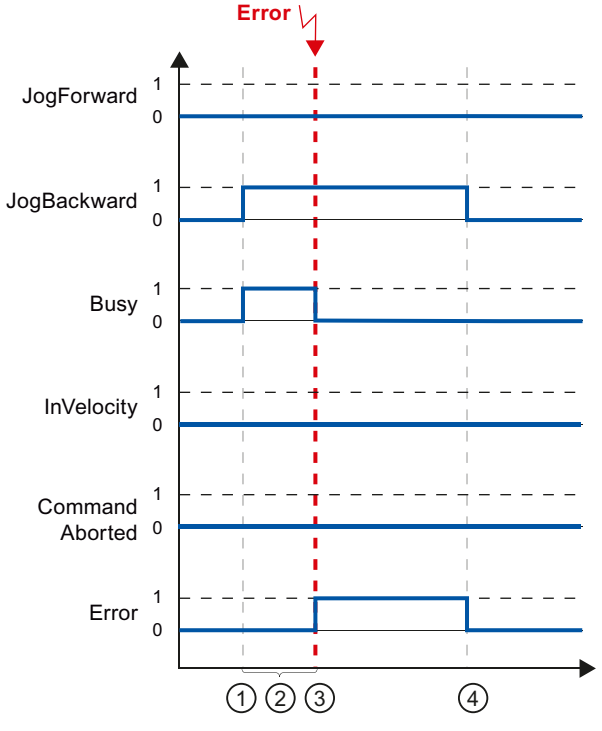
Note

The command abort is indicated in the output parameter "CommandAborted" for only one execution cycle, if all conditions below are met:

The input parameters "JogForward" and "JogBackward" have the value FALSE (but the axis is still decelerating) and a new motion control command is initiated.

An error has occurred during command execution

If an error occurs during execution of the motion control command, this is indicated by the value TRUE in output parameter "Error". The behavior is independent of whether or not the parameterized velocity has been reached.

Jog mode is controlled by input parameter "JogForward".	Jog mode is controlled by input parameter "JogBackward".
	

①	The command is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during command execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the output parameter "Error" changes its value to FALSE.

Note

An error occurrence is indicated in the output parameter "Error" for only one execution cycle, if all the conditions below are met:

The input parameters "JogForward" and "JogBackward" have the value FALSE (but the axis is still decelerating) and a new error occurs (software limit switch is approached, for example).

13.2.9.6 Error displays of the Motion Control statements

The Motion Control instructions indicate any errors in motion control commands and the technology object at the output parameters "Error", "ErrorID" and "ErroInfo" of the Motion Control instructions.

Error display at output parameters "Error", "ErrorID" and "ErrorInfo"

If the output parameter "Error" indicates the value TRUE, the complete command, or portions thereof, could not be executed. The cause of the error is indicated by the value in output parameter "ErrorID". Detailed information about the cause of the error is returned by the value in output parameter ErrorInfo. We distinguish between the following error classes for error indication:

- **Operating error with axis stop (for example, "HW limit switch was approached")**
  Operating errors with axis stop are errors that occur during runtime of the user program. If the axis is in motion, it is stopped with the configured deceleration or emergency stop deceleration, depending on the error. The errors are indicated in the error-triggering Motion Control instruction and in the Motion Control instruction "MC_Power".
- **Operating error without axis stop (for example, "Axis is not homed")**
  Operating errors without axis stop are errors that occur during runtime of the user program. If the axis is in motion, the motion is continued. The errors are only indicated in the Motion Control instruction which triggers the error.
- **Configuration error in Motion Control instruction (for example "Incorrect value in parameter "Velocity")**
  Parameterization errors occur when incorrect information is specified in the input parameters of Motion Control instructions. If the axis is in motion, the motion is continued. The errors are only indicated in the Motion Control instruction which triggers the error.
- **Configuration error on technology object "Axis" (for example, "Value for "Acceleration" is invalid")**
  A configuration error exists if one or more parameters are incorrectly configured in the axis configuration or if editable configuration data have been modified incorrectly during runtime of the program. An axis in motion is stopped with the configured emergency stop deceleration. The error is indicated in the error-triggering Motion Control instruction and in Motion Control instruction "MC_Power".

- **Configuration error on technology object "Command table" (for example "Value for "Velocity" is invalid")**
  There is a configuration error if one or more parameters are incorrectly set in the axis command table or if programmable configuration data have been modified incorrectly during runtime of the program. If the axis is in motion, the motion is continued. The errors are only indicated in the "MC_CommandTable" Motion Control instruction.
- **Internal error**
  When an internal error occurs, the axis is stopped. The errors are indicated in the error-triggering Motion Control instruction and, in some cases, in the Motion Control instruction "MC_Power".

A detailed description of the ErrorIDs and ErrorInfos, as well as their remedies, is available in the Appendix (Page 7477).

See also

Overview of the Motion Control statements (Page 7438)

Creating a user program (Page 7439)

Programming notes (Page 7442)

Behavior of the Motion Control commands after POWER OFF and restart (Page 7444)

Monitoring active commands (Page 7444)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

13.2.9.7 Restart of technology objects

Description

After the CPU is switched on, or after technology objects are downloaded into the CPU, the system automatically initializes the technology objects with the start values from the technology data block. If restart-relevant changes are detected during a reload into the CPU, a restart of the technology object is automatically performed.

If restart-relevant data have been changed in RUN mode by the user program, then the technology object must be reinitialized by the user in order for the changes to be used.

If changes in the technology data block should also be retained after the restart of the technology object, then you must write the changes to the start value in load memory using the extended instruction "WRIT_DBL".

Restart necessary

If a restart of the technology object is necessary, this will be indicated under "Technology object > Diagnostics > Status and error bits > Status messages > Restart required", and in the tags of the technology object <Axis name>.StatusBits.RestartRequired.

Restarting a technology object

A restart of the technology object is triggered by the user by means of the "MC_Reset" Motion Control instruction, with parameter "Restart" = TRUE.

A restart resets the "Homed" status of a technology object with incremental actual values (<Axis name>.StatusBits.HomingDone).

13.2.10 Axis - Diagnostics

13.2.10.1 Status and error bits (technology objects as of V4)

You use the "Status and error bits" diagnostic function to monitor the most important status and error messages for the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The status error messages have the following meaning:

Status messages

Status message - Axis	Description
Enabled	The axis is enabled and ready to be controlled via Motion Control commands. (Tag of technology object: <Axis name>.StatusBits.Enable)
Homed	The axis is homed and is capable of executing absolute positioning commands of Motion Control instruction "MC_MoveAbsolute". The axis does not have to be homed for relative positioning. Special situations: • During active homing, the status is FALSE. • If a homed axis undergoes passive homing, the status is set to TRUE during passive homing. (Tag of technology object: <Axis name>.StatusBits.HomingDone)
Axis error	An error has occurred in the "Axis" technology object. Additional information about the error is available in automatic control at the ErrorID and ErrorInfo parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error. (Tag of technology object: <Axis name>.StatusBits.Error)
Control panel active	The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the "Axis" technology object. The axis cannot be controlled from the user program. (Tag of technology object: <Axis name>.StatusBits.ControlPanelActive)
Restart required	A modified configuration of the axis was downloaded to the load memory in CPU RUN mode. To download the modified configuration to the work memory, you need to restart the axis. Use the Motion Control instruction MC_Reset to do this. (Tag of technology object: <Axis name>.StatusBits.RestartRequired)

Status message - Drive	Description
Ready	The drive is ready for operation. (Tag of technology object: <Axis name>.StatusBits.DriveReady)
Drive error	The drive has reported an error due to loss of its "Drive ready" signal. (Tag of technology object: <Axis name>.ErrorBits.DriveFault)

Status message - Motion	Description
Standstill	The axis is at a standstill. (Tag of technology object: <Axis name>.StatusBits.StandStill)
Acceleration	The axis accelerates. (Tag of technology object: <Axis name>.StatusBits.Accelerating)
Constant velocity	The axis travels at constant velocity. (Tag of technology object: <Axis name>.StatusBits.ConstantVelocity)
Deceleration	The axis decelerates (slows down). (Tag of technology object: <Axis name>.StatusBits.Decelerating)

Status message - Motion type	Description
Positioning	The axis executes a positioning command of the Motion Control instruction "MC_MoveAbsolute", "MC_MoveRelative" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.PositioningCommand)
Travel with velocity specification	The axis executes a command with velocity specification of the Motion Control instruction "MC_MoveVelocity", "MC_MoveJog" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.VelocityCommand)
Homing	The axis executes a homing command of the Motion Control instruction "MC_Home" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.HomingCommand)
Command table active	The axis is controlled by Motion Control instruction "MC_CommandTable". (Tag of technology object: <Axis name>.StatusBits.CommandTableActive)

Limit switch status messages

Limit switch status message	Description
Low SW limit switch has been reached	A software limit switch was reached or exceeded. (Tag of technology object: <Axis name>.StatusBits.SWLimitMinActive)
High SW limit switch has been reached	A hardware limit switch was reached or exceeded. (Tag of technology object: <Axis name>.StatusBits.SWLimitMaxActive)
Low HW limit switch was reached	The low hardware limit switch was reached or exceeded. (Tag of technology object: <Axis name>.StatusBits.HWLimitMinActive)
High HW limit switch was reached	The high hardware limit switch was reached or exceeded. (Tag of technology object: <Axis name>.StatusBits.HWLimitMaxActive)

Error messages

Error message	Description
SW limit switch has been reached	A software limit switch was reached or exceeded. (Tag of technology object: <Axis name>.ErrorBits.SWLimit)
HW limit switch has been reached	A hardware limit switch was reached or exceeded. (Tag of technology object: <Axis name>.ErrorBits.HWLimit)
Invalid direction of motion	The motion direction of the command does not match the configured motion direction. (Tag of technology object: <Axis name>.ErrorBits.DirectionFault)
PTO already in use	A second axis is using the same PTO (Pulse Train Output) and HSC (High Speed Counter) and is enabled with "MC_Power". (Tag of technology object: <Axis name>.ErrorBits.HWUsed)
Encoder	Error in the encoder system. (Tag of technology object: <Axis name>.ErrorBits.SensorFault)
Data exchange	Error in communication with a connected device. (Tag of technology object: <Axis name>.ErrorBits.CommunicationFault)
Positioning	The axis was not correctly positioned at the end of a positioning motion. (Tag of technology object: <Axis name>.ErrorBits.PositionigFault)
Following error	The maximum permitted following error was exceeded. (Tag of technology object: <Axis name>.ErrorBits.FollowingErrorFault)
Encoder values are invalid	The encoder values are invalid. (Tag of technology object: <Axis name>.StatusSensor.State)
Configuration error	The "Axis" technology object was incorrectly configured or editable configuration data were modified incorrectly during runtime of the user program. (Tag of technology object: <Axis name>.ErrorBits.ConfigFault)
Internal error	An internal error has occurred. (Tag of technology object: <Axis name>.ErrorBits.SystemFault)

The output window below shows the first reported and still unacknowledged error.

See also

StatusBits tags as of V4 (Page 7525)

ErrorBits tags as of V4 (Page 7529)

Diagnostics - Status and error bits ("Axis" technology object V1...3) (Page 7552)

Compatibility list of tags (Page 7350)

Motion status (Page 7462)

13.2.10.2 Motion status

Use the "Motion status" diagnostic function to monitor the motion status of the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The displayed status information has the following meaning:

Status	Description
Current position	The "Current position" box indicates the current axis position. If the axis is not homed, the value indicates the position value relative to the enable position of the axis. (Tag of technology object: <Axis name>.Position)
Current velocity	The "Current velocity" box indicates the current axis velocity. (Tag of technology object: <Axis name>.Velocity)
Target position	The "Target position" box indicates the current target position of an active positioning command or of the axis command table. The value of the "Target position" is only valid during execution of a positioning command. (Tag of technology object: <Axis name>.StatusPositioning.TargetPosition)
Remaining travel distance	The "Remaining travel distance" box indicates the travel distance currently remaining for an active positioning command or the axis command table. The "Remaining travel distance" value is only valid during execution of a positioning command. (Tag of technology object: <Axis name>.StatusPositioning.Distance)

See also

Position tag as of V4 (Page 7498)

Velocity tag as of V4 (Page 7499)

StatusPositioning tags as of V4 (Page 7522)

Compatibility list of tags (Page 7350)

Status and error bits (technology objects as of V4) (Page 7457)

13.2.10.3 Dynamics settings

Use the "Dynamics settings" diagnostic function to monitor the dynamic limits of the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The displayed status information has the following meaning:

Dynamic limit	Description
Acceleration	The "Acceleration" box indicates the currently configured acceleration of the axis. (Tag of technology object: <Axis name>.DynamicDefaults.Acceleration)
Deceleration	The "Deceleration" box indicates the currently configured deceleration of the axis. (Tag of technology object: <Axis name>.DynamicDefaults.Deceleration)

Dynamic limit	Description
Emergency deceleration	The "Emergency deceleration" box indicates the currently configured emergency stop deceleration of the axis. (Tag of technology object: <Axis name>.DynamicDefaults.EmergencyDeceleration)
Jerk (axis technology object as of V2)	The "Velocity" box indicates the current axis step velocity configured. (Tag of technology object: <Axis name>.DynamicDefaults.Jerk)

See also

DynamicDefaults tags as of V4 (Page 7513)

Compatibility list of tags (Page 7350)

13.2.10.4 PROFIdrive frame

The "Technology object > Diagnostics > PROFIdrive telegram" diagnostics function is used in the TIA Portal to monitor the PROFIdrive telegrams returned by the drive and encoder. The display of the Diagnostics function is available in online operation.

"Drive" area

This area displays the following parameters contained in the PROFIdrive telegram from the drive to the controller:

- Status words "SW1" and "SW2"
- The setpoint speed that was output to the drive (NSET)
- The actual speed that was signaled from the drive (NACT)

"Encoder" area

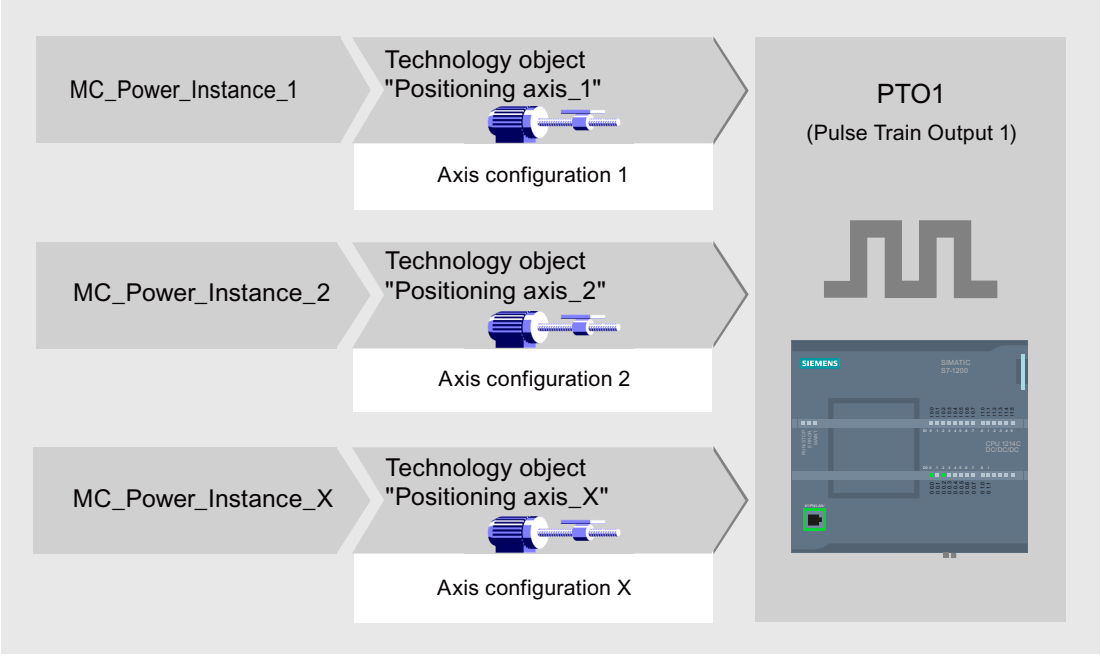
This area displays the following parameters contained in the PROFIdrive telegram from the encoder to the controller:

- Status word "G1_ZSW"
- The actual position value "G1_XIST1" (cyclic actual encoder value)
- The actual position value "G1_XIST2" (absolute value of the encoder)

13.2.11 Appendix

13.2.11.1 Using multiple axes with the same PTO

Use the Motion Control functionality of the CPU S7-1200 to run multiple positioning axis technology objects with the same PTO (Pulse Train Output) and thus with the same CPU outputs. This is appropriate, for example, if different axis configurations are to be used for different production sequences via one PTO. As described below, it is possible to switch between these axis configurations as often as necessary. The following diagram presents the basic functional relationships:

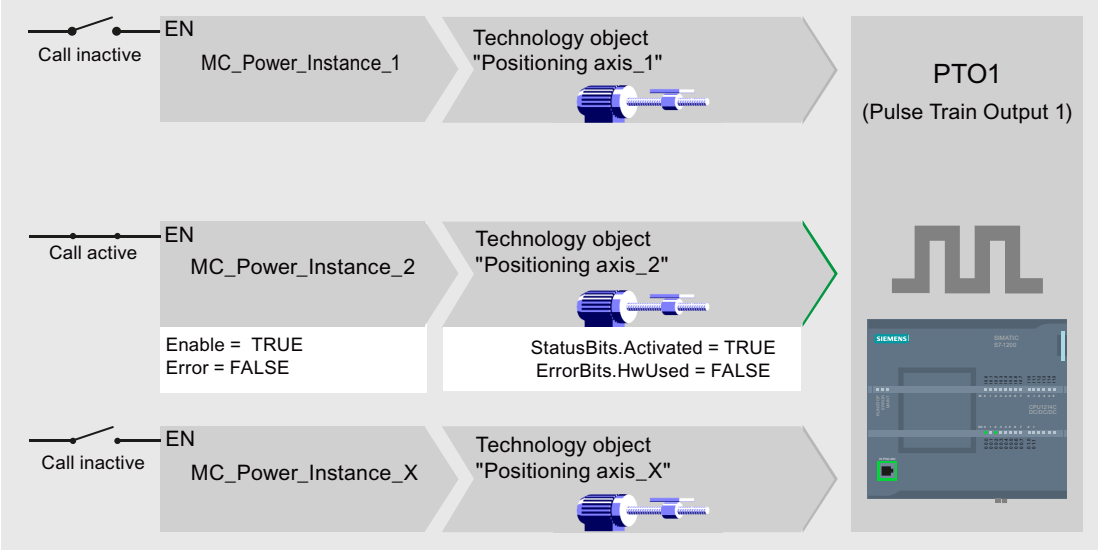


In this example, several positioning axis technology objects, each with its own axis configuration, use the same PTO. Each axis must be called in the user program with a separate call of Motion Control instruction "MC_Power" with a separate instance data block. Only one axis at a time may use the PTO. The axis that is currently using the PTO indicates this with tag <Axis name>.StatusBits.Activated = TRUE.

Switchover of the positioning axis technology object

The program scheme described below shows you how to switch between different technology objects and, thus, between different axis configurations. To use the same PTO with multiple axes without error indications, only the Motion Control instructions of the axis currently being used may be called.

The following diagram presents this principle using Motion Control instruction "MC_Power" as an example:



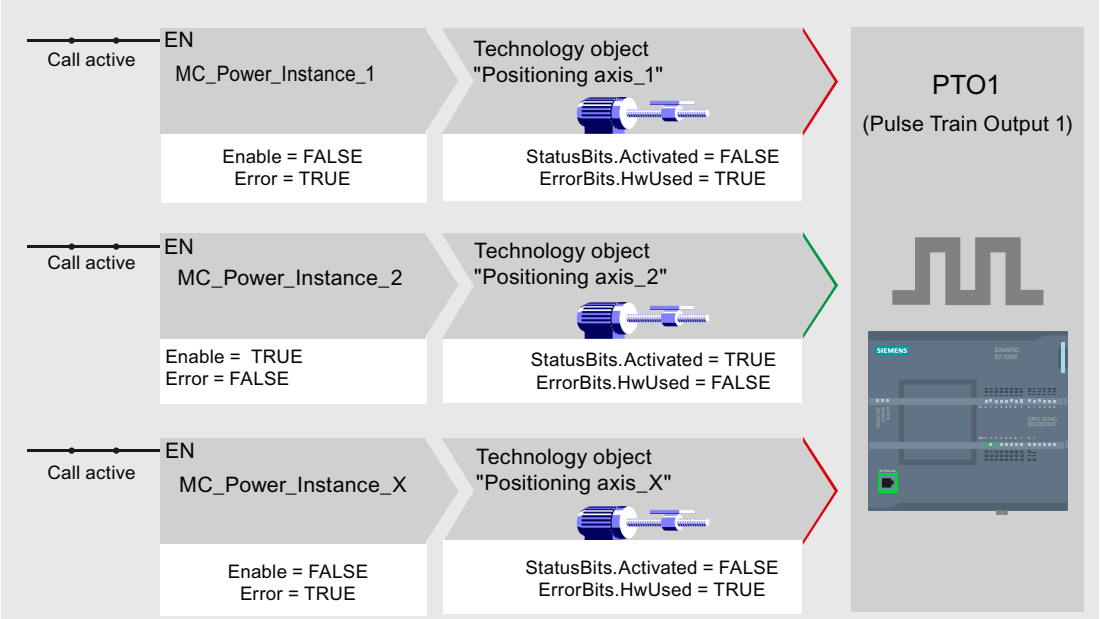
The tags of the activated axis ("Positioning axis_2" here) show the following typical indicators in the user program:

- <Axis name>.StatusBits.Activated = TRUE
- <Axis name>.ErrorBits.HWUsed = FALSE

To switch from one positioning axis technology object to another, follow the steps described below. In the example, a switch is made from "Positioning axis_2" to "Positioning axis_1":

1. End any active traversing motions of activated "Positioning axis_2"
2. Disable "Positioning axis_2" with the associated Motion Control instruction "MC_Power" using input parameter Enable = FALSE
3. To verify that "Positioning axis_2" has been disabled, use an AND operation of output parameter Status = FALSE of Motion Control instruction "MC_Power" and technology object tag <Axis name>.StatusBits.Enable = FALSE.
4. Deactivate the conditional call of the Motion Control instructions for "Positioning axis_2".
5. Activate the conditional call of the Motion Control instruction for "Positioning axis_1". At the first call of the corresponding Motion Control instruction "MC_Power", "Positioning axis_2" is deactivated and "Positioning axis_1" is activated.
6. Enable "Positioning axis_1" with the associated Motion Control instruction "MC_Power" using input parameter Enable = TRUE.
7. To verify that "Positioning axis_1" has been enabled, use an AND operation of output parameter Status = TRUE of Motion Control instruction "MC_Power" and technology object tag <Axis name>.StatusBits.Enable = TRUE.

It is also always possible to cyclically call all Motion Control instructions of all axes working with a single PTO.



When an axis is enabled (here "Positioning axis_2"), this axis becomes active.

In contrast to the conditional call, the Motion Control instructions of the deactivated axes (here "Positioning axis_1" and "Positioning axis_x") indicate errors. The tags of these axes indicate the status <Axis name>.StatusBits.Activated = FALSE and <Axis name>.ErrorBits.HWUsed = TRUE.

Use the conditional call of the Motion Control instructions if you want to implement the user program without error indicators.

See also

Tags of the positioning axis technology object as of V4 (Page 7498)

Using multiple drives with the same PTO (Page 7467)

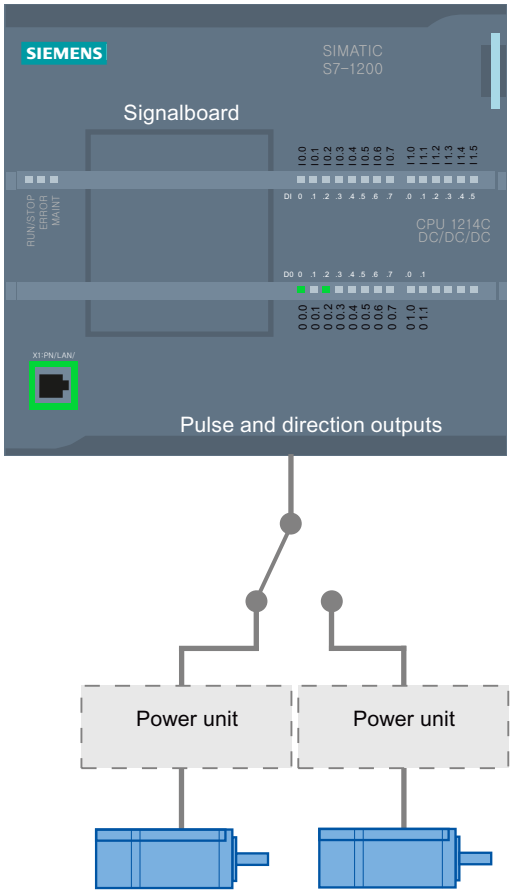
Tracking jobs from higher priority classes (execution levels) (Page 7468)

Special cases when using software limit switches for drive connection via PTO (Page 7470)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

13.2.11.2 Using multiple drives with the same PTO

If multiple drives are to be used, they can be run with a common PTO (Pulse Train Output) using changeover. The following diagram represents the basic circuit design:



The changeover between drives can be controlled, if required, by the user program via a digital output. If different axis configurations are required for the different drives, a changeover between these configurations is required for the PTO. For additional information on this topic, refer to "Using multiple axes with the same PTO (Page 7462)".

See also

Tags of the positioning axis technology object as of V4 (Page 7498)

Using multiple axes with the same PTO (Page 7462)

Tracking jobs from higher priority classes (execution levels) (Page 7468)

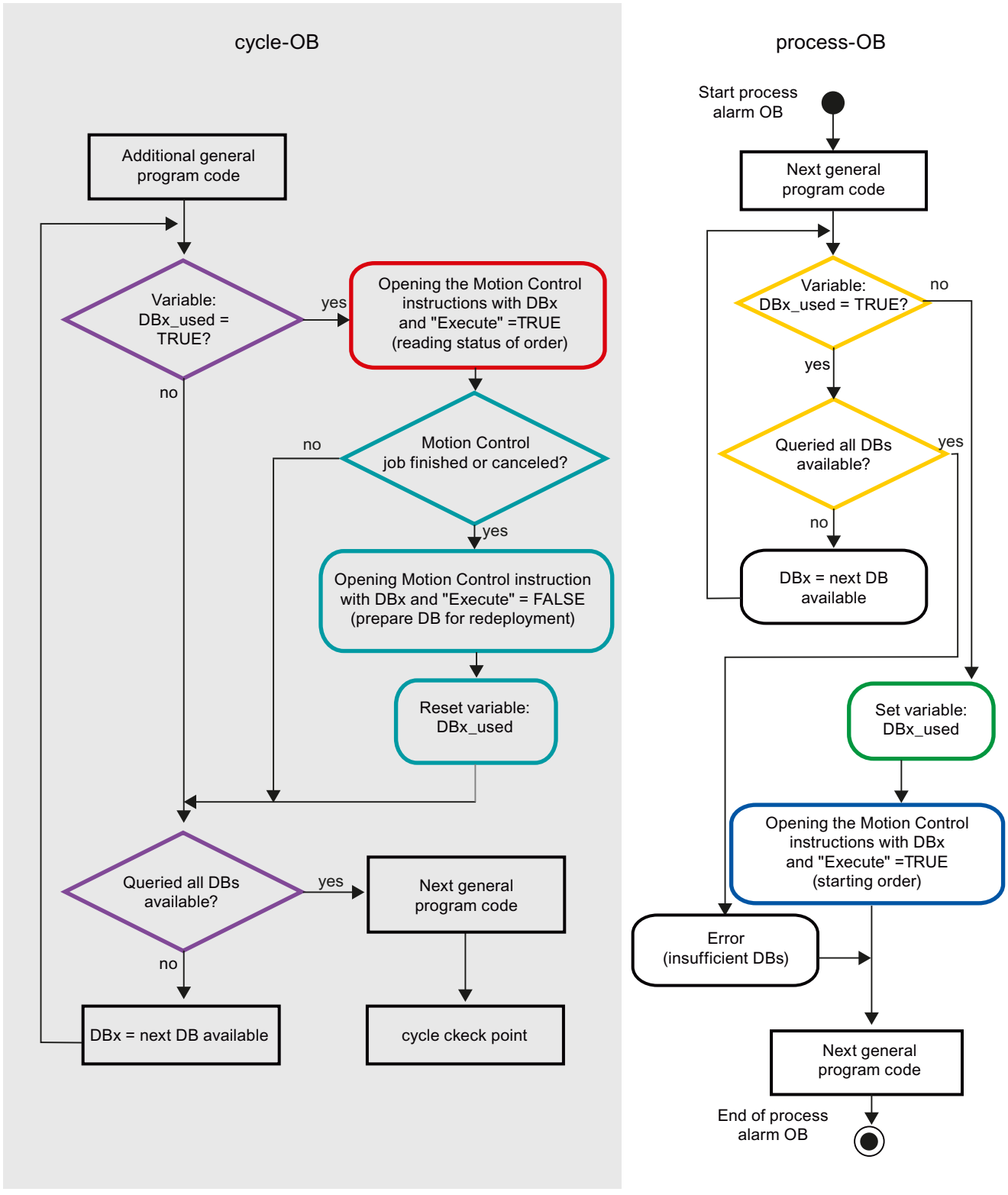
Special cases when using software limit switches for drive connection via PTO (Page 7470)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

13.2.11.3 Tracking jobs from higher priority classes (execution levels)

Depending on the application, it may be necessary to start Motion Control commands (for example, interrupt-controlled) in a higher priority class (execution level).

The Motion Control instructions must be called at short intervals for status monitoring. Motion Control commands cannot be sufficiently closely monitored if the higher priority Motion Control commands are called only once or at too great an interval. Tracking in such cases can be carried out in the cycle OB. An instance data block that is not currently being utilized must be available for each start of a higher priority Motion Control command. Refer to the following flow chart to see how you start Motion Control commands in a higher priority class (for example, hardware interrupt OB) and continue tracking in the program cycle OB:



Depending on the frequency of the Motion Control commands you want to start, you will have to generate a sufficient number of instance data blocks. Users determine which instance data block is currently used in the DBx_used tags.

Start of Motion Control command in the hardware interrupt OB

Binary queries of the DBx_used tags (orange) are used to find an instance data block not currently in use. If such an instance data block is found, the utilized instance data block is marked as "used" (green) and the Motion Control command is started with this instance data block (blue).

Any other program sections of the hardware interrupt OB are then executed, followed by a return to the program cycle OB.

Tracking of started Motion Control commands in the program cycle OB

All instance data blocks available in the cycle OB are checked to determine if they are currently in use by means of the DBx_used tag (violet).

If an instance data block is in use (Motion Control command is being processed), the Motion Control instruction with this instance data block and input parameter Execute = TRUE is called to read out the status messages (red).

If the command is complete or has been aborted, the following actions are taken next (blue green):

- Call of Motion Control instruction with input parameter Execute = FALSE
- Resetting the DBx_used tag

This completes the command tracking, and the instance data block is now available for use again.

See also

Tags of the positioning axis technology object as of V4 (Page 7498)

Using multiple axes with the same PTO (Page 7462)

Using multiple drives with the same PTO (Page 7465)

Special cases when using software limit switches for drive connection via PTO (Page 7470)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7477)

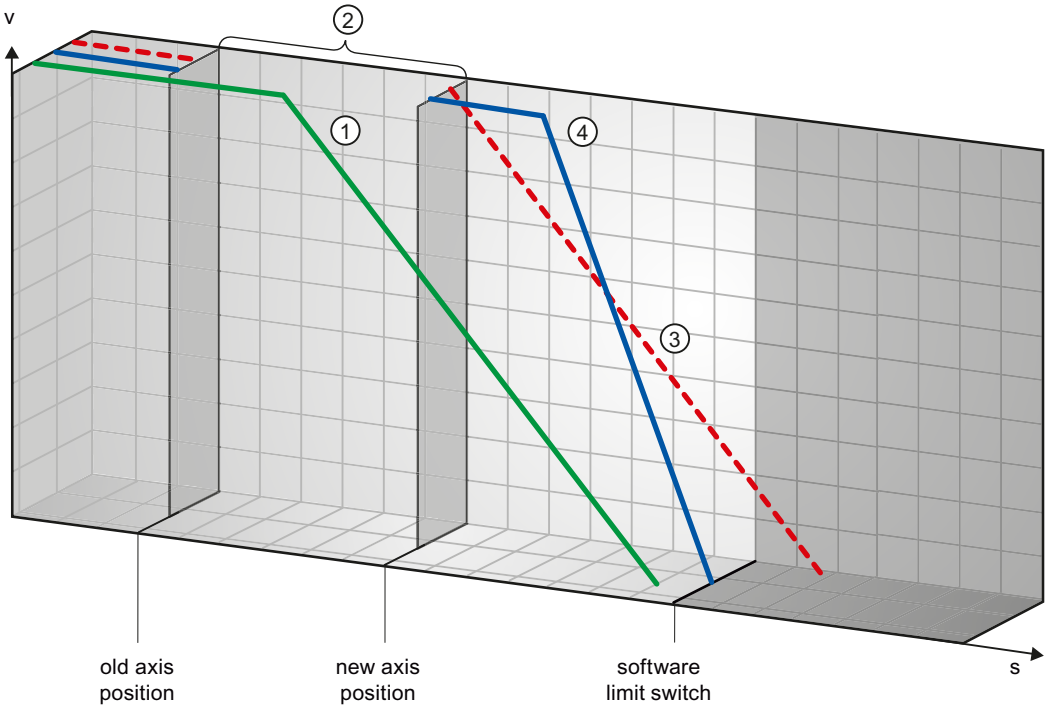
13.2.11.4 Special cases when using software limit switches for drive connection via PTO

Software limit switches in conjunction with a homing operation

Due to unfavorably parameterized homing jobs, the braking action of the axis may be influenced at the software limit switch. Take the following examples into consideration when developing your program.

Example 1:

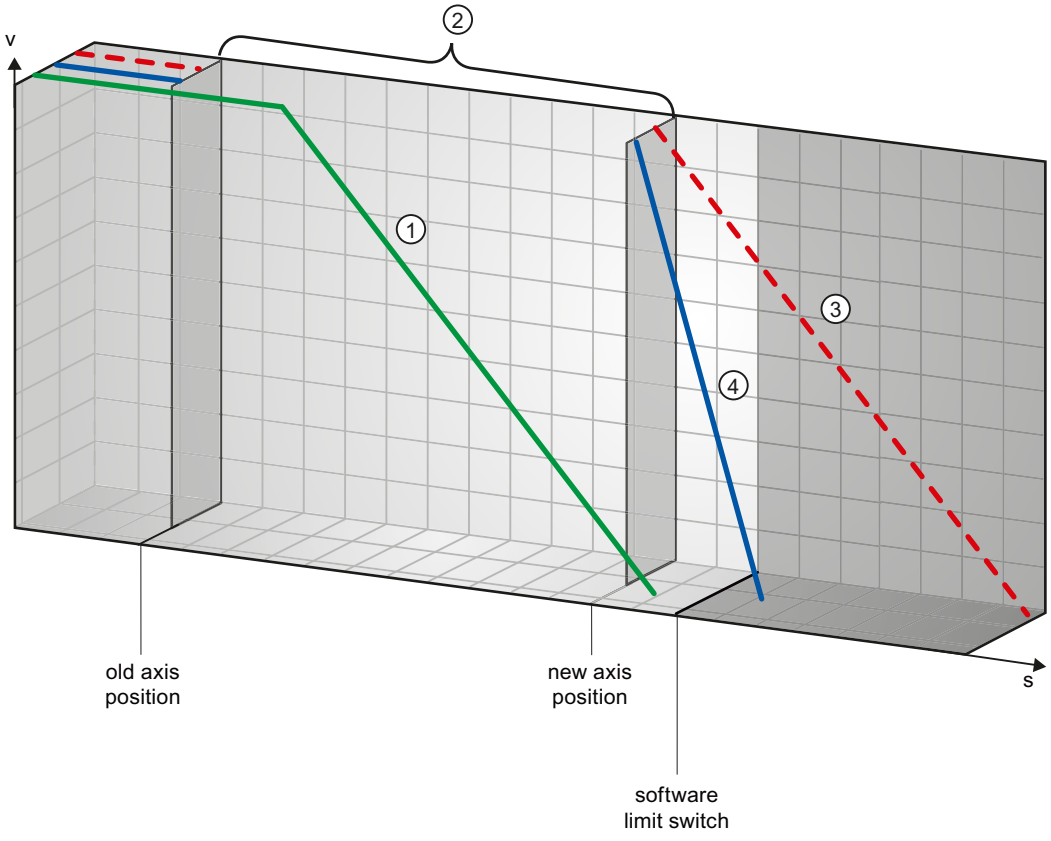
During a travel command, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. It is still possible to bring the axis to a standstill before reaching the software limit switch:



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. Following constant motion, the axis brakes at the configured emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 2:

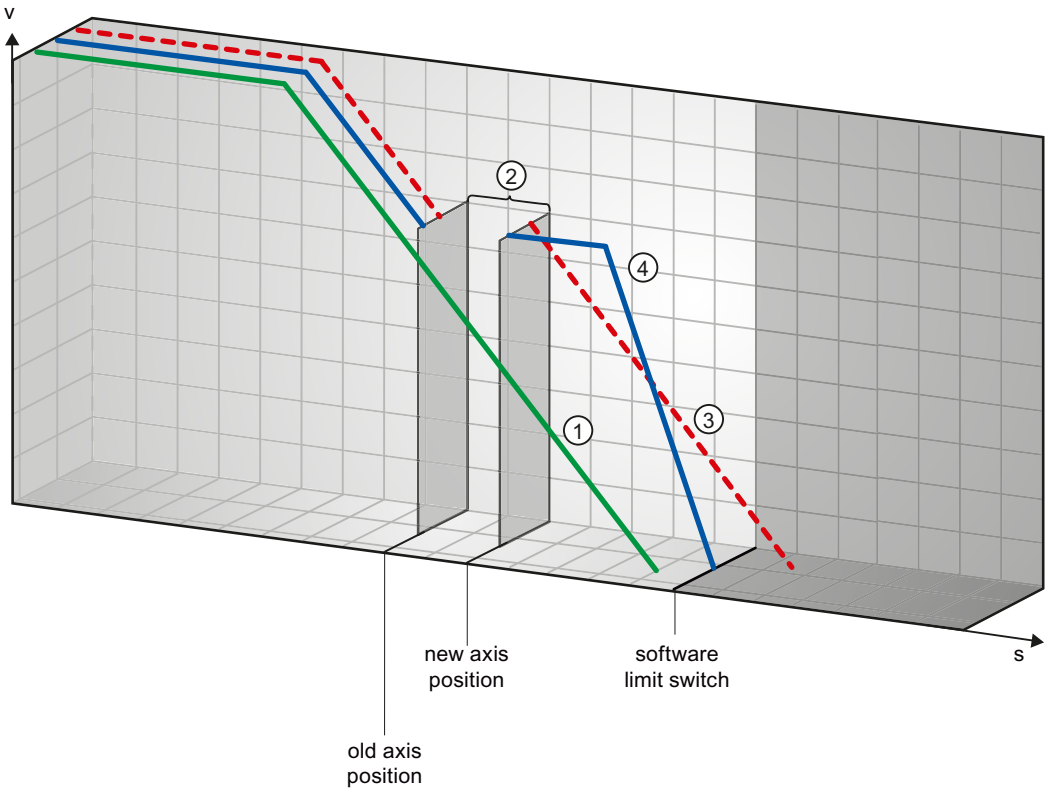
During a travel command, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. In contrast to example 1, it is no longer possible to bring the axis to a standstill before reaching the software limit switch. The axis overruns the position of the software limit switch.



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position well after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. The axis brakes with the configured emergency stop deceleration. However, the emergency stop deceleration is not sufficient to stop the axis at the position of the software limit switch. The position of the software limit switch is overrun.

Example 3:

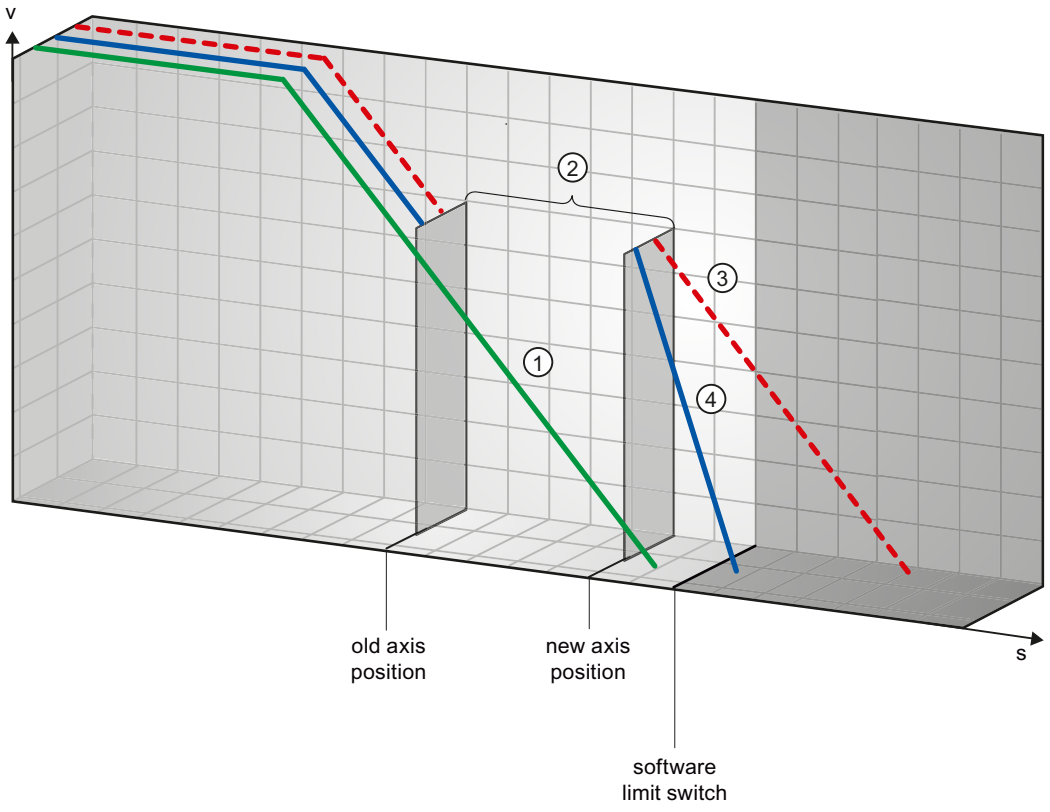
During a braking operation, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. It is still possible to bring the axis to a standstill before reaching the software limit switch:



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. Following constant motion, the axis brakes at the configured emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 4:

During a braking operation, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. In contrast to example 3, it is no longer possible to bring the axis to a standstill before reaching the software limit switch. The axis overruns the position of the software limit switch.



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position well after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. The axis brakes with the configured emergency stop deceleration. However, the emergency stop deceleration is not sufficient to stop the axis at the position of the software limit switch. The position of the software limit switch is overrun.

See also

Software limit switches and software limit switch position changes. (Page 7474)

Software limit switches in conjunction with dynamic changes (Page 7475)

Behavior of axis when position limits is tripped (Page 7377)

Software limit switches and software limit switch position changes.

An incorrect change in the position of the software limit switch during the runtime of the user program can abruptly reduce the distance between the current axis position and the position of the software limit switch.

The axis response is similar to that described in Software limit switches in conjunction with a homing operation (Page 7468).

See also

Software limit switches in conjunction with a homing operation (Page 7468)

Software limit switches in conjunction with dynamic changes (Page 7475)

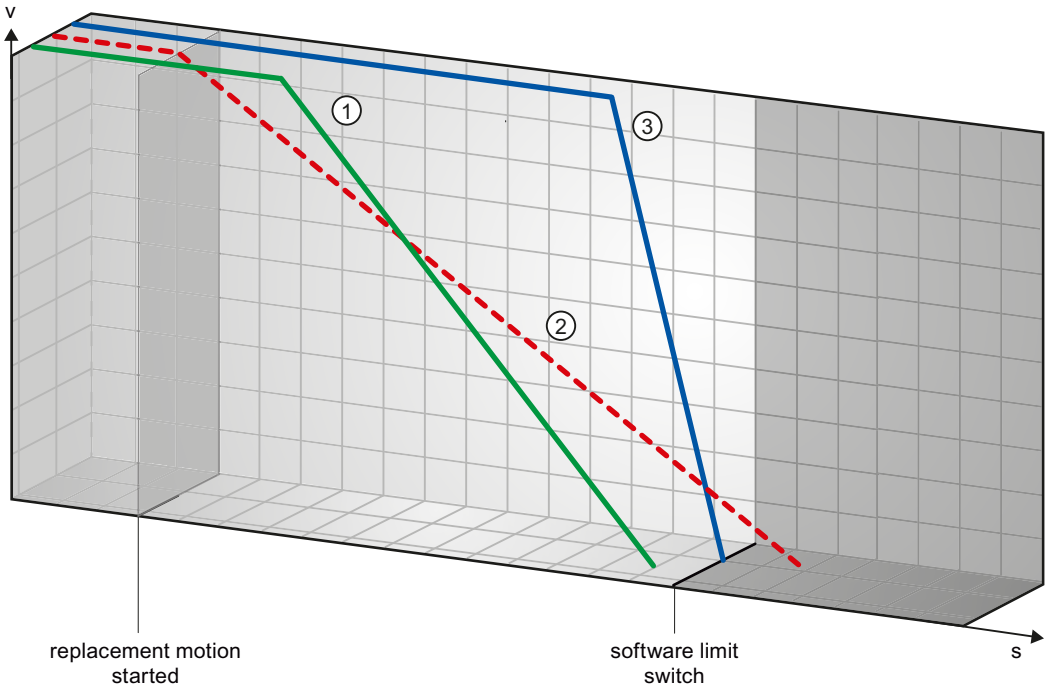
Behavior of axis when position limits is tripped (Page 7377)

Software limit switches in conjunction with dynamic changes

It is possible to influence the deceleration of the axis in the area of the software limit switches in conjunction with overriding motion commands. This applies when the overriding motion command is started with a lower deceleration (tag <Axis name>.DynamicDefaults.Deceleration). Take the following examples into consideration when developing your program.

Example 1:

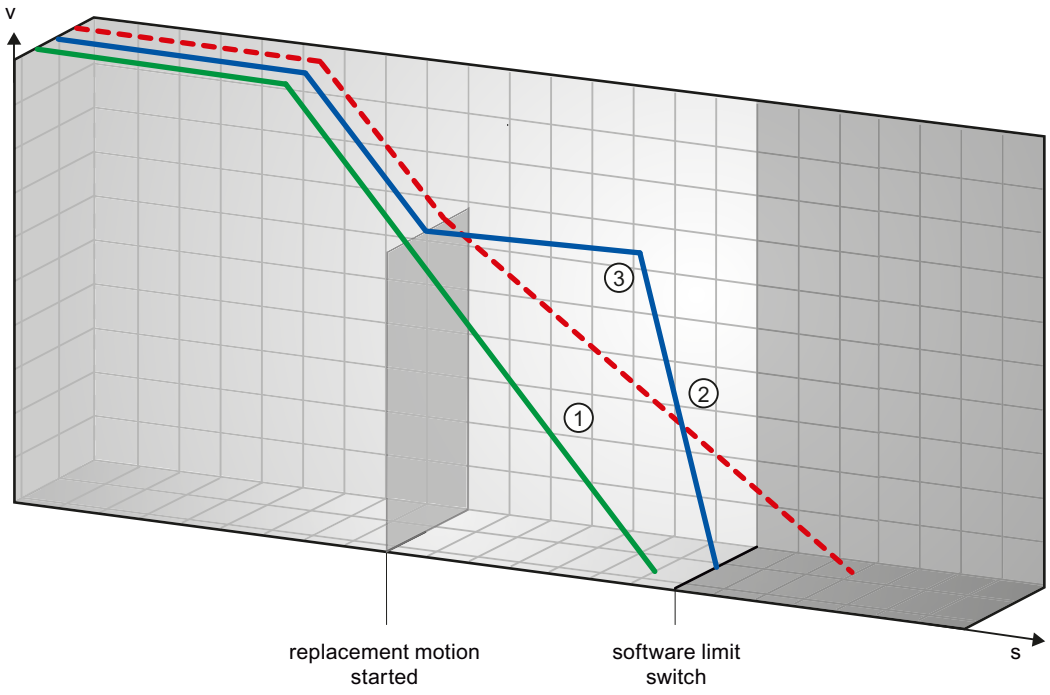
During axis motion, an active motion command is overridden by another motion command with a lower deceleration:



①	The green curve shows the motion of an active command **without** this command being overridden. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	Based on the overriding motion command with lower deceleration, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
③	Because braking with the configured deceleration of the overriding motion command is no longer sufficient, the axis actually follows the blue curve. Following a constant motion, the axis brakes at the emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 2:

During braking of the axis, an active motion command is overridden by another motion command with a lower deceleration:



①	The green curve shows the motion of an active command **without** this command being overridden. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	Based on the overriding motion command with lower deceleration, the axis would theoretically be stopped at a position well after the software limit switch (red curve).
③	Because braking with the configured deceleration of the overriding motion command is no longer sufficient, the axis actually follows the blue curve. Following a constant motion, the axis brakes at the emergency stop deceleration and comes to a standstill at the position of the software limit switch.

See also

Software limit switches in conjunction with a homing operation (Page 7468)

Software limit switches and software limit switch position changes. (Page 7472)

Behavior of axis when position limits is tripped (Page 7377)

13.2.11.5 Reducing velocity for a short positioning duration

The CPU can reduce the velocity of a positioning command when the planned positioning duration is < 2 ms.

The velocity of command execution will then be reduced for the entire duration. The reduced velocity (pulses per s) is calculated as follows:

- Reduced velocity = Number of pulses to be output * 500Hz

Velocity is **not** reduced if the planned positioning duration is >= 2 ms.

13.2.11.6 Dynamic adjustment of start/stop velocity

The configuration of your velocity limits (start/stop velocity and maximum velocity), the dynamic values (acceleration, deceleration and jerk) and the target speed of the traversing command may under certain circumstances result in the start/stop velocity being dynamically adjusted by the CPU.

This is for example the case when, due to a low configured start/stop velocity, the time required for the first pulses would be longer than that possible for the entire acceleration. The first pulse is in these cases output at a greater velocity than the configured start/stop velocity. The subsequent pulses are also dynamically adjusted to ensure the acceleration process can be completed in the specified time.

Ensure in the event of any pulse loss that the hardware (drive) you use is adjusted to this situation, or change the dynamic settings of your axis to avoid dynamic adjustment of the start/stop velocity.

13.2.11.7 List of ErrorIDs and ErrorInfos (technology objects as of V4)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Depending on the error reaction, the axis is stopped in the case of operating errors with stop of axis. The following error reactions are possible:

- **Remove enable**
  The setpoint zero is output and the enable is removed. The axis is braked depending on the configuration in the drive, and is brought to a standstill.
- **Stop with emergency stop ramp**
  Active motion commands are aborted. The axis is braked with the emergency stop deceleration configured under "Technology object > Extended parameters > Dynamics > Emergency stop ramp" without any jerk limit and brought to a standstill.

Operating error with axis stop

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8000		**Drive error, loss of "Drive ready"**		-
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; restart command, if necessary	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8001		**Low SW limit switch has been tripped**		Stop with emergency stop ramp
	16#000E	The position of the low SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the SW limit switch	
	16#000F	The position of the low SW limit switch was reached with the emergency stop deceleration		
	16#0010	The position of the low SW limit switch was exceeded with the emergency stop deceleration		
16#8002		**High SW limit switch has been tripped**		Stop with emergency stop ramp
	16#000E	The position of the high SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the SW limit switch	
	16#000F	The position of the high SW limit switch was reached with the emergency stop deceleration		
	16#0010	The position of the high SW limit switch was exceeded with the emergency stop deceleration		
16#8003		**Low HW limit switch was reached**		For drive connection via PTO (Pulse Train Output): Stop with emergency stop ramp For drive connection via PROFIdrive/analog output: Remove enable
	16#000E	The low HW limit switch was reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the HW limit switch.	
16#8004		**High HW limit switch was reached**		For drive connection via PTO (Pulse Train Output): Stop with emergency stop ramp For drive connection via PROFIdrive/analog output: Remove enable
	16#000E	The high HW limit switch has been reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the HW limit switch.	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8005		**PTO/HSC are already being used by another axis**		-
	16#0001	-	**The axis was configured incorrectly:** Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller	
			More than one axis is to run with one PTO: Another axis is using the PTO / HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 7462))	
16#8006		**A communication error in the control panel has occurred**		Remove enable
	16#0012	A timeout has occurred	Check the cable connection and press the "Manual control" button again	
16#8007		**The axis cannot be enabled**		-
	16#0025	Restarting	Wait until the axis restart is complete.	
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.	
16#8008		**Invalid direction of motion**		-
	16#002E	The selected motion direction is not allowed.	• Adjust the motion direction and restart the command. • Adjust the allowed direction of rotation in the technology object configuration under "Extended parameters > Mechanics". Restart the command.	
	16#002F	A reversing motion is not possible with the selected motion direction.		
16#8009		**Reference switch/encoder zero mark not found**		Stop with emergency stop ramp
	16#0033	Error in the configuration, hardware or installation of the encoder or at the homing switch.	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#800A		**Alarm message from encoder**		Remove enable
	16#0001	-	Check the device with regard to function, connections and I/Os.	
	16#0034	Hardware error at encoder		
	16#0035	Encoder dirty		
	16#0036	Error during reading of encoder absolute value	Compare the encoder type in the drive or encoder parameter P979 with the configuration data of the technology object.	
	16#0037	Zero mark monitoring of the encoder	Encoder reports error in zero mark monitoring (fault code 0x0002 in Gx_XIST2, see PROFIdrive profile). Check the plant for electromagnetic compatibility (EMC).	
	16#0038	Encoder is in "Parking" state	• Search for the cause of the error in the connected drive or encoder. • Check whether the error message was possibly triggered by a commissioning action at the drive or encoder.	
	16#0040	PROFIdrive: Encoder at bus failed (station failure)	Check the device with regard to function, connections and I/Os.	
	16#0041	PROFIdrive: Signs of life of encoder faulty		
16#800B		**Range violation of the position**		Remove enable
	16#0039	Range violation in positive direction	Home the axis to a valid actual value range.	
	16#003A	Range violation in negative direction		
	16#003B	The change of the actual position in a position control clock cycle is greater than the modulo length.	Adjust the modulo length of the employed encoder.	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#800C		**Alarm message from drive**		Remove enable
	16#0001	-	Check the device with regard to function, connections and I/Os.	
	16#003C	PROFIdrive: Drive signal "Control requested" failed		
	16#003D	PROFIdrive: Drive has shut down		
	16#003E	PROFIdrive: Drive at bus failed (station failure)		
	16#003F	PROFIdrive: Signs of life of drive faulty	• Check the device with regard to function, connections and I/Os. • Compare the clock parameters of HW Config (PROFIBUS line, slave OM for drive or encoder) and the execution system. Tmapc and servo must be configured with the same clock cycle time.	
16#800D		**The permitted following error was exceeded**		Remove enable
	16#0001	-	• Check the configuration of the control loop. • Check the direction signal of the encoder. • Check the configuration of following error monitoring.	
16#800E		**Error at the hardware limit switch**		Remove enable
	16#0042	Illegal free travel direction with active hardware limit switch	The programmed direction of movement is disabled due to the active hardware limit switch. Retract the axis in the opposite direction.	
	16#0043	Hardware limit switch polarity is reversed, axis cannot be freed	Check the mechanical configuration of the hardware limit switch.	
	16#0044	Both hardware limit switches are active, axis cannot be freed		
16#800F		**Error in target range**		Remove enable
	16#0045	Target range not reached	Target range was not reached within the positioning tolerance time. • Check the configuration of the position monitoring. • Check the configuration of the control loop.	
	16#0046	Exit target range again	The target range was exited within the minimum dwell time. • Check the configuration of the position monitoring. • Check the configuration of the control loop.	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8010		**Position of the low SW limit switch is greater than that of the high SW limit switch when the axis is not a modulo axis**		Remove enable
	16#0001	-	Change the position of the software limit switches.	

Operating error without axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8200		**Axis is not enabled**	
	16#0001	-	Enable the axis; restart the command
16#8201		**Axis has already been enabled by another "MC_Power" instance**	
	16#0001	-	Enable the axis with only one "MC_Power" instance
16#8202		**The maximum number of simultaneous motion control commands has been exceeded (max. 200 commands for drive connection via PTO (Pulse Train Output), max. 100 commands for drive connection via PROFIdrive/analog output)**	
	16#0001	-	Reduce the number of simultaneously active commands; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		**Axis is currently operated in "Manual control" (axis control panel)**	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		**Axis is not homed**	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		**The axis is currently controlled by the user program (the error is only displayed in the axis control panel)**	
	16#0013	The axis is enabled in the user program	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis control panel
16#8206		**Technology object not activated yet**	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis control panel.
16#8207		**Command rejected**	

ErrorID	ErrorInfo	Description	Remedy
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt".
	16#0018	The axis cannot be moved with a command table while it is being actively or passively homed.	Wait until direct or passive homing is complete.
	16#0019	The axis cannot be actively or passively homed while a command table is being processed.	Wait for command table to finish or abort the command table with a motion command, for example, "MC_Halt".
	16#0052	The specified position exceeds the numerical limit.	Enter a valid position value at the Motion Control instruction.
	16#0053	The axis is ramping up.	Wait until the axis is ready for operation.
	16#0054	Actual value is invalid	To execute a "MC_Home" command, the actual values must be valid. Check the status of the actual values. The tag of the technology object <Axis name>."StatusSensor.State" must show the value 2 (valid).
16#8208		**Difference between maximum and start/stop velocity is invalid**	
	16#0002	Value is not a valid number	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#8209		**Invalid acceleration for technology object "Axis"**	
	16#0002	Value is not a valid number	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#820A		**It is not possible to restart the axis**	
	16#0013	The axis is enabled in the user program	Disable the axis with the "MC_Power" instruction; restart again
	16#0027	The axis is currently being operated in "Manual control" (axis control panel)	Exit "Manual control"; restart again
	16#0047	The technology object is not ready for restart.	Download the project again.
	16#0048	Condition for restart of the technology object is not satisfied.	Disable the technology object.
16#820B		**It is not possible to execute the command table**	
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.
16#820C		**No configuration available**	
	16#0001	-	Internal error Contact the hotline.

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	Value is not a valid number	Correct the value; restart the command
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than 1E +12)	
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Value is not a valid number	Correct the value; restart the command
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than 1E +12)	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	Value is not a valid number	Correct the value; restart the command
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
	16#0024	Value is less than 0	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	The direction reversal is activated at the hardware limit switch, despite the fact that the hardware limit switches are disabled	• Activate the HW limit switch using the tag <Axis name>.PositionLimitsHW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0055	Invalid mode at incremental encoder	Start a homing process for an incremental encoder using parameter "Mode" = 0, 1, 2, 3.
	16#0056	Invalid mode at absolute encoder	Passive and active homing ("Mode" = 2, 3) are not possible for an absolute value encoder. Start a homing process for an absolute encoder using parameter "Mode" = 0, 1.
16#8405		**Invalid value at parameter "StopMode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; enable the axis again
16#8406		**Simultaneous forward and backward jogging is not allowed**	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		**Switching to another axis with instruction "MC_Power" is only permitted after disabling the active axis.**	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.
16#8408		**Invalid value at parameter "Axis" of the Motion Control instruction**	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	

ErrorID	ErrorInfo	Description	Remedy
16#8409		**Invalid value at parameter "CommandTable" of the Motion Control instruction**	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#840A		**Invalid value at parameter "StartStep" of the Motion Control instruction**	
	16#000A	Value is less than or equal to 0.	Correct the value; restart the command
	16#001D	The start step is greater than the end step	
	16#001E	Value is greater than 32	
16#840B		**Invalid value at parameter "EndStep" of the Motion Control instruction**	
	16#000A	Value is less than or equal to 0.	Correct the value; restart the command
	16#001E	Value is greater than 32	
16#840C		**Invalid value at parameter "RampUpTime" of the Motion Control instruction**	
	16#0002	Value is not a valid number	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840D		**Invalid value at parameter "RampDownTime" of the Motion Control instruction**	
	16#0002	Value is not a valid number	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840E		**Invalid value at parameter "EmergencyRampTime" of the Motion Control instruction**	
	16#0002	Value is not a valid number	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840F		**Invalid value at parameter "JerkTime" of the Motion Control instruction**	
	16#0002	Value is not a valid number	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#8410		**Invalid value at parameter "Parameter" of the Motion Control instruction**	
	16#0002	Value is not a valid number	Correct the value; restart the command
	16#000B	Address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0028	Data type of VARIANT pointer "Parameter" and "Value" do not match.	Use a suitable data type; restart command
	16#0029	VARIANT pointer "Parameter" does not point to a data block of the technology object.	Correct the VARIANT pointer; restart the command
	16#002A	The value at the VARIANT pointer "Parameter" cannot be read.	Correct the VARIANT pointer; restart the command
	16#002B	The value at the VARIANT pointer "Parameter" cannot be written.	Correct the VARIANT pointer or value; restart the command
	16#002C	The axis is not disabled.	Disable the axis; restart the command
16#8411		**Invalid value at parameter "Value" of the Motion Control instruction**	
	16#0002	Value is not a valid number	Correct the value; restart the command

Configuration error of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8600		**Parameter assignment of pulse generator (PTO is invalid**	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8601		**Parameter assignment of the high-speed counter (HSC) is invalid**	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8602		**Invalid parameter assignment of "Enable output"**	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8603		**Invalid parameter assignment of "Ready input"**	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8604		**Invalid "Pulses per motor revolution" value**	
	16#000A	Value is less than or equal to zero	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8605		**Invalid "Distance per revolution" value**	
	16#0002	Value is not a valid number	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#000A	Value is less than or equal to zero	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8606		**Invalid "Start/stop velocity" value**	
	16#0002	Value is not a valid number	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value is higher than the high hardware limit	
	16#0004	Value is lower than the low hardware limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		**Invalid "maximum velocity" value**	
	16#0002	Value is not a valid number	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value is higher than the high hardware limit	
	16#0004	Value is lower than the low hardware limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary

ErrorID	ErrorInfo	Description	Remedy
16#8608		**Invalid "Acceleration" value**	
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the high hardware limit	
	16#0004	Value is lower than the low hardware limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8609		**Invalid "Deceleration" value**	
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the high hardware limit	
	16#0004	Value is lower than the low hardware limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#860A		**Invalid "Emergency stop deceleration" value**	
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the high hardware limit	
	16#0004	Value is lower than the low hardware limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#860B		**Value for position of the low SW limit switch is invalid**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than 1E +12)	
	16#0030	The position value of the low software limit switch is greater than that of the high software limit switch	
16#860C		**Value for position of the high SW limit switch is invalid**	
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than 1E +12)	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#860D		**Invalid address of the low HW limit switch**	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#860E		**Invalid address of the high HW limit switch**	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#860F		**Invalid "home position offset" value**	
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than 1E +12)	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8610		**Invalid "approach velocity" value**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8611		**Invalid "Homing velocity" value**	
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8612		**Invalid address of the homing switch**	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#8613		**During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured**	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8614		**Invalid "Jerk" value**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#001F	Value is greater than the maximum jerk	
	16#0020	Value is less than the minimum jerk	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8615		**Value for "Unit of measurement" is invalid**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8616		**Address of homing switch is invalid (passive homing as of V4)**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8617		**Value of tag <Axis name>.Sensor.Sensor[1].ActiveHoming.Mode is invalid**	
	16#0011	The selection value is invalid (Valid value: 2 = Homing via digital input)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8618		**Value of tag <Axis name>.Sensor.Sensor[1].PassiveHoming.Mode is invalid**	
	16#0011	The selection value is invalid (Valid value: 2 = Homing via digital input)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8619		**Value of tag <Axis name>.Actor.Type is invalid**	

ErrorID	ErrorInfo	Description	Remedy
	16#0011	The selection value is invalid (Valid value: 2 = Connection via pulse interface)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861A		**Value for "Permitted direction of rotation" is invalid**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#002D	"Both directions" not allowed when direction output is deactivated	
16#861B		**Faulty load gear factors**	
	16#0031	Valid is invalid.	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#861C		**Illegal combination of data for homing with incremental encoder**	
	16#0031	Valid is invalid.	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861D		**The set encoder mounting type is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Mounting-Mode**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861E		**The configuration of the measuring wheel circumference of the encoder is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.DistancePerRevolution**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861F		**The configuration for the resolution of the linear encoder is faulty. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.Resolution**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary

ErrorID	ErrorInfo	Description	Remedy
16#8620		**The set fine resolution for Gn_XIST1 is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.FineResolutionXist1**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8621		**The set fine resolution for Gn_XIST1 in <Axis name>.Sensor.Sensor[1].Parameter.FineResolutionXist1 is not consistent with the setting in PROFIdrive parameter P979**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8622		**Invalid value for the configuration date <Axis name>.Actor.Interface.AddressIn or <Axis name>.Actor.Interface.AddressOut**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8623		**The value set in the tag <Axis name>.Sensor.Sensor[1].Type is invalid.**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8624		**The set encoder system is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].System**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8625		**Parameter of position monitoring is faulty. Invalid value in <Axis name>.PositioningMonitoring.MinDwellTime**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8626		**Parameter of position monitoring is faulty. Invalid value in <Axis name>.PositioningMonitoring.Window**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary

ErrorID	ErrorInfo	Description	Remedy
16#8627		**The configuration of the PROFIdrive interface of the actual value is faulty. Invalid value in <Axis name>.Sensor.Sensor[1].Interface.AddressIn or <Axis name>.Sensor.Sensor[1].Interface.AddressOut**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8628		**Faulty controller factors**	
	16#0030	Value has an incorrect number format or is outside the valid number range	The value for the gain or the precontrol of the control loop is faulty. • Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary (<Axis name>.PositionControl.Kv, <Axis name>.PositionControl.Kpc)
16#8629		**Limit for standstill signal is faulty. Invalid value in <Axis name>.StandStillSignal.VelocityThreshold**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862A		**Parameter of position monitoring is faulty. Invalid value in <Axis name>.PositioningMonitoring.ToleranceTime**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862B		**Inconsistent PROFIBUS parameterization; the sum of Ti and To is greater than one DP cycle**	
	16#0030	Value has an incorrect number format or is outside the valid number range	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#862C		**Parameter of standstill monitoring is faulty. Invalid value in <Axis name>.StandStillSignal.MinDwellTime**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862D		**Parameter of following error monitoring is faulty. Invalid value in <Axis name>.FollowingError.MinValue**	

ErrorID	ErrorInfo	Description	Remedy
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862E		**Invalid value for the configuration date <Axis name>.Modulo.Length**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862F		**Invalid value for the configuration date <Axis name>.Modulo.StartValue**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8630		**Invalid value for the configuration date <Axis name>.Actor.DriveParameter.ReferenceSpeed**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8631		**The set fine resolution for Gn_XIST2 is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.FineResolutionXist2**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8632		**The number of determinable encoder revolutions is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.DeterminableRevolutions**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8633		**The specified approach direction of the homing switch for passive homing is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].PassiveHoming.Direction**	

ErrorID	ErrorInfo	Description	Remedy
16#8634		**Parameter of the following error monitoring is faulty. Invalid value in <Axis name>.FollowingError.MaxValue**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8635		**Parameter of the following error monitoring is faulty. Invalid value in <Axis name>.FollowingError.MinVelocity**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8636		**Controller factor is incorrect. Invalid value of the precontrol factor <Axis name>.PositionControl.Kpc**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8637		**Invalid value for the configuration date <Axis name>.Sensor.Sensor[1].Interface.Type**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8638		**Invalid value for the configuration date <Axis name>.Sensor.Sensor[1].Interface.HSC**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8639		**Error at the drive**	
	16#0049	Configuration error at device	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#004A	The technology needs a smaller servo clock.	Internal system error. Check the project for consistency and reload it into the controller.
	16#004B	Device driver not initialized during ramp-up.	To enable a technology object, the actuator driver must be initialized. Execute the command again later.
16#863A		**Communication to the drive is faulty**	

ErrorID	ErrorInfo	Description	Remedy
	16#004C	Configuration error at device	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004D	The device driver needs a smaller servo clock.	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004E	Error in internal communication with the device	Check the project for consistency and reload it into the controller.
16#863B		**Error at encoder**	
	16#0049	Configuration error at device	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#004A	The technology needs a smaller servo clock.	Internal system error. Check the project for consistency and reload it into the controller.
	16#004B	Device driver not initialized during ramp-up.	To enable a technology object, the actuator driver must be initialized. Execute the command again later.
16#863C		**Communication with encoder is faulty**	
	16#004C	Configuration error at device	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004D	The device driver needs a smaller servo clock.	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004E	Error in internal communication with the device	Check the project for consistency and reload it into the controller.
16#863D		**Communication to the device (drive or encoder) is faulty**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0055	The requested logical address is invalid.	• Connect a suitable device. • Check the device (I/Os). • Check the topological configuration in HW Config. • Compare the configuration of HW Config and the technology object.
	16#0056	The requested logical output address is invalid.	
	16#0057	The requested logical output address is invalid.	
16#863E		**Value of tag "ControlPanel.Input.TimeOut" is invalid (axis control panel)**	

ErrorID	ErrorInfo	Description	Remedy
	16#0030	Value has an incorrect number format or is outside the valid number range	Correct the value in the tags of the technology object <Axis name>.ControlPanel.Input.TimeOut. The value is specified in milliseconds (ms).
16#863F		**Invalid value for the configuration date <Axis name>.Actor.DriveParameter.MaxSpeed**	
	16#0030	Value has an incorrect number format or is outside the valid number range	Correct the reference value in the drive and in the configuration of the technology object to Actuator.MaxSpeed/2. With analog drive connection, correct the reference value in the drive and in the configuration of the technology object to Actuator.MaxSpeed/1.17.

Configuration error of the command table

ErrorID	ErrorInfo	Description	Remedy
16#8700		**Value for "Command type" in the command table is invalid**	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
16#8701		**Value for "Position / travel path" in the command table is invalid**	
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than 1E +12)	
16#8702		**Value for "Velocity" in the command table is invalid**	
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
16#8703		**Value for "Duration" in the command table is invalid**	
	16#0002	Value is not a valid number	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0021	Value is greater than 64800 s	
	16#0022	Value is less than 0.001 s	
16#8704		**Value for "Next step" in the command table is invalid**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0023	The command transition is not permitted for this command	

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		**Internal error**	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

Tags of the positioning axis technology object as of V4 (Page 7498)

ErrorIDs and ErrorInfos (Page 7555)

Error displays of the Motion Control statements (Page 7455)

Using multiple axes with the same PTO (Page 7462)

Using multiple drives with the same PTO (Page 7465)

Tracking jobs from higher priority classes (execution levels) (Page 7466)

Special cases when using software limit switches for drive connection via PTO (Page 7468)

13.2.11.8 Tags of the positioning axis technology object as of V4

Position tag as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle control point.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tags is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Position				
Setpoint position of the axis (indicated in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

See also

Motion status (Page 7460)

Tag of the axis technology object V1...3 (Page 7568)

Velocity tag as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle control point.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tags is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Velocity				
Setpoint velocity of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

See also

Motion status (Page 7460)

Tag of the axis technology object V1...3 (Page 7568)

ActualPosition tag as of V5

Legend

Data type	Data type of tag
Start value	Start value of tag

Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle control point.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tags is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.ActualPosition				
Actual position of the axis (indicated in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

ActualVelocity tag as of V5

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle control point.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tags is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.ActualVelocity				
Actual velocity of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

Actor tags as of V4

Legend

Data type	Data type of tag
Start value	Start value of tag

Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect:	
	2	For drive connection via PTO (Pulse Train Output): When axis is enabled
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Actor.Type

Drive connection

- Positioning axis technology object as of V5:
  0 = The drive is connected via an analog output. All movements of the axis are position-controlled.
  1 = The drive is connected via PROFIdrive telegrams. All movements of the axis are position-controlled.
  2 = The drive is connected via a pulse interface.
- Positioning axis technology object V4:
  The drive is connected via a pulse interface.

Data type	Start value	Access	Effective	HMI
DInt	2	R	-	X

<Axis name>.Actor.InverseDirection

Invert direction

FALSE = The direction is not inverted.

TRUE = The direction is inverted.

Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X
		WP	2, 9	-

<Axis name>.Actor.DirectionMode

Permitted direction of rotation

0 = Both directions

1 = Positive direction

2 = Negative direction

Data type	Start value	Access	Effective	HMI
Int	0	R	-	X
		WP	2, 9	-

<AxisName>.Actor.Interface.AddressIn (Positioning axis technology object as of V5)				
Input address for the PROFIdrive telegram (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<AxisName>.Actor.Interface.AddressOut (Positioning axis technology object as of V5)				
Output address for the PROFIdrive telegram (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.Actor.Interface.EnableDriveOutput				
Enable output (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.Actor.Interface.DriveReadyInput				
Ready input (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.Actor.Interface.PTO				
Pulse output (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	0	-	-	-

<AxisName>.Actor.DriveParameter.ReferenceSpeed (Positioning axis technology object as of V5)				
Reference value (100%) for the setpoint speed of the drive (N-set) The setpoint speed is transmitted in the PROFIdrive telegram as a standardized value from -200% to 200% of the "ReferenceSpeed". For setpoint specification via an analog output, the analog output can be operated in the range from -117% to 117%, provided the drive permits this.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	3000.0	R	-	X
		WP	2, 9	-

<AxisName>.Actor.DriveParameter.MaxSpeed (Positioning axis technology object as of V5)				
Maximum value for the setpoint speed of the drive (N-set) (PROFIdrive: MaxSpeed ≤ 2 × ReferenceSpeed Analog setpoint: MaxSpeed ≤ 1.17 × ReferenceSpeed)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	3000.0	R	-	X
		WP	2, 9	-

<Axis name>.Actor.DriveParameter.PulsesPerDriveRevolution				
Pulses per motor revolution				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	1000.0	R	-	X
		WP	2, 9	-

See also

Tag of the axis technology object V1...3 (Page 7568)

Sensor[1] tags

Sensor[1] tags as of V5

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect:	
	2	For drive connection via PTO (Pulse Train Output): When axis is enabled
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Sensor[1].Type				
Encoder type (internal parameter) 0 = incremental 1 = Absolute				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	0	-	-	-

<Axis name>.Sensor[1].InverseDirection				
Inversion of the actual value FALSE: Actual value is not inverted TRUE: Actual value is inverted				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X
		WP	2, 9	-

<Axis name>.Sensor[1].System				
Encoder system 0 = Linear encoder 1 = Rotary encoder				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	1	R	-	X
		WP	2, 9	-

<Axis name>.Sensor[1].MountingMode				
Type of encoder mounting 0 = Drive end 2 = External				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	0	R	-	X
		WP	2, 9	-

Sensor[1].Interface tags as of V5

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Sensor[1].Interface.Type				
Encoder connection (internal parameter) 0 = PROFIdrive encoder on PROFINET 1 = Encoder on technology module (TM) 2 = Encoder on drive 4 = Encoder on high-speed counter				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	4	-	-	-

<Axis name>.Sensor[1].Interface.AddressIn				
Input address for the PROFIdrive telegram (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.Sensor[1].Interface.AddressOut				
Output address for the PROFIdrive telegram (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.Sensor[1].Interface.HSC				
High-speed counter to which the encoder transfers the actual value (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	-	-	-	-

Sensor[1].Parameter tags as of V5

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	2	For drive connection via PTO (Pulse Train Output): When axis is enabled
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Sensor[1].Parameter.Resolution				
Resolution of a linear encoder (offset between two encoder pulses)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.001	R	-	X
		WP	2, 9	-

<Axis name>.Sensor[1].Parameter.StepsPerRevolution				
Increments per rotary encoder revolution				
Data type	**Start value**	**Access**	**Effective**	**HMI**
UDInt	2048	R	-	X
		WP	2, 9	-

<Axis name>.Sensor[1].Parameter.FineResolutionXist1				
Number of bits for fine resolution Gn_XIST1 (cyclic actual encoder value)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
UDInt	11	R	-	X
		WP	2, 9	-

<Axis name>.Sensor[1].Parameter.FineResolutionXist2				
Number of bits for fine resolution Gn_XIST2 (absolute value of the encoder)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
UDInt	9	R	-	X
		WP	2, 9	-

<Axis name>.Sensor[1].Parameter.DeterminableRevolutions				
Number of differentiable encoder revolutions for a multi-turn absolute value encoder (For a single-turn absolute value encoder = 1; for an incremental encoder = 0)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
UDInt	1	R	-	X
		WP	2, 9	-

<Axis name>.Sensor[1].Parameter.DistancePerRevolution				
Load distance per revolution of an externally mounted encoder				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	100.0	R	-	X
		WP	2, 9	-

Sensor[1].ActiveHoming tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect:	
	1	For drive connection via PTO (Pulse Train Output): When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE → TRUE), disabled, or enabled
	2	For drive connection via PTO (Pulse Train Output): When axis is enabled
	8	For drive connection via PTO (Pulse Train Output): When an active homing command is started
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Sensor[1].ActiveHoming.Mode

Active homing mode

- Positioning axis technology object as of V5:
  0 = Zero mark via PROFIdrive telegram (not PTO)
  1 = Zero mark via PROFIdrive telegram and proximity switch (not PTO)
  2 = Homing via digital input
- Positioning axis technology object V4:
  2 = Homing via digital input

Data type	Start value	Access	Effective	HMI
DInt	2	R	-	X
		WP	2, 9	-

<Axis name>.Sensor[1].ActiveHoming.SideInput				
End of the homing switch to which the axis is homed during active homing FALSE = Lower end TRUE = Upper end				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	1, 8, 10	X

<Axis name>.Sensor[1].ActiveHoming.DigitalInputAddress				
Symbolic input address of the homing switch (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.Sensor[1].ActiveHoming.HomePositionOffset				
Home position offset (active homing) (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	1, 8, 10	X

<Axis name>.Sensor[1].ActiveHoming.SwitchLevel				
Selection of signal level that is present at the CPU input when the homing switch is reached FALSE = Low level (Low-active) TRUE = High level (High-active)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	TRUE	RW	1, 8, 10	X

See also

Tag of the axis technology object V1...3 (Page 7568)

Sensor[1].PassiveHoming tags as of V4

Legend

Data type	Data type of tag
Start value	Start value of tag The start value can be overwritten by the axis configuration.

Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect:	
	1	For drive connection via PTO (Pulse Train Output): When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE → TRUE), disabled, or enabled
	2	For drive connection via PTO (Pulse Train Output): When axis is enabled
	7	For drive connection via PTO (Pulse Train Output): When a passive homing command is started
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Sensor[1].PassiveHoming.Mode

Passive homing mode

- Positioning axis technology object as of V5:
  0 = Zero mark via PROFIdrive telegram (not PTO)
  1 = Zero mark via PROFIdrive telegram and proximity switch (not PTO)
  2 = Homing via digital input
- Positioning axis technology object V4:
  2 = Homing via digital input

Data type	Start value	Access	Effective	HMI
DInt	2	R	-	X
		WP	2, 9	-

<Axis name>.Sensor[1].PassiveHoming.SideInput

End of the homing switch to which the axis is homed during passive homing

FALSE = Lower end

TRUE = Upper end

Data type	Start value	Access	Effective	HMI
Bool	FALSE	RW	1, 7, 10	X

<Axis name>.Sensor[1].PassiveHoming.DigitalInputAddress				
Symbolic input address of the homing switch (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.Sensor[1].PassiveHoming.SwitchLevel				
Selection of level that is present at the CPU input when the homing switch is reached FALSE = Low level (Low-active) TRUE = High level (High-active)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	TRUE	RW	1, 7, 10	X

See also

Tag of the axis technology object V1...3 (Page 7568)

Units tag as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	2	For drive connection via PTO (Pulse Train Output): When axis is enabled
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Units.LengthUnit				
Configured unit of measurement of the parameter -1 = "Pulse" 1005 = "°" (degrees) 1013 = "mm" 1010 =: "m" 1018 = "ft" 1019 = "in"				
Data type	**Start value**	**Access**	**Effective**	**HMI**

<Axis name>.Units.LengthUnit				
Int	1013	R	-	X
		WP	2, 9	-

See also

Tag of the axis technology object V1...3 (Page 7568)

Mechanics tag as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	2	For drive connection via PTO (Pulse Train Output): When axis is enabled
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Mechanics.LeadScrew				
Distance per revolution (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	10.0	R	-	X
		WP	2, 9	-

See also

Tag of the axis technology object V1...3 (Page 7568)

Modulo tags as of V5

Legend

Data type	Data type of tag
Start value	Start value of tag The start value can be overwritten by the axis configuration.

Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	2	For drive connection via PTO (Pulse Train Output): When axis is enabled
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Modulo.Enable				
Enable modulo FALSE = Modulo conversion deactivated TRUE = Modulo conversion activated When modulo conversion is enabled, a check is made for modulo length > 0.0				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X
		WP	2, 9	-

<Axis name>.Modulo.Length				
Modulo length				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	360.0	R	-	X
		WP	2, 9	-

<Axis name>.Modulo.StartValue				
Modulo start value				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	R	-	X
		WP	2, 9	-

DynamicLimits tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".

Effective	Specifies when a change to the tag takes effect:	
	2	For drive connection via PTO (Pulse Train Output): When axis is enabled
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.DynamicLimits.MaxVelocity				
Maximum velocity of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	250.0	R	-	X
		WP	2, 9	-

<Axis name>.DynamicLimits.MinVelocity				
Start/stop velocity of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	10.0	R	-	X
		WP	2, 9	-

See also

Tag of the axis technology object V1...3 (Page 7568)

DynamicDefaults tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".

Effective	Specifies when a change to the tag takes effect:	
	1	For drive connection via PTO (Pulse Train Output): When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE → TRUE), disabled, or enabled
	5	For drive connection via PTO (Pulse Train Output): The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable, or active MC_Home command is started (Mode = 3)
	6	For drive connection via PTO (Pulse Train Output): When a MC_MoveJog command is stopped
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.DynamicDefaults.Acceleration				
Acceleration of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	48.0	RW	1, 5, 6, 10	X

<Axis name>.DynamicDefaults.Deceleration				
Deceleration of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	48.0	RW	1, 5, 6, 10	X

<Axis name>.DynamicDefaults.EmergencyDeceleration				
Emergency stop deceleration of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	120.0	RW	1, 5, 6, 10	X

<Axis name>.DynamicDefaults.Jerk				
Jerk during acceleration and deceleration ramp of the axis (indicated in the configured unit of measurement) The jerk is activated if the configured jerk is greater than 0.00004 mm/s².				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	192.0	RW	1, 5, 10	X

See also

Tag of the axis technology object V1...3 (Page 7568)

PositionLimitsSW tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	1	For drive connection via PTO (Pulse Train Output): When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE → TRUE), disabled, or enabled
	5	For drive connection via PTO (Pulse Train Output): The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable, or active MC_Home command is started (Mode = 3)
	6	For drive connection via PTO (Pulse Train Output): When a MC_MoveJog command is stopped
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.PositionLimitsSW.Active				
Activation of the software limit switches FALSE = The software limit switches are deactivated. TRUE = The software limit switches are activated.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	1, 5, 6. 10	X

<Axis name>.PositionLimitsSW.MinPosition				
Position of the low software limit switch (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	-10000.0	RW	1, 5, 6, 10	X

<Axis name>.PositionLimitsSW.MaxPosition				
Position of the high software limit switch (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	10000.0	RW	1, 5, 6, 10	X

See also

Tag of the axis technology object V1...3 (Page 7568)

PositionLimitsHW tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect:	
	1	For drive connection via PTO (Pulse Train Output): When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE → TRUE), disabled, or enabled
	2	For drive connection via PTO (Pulse Train Output): When axis is enabled
	5	For drive connection via PTO (Pulse Train Output): The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable, or active MC_Home command is started (Mode = 3)
	6	For drive connection via PTO (Pulse Train Output): When a MC_MoveJog command is stopped
	9	For drive connection via PROFIdrive/analog output: With the restart of the technology object
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.PositionLimitsHW.Active				
Activation of the hardware limit switches FALSE = The hardware limit switches are deactivated. TRUE = The hardware limit switches are activated.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	1, 5, 6, 10	X

<Axis name>.PositionLimitsHW.MinSwitchLevel				
Selection of signal level that is present at the CPU input when the low hardware limit switch is reached FALSE = Low level (Low-active) TRUE = High level (High-active)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X
		WP	2, 9	-

<Axis name>.PositionLimitsHW.MinSwitchAddress				
Symbolic input address of the low hardware limit switch (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.PositionLimitsHW.MaxSwitchLevel				
Selection of signal level that is present at the CPU input when the high hardware limit switch is reached FALSE = Low level (Low-active) TRUE = High level (High-active)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X
		WP	2, 9	-

<Axis name>.PositionLimitsHW.MaxSwitchAddress				
Input address of the high hardware limit switch (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

See also

Tag of the axis technology object V1...3 (Page 7568)

Homing tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".

Effective	Specifies when a change to the tag takes effect:	
	1	For drive connection via PTO (Pulse Train Output): When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE → TRUE), disabled, or enabled
	8	For drive connection via PTO (Pulse Train Output): When an active homing command is started
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Homing.AutoReversal				
Activation of auto reverse at the hardware limit switch during active homing FALSE = Auto reversal at the hardware limit switch is deactivated. TRUE = Auto reversal at the hardware limit switch is activated.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	1, 8, 10	X

<Axis name>.Homing.ApproachDirection				
Approach and homing direction of axis during active homing FALSE = Negative approach direction for finding the homing switch and negative homing direction TRUE = Positive approach direction for finding the homing switch and positive homing direction				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	TRUE	RW	1, 8, 10	X

<Axis name>.Homing.ApproachVelocity				
Approach velocity of the axis during active homing (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	200.0	RW	1, 8, 10	X

<Axis name>.Homing.ReferencingVelocity				
Homing velocity of the axis during active homing (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	40.0	RW	1, 8, 10	X

See also

Tag of the axis technology object V1...3 (Page 7568)

PositionControl tag as of V5

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.PositionControl.Kv				
Proportional gain in the position control ("Kv" > 0.0)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	10.0	R	-	X
		WP	10	-

<Axis name>.PositionControl.Kpc				
Velocity precontrol of the position control as a percentage				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	100.0	R	-	X
		WP	10	-

FollowingError tags as of V5

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".

Effective	Specifies when a change to the tag takes effect:	
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.FollowingError.EnableMonitoring				
Enable following error monitoring FALSE = Following error monitoring disabled TRUE = Following error monitoring enabled				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	TRUE	R	-	X
		WP	9	-

<Axis name>.FollowingError.MinValue				
Permissible following error at velocities below the value of "MinVelocity".				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	10.0	R	-	X
		WP	10	-

<Axis name>.FollowingError.MaxValue				
Maximum permissible following error which may be reached at the maximum velocity.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	100.0	R	-	X
		WP	10	-

<Axis name>.FollowingError.MinVelocity				
"MinValue" is permissible below this velocity, and is held constant.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	10.0	R	-	X
		WP	10	-

PositionMonitoring tags as of V5

Legend

Data type	Data type of tag
Start value	Start value of tag The start value can be overwritten by the axis configuration.

Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	9	For drive connection via PROFIdrive/analog output: After restart of the technology object with the enabling of the axis
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.PositioningMonitoring.ToleranceTime				
Tolerance time Maximum permitted duration from reaching velocity setpoint zero until entrance into the positioning window.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	1.0	R	-	X
		WP	10	-

<Axis name>.PositioningMonitoring.MinDwellTime				
Minimum dwell time in positioning window				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.1	R	-	X
		WP	10	-

<Axis name>.PositioningMonitoring.Window				
Positioning window				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	1.0	R	-	X
		WP	10	-

<Axis name>.PositioningMonitoring.Enable				
Enable position monitoring FALSE = Position monitoring disabled TRUE = Position monitoring enabled				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X
		WP	9	-

StandstillSignal tags as of V5

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.StandstillSignal.VelocityThreshold				
Velocity threshold If velocity is below this threshold, then the minimum dwell time begins.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	5.0	R	-	X
		WP	10	-

<Axis name>.StandstillSignal..MinDwellTime				
Minimum dwell time				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.01	R	-	X
		WP	10	-

StatusPositioning tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle control point.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tags is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.StatusPositioning.Distance				
Current distance of axis from target position (indicated in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute", "MC_MoveRelative", or the axis control panel.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

<Axis name>.StatusPositioning.TargetPosition				
Target position of the axis (indicated in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute", "MC_MoveRelative", or the axis control panel.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

<AxisName>.StatusPositioning.FollowingError (Positioning axis technology object as of V5)				
Current following error of the axis (indicated in the configured unit of measurement) FollowingError = 0.0 for drive connection via PTO (Pulse Train Output).				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

See also

Motion status (Page 7460)

Tag of the axis technology object V1...3 (Page 7568)

StatusDrive tags as of V5

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle control point.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tags is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect:	
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.StatusDrive.InOperation				
Operational status of the drive FALSE = Drive not ready. Setpoints will not be executed. TRUE = Drive ready. Setpoints can be executed.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusDrive.CommunicationOK				
Cyclic BUS communication between controller and drive FALSE = Communication not established TRUE = Communication established				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

StatusSensor tags as of V5

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle control point.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tags is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect:	
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.StatusSensor.State				
Status of the encoder value 0 = Invalid 1 = Wait for status valid 2 = Valid				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	0	RCCP, RP	-	X

<Axis name>.StatusSensor.CommunicationOK				
Cyclic BUS communication between controller and encoder FALSE = Communication not established TRUE = Communication established				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusSensor.AbsEncoderOffset				
Home point offset to the value of an absolute value encoder. The value will be retentively stored in the CPU.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

StatusBits tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle control point.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tags is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.StatusBits.Activated				
Activation of the axis FALSE = The axis is not activated. TRUE = The axis is activated. It is connected to the assigned PTO (Pulse Train Output). The data of the technology data block will be updated cyclically.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.Enable				
Enable status of the axis FALSE = The axis is not enabled. TRUE = The axis is enabled and ready to accept Motion Control commands.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.HomingDone				
Homing status of the axis FALSE = The axis is not homed. TRUE = The axis is homed and is capable of executing absolute positioning commands. The axis does not have to be homed for relative positioning. The status is FALSE during active homing. The status remains TRUE during passive homing if the axis was already homed beforehand.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.Done				
Command execution on the axis FALSE = A Motion Control command is active on the axis. TRUE = A Motion Control command is not active on the axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.Error				
Error status on the axis FALSE = No error is active on the axis. TRUE = An error occurred on the axis. Additional information about the error is available in automatic control at the "ErrorID" and "ErrorInfo" parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.StandStill				
Standstill of the axis FALSE = The axis is in motion. TRUE = The axis is at a standstill.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.PositioningCommand				
Execution of a positioning command FALSE = A positioning command is not active on the axis. TRUE = The axis is executing a positioning command of the "MC_MoveRelative" or MC_MoveAbsolute Motion Control instruction.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.VelocityCommand				
Execution of a command with velocity specification FALSE = A command with velocity specification is not active on the axis. TRUE = The axis is executing a motion command with velocity specification of the "MC_MoveVelocity" or MC_MoveJog Motion Control instruction.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.HomingCommand				
Execution of a homing command FALSE = A homing command is not active on the axis. TRUE = The axis is executing a homing command of the "MC_Home" Motion Control instruction.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.CommandTableActive				
Execution of a command table FALSE = A command table is not active on the axis. TRUE = The axis is controlled by "MC_CommandTable" Motion Control instruction.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.ConstantVelocity				
Constant velocity FALSE = The axis is accelerating, decelerating, or at a standstill. TRUE = The setpoint velocity has been reached. The axis is moving at constant velocity.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.Accelerating				
Acceleration process FALSE = The axis is decelerating, moving at constant velocity, or at a standstill. TRUE = The axis is accelerating.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.Decelerating				
Deceleration process FALSE = The axis is accelerating, moving at constant velocity, or at a standstill. TRUE = The axis is decelerating.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.ControlPanelActive				
Activation status of the axis control panel FALSE = "Automatic mode" is activated. The user program has control priority over the axis. TRUE = The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the axis. The axis cannot be controlled from the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.DriveReady				
Operational status of the drive FALSE = The drive is not ready. Setpoints will not be executed. TRUE = The drive is ready. Setpoints can be executed.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.RestartRequired				
Restart of axis required FALSE = A restart of the axis is not required. TRUE = Values were modified in the load memory. To download the values to the work memory while the CPU is in RUN mode, the axis must be restarted. Use the MC_Reset Motion Control instruction to do this.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.SWLimitMinActive				
Status of the low software limit switch FALSE = The axis is kept within its configured work area. TRUE = The low software limit switch was reached or exceeded.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.SWLimitMaxActive				
Status of the high software limit switch FALSE = The axis is kept within its configured work area. TRUE = The high software limit switch was reached or exceeded.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.HWLimitMinActive				
Status of the low hardware limit switch FALSE = The axis is kept within its configured permitted traversing range. TRUE = The low hardware limit switch was reached or exceeded.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.StatusBits.HWLimitMaxActive				
Status of the high hardware limit switch FALSE = The axis is kept within its configured permitted traversing range. TRUE = The high hardware limit switch was reached or exceeded.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

See also

Status and error bits (technology objects as of V4) (Page 7457)

Tag of the axis technology object V1...3 (Page 7568)

ErrorBits tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle control point.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tags is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.ErrorBits.SystemFault				
Internal system error				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.ErrorBits.ConfigFault				
Incorrect configuration of the axis				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.ErrorBits.DriveFault				
Error in the drive. Loss of the "Drive ready" signal.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.ErrorBits.SWLimit				
Software limit switch reached or exceeded				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.ErrorBits.HWLimit				
Hardware limit switch reached or exceeded				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.ErrorBits.DirectionFault				
Impermissible motion direction				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP, RP	-	X

<Axis name>.ErrorBits.HWUsed				
Another axis is using the same PTO (Pulse Train Output) and is enabled with "MC_Power".				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<AxisName>.ErrorBits.SensorFault (Positioning axis technology object as of V5)				
Error in the encoder system				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.ErrorBits.CommunicationFault (Positioning axis technology object as of V5)				
Communication error Error in communication with a connected device.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.ErrorBits.FollowingErrorFault (Positioning axis technology object as of V5)				
Maximum permitted following error exceeded				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

<Axis name>.ErrorBits.PositioningFault (Positioning axis technology object as of V5)				
Positioning error The axis was not correctly positioned at the end of a positioning motion.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP, RP	-	X

See also

Status and error bits (technology objects as of V4) (Page 7457)

Tag of the axis technology object V1...3 (Page 7568)

ControlPanel tags as of V4

The "ControlPanel" tags do not contain any user-relevant data. These tags cannot be accessed in the user program.

See also

Tag of the axis technology object V1...3 (Page 7568)

Internal tags as of V4

The "Internal" tags do not contain any user-relevant data. These tags cannot be accessed in the user program.

See also

Tag of the axis technology object V1...3 (Page 7568)

Update of the technology object tags

The status and error information of the axis indicated in the technology object tags is updated at each cycle control point.

Changes to the values of configuration tags do not take effect immediately. For information on the conditions under which a change takes effect, refer to the detailed description of the relevant tag.

13.2.11.9 Tags of the command table technology object as of V4

Command[1...32] tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the configuration of the command table.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Command table>.Command[n].Type				
Command type • 0 = "Empty" command • 2 = "Hold" command • 5 = "Relative positioning" command • 6 = "Absolute positioning" command • 7 = "Velocity setpoint" command • 151 = "Wait" command				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Int	0	RW	-	X

<Command table>.Command[n].Position				
Target position/traversing distance of the command				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Axis name>.Command[n].Velocity				
Command velocity				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Command table>.Command[n].Duration				
Command duration				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Command table>.Command[n].NextStep				
Mode for the transition to the next command • 0 = "Complete command" • 1 = "Blend movement"				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Int	0	RW	-	X

<Command table>.Command[n].StepCode				
Command step code				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Word	0	RW	-	X

See also

Tags of the command table technology object V1...3 (Page 7585)

13.2.11.10 Versions V1...4

CPU outputs relevant for motion control (technology version V1...3)

The number of usable drives depends on the CPU, the number of PTOs (pulse train outputs) and the number of available pulse generator outputs.

The following tables provide information about the relevant dependencies:

Maximum number of PTOs

The maximum number of controllable PTOs (drives) depends on the article number of the CPU:

CPU article number	Number of PTOs
xxxxxxx-1xx**30**-xxxx	2
xxxxxxx-1xx**31**-xxxx	4

The maximum number of controllable PTOs (drives) applies regardless of the use of a signal board.

Usable pulse generator outputs

The CPU provides one pulse output and one direction output for controlling a stepper motor drive or a servo motor drive with pulse interface. The pulse output provides the drive with the pulses required for motor motion. The direction output controls the travel direction of the drive.

Pulse and direction outputs are permanently assigned to one another and form a pulse generator output. Onboard CPU outputs or outputs of a signal board can be used as pulse generator outputs. You select between onboard CPU outputs and outputs of the signal board during device configuration under Pulse generators (PTO/PWM) on the "Properties" tab.

The following table shows the number of usable drives per CPU or signal board:

CPU		On-board	Signal board				
			DI2/DO2 x DC24V 20kHz	DI2/DO2 x DC24V 200kHz	DO4 x DC24V 200kHz	DI2/DO2 x DC5V 200kHz	DO4 x DC5V 200kHz
CPU 1211C, CPU 1212C, CPU 1214C (MLFB - article number xxxxxxx-1xx**30**-xxxx)	DC/DC/DC	2	2	2	2	2	2
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1211C (MLFB - article number xxxxxxx-1xx**31**-xxxx)	DC/DC/DC	2	3	3	4	3	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1212C (MLFB - article number xxxxxxx-1xx**31**-xxxx)	DC/DC/DC	3	4	4	4	4	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1214C (MLFB - article number xxxxxxx-1xx**31**-xxxx)	DC/DC/DC	4	4	4	4	4	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1215C	DC/DC/DC	4	4	4	4	4	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2

The following table shows the address assignment of the pulse and direction outputs:

CPU S7-1200		Outputs PTO1 *)		Outputs PTO2 **)		Outputs PTO3 *)		Outputs PTO4 **)	
		Pls.	Dir.	Pls.	Dir.	Pls.	Dir.	Pls.	Dir.
CPU 1211C, CPU 1212C, CPU 1214C, CPU1215C (DC/DC/DC)	CPU	Ax.0	Ax.1	Ax.2	Ax.3	Ax.4	Ax.5	Ax.6	Ax.7
	Signal board	Ay.0	Ay.1	Ay.2	Ay.3	Ay.0	Ay.1	Ay.2	Ay.3
CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C (AC/DC/ RLY)	CPU	-	-	-	-	-	-	-	-
	Signal board	Ay.0	Ay.1	Ay.2	Ay.3	Ay.0	Ay.1	Ay.2	Ay.3
CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C (DC/DC/ RLY)	CPU	-	-	-	-	-	-	-	-
	Signal board	Ay.0	Ay.1	Ay.2	Ay.3	Ay.0	Ay.1	Ay.2	Ay.3

x = Initial byte address of onboard CPU outputs (default value = 0)

y = Initial byte address of signal board outputs (default value = 4)

* If a DC/DC/DC CPU variant is used together with a DI2/DO2 signal board, the signals of the PTO1/3 can be generated via the onboard CPU outputs or via the signal board.

** If a DC/DC/DC CPU variant is used together with a DO4 signal board, the signals for PTO1/3 or PTO2/4 can be generated via the onboard CPU outputs or via the signal board.

PTO3 and PTO4 are only available for CPUs with the article numbers xxxxxxx-1xx**31**-xxxx.

Note

Access to pulse generator outputs via the process image

The firmware takes control via the corresponding pulse generator outputs if the PTO (Pulse Train Output) is selected and assigned to an axis.

With this takeover of the control function, the connection between the process image and I/O output is also disconnected. Although the user has the option of writing the process image of pulse generator outputs via the user program or watch table, this is not transferred to the I/O output. Accordingly, it is also not possible to monitor the I/O output via the user program or watch table. The information read reflects the value of the process image and does not match the real status of the I/O output.

For all other CPU outputs that are not used permanently by the CPU firmware, the status of the I/O output can be controlled or monitored via the process image, as usual.

Limit frequencies of pulse outputs

The following limit frequencies apply to the pulse outputs:

Pulse output	Limit frequencies for technology object positioning axis V1	Limiting frequencies of the technology object positioning axis V2/V3 with CPU < V3.0	Limiting frequencies of the technology object positioning axis V2/V3 with CPU V3.0
On-board (MLFB article number xxxxxxx-1xx**30**-xxxx)	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 100 kHz	1 Hz ≤ f ≤ 100 kHz
On-board (MLFB - article number xxxxxxx-1xx**31**xxxx)	2 Hz ≤ f ≤ 100 kHz (PTO 1+2) 2 Hz ≤ f ≤ 20 kHz (PTO 3+4)	2 Hz ≤ f ≤ 100 kHz (PTO 1+2) 2 Hz ≤ f ≤ 20 kHz (PTO 3+4)	1 Hz ≤ f ≤ 100 kHz (PTO 1+2) 1 Hz ≤ f ≤ 20 kHz (PTO 3+4)
Signal board DI2/DO2 x DC24V 20kHz	2 Hz ≤ f ≤ 20 kHz	2 Hz ≤ f ≤ 20 kHz	1 Hz ≤ f ≤ 20 kHz
Signal board DI2/DO2 x DC24V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz
Signal board DO4 x DC24V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz
Signal board DI2/DO2 x DC5V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz
Signal board DO4 x DC5V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz

Drive signals

For motion control, you can optionally parameterize a drive interface for "Drive enabled" and "Drive ready". When using the drive interface the digital output for the drive enable and the digital input for "drive ready" can be freely selected.

Acceleration/deceleration limits

The following limits apply to acceleration and deceleration:

Acceleration/deceleration	Value (CPU < V3.0)	Value (CPU V3.0)
Minimum acceleration/deceleration	2.8E-1 pulses/s^2	5.0E-3 pulses/s^2
Maximum acceleration/deceleration	9.5E+9 pulses/s^2	9.5E+9 pulses/s^2

Jerk limits

The following limits apply to the jerk:

Jerk	Value (CPU < V3.0)	Value (CPU V3.0)
Minimum jerk	4.0E-2 pulses/s^3	4.0E-3 pulses/s^3
Maximum jerk	1.5E+8 pulses/s^3	1.0E+10 pulses/s^3

See also

CPU outputs relevant for motion control (Page 7331)

Configuration dialogs

V1...3

Configuration - General ("Axis" technology object V1...3)

Configure the basic properties of the "Axis" technology object in the "General" configuration window.

Axis name:

Define the name of the axis or the name of the "Axis" technology object in this box. The technology object is listed under this name in the project navigation.

Hardware interface

The pulses are output to the power unit of the drive by fixed assigned digital outputs.

In CPUs with relay outputs, the pulse signal cannot be output on these outputs because the relays do not support the necessary switching frequencies. A signal board with digital outputs must be used to enable you to work with the PTO (Pulse Train Output) on these CPUs.

Note

The PTO requires the functionality of a fast counter (HSC). An HSC is used for this purpose with CPU version <V3.0. The HSC is then no longer available to the user. An internal HSC is used for this with CPU version ≥ V3.0.

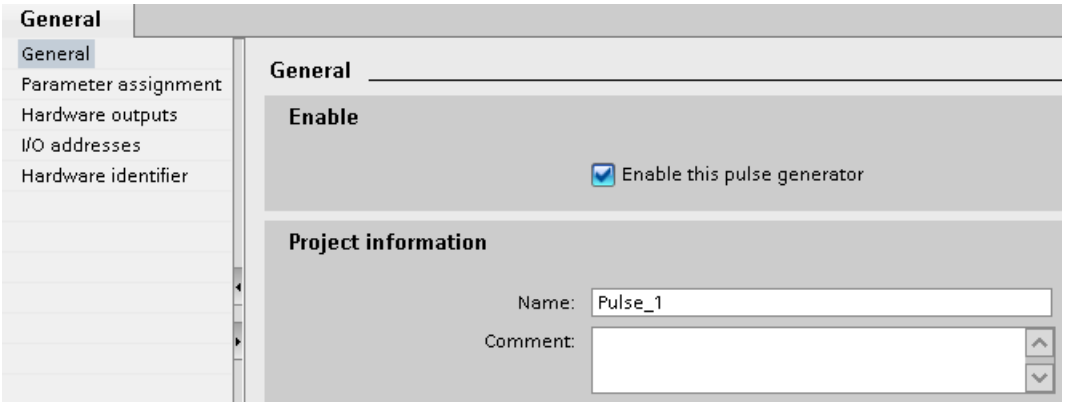
The count can not be evaluated from its input address.

The assignment between PTO and HSC is fixed. When the user activates PTO1, it is connected to the HSC1. If the PTO2 is activated, this is connected with the HSC2.

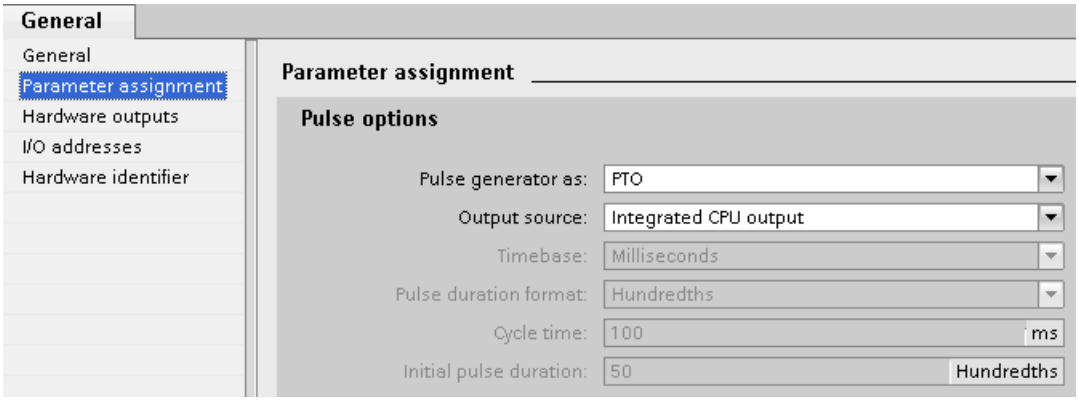
In the "Pulse generator selection" drop-down list, select the PTO (Pulse Train Output) which is to provide the pulses for controlling the stepper motors or servo motors with pulse interface. If the pulse generators and high-speed counters are not used elsewhere in the device configuration, the hardware interface can be configured automatically. In this case, the PTO selected in the drop-down list is displayed with a white background. The interfaces used are listed in the output boxes "Output source", "Pulse output", "Direction output" and "Assigned fast counter".

Proceed as follows if you wish to change the interfaces or if the PTO could not be automatically configured (entry in the "Pulse generator selection" drop-down list is highlighted in red):

1. Click on the "Device configuration" button.
   The pulse generator device configuration opens.
   Enlarge the property window of the device configuration if the configuration of the pulse generator is not visible.



2. Select the "Enable this pulse generator" check box.
3. Select the "Parameter assignment" entry in the area navigation.
   The "Parameter assignment" opens.



4. In the "Pulse generator as:" dropdown list select the "PTO" entry.
5. In the "Output source:" dropdown list select the "Integrated CPU output" or "Signal board output" entry. The "Signal board output" entry can only be selected for PTO1 or for PTO1 and PTO2 depending on the installed signal board. For more detailed information, see section: CPU outputs relevant for motion control (Page 7331)
6. Go back to the axis configuration.
   Unless the corresponding fast counter has already been used elsewhere, the PTO boxes of the "General" axis configuration are not shaded red. If this is not the case, correct the configuration based on the error messages.

User unit

Select the desired unit for the dimension system of the axis in the dropdown list. The selected unit is used for additional configuration of the "Axis" technology object and for displaying the current axis data.

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this unit.

Note

Later changing of the dimension system may not be converted correctly in all the configuration windows of the technology object. In this case check the configuration of all axis parameters.

The values of the input parameters of the Motion Control instructions may have to be adapted to the new unit of measurement in the user program.

See also

Configuration - General (Page 7362)

Configuration - Homing ("Axis" technology object V1)

Configure the parameters for active and passive homing in the "Homing" configuration window. The homing method is set using the "Mode" input parameter of the Motion Control instruction. Here, Mode = 2 means passive homing and Mode = 3 means active homing.

Reference point switch input

Select the digital input for the reference point switch from the drop-down list. The input must be interrupt-capable. The onboard CPU inputs and the inputs of an inserted signal board can be selected as inputs for the reference point switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a reference point switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the reference point switch, the reference point may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the reference point switch.

Permitting direction reversal after reaching the HW limit switch (active homing only)

Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be activated for direction reversal. If the CPU firmware V1.0 is used, both hardware limit switches must be configured. If CPU firmware as of V2.0 is used, only the hardware limit switch in the approach direction must be configured.

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The reference point switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

Note

Use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low
- Increase the configured acceleration/deceleration
- Increase the distance between hardware limit switch and mechanical stop

Approach / homing direction (active and passive homing)

With the direction selection, you determine the "approach direction" used during active homing to search for the reference point switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the reference point switch to carry out the homing operation.

Refer to the table under "Reference point switches" for the effect of the approach direction setting on passive homing.

Side of the reference point switch (active and passive homing)

- Active homing
  This is where you select whether the axis is homed on the low or high end of the reference point switch.

  Note

  Depending on the start position of the axis and the configuration of the homing parameters, the homing procedure sequence can differ from the diagram in the configuration window.

- Passive homing
  With passive homing, the traversing motions for purposes of homing must be implemented by the user via motion commands. The end of the reference point switch on which homing occurs depends on the following factors:
  - "Approach direction" configuration
  - "Reference point switch" configuration
  - Current travel direction during passive homing

  The table below presents details on the effect of factors:

Influencing factors:			Result:
Configuration Approach direction	Configuration Reference point switch	Current travel direction	Homing on Reference point switch
Positive	"Bottom side"	Positive direction	**Top side**
		Negative direction	**Bottom side**
Positive	"Top side"	Positive direction	**Bottom side**
		Negative direction	**Top side**
Negative	"Bottom side"	Positive direction	**Bottom side**
		Negative direction	**Top side**
Negative	"Top side"	Positive direction	**Top side**
		Negative direction	**Bottom side**

Velocity (active homing only)

In this box, specify the velocity at which the reference point switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity (active homing only)

In this box, specify the velocity at which the axis approaches the reference point switch for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset (active homing only)

If the desired home position deviates from the position of the reference point switch, the home position offset can be specified in this box.

If the value does not equal 0, the axis executes the following actions following homing at the reference point switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Configuration - Homing ("Axis" technology object V2...3)

Configuration - Homing - General (Axis technology object V2...3)

Configure the reference point switch input for active and passive homing in the "Homing - General" configuration window.

Reference point switch input

Select the digital input for the homing switch from the drop-down list. The input must be interrupt-capable. The on-board CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the homing switch.

Active level

In the drop-list, select the level of the homing switch that is to be used for homing.

See also

Sequence - Active homing (Page 7390)

Configuration - Homing - Passive (Axis technology object V2...3)

Configure the necessary parameters for passive homing in the "Homing - Passive" configuration window.

The movement for passive homing must be triggered by the user (e.g. using an axis motion command). Passive homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 2.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Note

If passive homing is carried out without an axis motion command (axis at a standstill), homing will be executed upon the next rising or falling edge at the homing switch.

Configuration - Homing - Active (Axis technology object V2...3)

Configure the necessary parameters for active homing in the "Active homing" configuration window. Active homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 3.

Permit auto reverse at the hardware limit switch

Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be enabled for the reversal of direction (at least the hardware limit switch in the direction of approach must be configured).

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The homing switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

Note

If possible, use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low.
- Increase the configured acceleration/deceleration.
- Increase the distance between the hardware limit switch and the mechanical endstop.

Approach/homing direction

With the direction selection, you determine the approach direction used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the homing switch to carry out the homing operation.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Velocity

In this box, specify the velocity at which the homing switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity

In this box, specify the velocity at which the reference point switch is to be approached for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset

If the desired home position deviates from the position of the homing switch, the home position offset can be specified in this box.

If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Changing configuration of dynamic values in user program ("Axis" technology object V1...3)

You can change the following configuration parameters during runtime of the user program in the CPU:

Acceleration and deceleration

You can also change the values for acceleration and deceleration during runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.Config.DynamicDefaults.Acceleration
  for changing acceleration
- <Axis name>.Config.DynamicDefaults.Deceleration
  for changing deceleration

Refer to the description of the technology object tags (Page 7568) in the appendix for information on when changes to the configuration parameters take effect.

Emergency stop deceleration

You can also change the value for the emergency stop deceleration during runtime of the user program. Use the following technology object tag for this purpose:

- <Axis name>.Config.DynamicDefaults.EmergencyDeceleration

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

Notice

After changes to this parameter, it may be necessary to adapt the positions of the hardware limit switches and other safety-relevant settings.

Jerk limit (as of technology object "Axis" V2.0)

You can also activate and deactivate the jerk limit at runtime of the user program and change the value for the jerk. Use the following technology object tags for this purpose:

- <Axis name>.Config.DynamicDefaults.JerkActive
  for activating and deactivating the jerk limit
- <Axis name>.Config.DynamicDefaults.Jerk
  for changing the jerk

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

See also

Changing the configuration of dynamics in the user program (Page 7385)

V4

Configuration - General (positioning axis technology object as of V4)

Configure the basic properties of the positioning axis technology object in the "General" configuration window.

Axis name

Define the name of the axis or the name of the positioning axis technology object in this field. The technology object is listed under this name in the project tree.

Hardware interface

The pulses are output to the power unit of the drive by fixed assigned digital outputs.

In CPUs with relay outputs, the pulse signal cannot be output on these outputs because the relays do not support the necessary switching frequencies. A signal board with digital outputs must be used to enable you to work with the PTO (Pulse Train Output) on these CPUs.

Note

The PTO requires the functionality of a high-speed counter (HSC). An internal HSC is used for this, the count of which cannot be evaluated.

Selecting the pulse generator

In the drop-down list, select the PTO (Pulse Train Output) which is to provide the pulses for controlling the stepper motors or servo motors with pulse interface. If the pulse generators and high-speed counters are not used elsewhere in the device configuration, the hardware interface can be configured automatically. In this case, the PTO selected in the drop-down list is displayed with a white background.

"Device configuration" button

This button takes you to the settings for the pulse options in the device configuration of the CPU.

Signal type

Select the desired signal type in the drop-down list. The following signal types are available:

- **PTO (pulse A and direction B)**
  A pulse output and a direction output are used for controlling the stepper motor.
- **PTO (count up A, count down B)**
  One pulse output each for motion in positive direction and negative direction is used for controlling the stepper motor.
- **PTO (A/B phase offset)**
  Both pulse outputs for Phase A and for Phase B run at the same frequency.
  The period of the pulse outputs is evaluated at the drive end as a step.
  The phase offset between Phase A and Phase B determines the direction of the motion.
- **PTO (A/B phase offset - quadruple)**
  Both pulse outputs for Phase A and for Phase B run at the same frequency.
  All positive edges and all negative edges of Phase A and Phase B are evaluated as a step at the drive end.
  The phase offset between Phase A and Phase B determines the direction of the motion.

Pulse output (signal type "PTO (pulse A and direction B)")

Select the desired output for the pulse output in this field.

You can select the output using a symbolic address or assign it to an absolute address.

Activate direction output (signal type "PTO (pulse A and direction B)")

You can deactivate and activate the direction output in "pulse and direction" mode. This option enables you to limit the motion direction.

Direction output (signal type "PTO (pulse A and direction B)")

Select the desired output for the direction output in this field.

You can select the output using a symbolic address or assign it to an absolute address.

Pulse output up (signal type "PTO (count up A, count down B)")

Select the desired pulse output for motions in positive direction in this field.

You can select the output using a symbolic address or assign it to an absolute address.

Pulse output down (signal type "PTO (count up A, count down B)")

Select the desired pulse output for motions in negative direction in this field.

You can select the output using a symbolic address or assign it to an absolute address.

Phase A (signal types "PTO (A/B phase offset)" and "PTO (A/B phase offset - quadruple)")

Select the desired pulse output for Phase A signals in this field.

You can select the output using a symbolic address or assign it to an absolute address.

Phase B (signal types "PTO (A/B phase offset)" and "PTO (A/B phase offset - quadruple)")

Select the desired pulse output for Phase B signals in this field.

You can select the output using a symbolic address or assign it to an absolute address.

User unit

Select the desired unit for the dimension system of the axis in the drop-down list. The selected unit is used for further configuration of the positioning axis technology object and for displaying the current axis data.

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this unit.

Note

Later changing of the dimension system may not be converted correctly in all the configuration windows of the technology object. In this case check the configuration of all axis parameters.

The values of the input parameters of the Motion Control instructions may have to be adapted to the new unit of measurement in the user program.

Configuration - Drive signals (positioning axis technology object V4)

Configure the output for drive enable and the input for the "Drive ready" feedback signal of the drive in the "Drive signals" configuration window.

Drive enable is controlled by Motion Control instruction "MC_Power" and enables power to the drive. The signal is provided to the drive via the output to be configured.

The drive signals "Drive ready" to the CPU if it is ready to start executing travel after receipt of drive enable. The "Drive ready" signal is reported back to the CPU via the input to be configured.

If the drive does not have any interfaces of this type, you will not have to configure the parameters. In this case, select the value TRUE for the ready input.

See also

Configuration - Mechanics (positioning axis technology object V4) (Page 7548)

Position limits (Page 7375)

Dynamics (Page 7379)

Homing (positioning axis technology object as of V2) (Page 7386)

Configuration - Mechanics (positioning axis technology object V4)

Configure the mechanical properties of the drive in the "Mechanics" configuration window.

Increments per motor revolution

Configure the number of pulses required for one revolution of the motor in this box.

Limits (independent of the selected unit of measurement):

- 0 < Pulse per motor revolution ≤ 2147483647

Distance per revolution

In this box, configure the load distance per motor revolution covered by the mechanical system of your unit.

Limits (independent of the selected unit of measurement):

- 0.0 < Distance per revolution ≤ 1.0e12

Permitted direction of rotation (technology version as of V4)

Configure this box to determine whether the mechanics of your system are to move in both directions or only in the positive or negative direction.

If you have not activated the direction output of the pulse generator in the "PTO (pulse A and direction B)" mode, the selection is limited to the positive or negative direction.

Invert direction

You can use the "Invert direction" check box to adapt the control system to the direction logic of the drive.

The direction logic is inverted according to the selected mode of the pulse generator:

- **PTO (pulse A and direction B)**
  - 0 V at direction output ⇒ positive direction of rotation
  - 5 V/24 V at direction output ⇒ negative direction of rotation

  The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.
- **PTO (count up A, count down B)**
  The outputs "Pulse output down" and "Pulse output up" are swapped.
- **PTO (A/B phase offset)**
  The "Phase A" and "Phase B" outputs are swapped.
- **"PTO (A/B phase offset - quadruple)**
  The "Phase A" and "Phase B" outputs are swapped.

See also

Configuration - Drive signals (positioning axis technology object V4) (Page 7546)

Position limits (Page 7375)

Dynamics (Page 7379)

Homing (positioning axis technology object as of V2) (Page 7386)

Relationship between the signal type and the direction of travel (Page 7334)

Configuration - Homing - Passive (Positioning axis technology object V4)

Configure the necessary parameters for passive homing in the "Homing - Passive" configuration window.

The movement for passive homing must be triggered by the user (e.g. using an axis motion command). Passive homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 2.

Homing switch input

Select the digital input for the homing switch from the drop-down list. The input must be interrupt-capable. The on-board CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the homing switch.

Active level

In the drop-list, select the level of the homing switch that is to be used for homing.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Note

If passive homing is carried out without an axis motion command (axis at a standstill), homing will be executed upon the next rising or falling edge at the homing switch.

Configuration - Homing - Active (Positioning axis technology object V4)

Configure the necessary parameters for active homing in the "Active homing" configuration window. Active homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 3.

Homing switch input

Select the digital input for the homing switch from the drop-down list. The input must be interrupt-capable. The on-board CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the homing switch.

Active level

In the drop-list, select the level of the homing switch that is to be used for homing.

Allowing direction reversal at the hardware limit switch

Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be enabled for the reversal of direction (at least the hardware limit switch in the direction of approach must be configured).

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The homing switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

Note

If possible, use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low.
- Increase the configured acceleration/deceleration.
- Increase the distance between the hardware limit switch and the mechanical endstop.

Approach/homing direction

With the direction selection, you determine the approach direction used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the homing switch to carry out the homing operation.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Approach velocity

In this box, specify the velocity at which the homing switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity

In this box, specify the velocity at which the homing switch is to be approached for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset

If the desired home position deviates from the position of the homing switch, the home position offset can be specified in this box.

If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Diagnostics - Status and error bits ("Axis" technology object V1...3)

You use the "Status and error bits" diagnostic function to monitor the most important status and error messages for the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The status error messages have the following meaning:

Status of the axis

Status	Description
Enabled	The axis is enabled and ready to be controlled via Motion Control commands. (Tag of technology object: <Axis name>.StatusBits.Enable)
Homed	The axis is homed and is capable of executing absolute positioning commands of Motion Control instruction "MC_MoveAbsolute". The axis does not have to be homed for relative positioning. Special situations: • During active homing, the status is FALSE. • If a homed axis undergoes passive homing, the status is set to TRUE during passive homing. (Tag of technology object: <Axis name>.StatusBits.HomingDone)
Axis error	An error has occurred in the "Axis" technology object. Additional information about the error is available in automatic control at the ErrorID and ErrorInfo parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error. (Tag of technology object: <Axis name>.StatusBits.Error)
Axis control panel enabled	The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the "Axis" technology object. The axis cannot be controlled from the user program. (Tag of technology object: <Axis name>.StatusBits.ControlPanelActive)
Restart necessary	A modified configuration of the axis was downloaded to the load memory in CPU RUN operating mode. To download the modified configuration to the work memory, you need to restart the axis. Use the Motion Control instruction MC_Reset to do this.

Drive status

Status	Description
Drive ready	The drive is ready for operation. (Tag of technology object: <Axis name>.StatusBits.DriveReady)
Drive error	The drive has reported an error due to loss of its "Drive ready" signal. (Tag of technology object: <Axis name>.ErrorBits.DriveFault)

Status of the axis motion

Status	Description
Standstill	The axis is at a standstill. (Tag of technology object: <Axis name>.StatusBits.StandStill)
Acceleration	The axis accelerates. (Tag of technology object: <Axis name>.StatusBits.Acceleration)
Constant velocity	The axis travels at constant velocity. (Tag of technology object: <Axis name>.StatusBits.ConstantVelocity)
Deceleration	The axis decelerates (slows down). (Tag of technology object: <Axis name>.StatusBits.Deceleration)

Status of the motion mode

Status	Description
Positioning	The axis executes a positioning command of the Motion Control instruction "MC_MoveAbsolute", "MC_MoveRelative" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.PositioningCommand)
Travel with velocity specification	The axis executes a command with velocity specification of the Motion Control instruction "MC_MoveVelocity", "MC_MoveJog" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.SpeedCommand)
Homing	The axis executes a homing command of the Motion Control instruction "MC_Home" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.Homing)
Command table active (as of technology object Axis V2.0)	The axis is controlled by Motion Control instruction "MC_CommandTable". (Tag of technology object: <Axis name>.StatusBits.CommandTableActive)

Error messages

Error	Description
Lower SW limit switch was reached	The lower software limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMinReached)
Lower SW limit switch was exceeded	The lower software limit switch has been exceeded. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMinExceeded)
Upper SW limit switch was reached	The upper software limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMaxReached)
Upper SW limit switch was exceeded	The upper software limit switch has been exceeded. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMaxExceeded)
Lower HW limit switch was reached	The lower hardware limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.HwLimitMin)
Upper HW limit switch was reached	The upper hardware limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.HwLimitMax)
PTO and HSC already in use	A second axis is using the same PTO (Pulse Train Output) and HSC (High Speed Counter) and is enabled with "MC_Power". (Tag of technology object: <Axis name>.ErrorBits.HwUsed)
Configuration error	The "Axis" technology object was incorrectly configured or editable configuration data were modified incorrectly during runtime of the user program. (Tag of technology object: <Axis name>.ErrorBits.ConfigFault)
Internal error	An internal error has occurred. (Tag of technology object: <Axis name>.ErrorBits.SystemFault)

See also

Status and error bits (technology objects as of V4) (Page 7457)

ErrorIDs and ErrorInfos

List of ErrorIDs and ErrorInfos (technology objects V2...3)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Operating error with axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8000		**Drive error, loss of "Drive ready"**	
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; restart command, if necessary
16#8001		**Lower SW limit switch has been tripped**	
	16#000E	The position of the lower SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the SW limit switch
	16#000F	The position of the lower SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the lower SW limit switch was exceeded with the emergency stop deceleration	
16#8002		**Upper SW limit switch has been tripped**	
	16#000E	The position of the upper SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the SW limit switch
	16#000F	The position of the upper SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the upper SW limit switch was exceeded with the emergency stop deceleration	
16#8003		**Lower HW limit switch was reached**	
	16#000E	The lower HW limit switch was reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the reference point switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the HW limit switch.
16#8004		**Upper HW limit switch was reached**	
	16#000E	The upper HW limit switch has been reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the reference point switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the HW limit switch.
16#8005		**PTO/HSC are already being used by another axis**	

ErrorID	ErrorInfo	Description	Remedy
	16#0001	-	**The axis was configured incorrectly:** Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller
			More than one axis is to run with one PTO: Another axis is using the PTO / HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 7462))
16#8006		**A communication error in the control panel has occurred**	
	16#0012	A timeout has occurred	Check the cable connection and press the "Manual control" button again.
16#8007		**The axis cannot be enabled.**	
	16#0025	Restarting	Wait until the axis restart is complete.
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.

Operating error without axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8200		**Axis is not enabled**	
	16#0001	-	Enable the axis; restart the command
16#8201		**Axis has already been enabled by another "MC_Power" instance**	
	16#0001	-	Enable the axis with only one "MC_Power" instance
16#8202		**The maximum number of simultaneously active Motion Control commands has been exceeded (maximum of 200 commands for all motion control technology objects)**	
	16#0001	-	Reduce the number of simultaneously active commands; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		**Axis is currently operated in "Manual control" (axis control panel)**	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		**Axis is not homed**	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		**The axis is currently controlled by the user program (the error is only displayed in the axis control panel)**	
	16#0013	The axis is enabled in the user program.	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis control panel
16#8206		**Technology object not activated yet**	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis control panel.
16#8207		**Command rejected**	

ErrorID	ErrorInfo	Description	Remedy
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt".
	16#0018	The axis cannot be moved with a command table while it is being actively or passively homed.	Wait until direct or passive homing is complete.
	16#0019	The axis cannot be actively or passively homed while a command table is being processed.	Wait for command table to finish or abort the command table with a motion command, for example, "MC_Halt".
16#8208		**Difference between maximum and start/stop velocity is invalid**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#8209		**Invalid acceleration for technology object "Axis"**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#820A		**It is not possible to restart the axis**	
	16#0013	The axis is enabled in the user program.	Disable the axis with the "MC_Power" instruction; restart again
	16#0027	The axis is currently being operated in "Manual control" (axis control panel)	Exit "Manual control"; restart again
16#820B		**It is not possible to execute the command table**	
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
	16#0024	Value is less than 0	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	

ErrorID	ErrorInfo	Description	Remedy
	16#0011	The selection value is invalid	Correct the selection value; restart the command
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	The direction reversal is activated at the hardware limit switch, despite the fact that the hardware limit switches are disabled	• Activate the hardware limit switch using the tag <axis>.Config.PositionLimits_HW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
16#8405		**Invalid value at parameter "StopMode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; enable the axis again
16#8406		**Simultaneous forward and backward jogging is not allowed**	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		**Switching to another axis with instruction "MC_Power" is only permitted after disabling the active axis.**	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.
16#8408		**Invalid value at parameter "Axis" of the Motion Control instruction**	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#8409		**Invalid value at parameter "CommandTable" of the Motion Control instruction**	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#840A		**Invalid value at parameter "StartStep" of the Motion Control instruction**	
	16#000A	Value is less than or equal to 0	Correct the value; restart the command
	16#001D	The start step is greater than the end step	
	16#001E	Value is greater than 32	
16#840B		**Invalid value at parameter "EndStep" of the Motion Control instruction**	
	16#000A	Value is less than or equal to 0	Correct the value; restart the command
	16#001E	Value is greater than 32	
16#840C		**Invalid value at parameter "RampUpTime" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#840D		**Invalid value at parameter "RampDownTime" of the Motion Control instruction**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#840E		**Invalid value at parameter "EmergencyRampTime" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#840F		**Invalid value at parameter "JerkTime" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	

Configuration error of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8600		**Parameter assignment of pulse generator (PTO is invalid**	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8601		**Parameter assignment of the high-speed counter (HSC) is invalid**	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8602		**Invalid parameter assignment of "Enable output"**	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8603		**Invalid parameter assignment of "Ready input"**	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8604		**Invalid "Pulses per motor revolution" value**	
	16#000A	Value is less than or equal to zero	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8605		**Invalid "Distance per revolution" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#000A	Value is less than or equal to zero	
16#8606		**Invalid "Start/stop velocity" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		**Invalid "maximum velocity" value**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#8608		**Invalid "Acceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#8609		**Invalid "Deceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#860A		**Invalid "Emergency stop deceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#860B		**Value for position of the lower SW limit switch is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
	16#0007	The position value of the lower software limit switch is greater than that of the upper software limit switch	
16#860C		**Value for position of the upper SW limit switch is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#860D		**Invalid address of the lower HW limit switch**	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#860E		**Invalid address of the upper HW limit switch**	

ErrorID	ErrorInfo	Description	Remedy
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#860F		**Invalid "home position offset" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8610		**Invalid "approach velocity" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
16#8611		**Invalid "Homing velocity" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
16#8612		**Invalid address of the reference point switch**	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#8613		**During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured**	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8614		**Invalid "Jerk" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#001F	Value is greater than the maximum jerk	
	16#0020	Value is less than the minimum jerk	
16#8615		**Value for "Unit of measurement" is invalid**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"

Configuration error of the command table

ErrorID	ErrorInfo	Description	Remedy
16#8700		**Value for "Command type" in the command table is invalid**	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
16#8701		**Value for "Position / travel path" in the command table is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$	
16#8702		**Value for "Velocity" in the command table is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
16#8703		**Value for "Duration" in the command table is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0021	Value is greater than 64800 s	
	16#0022	Value is less than 0.001 s	
16#8704		**Value for "Next step" in the command table is invalid**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0023	The command transition is not permitted for this command	

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		**Internal error**	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7475)

Using multiple axes with the same PTO (Page 7462)

List of ErrorIDs and ErrorInfos (technology objects V1)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Operating error with axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8000		**Drive error, loss of "Drive ready"**	
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; restart command, if necessary
16#8001		**Lower SW limit switch has been tripped**	
	16#000E	The position of the lower SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the SW limit switch
	16#000F	The position of the lower SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the lower SW limit switch was exceeded with the emergency stop deceleration	
16#8002		**Upper SW limit switch has been tripped**	
	16#000E	The position of the upper SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the SW limit switch
	16#000F	The position of the upper SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the upper SW limit switch was exceeded with the emergency stop deceleration	
16#8003		**Lower HW limit switch was reached**	
	16#000E	The lower HW limit switch was reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the reference point switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the HW limit switch.
16#8004		**Upper HW limit switch was reached**	
	16#000E	The upper HW limit switch has been reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the reference point switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the HW limit switch.
16#8005		**PTO/HSC are already being used by another axis**	

ErrorID	ErrorInfo	Description	Remedy
	16#0001	-	**The axis was configured incorrectly:** Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller
			More than one axis is to run with one PTO: Another axis is using the PTO / HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 7462))

Operating error without axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8200		**Axis is not enabled**	
	16#0001	-	Enable the axis; restart the command
16#8201		**Axis has already been enabled by another "MC_Power" instance**	
	16#0001	-	Enable the axis with only one "MC_Power" instruction
16#8202		**The maximum number of simultaneously active Motion Control commands was exceeded (maximum of 200 commands for all motion control technology objects)**	
	16#0001	-	Reduce the number of simultaneously active commands; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		**Axis is currently operated in "Manual control" (axis control panel)**	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		**Axis is not homed**	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		**The axis is currently controlled by the user program (the error is only displayed in the axis control panel)**	
	16#0001	-	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis control panel
16#8206		**Technology object Axis not yet enabled**	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis control panel.
16#8207		**Command rejected**	
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt". The other homing type can then be started.

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "position" value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "Distance" value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "Velocity" value; restart the command
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; restart the command
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	Axis reversal is activated at the HW limit switch, despite the fact that the hardware limit switches are disabled	• Activate the hardware limit switch using the tag <axis>.Config.PositionLimits_HW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
16#8405		**Invalid value at parameter "StopMode" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; enable the axis again
16#8406		**Simultaneous forward and backward jogging is not allowed**	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		**Switching the axis with Motion Control instruction "MC_Power" is only permitted after disabling the axis.**	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.

Configuration error

ErrorID	ErrorInfo	Description	Remedy
16#8600		**Parameter assignment of pulse generator (PTO is invalid**	
	16#000B	Address is invalid	Correct the configuration of the PTO (Pulse Train Output) and download it to the controller
16#8601		**Parameter assignment of the high-speed counter (HSC) is invalid**	
	16#000B	Address is invalid	Correct the configuration of the HSC (High Speed Counter) and download it to the controller
16#8602		**Invalid parameter assignment of "Enable output"**	
	16#000D	Address is invalid	Correct the configuration and download it to the controller
16#8603		**Invalid parameter assignment of "Ready input"**	
	16#000D	Address is invalid	Correct the configuration and download it to the controller
16#8604		**Invalid "Pulses per motor revolution" value**	
	16#000A	Value is less than or equal to zero	Correct the configuration and download it to the controller
16#8605		**Invalid "Distance per revolution" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#000A	Value is less than or equal to zero	
16#8606		**Invalid "Start/stop velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		**Invalid "Maximum velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#8608		**Invalid "Acceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#8609		**Invalid "Deceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#860A		**Invalid "Emergency stop deceleration" value**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#860B		**Value for position of the lower SW limit switch is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
	16#0007	The position value of the lower SW limit switch is greater than that of the upper SW limit switch	
16#860C		**Value for position of the upper SW limit switch is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#860D		**Invalid address of the lower HW limit switch**	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	
16#860E		**Invalid address of the upper HW limit switch**	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	
16#860F		**Invalid "home position offset" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8610		**Invalid "approach velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8611		**Invalid "Homing velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8612		**Invalid address of the reference point switch**	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	

ErrorID	ErrorInfo	Description	Remedy
16#8613		**During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured**	
	16#0001	-	Correct the configuration and download it to the controller

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		**Internal error**	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 7475)

Using multiple axes with the same PTO (Page 7462)

Tag of the axis technology object V1...3

Config tags V1...3

Config.General tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.General.PTO				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWORD	DW#16#00000000	-	-	-

<Axis name>.Config.General.HSC				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWORD	DW#16#00000000	-	-	-

<Axis name>.Config.General.LengthUnit ("Axis" technology object as of V2.0)				
The unit of measurement for the parameter selected in the configuration: • 1013 = "mm" • 1010 =: "m" • 1019 = "in" • 1018 = "ft" • 1005 = "°" (degrees) • -1 = "Pulse"				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Int	1013	R	-	X

See also

Tags of the positioning axis technology object as of V4 (Page 7496)

Config.DriveInterface tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.DriveInterface.EnableOutput				
Tags cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
-	-	-	-	-

<Axis name>.Config.DriveInterface.ReadyInput				
Tags cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
-	-	-	-	-

See also

Tags of the positioning axis technology object as of V4 (Page 7496)

Config.Mechanics tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.Mechanics.PulsesPerDriveRevolution				
Increments per motor revolution				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	L#1000	R	-	X

<Axis name>.Config.Mechanics.LeadScrew				
Distance per revolution (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	1.0E+001	R	-	X

<Axis name>.Config.Mechanics.InverseDirection				
Invert direction				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

See also

Tags of the positioning axis technology object as of V4 (Page 7496)

Config.DynamicLimits tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.DynamicLimits.MinVelocity				
Start/stop velocity of axis (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	1.0E+001	R	-	X

<Axis name>.Config.DynamicLimits.MaxVelocity				
Maximum velocity of axis (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	2.5E+002	R	-	X

See also

Tags of the positioning axis technology object as of V4 (Page 7496)

Config.DynamicDefaults tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	2	When axis is enabled
	5	The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable or active MC_Home command is started (Mode = 3).
	6	When a MC_MoveJog command is stopped
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.DynamicDefaults.Acceleration					
Acceleration of axis (specified in the configured dimension unit)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	4.8E+001	RW	5	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.DynamicDefaults.Deceleration					
Deceleration of axis (specified in the configured dimension unit)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	4.8E+001	RW	5, 6	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.DynamicDefaults.EmergencyDeceleration				
Emergency stop deceleration of axis (specified in the configured dimension unit)				
Data type	**Start value**	**Access**	**Effective**	**HMI**

<Axis name>.Config.DynamicDefaults.EmergencyDeceleration					
Real	1.2E+002	RW	2, 5, 6	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.DynamicDefaults.JerkActive ("Axis" technology object as of V2.0)				
TRUE = the jerk limit is activated				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	1, 5	X

<Axis name>.Config.DynamicDefaults.Jerk ("Axis" technology object as of V2.0)				
Jerk during axis acceleration and deceleration ramp (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	1.92E+002	RW	1, 5	X

See also

Tags of the positioning axis technology object as of V4 (Page 7496)

Config.PositionLimits_SW tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	4	Upon the next start of a Motion Control command after a standstill of the axis. The axis standstill can be checked with tag <Axis name>. StatusBits.Standstill.
	5	The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable or active MC_Home command is started (Mode = 3).
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.PositionLimits_SW.Active					
TRUE = The software limit switches are activated					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Bool	FALSE	RW	4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.PositionLimits_SW.MinPosition					
Position of lower software limit switch (specified in the configured unit of measurement)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	-1.0E+004	RW	4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.PositionLimits_SW.MaxPosition					
Position of upper software limit switch (specified in the configured unit of measurement)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	1.0E+004	RW	4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

See also

Tags of the positioning axis technology object as of V4 (Page 7496)

Config.PositionLimits_HW tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.

Effective	Specifies when a change to the tag takes effect.	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	3	After axis enable (the axis must have previously been at a standstill). The axis standstill can be checked with tag <Axis name>. StatusBits.Standstill.
	4	Upon the next start of a Motion Control command after a standstill of the axis. The axis standstill can be checked with tag <Axis name>. StatusBits.Standstill.
	5	The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable or active MC_Home command is started (Mode = 3).
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.PositionLimits_HW.Active				
TRUE = The hardware limit switches are active.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	3, 4: CPU Firmware V1.0	X
			1, 5, 6: CPU firmware as of V2.0	

<Axis name>.Config.PositionLimits_HW.MinSwitchedLevel				
TRUE = 24 V at CPU input corresponds to lower hardware limit switch approached FALSE = 0 V at CPU input corresponds to lower hardware limit switch approached				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.Config.PositionLimits_HW.MinFallingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	DW#16#00000000	-	-	-

<Axis name>.Config.PositionLimits_HW.MinRisingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	DW#16#00000000	-	-	-

<Axis name>.Config.PositionLimits_HW.MaxSwitchedLevel				
TRUE = 24 V at CPU input corresponds to upper hardware limit switch approached FALSE = 0 V at CPU input corresponds to upper hardware limit switch approached				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.Config.PositionLimits_HW.MaxFallingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	DW#16#00000000	-	-	-

<Axis name>.Config.PositionLimits_HW.MaxRisingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	DW#16#00000000	-	-	-

See also

Tags of the positioning axis technology object as of V4 (Page 7496)

Config.Homing tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	7	When a passive homing command is started
	8	When an active homing command is started
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.Homing.AutoReversal					
TRUE = Direction reversal at hardware limit switch enabled (active homing) FALSE = Direction reversal at hardware limit switch disabled (active homing)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Bool	TRUE	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.Direction					
TRUE = Positive approach direction to search for reference point switch and positive homing direction (active homing) FALSE = Negative approach direction to search for reference point switch and positive homing direction (active homing)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Bool	TRUE	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.SideActiveHoming ("Axis" technology object as of V2.0)				
TRUE = Homing on high end of the reference point switch (active homing) TRUE = Homing on lower end of the reference point switch (active homing)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	TRUE	RW	1, 8	X

<Axis name>.Config.Homing.SidePassiveHoming ("Axis" technology object as of V2.0)				
TRUE = Homing on high end of the reference point switch (passive homing) TRUE = Homing on lower end of the reference point switch (passive homing)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	TRUE	RW	1, 7	X

<Axis name>.Config.Homing.RisingEdge (as of technology object "Axis" V1.0)				
TRUE = Homing with negative signal edge of the reference point switch (active homing) FALSE = Homing with positive signal edge of the reference point switch (active homing) For information on the effect of the tag on passive homing, refer to the description in "Configuration - Homing".				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.Config.Homing.Offset					
Home position offset /specified in the configured unit of measurement (active homing)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	0.0	R RW	-	Technology object "Axis" V1.0	X
			1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.FastVelocity				
Approach velocity / specified in the configured unit of measurement (active homing)				
Data type	**Start value**	**Access**	**Effective**	**HMI**

<Axis name>.Config.Homing.FastVelocity					
Real	2.0E+002	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.SlowVelocity					
Homing velocity / specified in the configured unit of measurement (active homing)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	4.0E+001	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.FallingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	DW#16#00000000	-	-	-

<Axis name>.Config.Homing.RisingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	DW#16#00000000	-	-	

See also

Tags of the positioning axis technology object as of V4 (Page 7496)

MotionStatus tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.MotionStatus.Position				
Current position of the axis (specified in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	R	-	X

<Axis name>.MotionStatus.Velocity				
Current velocity of the axis (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	R	-	X

<Axis name>.MotionStatus.Distance				
Current distance to the target position of the axis (specified in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute" or "MC_MoveRelative" or of the axis command table.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	R	-	X

<Axis name>.MotionStatus.TargetPosition				
Target position of axis (specified in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute" or "MC_MoveRelative" or of the axis command table.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	R	-	X

See also

Motion status (Page 7460)

Tags of the positioning axis technology object as of V4 (Page 7496)

StatusBits tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.StatusBits.Activated				
TRUE = The axis is activated. It is connected to the assigned PTO (Pulse Train Output). The data of the technology data block will be updated cyclically.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Enable				
TRUE = The axis is enabled and ready to take on Motion Control commands.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.HomingDone				
TRUE = The axis is homed and is capable of executing absolute positioning commands. The axis does not have to be homed for relative positioning. The status is FALSE during active homing. The status remains TRUE during passive homing if the axis has already been homed.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Done				
TRUE = No Motion Control command is active on the axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Error				
TRUE = An error occurred in the axis technology object. Detailed information about the error is available in automatic mode in the "ErrorID" and "ErrorInfo" parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.StandStill				
TRUE = The axis is at a standstill.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.PositioningCommand				
TRUE = The axis is executing a positioning command.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.SpeedCommand				
TRUE = The axis is executing a travel command at predefined velocity.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Homing				
TRUE = The axis is executing a homing command of the "MC_Home" Motion Control instruction or axis command table.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.CommandTableActive				
TRUE = The axis is controlled by Motion Control instruction "MC_CommandTable".				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.ConstantVelocity				
TRUE = The axis travels at constant velocity.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Acceleration				
TRUE = The axis accelerates.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Deceleration				
TRUE = The axis decelerates (slows down).				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.ControlPanelActive				
TRUE = The "Manual control" mode has been enabled in the axis command table. The axis command table has control priority over the "Axis" technology object. The axis cannot be controlled from the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.DriveReady				
TRUE = The drive is ready.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.RestartRequired				
TRUE = Values were modified in the load memory. To download the values in the CPU RUN operating mode to the work memory, you need to restart the axis. Use the Motion Control instruction MC_Reset to do this.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

See also

Status and error bits (technology objects as of V4) (Page 7457)

Tags of the positioning axis technology object as of V4 (Page 7496)

ErrorBits tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.ErrorBits.SystemFault				
TRUE = Internal system error.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.ConfigFault				
TRUE = Incorrect configuration of axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.DriveFault				
TRUE = The drive has reported an error due to loss of its "Drive ready" signal.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.SwLimitMinReached				
TRUE = The lower software limit switch has been reached.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.SwLimitMinExceeded				
TRUE = The lower software limit switch has been exceeded.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.SwLimitMaxReached				
TRUE = The upper software limit switch has been reached.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.SwLimitMaxExceeded				
TRUE = The upper software limit switch has been exceeded.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.HwLimitMin				
TRUE = The lower hardware limit switch has been approached.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.HwLimitMax				
TRUE = The upper hardware limit switch has been approached.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.HwUsed				
TRUE = A second axis is using the same PTO (Pulse Train Output) and is enabled with "MC_Power".				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

See also

Status and error bits (technology objects as of V4) (Page 7457)

Tags of the positioning axis technology object as of V4 (Page 7496)

Internal tags V1...3

The "Internal" tags contain no user-relevant data; these tags cannot be accessed in the user program.

See also

Tags of the positioning axis technology object as of V4 (Page 7496)

ControlPanel tags V1...3

The "ControlPanel" tags contain no user-relevant data; these tags cannot be accessed in the user program.

See also

Tags of the positioning axis technology object as of V4 (Page 7496)

Update of the technology object tags

The status and error information of the axis indicated in the technology object tags is updated at each cycle control point.

The change in values of editable configuration tags does not take effect immediately. For information on the conditions under which a change takes effect, refer to the detailed description of the relevant tag.

Tags of the command table technology object V1...3

Config.Command.Command[1 ... 32] tags V1...3

Legend

Data type	Data type of the tag	
Start value	Start value of tag The start value can be overwritten by the configuration of the command table.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Command table>.Config.Command.Command[x].Type				
Command type • 0 = "Empty" command • 2 = "Hold" command • 5 = "Relative positioning" command • 6 = "Absolute positioning" command • 7 = "Velocity setpoint" command • 151 = "Wait" command				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Int	0	RW	-	X

<Command table>. Config.Command.Command[x].Position				
Command target position / travel path				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real		RW	-	X

<Command table>. Config.Command.Command[x].Velocity				
Command velocity				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Command table>. Config.Command.Command[x].Duration				
Command duration				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Command table>. Config.Command.Command[x].BufferMode				
Value for command "Next step" • 0 = "Complete command" • 1 = "Blend movement"				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Int	0	RW	-	X

<Command table>. Config.Command.Command[x].StepCode				
Command step code				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Word	0	RW	-	X

See also

Tags of the command table technology object as of V4 (Page 7530)

Using online and diagnostics functions 14

14.1 Displaying accessible devices

Accessible devices are all devices connected to an interface of the programming device / PC that are turned on. Devices that allow only restricted configuration using the currently installed products or that cannot be configured at all can also be displayed.

To speed up the display in large networks with many network devices, the PROFINET device names are displayed in a slightly adapted form for PROFINET devices. Unicode-specific characters in the PROFINET device name are converted into ASCII-compatible characters using the Punycode method (according to RFC 3492).

Displaying accessible devices in the project tree

To display accessible devices at an interface of the programming device / PC, follow these steps:

1. Open the "Online access" folder in the project tree.
2. Click on the arrow to the left of the interface to show all the objects arranged below the interface.
3. Double-click "Update accessible devices" below the interface.
   You can see the progress of the search in the status bar. If you have found the desired device before the search is completed, you can cancel the search. To do this, click on the cross to the right of the progress bar.
   All network devices that are accessible over the selected interface are displayed.
4. Optional: Select the interface in the project tree. Then open the overview window in the detail view to display more details for the respective devices. More details include, for example, the device type, the MAC address or the configured device name.
   The detailed display of the devices in the overview window can take some time in networks with many devices because each individual device must submit the details.

Displaying accessible devices in a table

To display accessible devices in an easy to read table, follow these steps:

1. Select the "Accessible devices" command in the "Online" menu.
   The "Accessible devices" dialog is displayed.
2. Select the type of interface from the "Type of the PG/PC interface" drop-down list.
3. Select the required interface of the programming device / PC from the "PG/PC interface" drop-down list.

4. Click the "Start Search" button.
   The accessible devices at the selected interface of the programming device/PC are displayed in the table. The connection line between PG/PC and the network is displayed as a solid line in the graphic. If no devices are accessible at an interface, the connection line is displayed as a dashed line.
5. To go to a device in the project tree, select the device from the list of accessible devices and click the "Show" button.
   The interface to which the selected device is connected is shown as selected in the project tree.

Displaying additional information about the devices in the project tree

To display additional information on the individual accessible devices in the project tree, follow these steps:

1. Click on the arrow to the left of one of the accessible devices in the project tree.
   All data available online, for example blocks and system data, is displayed for known devices. Devices that you cannot be edited directly at this point are grayed out. If additional editing options are available for a device with the shortcut menu, the device is shown in black text.

See also

Changing the device configuration online (Page 7588)

Default setting online connection data (Page 7592)

Overview window (Page 331)

14.2 Changing the device configuration online

You can assign parameters to some devices, preferably in small hardware setups, directly online. You do not need to create a project or have offline data to do this. In this way, you can quickly and easily change the device configuration. You do not need to compile the hardware configuration or perform a download. Depending on the device, either all changes become active immediately or they are written to the device only after confirmation.

Requirement

- The device must support online parameter assignment. You can learn whether or not your specific devices support this function in the device manual.
- The device must be connected to the PG/PC and available in the list of accessible devices.

Procedure

To change the device configuration online, follow these steps:

1. Show the accessible devices on the interface via which the device is connected. To learn how to show the accessible devices, see the chapter "Showing accessible devices (Page 7585)".
2. Expand the device to display the lower-level elements.
3. Double-click the "Parametrize device" item.
   A configuration page for the device opens in the work area.
4. Make all required settings.
   With some devices, the new settings take effect immediately.
5. Optionally, depending on the device: Click on the "Upload to device" button.
   The settings are transferred to the device.

14.3 Connecting devices online

14.3.1 General information about online mode

Online mode

In online mode, there is an online connection between your programming device / PC and one or more devices.

An online connection between the programming device/PC and the device is required, for example, for the following tasks:

- Testing user programs
- Displaying and changing the operating mode of the CPU
- Displaying and setting the date and time of day of the CPU
- Displaying module information
- Comparing blocks
- Hardware diagnostics

Before you can establish an online connection, the programming device/PC and the device must be physically or remotely connected. As an alternative, some devices support a simulation mode. In this case, a connection to the device is simulated via the PLCSIM virtual interface.

After establishing a connection, you can use the Online and Diagnostics view or the "Online tools" task card to access the data on the device. The current online status of a device is

indicated by an icon to the right of the device in the project tree. You will find the meaning of the individual status icons in the relevant tooltip.

Note

Some online functions depend on the scope of the installed software or whether a project is open.

Standby or hibernation of the programming device / PC

If the programming device / PC is changed to the standby or hibernation mode when there is an online connection, all online connections are terminated. When the programming device / PC wakes up from hibernation, the online connections are not automatically re-established.

Note that suddenly terminating an online connection can lead to loss of data or a connected device may interrupt program execution.

Performing an LED flash test

In many online dialogs you can perform an LED flash test, if the device connected online supports this feature. If you select the "Flash LED" check box, an LED flashes on the currently selected device. This feature is useful, for example, when you are not sure which device in the hardware configuration corresponds to the station currently selected in the software.

Read any additional information and learn about the possible limitations to the LED flash test in the respective device documentation.

See also

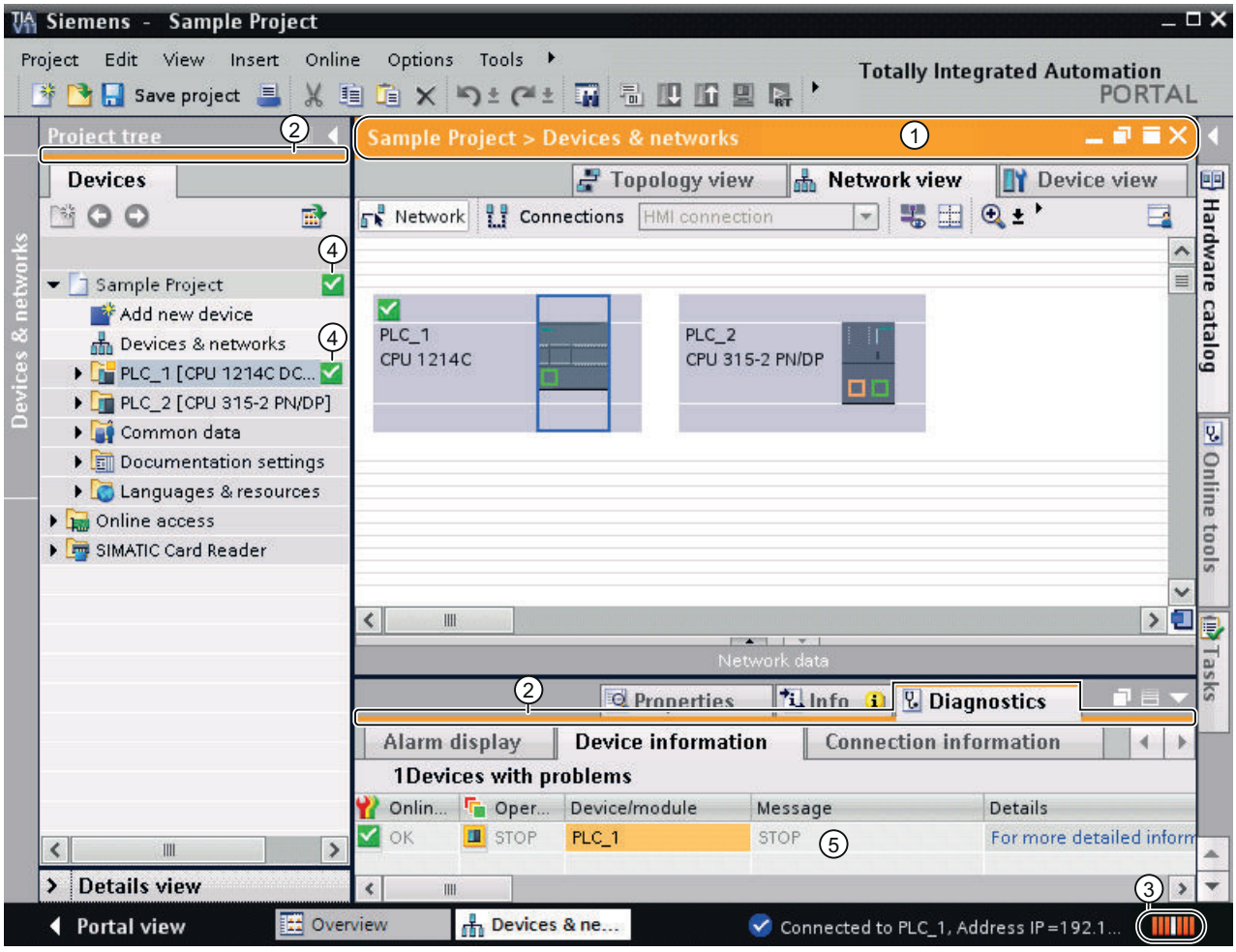
View in online mode (Page 7591)

Overview of the online and diagnostic settings (Page 305)

14.3.2 View in online mode

Online displays

After the online connection has been established successfully, the user interface changes. If a device is unavailable, this is indicated by a symbol. The following figure shows a device connected online and the corresponding user interface:



① The title bar of the active window gets an orange background as soon as at least one of the devices currently displayed in the editor has been successfully connected online. If one or more devices are unavailable, a symbol for a broken connection appears in the title bar of the editor.

② The title bars of inactive windows for the relevant station now have an orange line below them.

③ An orange, pulsing bar appears at the right-hand edge of the status bar. If the connection has been established but is functioning incorrectly, an icon for an interrupted connection is displayed instead of the bar. You will find more information on the error in "Diagnostics" in the Inspector window.

④ Operating mode symbols or diagnostics symbols for the stations connected online and their underlying objects are shown in the project tree. A comparison of the online and offline status is also made automatically. Differences between online and offline objects are also displayed in the form of symbols.

⑤ The "Diagnostics > Device information" area is brought to the foreground in the Inspector window.

Online connection abort

The online mode and its display are retained as long as at least one device is connected online. If the online connection of one or more devices breaks, the TIA Portal remains in online mode. The display of the TIA Portal changes to offline mode only when there is no longer an online connection for any device.

See also

General information about online mode (Page 7587)

Basics of project data comparison (Page 402)

14.3.3 Default setting online connection data

If you prefer to use a specific network interface of your programming device/PC to establish an online connection, you can preset this interface. The specified connection path is then already selected in the dialogs for online connection.

Procedure

To specify a default connection path, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are opened.
2. Select the "Online & Diagnostics" entry in the area navigation.
3. In the "Preset connection path for online access" area, specify the type of the PG/PC interface as well as the interface itself.
4. Select the "Use preset connection path for online access" check box.

See also

Establishing or changing an online connection (Page 7593)

Displaying accessible devices (Page 7585)

Downloading project data to a device (Page 396)

Online access (Page 7604)

Overview of the online and diagnostic settings (Page 305)

14.3.4 Establishing or changing an online connection

To use the online functions of a device, specify the connection path the first time you establish the connection. Once you have specified the connection path, it is saved and the online connection can be established immediately in the future.

Requirement

At least one PG/PC interface is installed and is physically connected to a device, for example, with an Ethernet cable. As an alternative, it is also possible to establish virtual online connections using PLCSIM.

Procedure

To establish or change an online connection, follow these steps:

1. Select one or more devices for the online connection in the project tree.
2. Select the "Go online" command in the "Online" menu.
   - If the device has already been connected online, the online connection is automatically established using the previously specified connection path.
   - If there was no previous connection, the "Go online" dialog opens.

   If you want to change a specified connection path, select the "Extended go online" command instead in the "Online" menu. The "Go online" dialog box is opened again in this case.
3. Select the connection path:
   - Select the type of interface from the "Type of the PG/PC interface" drop-down list.
   - Select the interface of your programming device/PC from the "PG/PC interface" drop-down list.
   - Select the interface or the subnet for the connection from the "Connection to interface/ subnet" drop-down list. Select a direct connection if no network node (e.g., switch) is connected in between. Select the matching subnet for the connection to the programming device or PC if the device can be accessed via a network node. Select the item "Try all interfaces" if the device has several interfaces and you do not know by which of the interfaces the device is connected to the programming device/PC.
     If you selected an MPI or PROFIBUS subnet, the bus parameters configured in the programming device/PC interface are applied at this point.
   - If the device is accessible via a gateway, select the gateway that connects the two subnets involved in the "1st gateway" drop-down list.
4. Click the "Start Search" button.
   Devices which can be reached by the set connection path are displayed in the "Compatible devices in target subnet". The connection line in the graphic on the left is displayed as solid.

5. Optional: If no devices can be accessed by the selected connection path, a dashed connecting line is displayed between the programming device/PC and the device. This is the case, for example, when the device is connected with the network by a NAT router. In this case, select the "Show all compatible devices" check box and click "Start Search" again to search the entire network. All devices that are compatible with the target system are displayed, even if the devices are not located in the same subnet as the programming device/PC.
6. Optional: You have two options to access the device for the load operation despite a different subnet:
   – Assign an address to the device in the same subnet as the programming device/PC. Manually enter an address in the "Address" column; the address must be located in the same subnet as the programming device/PC. This address is assigned to the device once for the load operation.
   – Assign an address to the programming device/PC in the same subnet as the device. An additional IP address in the same subnet as the device is temporarily assigned to the interface of the programming device/PC during the download. This happens automatically. Click "Go online" and confirm the next prompt.
7. Optional: Run a flash test. Select the "Flash LED" check box on the left in the graphic. You use the flash test to check that you have selected the correct device. The flash test is not supported by all devices.
8. Select the device in the "Compatible devices in target subnet" table, and confirm the selection with "Go online".
   The online connection to the selected target device is established.

Result

After the online connection has been established, the title bars of the editors change to orange. An orange progress bar is also shown in the title bar of an editor and in the status bar. In the project tree, status symbols show the difference between online and offline objects.

The set connection path is stored for future connection attempts.

Note
Undoing actions

You cannot undo actions once you have established an online connection.

See also

Overview of the online and diagnostic settings (Page 305)

Default setting online connection data (Page 7590)

Canceling an online connection (Page 7595)

Connecting online with several devices (Page 7595)

View in online mode (Page 7589)

Assigning a temporary IP address (Page 7610)

Influence of user rights (Page 349)

14.3.5 Canceling an online connection

Procedure

To terminate the existing online connection, follow these steps:

1. Select the device for which you want to disconnect the online connection in the project tree.
2. Select the "Go offline" command in the "Online" menu.
   The online connection is disconnected.

Note

Undoing actions

You cannot undo actions once you have disconnected an online connection.

See also

Establishing or changing an online connection (Page 7591)

14.3.6 Connecting online with several devices

You can establish an online connection to several devices at the same time without needing to select individual devices previously in the network view.

Requirement

- No device must be selected
- At least one PG/PC interface is installed and is physically connected to a device, for example with an Ethernet cable. As an alternative, it is also possible to establish a virtual online connection using PLCSIM or a remote connection.

Procedure

To establish an online connection to several devices at the same time, follow these steps:

1. Select the project in the project tree.
2. Select the "Go online" command in the "Online" menu.
   The "Select device for opening the online connection" dialog opens with a table of all available devices.

3. Select the devices to which you want to establish an online connection in the "Go online" column.
4. Click the "Go online" button.

Result

Without any further prompt for confirmation, a connection is established to all selected devices if a connection was already established to the selected devices at least once. If there was no previous online connection, the "Go online" dialog opens. In this case, first configure the online connection as described in the section "Go online and Go offline (Page 7591)".

See also

Establishing or changing an online connection (Page 7591)

Assigning a temporary IP address (Page 7610)

14.3.7 Disconnecting online connections of multiple devices

You can disconnect the online connections to multiple devices at one time without needing to select individual devices beforehand in the network view.

Requirement

- No device is selected.
- There is currently an online connection to at least one device.

Procedure

To terminate the online connections to multiple devices at one time, follow these steps:

1. Select the "Go offline" command in the "Online" menu.
   The "Select devices" dialog opens with a table of all available devices.
2. Select the device for which you want to terminate the online connection in the "Go offline" column.
3. Click the "Go offline" button.

Result

The online connection to the all the selected devices is terminated.

14.4 Creating a backup of an S7 CPU

14.4.1 Backup options for S7 CPUs

You will make a number of changes to your automation system over time, for example, add new devices, replace existing devices or adapt the user program. If these changes were to result in undesirable behavior, you can restore the automation plant to an earlier version. To continue working without interruptions after replacing individual devices, you can accept existing programs and values. The CPUs offer different options for backup and restoration of the hardware configuration and software.

Backup options

The table below provides an overview of the backup and restoration options of S7 CPUs:

	Snapshot of the monitored values	Upload from device (software)	Upload device as new station (hardware and software)	Download backup from online device
Application case	Restoring a specific status of a data block. The actual values of data blocks including time stamp are accepted in the program.	Download blocks on a CPU to the project.	Download of hardware configuration and software from a device to the project.	Create a complete backup of a CPU as restore point. The backup copy is consistent and cannot be changed or opened.
Requirement	The CPU has already been created in a project. The data blocks must be identical online and offline.	The CPU is created in the project.	The device is available in the hardware catalog of TIA Portal. Any necessary HSPs or GSD files are installed.	-
Possible in mode	RUN, STOP	RUN, STOP	RUN, STOP	STOP
Possible for F-CPUs	Yes	Yes	No	Yes

Backup contents

The table below shows which data you can download and back up with which options:

	Snapshot of the monitored values	Upload from device (software)	Upload device as new station (hardware and software)	Download backup from online device
Actual values of the data blocks	Snapshot is possible	Download is possible	Download is possible	Backup is possible
Software blocks	-	Download is possible	Download is possible	Backup is possible
PLC tags (tags and constants names)	-	Download possible for S7-1200 and S7-1500 CPUs	Download possible for S7-1200 and S7-1500 CPUs	Backup is possible
Hardware configuration	-	-	Download is possible	Backup is possible

	Snapshot of the monitored values	Upload from device (software)	Upload device as new station (hardware and software)	Download backup from online device
Monitoring tables (web server)	-	-	Download is not possible	Backup is possible
Local data, bit memories, timers, counters and process picture	Snapshot is not possible	Download is not possible	Download is not possible	Backup is possible
Archives and recipes (PLC)	-	-	-	Backup is possible
General data on the SIMATIC Memory Card, for example, help for program blocks or GSD files	-	-	-	Backup is possible

See also

Creating a backup of a device (Page 7598)

Creating a backup of a device (Page 7601)

General information on loading (Page 394)

Downloading project data to a device (Page 396)

Downloading project data to a memory card (Page 397)

Uploading project data from a device (Page 399)

Loading project data from a memory card (Page 401)

14.4.2 Backing up S7-300 and S7-400 CPUs

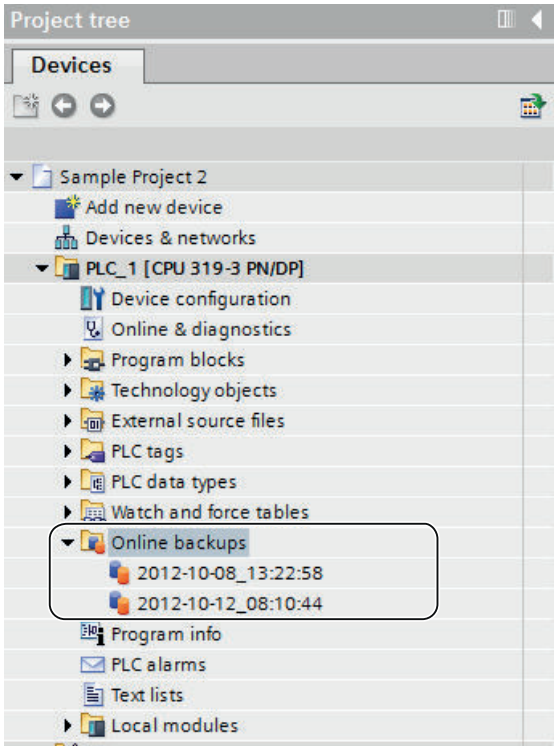
14.4.2.1 Creating a backup of a device

Backing up the software and hardware configuration of an S7-300/400 CPU

If you have already downloaded a configuration to an S7-300/400 CPU, it is advisable to make a backup. You may have modified the configuration and want to test the new configuration. Before you download the new configuration to the CPU, you can create a backup of the current device state and then restore the current configuration at a later date. The backup is performed with the current values of the CPU. In the case of S7-400 CPUs with fail-safe function, the initial values are backed up.

You can create as many backups as you want and store a variety of configurations for a CPU. The backups are named with the name of the CPU and the time and date of the backup. You can find the backup in the project tree under the CPU in the "Online backups" folder.

The following figure shows an S7-319 CPU for which two backups were created:



See also

Backup options for S7 CPUs (Page 7595)

14.4.2.2 Restoring the software and hardware configuration of a device

If you have backed up the configuration of a device at an earlier point in time, you can transfer the backup back to the device. The saved configuration is then restored on the device.

Requirement

You must have previously configure the device and stored a backup of the device in the project.

Procedure

To restore older software and hardware state on a device, follow these steps:

1. Open up the folder of the device in the project tree to display the lower-level objects.
2. Open the "Online backups" folder.
3. Select the backup you want to restore.

4. In the "Online" menu, select the "Download to device" command.
   - If you had previously established an online connection, the "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for downloading.
   - If you had not previously established an online connection, the "Extended download to device" dialog opens, and you must first select the interfaces via which you want to establish the online connection to the device.
     See also: Establishing and terminating an online connection
5. Check the messages in the "Load preview" dialog, and select the actions in the "Action" column, if necessary.

   Note

   Performing the proposed actions while the plant is in operation can cause serious bodily injury and property damage in the event of malfunctions or program errors.

6. As soon as loading becomes possible, the "Load" button is enabled.
7. Click the "Load" button.
   The backup is transferred to the device and device is restored. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
8. Click the "Finish" button.

14.4.2.3 Backing up a device configuration

You can back up the configuration of an S7-300/400 CPU in the TIA Portal. So you can download and test a new configuration to a device without any risk. If needed, you can restore the initial configuration of the CPU.

Requirement

- The CPU has already been created in the project.
- The CPU is online. If there is no online connection yet, an online connection is established during the backup.

Procedure

To create a backup of the current configuration of a CPU, follow these steps:

1. Select the CPU in the project tree.
2. Select the "Backup from online device" command in the "Online" menu.

Result:

A backup of the entire hardware configuration and software is created. The backup is stored in the project tree in the "Name of the CPU > Online backups" folder. The backup is assigned the name of the CPU with the time and date of the backup. You can rename the backup, but you cannot make any changes to the contents of the backup.

14.4.3 Backing up S7-1200 and S7-1500 CPUs

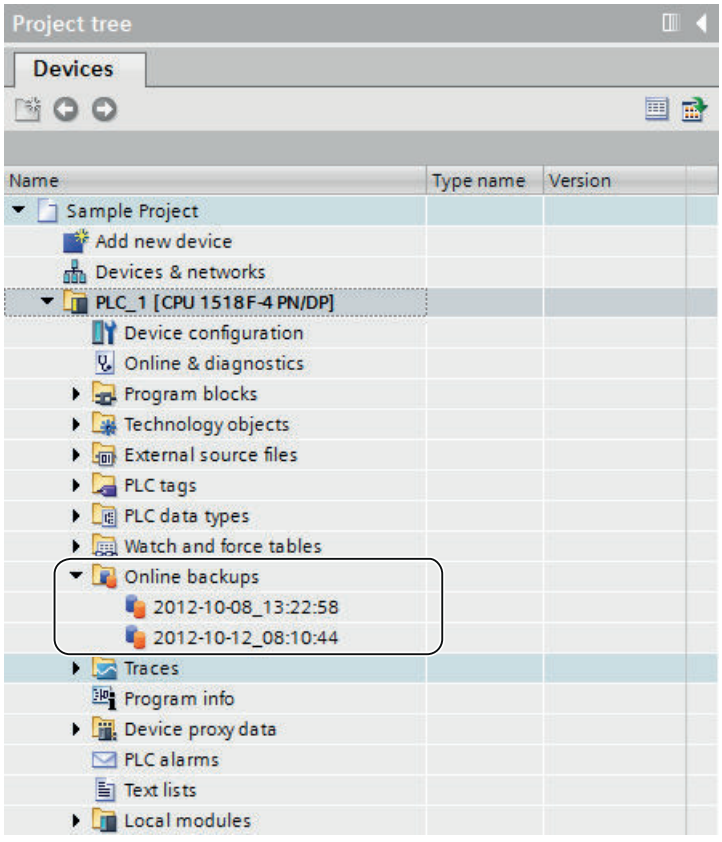
14.4.3.1 Creating a backup of a device

Backing up the software and hardware configuration of a CPU

If you have already downloaded a configuration to an S7-1200/S7-1500 CPU, it may be useful to make a backup. You may have modified the configuration and want to test the new configuration. Before you download the new configuration to the CPU, create a backup of the current device version. You can restore the current configuration at a later time.

You can create as many backups as you want and store a variety of configurations for a CPU. The backups are named with the name of the CPU and the time and date of the backup. You can find the backup in the project tree under the CPU in the "Online backups" folder.

The following figure shows an S7-1500 CPU for which two backups have been created:



Scope of the backup

The backup includes all data that are needed to restore a particular configuration version of a CPU. The following data is backed up, for example:

- Contents of the memory card
- Retentive memory areas of data blocks, counters, and bit memory, for example
- Other retentive memory contents, such as IP address parameters

The backup is performed with the current values of the CPU. Entries in the diagnostic buffer are not included in the backup. The current time of day is not backed up for an S7-1500 CPU.

See also

General information on loading (Page 394)

Backup options for S7 CPUs (Page 7595)

14.4.3.2 Backing up a device configuration

You can back up the configuration of a CPU in the TIA Portal. So you can download and test a new configuration to a device without any risk. If needed, you can restore the initial configuration of the CPU.

The CPU goes to STOP while a backup is being created. If an access level is configured for the CPU, you need the password for read access to the CPU.

Requirement

- The CPU has already been created in the project.
- The device is connected to the programming device/PC directly via the Industrial Ethernet interface of the CPU.
- The CPU is online. If there is no online connection yet, an online connection is established during the backup.

Procedure

To create a backup of the current configuration of a CPU, follow these steps:

1. Select the CPU in the project tree.
2. Select the "Backup from online device" command in the "Online" menu.
   If necessary, you must enter the password for read access to the CPU and confirm that the CPU should enter "STOP" mode.

Result:

A backup of the entire hardware configuration and software is created. The backup is stored in the project tree in the "Name of the CPU > Online backups" folder. The backup is assigned the name of the CPU with the time and date of the backup. You can rename the backup, but you cannot make any changes to the contents of the backup.

An entry is created for each backup operation in the diagnostic buffer of the CPU.

See also

Restoring the configuration of a device (Page 7603)

14.4.3.3 Restoring the configuration of a device

If you have backed up the configuration of a device at an earlier point in time, you can transfer the backup back to the device. The saved configuration is then restored on the device.

The CPU goes to STOP while a backup is being downloaded to the CPU. If an access level is configured for the CPU, you need the password for write access to the CPU.

Warning

Download backups with unknown content

If you activate the recommended actions during download during plant operation, you can cause serious damages or injuries in case of malfunctions or program errors.

Make sure that the backup contents do not include a configuration which results in unexpected plant behavior.

Requirement

- You must have previously configure the device and stored a backup of the device in the project.
- The device is connected to the programming device/PC directly via the Industrial Ethernet interface of the CPU.

Procedure

To restore older software and hardware state on a device, follow these steps:

1. Open up the folder of the device in the project tree to display the lower-level objects.
2. Open the "Online backups" folder.
3. Select the backup you want to restore.
4. In the "Online" menu, select the "Download to device" command.
   - If you had previously established an online connection, the "Load preview" dialog opens. This dialog displays alarms and recommends actions needed for the loading operation.
   - If you had not previously established an online connection, the "Extended download to device" dialog opens, and you must first select the interfaces via which you want to establish the online connection to the device.
     See also: Auto-Hotspot
5. Check the alarms in the "Load preview" dialog, and select the actions in the "Action" column, if necessary.
6. As soon as downloading becomes possible, the "Load" button is enabled.

7. Click the "Load" button.
   The backup is transferred to the device and device is restored. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
8. Click the "Finish" button.
   If necessary, you must enter the password for read access to the CPU and confirm that the CPU should enter "STOP" mode.
   The contents of the backup are restored on the CPU. The CPU is then restarted.

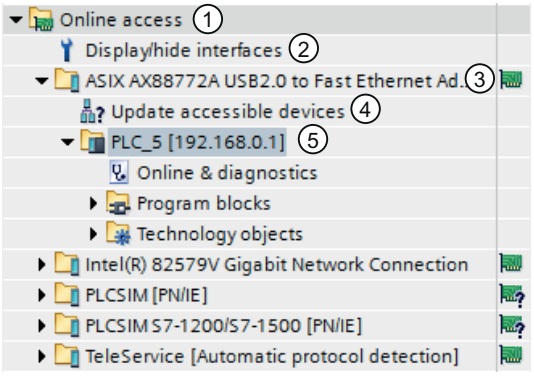
14.5 Configuring the PG/PC interface

14.5.1 Online access

Online access of the project

In the "Online access" folder of the project tree, you will find all active interfaces of your programming device/PC. Each interface icon provides you with information on the status of the interface. You can also display the accessible devices and display and edit the properties of an interface using the shortcut menu.

The following figure shows the "Online access" folder in the project tree:



① "Online access" folder in the project tree
All interfaces installed in the programming device/PC are displayed in the "Online access" folder.

② Show/hide interfaces
Use the function "Show/hide interfaces" to show or hide individual interfaces.

③ Status display for the interfaces
The current status of an interface is indicated by an icon to the right of the name. You can see the meaning of the icon in the tooltip.

④ Updating the list of accessible devices.
This function is available for each hardware interface of the programming device/PC. Software interfaces, such as a remote connection, do not offer this function.

⑤ Devices connected via the respective interface with the programming device/PC
The type of the respective device and its status are displayed by the preceding icon.

Displaying or updating accessible devices

You have the following options if you want to display all devices accessible online on your programming device/PC:

- Display of the accessible devices on a single interface of the programming device / PC in the project tree. In the project tree, you can also display additional information about the individual accessible devices.
- Display of the accessible devices of all interfaces in a list.

See also: Displaying accessible devices

Overview of icons for accessible devices

The accessible devices are identified with an icon according to their type and status. The following is an overview of all icons and their meaning.

	Icon for unidentified modules This icon is displayed whenever the identification of a module is not yet complete or when the identification of a module was not successful, for example, because the required online data could not be read.
	Icon for the following device types: • PLCs • SIMOCODE pro devices • IE/PB links • CPs of PC systems • SCALANCE head modules • S7-300 and S7-400 CPs • PROFINET IO devices and PROFINET CPs • SCALANCE modules and gateways that could not be identified
	PROFINET IO devices, encoders, switchgear, sensors and identification systems that were replaced by similar devices because these could not be identified
	Icon for the following device types: • HMI devices • PROFINET IO devices of the HMI type if these could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the drive type that could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the type development kit and network components that could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the type TeleService adapter that could not be identified and were therefore replaced by a similar device

See also

Displaying and modifying interface properties (Page 7607)

14.5.2 Basics of assigning parameters for the PG/PC interface

Options for connecting to target systems

If the devices of the project are connected via different subnets, you assign a suitable network access to each PG/PC interface to be able to establish online connections to the target systems. The following interfaces are automatically supported:

- MPI
- PROFIBUS
- Industrial Ethernet (ISO and TCP/IP)

You can make various settings for the interfaces. The following sections explain the parameter settings you can make.

Note

Note that changes to interface parameters have a direct influence on the operating system and the programming device / PC. Remember that some parameter settings can only be changed if you have adequate user rights.

See also

Setting parameters for the Industrial Ethernet interface (Page 7608)

Setting parameters for the MPI and PROFIBUS interfaces (Page 7612)

14.5.3 Showing or hiding interfaces

To improve clarity, it is possible to hide individual interfaces of the PG/PC in the project tree and to show them again when necessary.

Procedure

To show or hide individual interfaces, follow the steps below:

1. Open the "Online access" folder in the project tree.
2. Double click on the "Display/hide interfaces" icon.
   The "Display/hide interfaces" dialog opens.
3. Enable or disable the required interfaces in the "Display in the project tree" column.
4. Click the "Apply" button.
   The changes are applied. The view of the interfaces in the "Online access" folder is refreshed.

14.5.4 Displaying and modifying interface properties

Introduction

For each interface, you can display and, in some cases, modify properties, for example the network type, address, and status.

Procedure

To open the properties, follow these steps:

1. Right-click on the required interface below "Online access" in the project tree.
2. Select the "Properties" command from the shortcut menu.
   A dialog containing the properties of the interface opens. On the left of the dialog, you will see the area navigation. You can view the current parameter settings in the individual entries in the area navigation and, if necessary, change them.

14.5.5 Adding interfaces

You have the option of installing additional interfaces after installation of the TIA Portal.

Procedure

To install an interface at a later time and add it to the TIA Portal, follow these steps:

1. Install or update the drivers in the operating system once you have installed the interface hardware.
2. Close the TIA Portal if it is still open.
3. Open the Windows control panel.
4. Open the entry "Setting the PG/PC Interface" in the Control Panel.
   The "Setting the PG/PC Interface" dialog opens.
5. Make any necessary changes to the interface configuration and confirm them with "OK".
   You have to click "OK", even if you have not made any changes.
6. Restart the TIA Portal.

Result

The newly installed interface is now displayed in the project tree under the "Online access" folder.

14.5.6 Setting parameters for the Ethernet interface

14.5.6.1 Setting parameters for the Industrial Ethernet interface

Options in the parameter settings for the Industrial Ethernet interface

When setting parameters for the Industrial Ethernet interface, you have the following options:

- Parameters dependent on the operating system
  The Industrial Ethernet interface has parameters that are set in the operating system and are valid for all connected devices. These parameter settings are only displayed here, they can, however, be changed in the network settings of the operating system.
- Parameters that can be set in the software

Note

Note that changes to interface parameters have a direct influence on the operating system and the programming device / PC. Remember that some parameter settings can only be changed if you have adequate user rights.

Parameters for the Industrial Ethernet interface

The following table contains an overview of the parameters of the Industrial Ethernet interface that are set by the operating system and can be changed by the user.

Parameter settings that cannot be changed	Parameters that can be set
MAC address	Fast acknowledge at the IE-PG access and for TCP/IP
DHCP server activated/deactivated	Timeout at the IE-PG access and for TCP/IP
APIPA activated/deactivated	LLDP
IP address	Additional, dynamic IP addresses for the network adapter
Subnet mask	-
DNS addresses	-
DHCP addresses	-

See also

Basics of assigning parameters for the PG/PC interface (Page 7604)

Displaying operating system parameters (Page 7609)

Connecting the PG/PC interface to a subnet (Page 7609)

Setting parameters for the Ethernet interface (Page 7610)

Assigning a temporary IP address (Page 7610)

Managing temporary IP addresses (Page 7611)

Influence of user rights (Page 349)

14.5.6.2 Displaying operating system parameters

The Ethernet interface is part of the operating system. All parameters of the network adapter can therefore be adapted in the network settings of the operating system.

You can display the following parameters in the software:

- Physical address of the network adapter
- Assignment of the IP address by a DHCP server activated or deactivated
- Assignment of a private IP address by the operating system activated or deactivated
- Current static IP address
- Assigned subnet mask
- DNS addresses
- DHCP addresses

If you want to modify the parameter settings, please refer to the documentation of the operating system or the network adapter.

Displaying current parameters of the Ethernet interface

To display the current parameters of the Ethernet interface, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > Industrial Ethernet" in the area navigation.

See also

Setting parameters for the Ethernet interface (Page 7610)

14.5.6.3 Connecting the PG/PC interface to a subnet

If you have created several subnets, you can specify the subnet to which the Ethernet interface is connected.

Procedure

To select the subnet to which the Ethernet interface is connected, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Go to "General > Assignment" and select the subnet to which you want to connect the Ethernet interface of the programming device / PC in the "Connection to subnet" drop-down list.
4. Close the dialog with "OK".

14.5.6.4 Setting parameters for the Ethernet interface

You can adapt some parameter settings relating to the network protocol directly in the software.

Requirement

You must have adequate user rights.

See also: Influence of user rights (Page 349).

Procedure

To change parameter settings relating to the network protocol, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > IE-PG access" to adapt the protocol settings relevant to network management.
   - Select the "Fast acknowledge" check box to achieve faster reaction times with smaller network packets.
   - From the "Timeout" drop-down list, select the maximum time that can elapse before a network node is detected.
4. To activate the LLDP protocol and discover the network topology more accurately, set the "LLDP active" check box in "Configurations > LLDP".
5. Select "Configurations > TCP/IP" to adapt the TCP/IP protocol for network traffic during runtime.
   - Select the "Fast acknowledge" check box to achieve faster reaction times with smaller network packets.
   - From the "Timeout" drop-down list, select the maximum time that can elapse before there is a timeout during communication with a network node.

See also

Influence of user rights (Page 349)

Displaying operating system parameters (Page 7607)

14.5.6.5 Assigning a temporary IP address

Adding a dynamic IP address

If the IP address of a device is located in a different subnet from the IP address of the network adapter, you will first need to assign an additional IP address with the same subnet address as the device. Only then is communication between the device and the programming device / PC possible.

The assignment of an additional temporary IP address is also proposed automatically if you want to perform an online action and the current IP address of the programming device/PC is not yet in the correct subnet.

A temporarily assigned IP address remains valid until the next time the programming device/PC is restarted or until you delete it manually.

Note

You require adequate permissions to be able to assign a temporary IP address.

See also: Influence of user rights (Page 349)

See also

Managing temporary IP addresses (Page 7611)

14.5.6.6 Managing temporary IP addresses

If the IP address of a device is located in a different subnet from the current permanently assigned IP address of the network adapter, a suitable IP address from the subnet of the device is temporarily assigned to the network adapter.

You can display all temporarily assigned addresses and delete them. Note that IP addresses that you manually assigned in the operating system are not displayed in the TIA Portal.

Requirement

To delete, you require adequate permissions.

Procedure

To display and delete temporarily assigned addresses, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > IE-PG access".
   A table with the assigned IP addresses is displayed.
4. Click the "Delete project-specific IP addresses" button to delete all the IP addresses at one time.

See also

Influence of user rights (Page 349)

14.5.6.7 Resetting the TCP/IP configuration

If you have changed the TCP/IP protocol settings, you can reset them to the defaults.

Procedure

To restore the TCP/IP configuration to the default settings, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > TCP/IP".
4. Click the "Standard" button to reset all the settings.

14.5.7 Setting parameters for the MPI and PROFIBUS interfaces

14.5.7.1 Setting parameters for the MPI and PROFIBUS interfaces

Possible parameter settings for the MPI and PROFIBUS interfaces

The following parameter settings can be made for the MPI and PROFIBUS interfaces:

- Automatic configuration: You can use automatic detection functions to find out whether a device is connected to the PG/PC interface over PROFIBUS or MPI.
- Selecting a default configuration for PROFIBUS or MPI that can be adapted later.

Device- and network-related settings for MPI and PROFIBUS

You can set device- and network-related parameters for MPI and PROFIBUS interfaces. Device-related parameters are local settings for the interface. Network-related parameters, on the other hand, must match up on all devices.

MPI interface parameters you can modify

You can adapt the following default parameters for the MPI interface:

Device-related parameters	Network-related parameters
Is only master	Highest address
Own address	Transmission rate
Timeout	

PROFIBUS interface parameters you can modify

You can adapt the following default parameters for the PROFIBUS interface:

Device-related parameters	Network-related parameters
Is only master	Highest address
Own address	Transmission rate
Timeout	Profile

Device-related parameters	Network-related parameters
	Bus parameters
	Number of masters on bus
	Number of slaves on bus

Note

Behavior when a subnet is assigned

If you permanently assign a subnet from the project to the interface, the subnet settings of the project take precedence. In this case, the settings for the respective interface cannot be changed.

See also

Basics of assigning parameters for the PG/PC interface (Page 7604)

14.5.7.2 Setting MPI or PROFIBUS interface parameters automatically

Setting up automatic bus parameter detection

If you select an interface with automatic detection of the bus parameters (for example CP 5611 (Auto)), you can connect the programming device or PC to MPI or PROFIBUS without needing to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute.

Requirement

- Masters that distribute bus parameters cyclically are connected to the bus.
- In PROFIBUS networks, the cyclic distribution of bus parameters is enabled.
- The interface is not assigned to a subnet. If you assign a subnet in the properties of the interface, the settings for the subnet in the project take precedence. In this case, the settings for the automatic interface configuration cannot be changed.

Procedure

To enable automatic bus parameter detection, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Go to "General > Configurations > Active configuration" and select the setting "Automatic protocol detection".
4. Go to "Configurations > Auto configuration > Local settings" and select the address of the PG/PC interface in the "Own address" drop-down list.
5. If you then want to display the current bus settings, click the "Network detection" button.

See also

Setting parameters for the MPI interface (Page 7614)

Setting parameters for the PROFIBUS interface (Page 7616)

14.5.7.3 Setting parameters for the MPI interface

Changing the parameter settings of the MPI interface

The network-related parameters and bus parameters for the MPI network can be adapted. You should first select a default setting and then adapt this to the specific situation.

Requirement

The interface is not assigned to a subnet. If you assign a subnet in the properties of the interface, the settings for the subnet in the project take precedence. In this case, the interface configuration cannot be changed.

Setting defaults for the MPI interface

To adapt the parameters of the MPI interface, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Go to "General > Assignment" and select the subnet with which you want to connect the interface in the "Connection to subnet" drop-down list.
4. Under "General > Configuration", select a default for the device and network-related parameters. The defaults are suitable for most configurations. Select one of the following settings:
   - Automatic protocol detection
     You can connect the programming device to MPI or PROFIBUS without having to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute. Prerequisite for the automatic detection is a connection to the bus master, which distributes the bus parameters cyclically. With PROFIBUS subnets, cyclic distribution of bus parameters may not be deactivated (default PROFIBUS network setting).
   - MPI
     The "MPI" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.
   - PROFIBUS
     The "PROFIBUS" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.

Changing the default parameter settings

To adapt the default settings to your requirements, change the parameter setting where necessary in "Configurations > MPI".

You can set the following device-related parameters:

- Is only master
  An additional verification function to prevent bus disruptions when connecting the PG/PC to the network is disabled because the programming device or PC is the only master on the bus.
  - Do not enable this option unless you have only connected slaves to your programming device or PC.
  - If the "Is only master" check box is enabled, it is not possible to identify the directly connected device in the "Accessible devices" window.
- Own address
  This setting relates to the programming device or PC on which you call up the parameter settings of the interface. Set the local device address of your programming device or PC here.
  - This address must be unique throughout the network.
  - The programming device or PC is addressed using this address in the MPI network.
- Check
  This enables an additional safety function to prevent bus disruptions when connecting the PG/PC to the network. The driver checks whether the local address is already being used by another station. Active as well as passive stations are taken into consideration in this case. The driver monitors this on the PROFIBUS. The connection of the PG/PC to the network will take longer with the automatic check. To use the check, the driver must support the function. Furthermore, the "Is only master" option must not be selected.
- Timeout
  Set a higher timeout value if, for example, you have problems with long response times on the network.

You can set the following network-related parameters:

- Highest address:
  Select the configured highest device address. Make sure that the same highest device address is set for all devices of a PROFIBUS or MPI network.
- Transfer rate:
  Here, you select the transmission speed to be used on the MPI network.

See also

Setting MPI or PROFIBUS interface parameters automatically (Page 7611)

14.5.7.4 Setting parameters for the PROFIBUS interface

Changing the parameter settings of the PROFIBUS interface

The network-related parameters and bus parameters for the PROFIBUS network can be adapted more precisely. You should first select a default setting and then adapt this to the specific situation.

Requirement

The interface is not assigned to a subnet. If you assign a subnet in the properties of the interface, the settings for the subnet in the project take precedence. In this case, the interface configuration cannot be changed.

Setting defaults for the PROFIBUS interface

To adapt the parameters of the PROFIBUS interface, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Go to "General > Assignment" and select the subnet with which you want to connect the interface in the "Connection to subnet" drop-down list.
4. Under "General > Configuration", select a default for the device and network-related parameters. The defaults are suitable for most configurations. Select one of the following settings:
   - Automatic protocol detection
     You can connect the programming device to MPI or PROFIBUS without having to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute. Prerequisite for the automatic detection is a connection to the bus master, which distributes the bus parameters cyclically. With PROFIBUS subnets, cyclic distribution of bus parameters may not be deactivated (default PROFIBUS network setting).
   - MPI
     The "MPI" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.
   - PROFIBUS
     The "PROFIBUS" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.

Changing the default parameter settings

To adapt the default settings to your requirements, change the parameter setting where necessary in "Configurations > PROFIBUS".

You can set the following device-related parameters:

- Is only master
  An additional verification function to prevent bus disruptions when connecting the PG/PC to the network is disabled because the programming device or PC is the only master on the bus.
  - Do not enable this option unless you have only connected slaves to your programming device or PC.
  - If the "Is only master" check box is enabled, it is not possible to identify the directly connected device in the "Accessible devices" window.
- Own address
  This setting relates to the programming device or PC on which you call up the parameter settings of the interface. Set the local device address of your programming device or PC here.
  - This address must be unique throughout the network.
  - The programming device or PC is addressed using this address in the PROFIBUS network.
- Check
  This enables an additional safety function to prevent bus disruptions when connecting the PG/PC to the network. The driver checks whether the local address is already being used by another station. Active as well as passive stations are taken into consideration in this case. The driver monitors this on the PROFIBUS. The connection of the PG/PC to the network will take longer with the automatic check. To use the check, the driver must support the function. Furthermore, the "Is only master" option must not be selected.
- Timeout
  Set a higher timeout value if, for example, you have problems with long response times on the network.

You can set the following network-related parameters:

- Highest address:
  Select the configured highest device address. Make sure that the same highest station address is set for all devices of a PROFIBUS network.
- Transfer rate:
  Here, you select the transmission speed to be used on the PROFIBUS network.
- Profile:
  You have a choice of four alternatives for the PROFIBUS settings. "DP", "Standard" and "Universal (DP/FMS)" are predefined settings that you cannot change. If you select "User-defined", you can adapt the bus parameters yourself.
  - If you have selected "User-defined", go to "Configurations > PROFIBUS > Bus parameters" in area navigation.
  - If you have selected one of the defaults (DP, Standard or Universal (DP/FMS)), you should select the "Include" check box in "Configurations > PROFIBUS > Bus parameters > Additional parameters". You can then set the number of masters and slaves on the bus. This allows a more precise calculation of the bus parameters and potential bus disruptions can be prevented. The option cannot be selected with a user-defined profile.

See also

Overview of the bus parameters for PROFIBUS (Page 7618)

Setting MPI or PROFIBUS interface parameters automatically (Page 7611)

14.5.7.5 Overview of the bus parameters for PROFIBUS

Introduction

The PROFIBUS subnet will only function problem-free if the parameters for the bus profile are matched to one another. You should therefore only change the default values if you are familiar with how to configure the bus profile for PROFIBUS.

It may be possible for the bus parameters to be adjusted depending on the bus profile. The offline values of the bus parameters are always shown even if you are online and linked to the target system.

The displayed parameters are valid for the entire PROFIBUS subnet.

Meaning of the individual parameters

- Tslot: Wait-to-receive time (slot time)
  The wait-to-receive time (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner.
- Max. Tsdr: Maximum protocol processing time (max. station delay responder)
  The maximum protocol processing time defines the time after which the responding device must have processed the protocol.
- Min. Tsdr: Minimum protocol processing time (min. station delay responder)
  The minimum protocol processing time specifies the minimum time required by the responding device to process the protocol.
- Tset: Trigger time (setup time)
  The trigger time is the time that may lapse between the reception of a data frame frame and the reaction to it.
- Tqui: Quiet time for modulator
  The quiet time for modulator specifies the time required to change from sending to receiving.
- GAP factor: GAP update factor (GAP factor)
  The GAP factor specifies the number of token rotations before a new device is included in the token ring.
- Retry limit: Maximum number of repeated call attempts (retry limit)
  This parameter defines the maximum number of attempts made to reach a device.
- Trdy: Ready time
  The ready time is the time for an acknowledgment or response.
- Tid1: Idle time 1
  Idle time 1 specifies the delay time after receiving a response.
- Tid2: Idle time 2
  Idle time 2 specifies the delay time after sending a call without a response.

- Ttr: Target rotation time
  The target rotation time is the maximum time made available for a token rotation. During this time, all active devices (masters) receive the token once. The difference between the desired token round-trip time and the actual token round-trip time decides how much time is left for masters to send data frames to the slaves.
  As the minimum target rotation time (Ttr), select a value = 5000 times the HSA (Highest Station Address).
- Watchdog: Watchdog
  The watchdog time specifies the time after which a device must be addressed.
  As the minimum watchdog time, select a value = 6250 times the HSA.

Note

If you want to create a user-defined bus profile, please note that the minimum target rotation time (Ttr) should be 5000 times the HSA (highest PROFIBUS address). The minimum watchdog time should also be 6250 times the HSA.

See also

Setting parameters for the PROFIBUS interface (Page 7614)

14.5.7.6 Resetting the MPI or PROFIBUS configuration

If you have changed the MPI or PROFIBUS protocol settings, you can reset them to the defaults.

Requirement

The interface is not assigned to a subnet. If you assign a subnet in the properties of the interface, the settings for the subnet in the project take precedence. In this case, the interface configuration cannot be reset to the default values.

Procedure

To restore the MPI or PROFIBUS configuration to the default settings, follow these steps:

1. Select the MPI/PROFIBUS interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > MPI" or "Configurations > PROFIBUS", depending on the interface properties you want to reset.
4. Click the "Standard" button to reset all the settings.

14.6 Using the trace and logic analyzer function

Preface

Purpose of the documentation

The diagnostics options available with the trace and logic analyzer function are described in this documentation. Depending on the device used, the recording options can vary.

Required basic knowledge

In order to understand this documentation, the following knowledge is required:

- General knowledge in the field of automation
- Knowledge about the use of Windows-based computers
- S7-1200/1500 CPUs
  - Knowledge of the SIMATIC industrial automation system
  - Knowledge of working with STEP 7
- SINAMICS G120
  - Knowledge of working with the drive

Validity of the documentation

This documentation applies to all products of the S7-1200, S7-1500 and SINAMICS G120 product family as of TIA Portal V13 SP1.

Conventions

This documentation contains pictures of the devices described. The pictures may differ slightly from the devices supplied.

Please also observe notes marked as follows:

Note

A note contains important information on the product described in the documentation, on the handling of the product and on the section of the documentation to which particular attention should be paid.

Further support

- The range of technical documentation for the individual SIMATIC products and automation systems can be found on the Internet (http://www.siemens.com/simatic-tech-doku-portal).
- The online catalog and the online ordering system is available on the Internet (https://mall.industry.siemens.com).

14.6.1 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. You can find more information about industrial security on the Internet (http://www.siemens.com/industrialsecurity).

To stay informed about product updates as they occur, sign up for a product-specific newsletter. You can find more information on the Internet (http://support.automation.siemens.com).

14.6.1 Description

14.6.1.1 Supported hardware

If a device supports the trace and logic analyzer function, "Traces" is offered for selection in the project tree below the device.

The following devices (Page 7648) support the trace and logic analyzer function:

- SIMATIC S7-1200 CPUs (as of firmware version V4.0)
- SIMATIC S7-1500 CPUs
- SIMATIC S7-1500 software controller
- SINAMICS G120

14.6.1.2 Recording of measured values with the trace function

Introduction

The trace and logic analyzer function can be called in the project navigator (Page 7625) by double-clicking an entry in the "Traces" system folder.

You record device tags and evaluate the recordings with the trace and logic analyzer function. Tags are, for example, drive parameters or system and user tags of a CPU. The maximum

recording duration is limited by the memory size. How much memory is available for the recording depends on the hardware used.

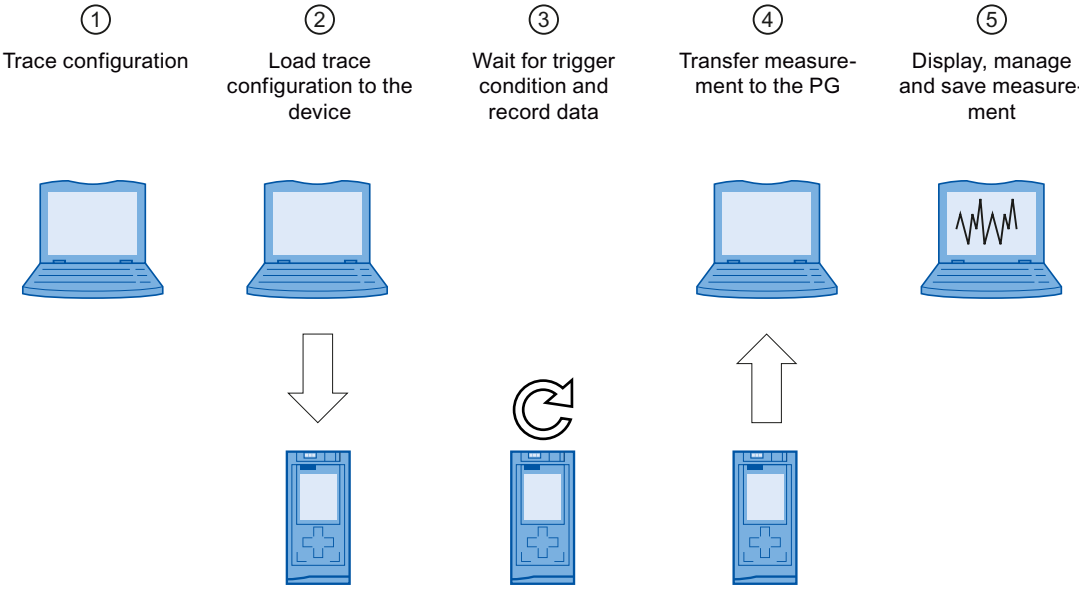
The recordings are saved on the device and, when required, can be read out with the engineering system (ES) and saved permanently. The trace and logic analyzer function is therefore suitable for monitoring highly dynamic processes. The recorded values are overwritten when the recording is activated again.

Active recordings from the axis control panel are displayed in the "Traces" system folder as installed traces. Recordings can be added to the measurements in the curve diagram of the axis control panel or the PID via a shortcut menu command.

Depending on the device (Page 7648) used, the recording options can vary.

A quick start (Page 7633) for working with the trace and logic analyzer function can be found in the Operation section.

The following figure shows the method of operation of the "Traces":



① Trace configuration on the programming device (PG) in the TIA Portal

You can specify the signals to be recorded, the duration of the recording and the trigger condition in the trace configuration. The trace configuration depends on the device and is described at the respective device (Page 7648).

② Transferring the trace configuration from the PG to the device

You can transfer the complete trace configuration (Page 7640) to the device when an online connection is established.

③ Waiting for the recording

If the installed trace configuration is activated (Page 7640), then the recording is performed independently of the PG. The recording is started as soon as the trigger condition is satisfied.

④ Transferring the measurement from the device to the PG

The saving of the measurement in the project (Page 7642) stores the measurement in the opened project of the TIA Portal. The measurement can be saved at any time after completing the recording, irrespective of the time of the measurement.

⑤ Evaluating, managing and saving the measurement

Numerous options are available for the evaluation of the measurement in the curve diagram and in the signal table (Page 7641). Various display types are possible, for example, a bit representation for binary signals.
Measurements can also be exported and imported as a file.
With the saving of the project (Page 7642) in the TIA Portal, the measurements transferred to the project are also saved.

14.6.1.3 Trace configuration, recording, installed trace and measurement

This section explains the meaning and relationships of the terms: trace configuration, recording, installed trace and measurement.

Trace configuration

You can make the following settings in the trace configuration :

- Signals to be recorded
- Recording conditions
  - Sampling
  - Trigger

Recording

A recording is performed in the device. There is only one recording for each installed trace configuration. When a new recording is started, the old recording is overwritten.

An installed recording is not retentive (it is lost when the device is switched off/on) and can be saved permanently in the project as a measurement.

Installed trace

An installed trace consists of a trace configuration and optionally a recording. The maximum number of installed traces depends on the device.

Measurement

A measurement always consists of a trace configuration with an associated recording. If an installed trace contains a recording, it can be saved in the project as a measurement.

The recording of a measurement can be viewed offline.

Trace configuration with an installed trace of the same name

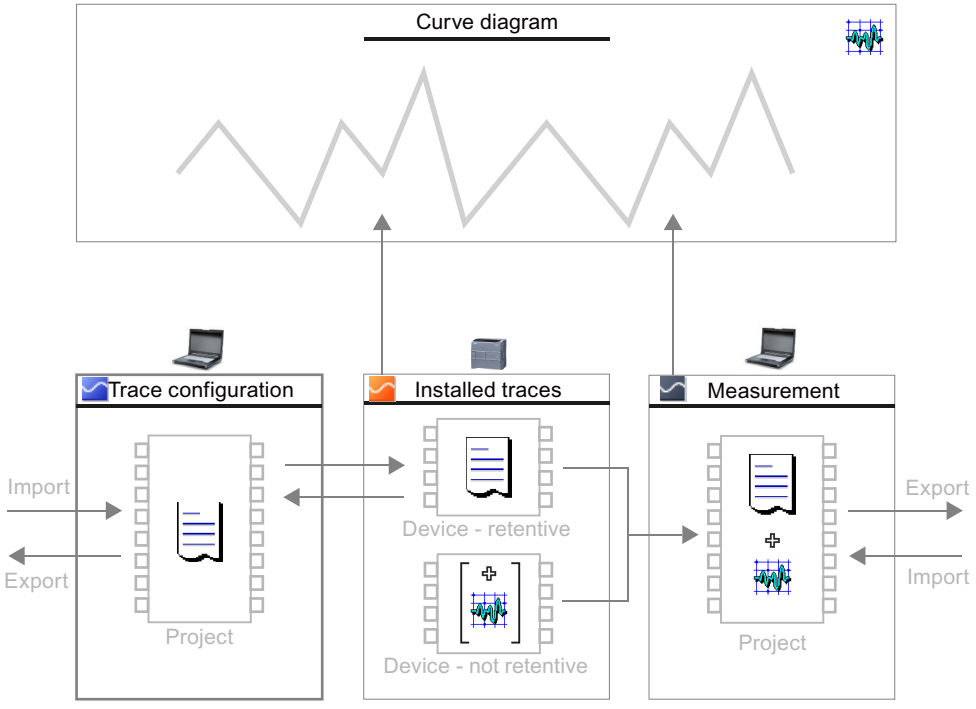
Usually, there is a trace configuration in the project with the same name for an installed trace. When there is an online connection, this trace is displayed with the icon in the project tree.

See also User interface - project tree (Page 7626).

14.6.1.4 Data storage

The trace toolbar and the curve diagram also enable the transfer of the trace configuration and the viewing of the recording.

The following figure is a schematic diagram of the data storage:



Note

Saving the trace configuration and measurement

You save the trace configuration and measurement with the project in the TIA Portal.

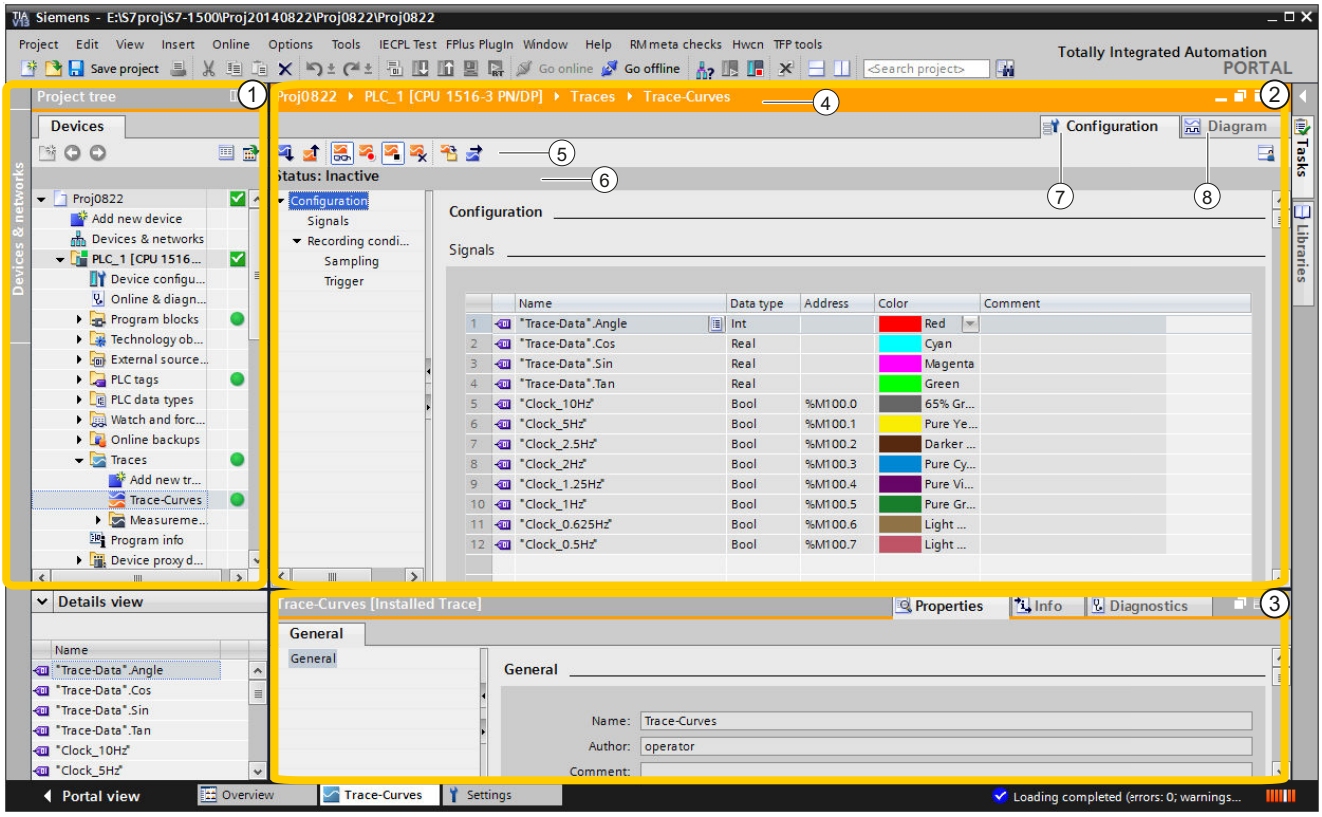
If you close the project without saving, the trace configurations and the measurements transferred to the project are discarded. The trace can be closed and reopened without loss of data until the project is closed.

14.6.2 Software user interface

Display areas

The user interface of the trace and logic analyzer function consists of several areas.

The example in the figure below shows the division of the user interface in the TIA Portal:



1	**Project navigator** Management and creation of the trace and measurements directly in the project tree and via context menu commands.
2	**Working area**

4		Title bar of the working area Shows the device to which the current display belongs.
5		Trace toolbar Buttons for managing the trace in the project and device: • Activation/deactivation of installed traces • Deletion of installed traces • Transfer of trace configurations and measurements between the device and the project • Export of trace configurations and measurements • Switchover between offline and online display
6		Status display of the trace Display of the current status of the recording.
7		Configuration tab Device-specific configuration of the recording duration, trigger condition and signal selection. See Device-specific descriptions (Page 7648).
8		Diagram tab Display of the recorded values as a curve diagram and the signals from the displayed measurement.
3	**Inspector window** Display of general information about the trace configuration	

14.6.2.1 Project navigator

User interface - project tree

Trace configurations, installed traces and measurements are displayed in the "Traces" folder.

Double-click an icon to open the appropriate "Diagram" or "Configuration" tab in the working area.

Icons in the "Traces" folder

The following table explains the icons in the "Traces" folder:

Icon	Description
	Add trace configuration Double-click the icon to add a new trace configuration.
	Trace configuration (offline) Double-click the icon to open the "Configuration" tab.
	Installed trace (online) The icon is only displayed when there is no offline trace configuration of the same name for the installed trace. Double-click the icon to open the "Diagram" tab.

Icon	Description
	Trace configuration with an installed trace of the same name If the button is deactivated, the trace configuration from the project is displayed. The trace corresponds to a trace configuration. If the button is activated, the trace configuration from the device is displayed. The trace corresponds to an installed trace. Double-click the icon to open the "Diagram" tab.
	"Measurements" system folder
	Measurement (offline) Double-click the icon to open the "Diagram" tab.

Status

When there is an online connection, the status is displayed in the right-hand column of the project tree. The status is also displayed as tooltip above the respective icon.

The following table shows the meaning of the icons:

Icon	Description
	Online and offline configuration are identical
	Online and offline configuration are different
	Configuration only exists online

Shortcut menu commands

The following table shows the shortcut menu commands for the "Traces" system folder:

Shortcut menu command	Description
"Import trace configuration"	Imports a trace configuration from a file.

The following table shows the shortcut menu commands for the "Measurements" system folder:

Shortcut menu command	Description
"Import measurement"	Imports a measurement from a file with the "*.ttrecx" file extension. For compatibility reasons, the "*.ttrec" file extension will be supported in V12, but does not contain any information about the device family.

The following table shows the shortcut menu commands for trace configurations, installed traces / and measurements:

Shortcut menu command	Trace configuration	Installed trace	Measurement	Description
"Cut"	-	-	-	Cannot be selected.
"Copy"	x	-	x	Copies the trace configuration of the selected objects to the clipboard.

Shortcut menu command	Trace configuration	Installed trace	Measurement	Description
"Paste"	x	-	x	Pastes the contents of the clipboard.
"Delete"	x	x	x	Deletes the selected objects from the project tree or from the device.
"Rename"	x	x	x	Switches the selected object to the editing mode.

The trace configuration can also be copied device-wide.

Several objects can be selected.

14.6.2.2 Working area

User interface - trace toolbar

Tools are available for handling the trace via buttons.

The following table shows the functions of the buttons:

Icon	Description
	Transfer the selected trace configuration to the device The selected trace configuration is transferred to the device.
	Transfer the selected trace configuration from the device The selected trace configuration is transferred from the device to the project.
	Observe on/off Change of the display between online and offline.
	Activate recording If the recording of an installed trace is repeated, then the settings relevant for the display (curve diagram and signal table) are also retained for the new recording.
	Deactivate recording
	Transfer the selected measurement from the device to the project The measurement is added to the "Measurements" system folder. **Note** In order to be able to save the installed trace data as a measurement, it must first have been displayed once in the curve diagram. The recording data is loaded from the device when displayed.
	Delete installed trace Deletes the selected trace from the device.
	Export trace configuration Exports a trace configuration as a file with the extension "*.ttcfgx". For compatibility reasons, the "*.ttcfgx" file extension will be supported in V12, but does not contain any information about the device family.
	Export measurement Exports a measurement as a file with the extension "*.ttrecx" or "*.csv". For compatibility reasons, the "*.ttrec" file extension will be supported in V12, but does not contain any information about the device family.

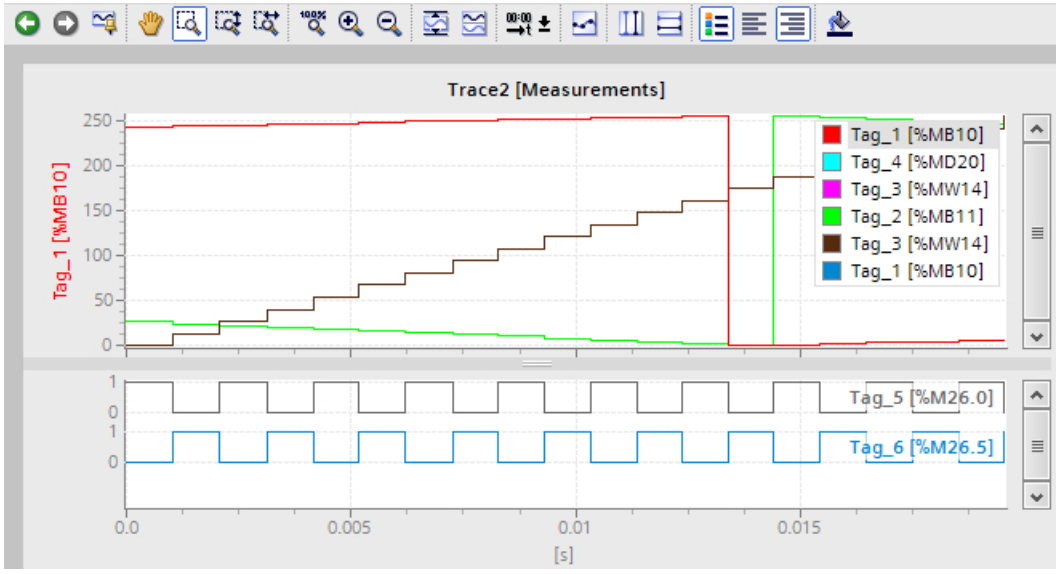
User interface - Diagram tab

User interface - curve diagram

The curve diagram displays the selected signals of a recording. Bits are shown in the lower diagram as a bit track. You can adjust the display of the signals in the signal table (Page 7631) and with the aid of the toolbar of the curve diagram.

Setting options and displays in the curve diagram

The following figure shows an example of the display in the TIA Portal:



The scale in the diagram applies to the selected (highlighted in gray) signal in the legend. The legend can be moved laterally with the mouse.

Shortcut menu commands

The following table shows the shortcut menu commands in the curve diagram:

Shortcut menu command	Description
"Save diagram as image"	Exports the current display as a bitmap.
"Copy image to clipboard"	Copies the current display to the clipboard.
"Center measurement cursors"	Positions the activated measurement cursors at a central point in the current display.
"Add to measurements" (only axis control panel and PID)	Adds the displayed recording to the "Measurements" system folder.

Toolbar of the curve diagram

Tools are available for adapting the display via buttons.

The following table shows the functions of the buttons:

Icon	Function	Description
	Undo zoom	Undoes the zoom function executed last. If several zoom functions have been executed, they can be undone step-by-step.
	Redo zoom	Redoes the last undone zoom function. If several zoom functions have been undone, they can be redone step-by-step.
	Standard view	Uses the current view as standard for this recording. If the trace recording is shown again later, the standard view is restored.
	Move view	Moves the display with the mouse button pressed.
	Zoom selection	Selection of an arbitrary range with the mouse button pressed. The display is scaled to the range selection.
	Vertical zoom selection	Selection of a vertical range with the mouse button pressed. The display is scaled to the range selection.
	Horizontal zoom selection	Selection of a horizontal range with the mouse button pressed. The display is scaled to the range selection.
	Display all	Scales the display of the available data so that the entire time range and all values are displayed. **Note** The dynamic progress indicator of a running trace is stopped when a zoom function is activated. This button activates the progress indicator again.
	Zoom in	Enlargement of the display. The ranges of the time axis and value axis are reduced every time the button is clicked. The curves are displayed larger.
	Zoom out	Reduction of the display. The ranges of the time axis and value axis are increased every time the button is clicked. The curves are displayed smaller.
	Scale automatically	Scaling of the display so that all values are displayed for the currently displayed time range. If signal groups are parameterized in the signal table, the minimum and maximum values of the Y-scale are determined for each signal group. In this way, the automatic scaling is always performed on the signal group and not on the individual signals of the signal group.
	Arrange in tracks	Arranges signals one beneath the other without overlaps.
	Unit changeover of the time axis	Changeover of the unit between time and cycles.
	Display samples	The samples are displayed as small circles on the curves.
	Display vertical measurement cursors	Display of the vertical measurement cursors. The vertical position of the two measurement cursors can be moved with the mouse. The associated measured values and the difference of the measurement cursors corresponding to the position are shown in the signal table.

Icon	Function	Description
	Display horizontal measurement cursors	Display of the horizontal measurement cursors. The horizontal position of the two measurement cursors can be moved with the mouse.
	Display chart legend	Display or hiding of the chart legend in the curve diagram.
	Align the chart legend to the left	Display of the chart legend on the left side of the curve diagram.
	Align the chart legend to the right	Display of the chart legend on the right side of the curve diagram.
	Change background color	Changeover between various background colors.

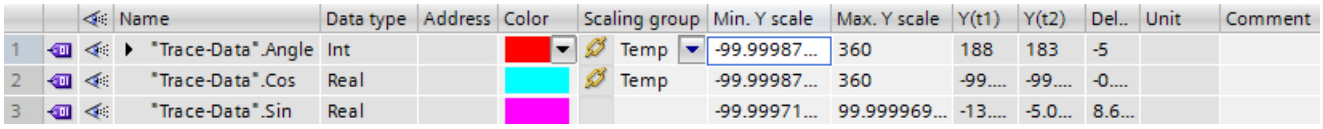
User interface - signal table

The signal table lists the signals of the selected measurement and provides setting options for some properties. If recording data of installed traces is displayed and the settings are changed in the signal table, these settings are retained until there is a change to the offline mode. If the installed trace is added to the measurements, the current settings of the signal table are saved in the measurement.

The signals can be sorted using drag-and-drop. The bits of a signal can be resorted within a signal.

Setting options and displays in the signal table

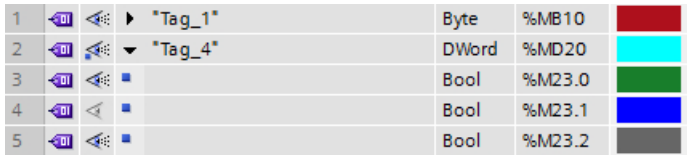
The following figure shows an example of the display in the TIA Portal:



			Name	Data type	Address	Color	Scaling group	Min. Y scale	Max. Y scale	Y(t1)	Y(t2)	Del..	Unit	Comment
1			"Trace-Data".Angle	Int			Temp	-99.99987...	360	188	183	-5		
2			"Trace-Data".Cos	Real			Temp	-99.99987...	360	-99....	-99....	-0....		
3			"Trace-Data".Sin	Real				-99.99971...	99.999969...	-13....	-5.0...	8.6...		

The following table shows the settings and displays of the recorded signals:

Column	Description
	Static display of the signal icon
	Selection for the display in the curve diagram The point indicates that at least one bit has been selected for display as bit track for the signal in the bit selection.
"Name"	Display of the signal name A click on the name of a displayed signal updates the scale in the curve diagram.

Column		Description
		Open bit selection Individual bits can also be selected for the following data types for display as a bit track in the lower curve diagram. • Byte, Word, DWord, LWord • SInt, USInt, Int, UInt, DInt, UDInt, LInt, ULInt Example of an opened bit selection for the DWORD data type:  Select or deselect the relevant bit for display by clicking the icon.
"Data type"		Display of the data type
"Address"		Display of the address (not for symbolic tags)
"Color"		Display and setting option for the color of the signal
"Signal group"		Display or input of the signal group name for one signal group The Y-scales are scaled identically for all signals of one signal group. Enter an identical signal group name for those signals that are to be scaled identically. Signals can be removed from the signal group by deleting the signal group name. The signal groups can be saved via the function "Use current view as standard" (button). **Note** Binary signals cannot be grouped.
	Gray field for chain icon	Move the cursor over the gray field or the chain icon (or) to add the signal to a signal group or delete the signal from the signal group. Clicking the chain icon adds the signal to a signal group or creates a new signal group. Clicking the chain icon removes the signal from the signal group. For a selected signal with signal group, the chain icon displays all signals of the same signal group.
	Input field	The input field displays the signal group name. As an alternative to the chain icon, you can assign or delete a group name via text input in this field.
"Min. Y-scale"		Display or input of the minimum value for the scaling of the signal
"Max. Y-scale"		Display or input of the maximum value for the scaling of the signal
"Y(t1)"		Display of the value at the position of the first measurement cursor
"Y(t2)"		Display of the value at the position of the second measurement cursor
"ΔY"		Display of the value difference between the first and the second measurement cursor
"Unit"		Display of the unit (e.g. for technology objects)
"Comment"		Display and input option for a comment about the signal

Shortcut menu commands

The following table shows the shortcut menu commands of the signal table:

Shortcut menu command	Description
"Cut"	Cannot be selected.
"Copy"	Copies the contents of the selected lines to the clipboard.
"Paste"	Cannot be selected.
"Delete"	Cannot be selected.
"Show/hide signal"	Shows/hides the signal in the curve diagram.

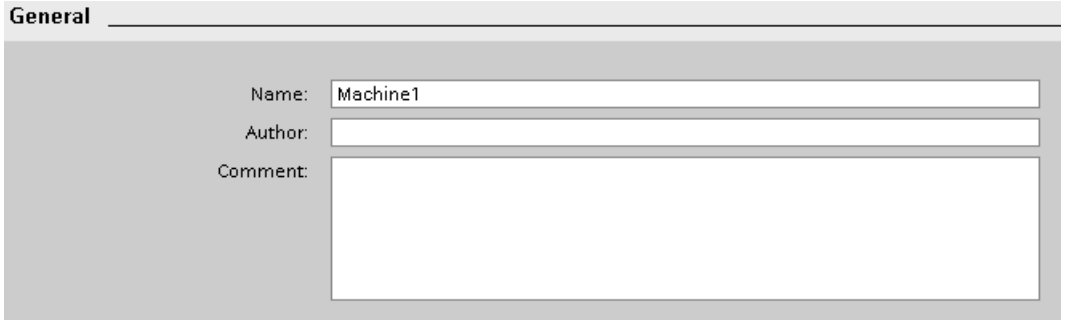
14.6.2.3 Inspector window

User interface - General

The "General" area shows the name of the trace configuration and input fields for the author and a comment.

Input options and displays in General

The following figure shows an example of the display in the TIA Portal:



The following table shows the input fields and displays:

Column	Icon	Description
"Name"	-	Name of the trace configuration.
"Author"	-	Author of the trace configuration
"Comment"	-	Input field for a comment.

14.6.3 Operation

14.6.3.1 Quick start

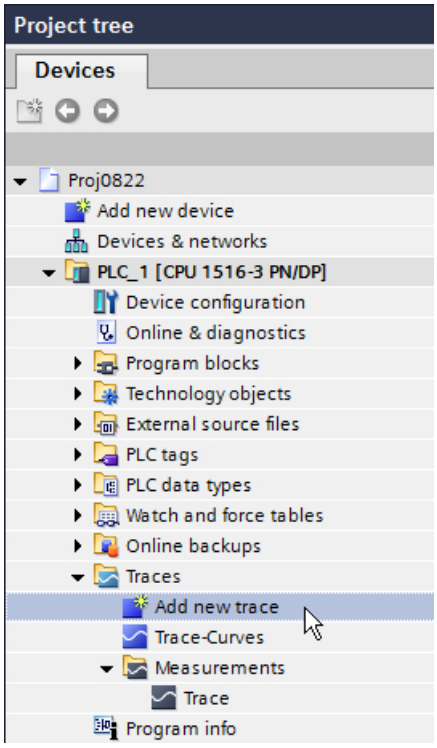
This description shows the steps for a recording of the S7-1500 CPU as an example. The displayed settings can differ depending on the device.

Requirement

A device is configured that supports the trace and logic analyzer function.

Creating a trace

The following figure shows the project tree with the "Traces" system folder below the device:

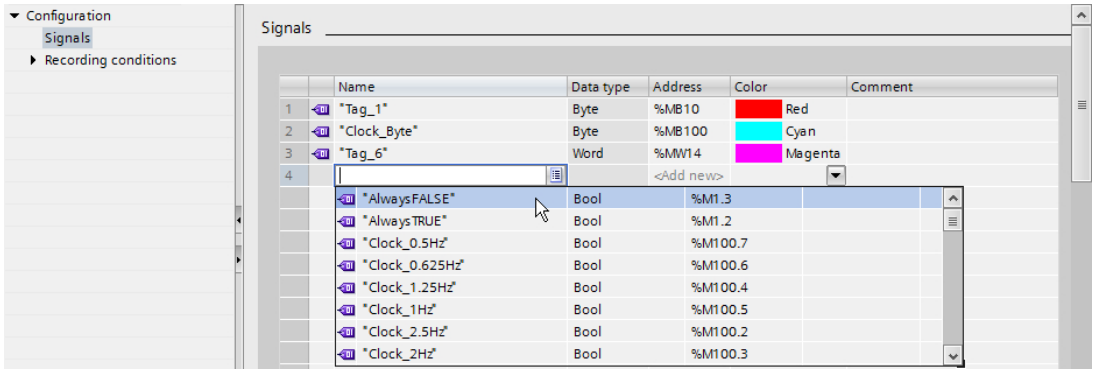


Procedure:

1. Double-click the "Add new trace" entry.
   A new trace configuration is created.
2. Adapt the name of the trace configuration by clicking the text.

Selecting signals

The following figure shows the configuration of the signals:

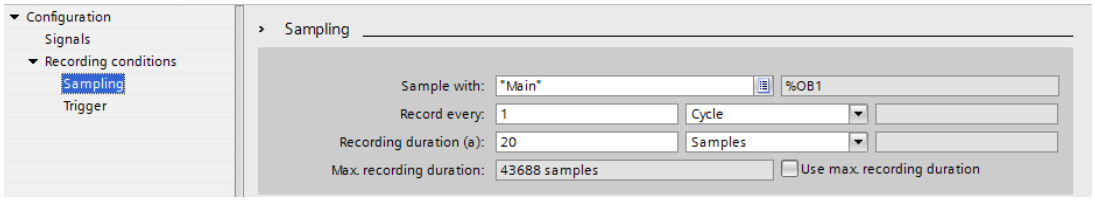


Procedure:

1. Double-click the trace configuration.
   The trace configuration is opened and displayed in the "Configuration" tab.
2. Select the signals to be recorded in the "Signals" area.
   Or:
3. Drag one or more signals, e.g. from a tag table, and drop them in the signal table.

Configuring the recording cycle

The following figure shows the configuration of the sampling:

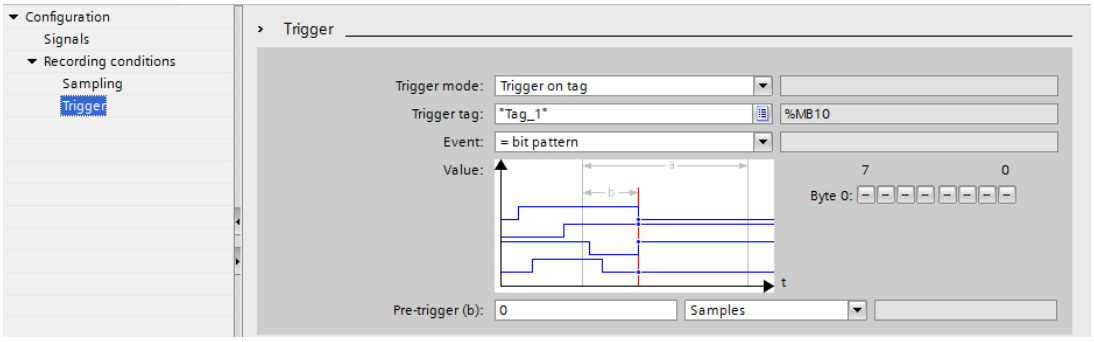


Procedure:

1. Configure the sampling.

Configuring the trigger

The following figure shows the configuration of the trigger:



Procedure:

1. Configure the trigger mode and the condition for the selected trigger.

Transferring the trace configuration to the device

Procedure:

1. Transfer the trace configuration to the device with the button.
   The following functions are executed:
   - An online connection is established to the device.
   - The trace configuration is transferred to the device.
   - The monitoring is activated.
   - The display switches to the "Diagram" tab.

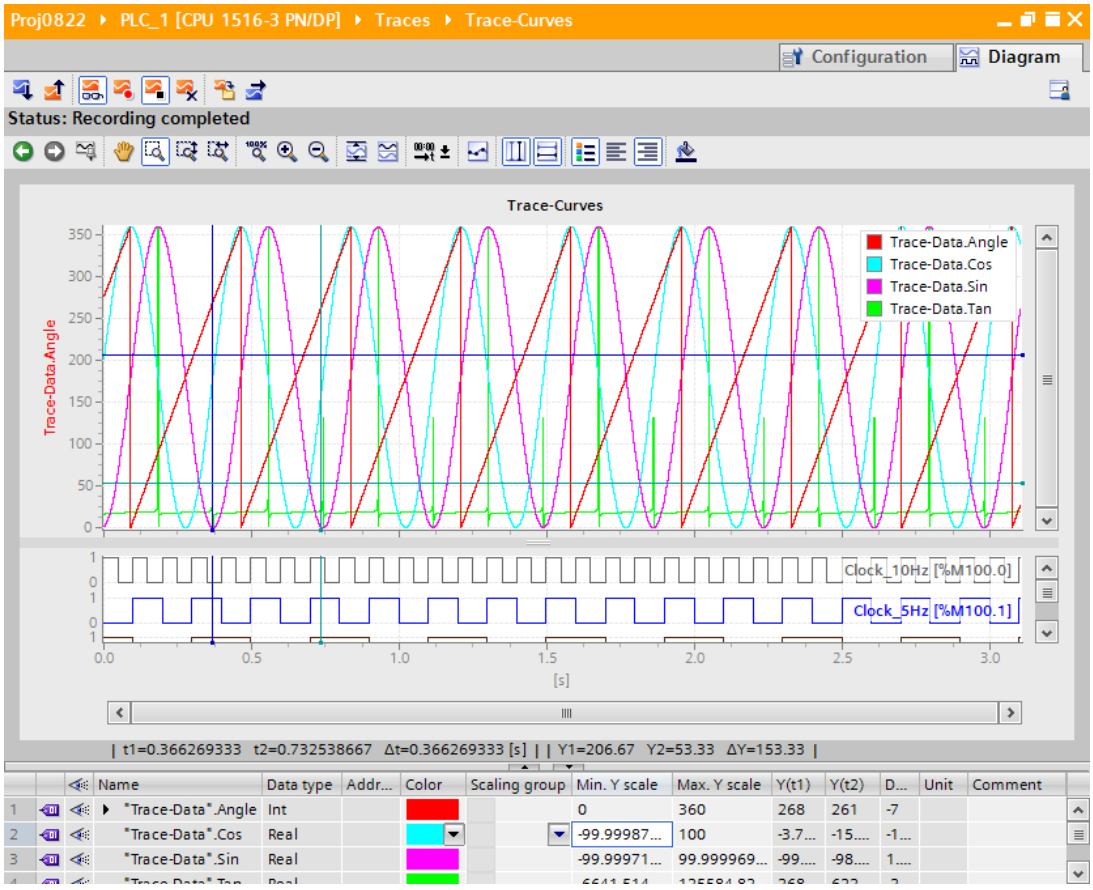
Activating a recording

Procedure:

1. Click the button.

Displaying the recording

The following figure shows the curve diagram with a recording:



Procedure:

1. Wait until the "Recording" or "Recording completed" status is displayed in the status display of the trace.
2. Switch to the "Diagram" tab.
3. Click the ▸ icon of a signal in the signal table.
   The individual bits of the signal are offered for display as a bit track.
4. In the signal table, select or deselect the individual signals and bits for display with the icon.

Saving the measurement in the project

Procedure:

1. Transfer the measurement to the project with the button.
   The measurement is displayed in the project tree under the "Measurements" system folder.

See also

User interface - trace toolbar (Page 7626)

14.6.3.2 Using the trace function - overview

Requirement

A device is configured in the TIA Portal that supports the trace and logic analyzer function and to which an online connection has been established.

Procedure

The following table shows a procedural overview with typical steps when working with the trace and logic analyzer function.

Step	Description
1	Creating a trace (Page 7638)
2	Configuring the trace (Page 7644)
3	Transferring the trace configuration to the device (Page 7640)
4	Activating/deactivating an installed trace (Page 7640)
5	Monitoring the recording (Page 7641)
6	Saving measurements in the project (Page 7642)
7	Displaying the recording (Page 7641)

See also

Displaying a configuration (Page 7639)

14.6.3.3 Project tree

Creating a trace

Traces can be created in the form of trace configurations in the project navigator.

The following instructions describe how you can create a trace configuration under the "Traces" system folder

Requirement

A device is configured that supports the trace and logic analyzer function.

Procedure

To create a trace configuration, proceed as follows:

1. Double-click the "Add new trace" entry.

A new trace configuration is created.

Displaying a configuration

Requirement

A trace configuration, an installed trace or a measurement is available in the "Traces" system folder.

Procedure

To display a trace configuration, proceed as follows:

1. Double-click the appropriate icon of a trace configuration, an installed trace or a measurement in the project tree.
   The "Configuration" or "Diagram" tab opens in the working area.
2. If required, click the "Configuration" tab for the display.

Note

Write protection

The configuration data of an installed trace or a measurement is displayed with write protection.

See also

User interface - project tree (Page 7624)

Displaying a diagram

Requirement

An installed trace or a measurement is available in the "Traces" system folder.

Procedure

To display a diagram, proceed as follows:

1. Double-click the appropriate icon of a trace configuration, an installed trace or a measurement in the project tree.
   The "Configuration" or "Diagram" tab opens in the working area.
2. If required, click the "Diagram" tab for the display.

See also

User interface - project tree (Page 7624)

14.6.3.4 Working area - general

Transferring the trace configuration to the device

Requirement

- A valid trace configuration is in the "Traces" system folder.
- The maximum number of installed traces has not been reached yet.

Procedure

To transfer a trace configuration to the device, proceed as follows:

1. Open a valid trace configuration in the working area.
2. Click the button.

Result

The trace configuration is transferred to the device.

Activating/deactivating an installed trace

Requirement

- There is an online connection to the device.
- There is a trace in the device.
- The installed trace is displayed in the working area.
- The button is activated for viewing the displayed trace.

Activating an installed trace

To activate the recording for an installed trace, proceed as follows:

1. Click the button.
   The installed trace is activated and starts the recording according to the configured trigger condition. The trigger condition is device-specific and described in Section "Configuration" below the respective device (Page 7648).
   The current status of the recording is displayed in the status display of the trace.

   Note

   When a recording is restarted, the previously recorded values are lost.

   To save the recorded values, save the measurement in the project (Page 7642) before you activate the recording again.

Deactivating an installed trace

To deactivate an activated installed trace, proceed as follows:

1. Click the button.
   The installed trace is deactivated.
   The status display of the trace changes to "Inactive".

Displaying the recording

Requirement

- There is an online connection to the device.
- There is a trace with recording in the device.

Or:

- A measurement is in the "Measurements" system folder.

Procedure

To display the recording, proceed as follows:

1. Select an installed trace.
2. Double-click the selected trace.
3. If required, activate the button for viewing.

Or:

1. Select a measurement in the "Measurements" system folder.
2. Double-click the selected measurement.

Result

The recording is displayed in the "Diagram" tab.

See also

User interface - project tree (Page 7624)

Saving measurements in the project

Requirement

- There is an online connection to the device.
- There is a trace with recording in the device.
- The installed trace data must have been displayed at least once in the curve diagram. The recording data is loaded from the device for the display.

Procedure

To save a recording in the project, proceed as follows:

1. Open the installed trace with the recorded data.
2. If required, make sure that the current data is loaded from the device by activating the button.
3. Click the button.
   The measurement is added to the "Measurements" system folder.
4. Save the project in the TIA Portal.

Exporting and importing measurements

Requirement

At least one measurement is in the "Measurements" system folder for export.

Exporting measurements

To export a measurement, proceed as follows:

1. Display the measurement in the working area.
2. Click the button.
3. Select a folder, a file name and a data type to save the measurement.
4. Click the "Save" button.

Importing measurements

To import a measurement, proceed as follows:

1. Right-click in the "Measurements" system folder and select the shortcut menu command "Import measurement".
2. Select the file of the "*.ttrecx" file type with the measurement to be imported.
3. Click the "Open" button.
   The imported measurement is displayed with the file name in the "Measurements" system folder.

Transferring the trace configuration from the device to the project

Requirement

- There is an online connection to the device.
- There is a trace in the device.

Procedure

To transfer a trace configuration to the project, proceed as follows:

1. Open an installed trace.
2. If required, activate the button for viewing.
3. Click the button to transfer the trace configuration from the device.

Result

The configuration is taken over as new trace configuration in the "Traces" system folder.

A trace configuration of the same name is overwritten in the system folder.

Deleting installed traces

Requirement

- There is an online connection to the device.
- There is a trace in the device.

Procedure

To delete an installed trace, proceed as follows:

1. Open an installed trace.
2. If required, activate the button for viewing.
3. Click the button.
   A confirmation prompt opens.
4. Confirm the prompt for deletion.

Or

1. Select one or more installed traces / in the project tree.
2. Press <Del> to delete the installed traces.
   A confirmation prompt opens.
3. If required, select an option for deletion and confirm the prompt.

14.6.3.5 Working area - Configuration tab

Configuring the trace

Requirement

The "Configuration" tab is open in the working area.

Configuring the trace

In the configuration, you specify the recording and trigger conditions and select the signals to be recorded.

See Section "Configuration" below the respective device (Page 7648).

Note

Saving the trace configuration

You save the trace configuration with the project in the TIA Portal.

If you close the project without saving, the configuration is discarded.

See also

Displaying a configuration (Page 7637)

14.6.3.6 Working area - Diagram tab

Use of the curve diagram

The curve diagram shows the signals of a recording selected in the signal table.

The display area can be zoomed as required. Measurement cursors can be used to select individual values for display in the signal table.

The following operating instructions describe the use of the measurement cursors.

Requirements

- An installed trace or a measurement has been selected for display.
- The button is activated for viewing an installed trace.
- The "Diagram" tab is open in the working area.

Evaluation of a certain instant of a recording

To display the values for a specific sample, proceed as follows:

1. Display the vertical measurement cursors via the button.
2. Move a measurement cursor with the mouse to the required position in the recording. The values of the signals are displayed in the signal table. The time or sample for the measurement cursors is displayed in the lower area of the curve diagram.

Evaluation of the difference between two samples

To display the difference, proceed as follows:

1. Display the vertical measurement cursors via the button.
2. Move both measurement cursors with the mouse to the required samples in the recording. The values of the signals and the difference are displayed in the signal table. The time or sample for the measurement cursors is displayed in the lower area of the curve diagram.

Using horizontal measurement cursors

To check whether a certain value has been reached, proceed as follows:

1. Display the horizontal measurement cursors via the button.
2. Move a measurement cursor with the mouse to the required value of the recording. The values of the measurement cursor for the selected signal are displayed in the lower area of the curve diagram.

Bringing a signal into the foreground

1. Display the legend via the button.
2. Click a signal in the legend. The signal is displayed in the foreground.

See also

Displaying a diagram (Page 7637)

Use of the signal table

The signal table shows the signals of an installed trace or a measurement. You can show or hide individual signals for the display in the table and adapt the properties for the display. Individual bits can be selected for some data types and displayed as a bit track.

The following operating instructions describe the operation of the signal table.

Requirements

- An installed trace or a measurement has been opened in the "Diagram" tab.
- The button is activated for viewing an installed trace.
- For the display of individual bits as a bit track:
  At least one recorded signal supports the display as a bit track.

Selecting individual signals and changing the format

To adapt the display to suit your requirements, proceed as follows:

1. Click the icon of the respective signal in the column to select or deselect it for the display.
2. Click in the "Color" column for the respective signal.
   The preset color of the signal is changed.

Bringing a signal into the foreground

1. In the signal table, select the line of the signal.
   The Y-scale of the signal is displayed.
   The signal curve is brought into the foreground in the curve diagram.

Selecting individual bits for display as a bit track

To display individual bits as a bit track in the lower curve diagram, proceed as follows:

1. Click the icon of a signal in the signal table.
2. Click the icon in the open bit selection of the signal.
   The bits are selected or deselected for display.

Using the signal group in the signal table

Individual signals can be scaled identically in a signal group, which makes it easier to compare the curve characteristics.

Binary signals cannot be grouped.

The following operating instructions describe how to work with the signal group.

Note

Saving signal groups

The signal groups can be saved individually for each measurement via the "Use current view as standard" function (button).

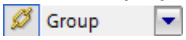
If they are not saved, the created signal groups are lost when the "Diagram" tab is closed.

Requirements

- An installed trace or a measurement is displayed.
- The button is activated for viewing an installed trace.
- The "Diagram" tab is open in the working area.
- There are at least two signals in the signal table that are not of the BOOL type.

Assigning signals to a signal group

To create a signal group and assign signals to this group, proceed as follows:

1. In the signal table, select the line or cell of the required signal.
2. Click the gray field in the "Signal group" column.
   The chain icon is displayed in the gray field and the name of the signal group is pre-assigned: 
3. Click the gray fields of further signals that are to be assigned to this signal group.

Or:

1. Click in the text field of the "Signal group" column for a signal to be grouped.
2. Enter a name for the group.
3. Enter the same group name in the respective text fields for further signals or select the group name via the drop-down list.

The Y-scales of the grouped signals are scaled with the values of the signal that was selected first. Changes to a scale value always affect the entire group.

Removing signals from a signal group

To delete the assignment of a signal to a signal group, proceed as follows:

1. Click the chain icon for the required signal in the "Signal group" column.

Or:

1. Click the text field for the required signal in the "Signal group" column.
2. Press the <Del> key.

Or:

1. Select the respective text field in the "Signal group" column for several signals using the <Shift> and <Ctrl> keys.
2. Press the <Del> key.

The signals are removed from the signal group or the signal group is deleted.

Printing a recording

The curve diagram supports the saving of the display as a bitmap and the copying of the display to the clipboard. Also use these functions (Page 7627) for printing.

14.6.4 Devices

14.6.4.1 S7-1200/1500 CPUs

Recordable variables

Device-dependent recording of tags

The following list shows the operand areas from which tags can be recorded:

- Process image input
- Process image output
- Bit memory
- Data blocks

Data types

Elementary data types can be recorded. The availability of the individual data types depends on the device used:

The following table lists the elementary data types:

Data types	Note
Binary numbers	
BOOL	-
Bit strings	
BYTE	-
WORD	-
DWORD	-
LWORD [1)]	Symbolic name required
Integers	
SINT	-
USINT	-
INT	-
UINT	-
DINT	-
UDINT	-
LINT [1)]	Symbolic name required
ULINT [1)]	Symbolic name required

Data types	Note
Floating-point numbers	
REAL	-
LREAL	Symbolic name required

1) Is not supported by all devices.

Lifetime of the installed trace configuration and recorded values

Installed trace configurations are retained after POWER OFF. The recording is activated again after the restart of the CPU.

Recorded values are lost during the restart.

Note

Downloading a configuration to the device in the "STOP" operating state

Note that after downloading a configuration in the "STOP" operating state, you must check the installed traces and, if required, reactivate them or transfer them again.

Note

If trigger tags that affect the address are changed, the trace configuration must also be transferred to the device again.

This is the case for example, when a data block is shortened or extended or the data type is changed.

Recording levels

The following list shows the execution levels that can be selected for the recording cycle:

- Program cycle - OB 1
- Time-of-day interrupt - OB 1x
- Time-delay interrupt - OB 2x
- Watchdog interrupt - OB 3x
- Synchronized processing cycles - OB 6x, not OB 60
- MC-Servo - OB 91
- MC-Interpolator - OB 92

Note

The measured values are recorded at the end of the OB after the processing of the user program.

Quantity structure

The following table shows the maximum quantity structure that can be recorded with the trace and logic analyzer function:

Device	Maximum number of installed traces	Maximum number of signals per trace configuration
S7-1200 (as of firmware version V4.0)	2	16
S7-1500	At least 4 (depending on the CPU type)	16

Example of CPU 1516-4 PN/DP

- Maximum of 3854 samples for 16 signals from PLC tags of the DWORD data type
- Maximum of 21844 samples for 16 signals from PLC tags of the BOOL data type
- Maximum of 58250 samples for 1 signal from a PLC tag of the BOOL data type

Refer to the respective manual for more detailed information.

CPU load through trace recording

A trace recording increases the runtime of the respective recording level that can result in an execution level overflow with high utilization of the CPU.

Remedy for execution level overflow:

- **Change the trace configuration**
  1) Configure fewer tags and signals.
  2) Then increase the number of tags and signals up to the maximum number of signals step-by-step without an execution level overflow.
- **Select a slower recording level**

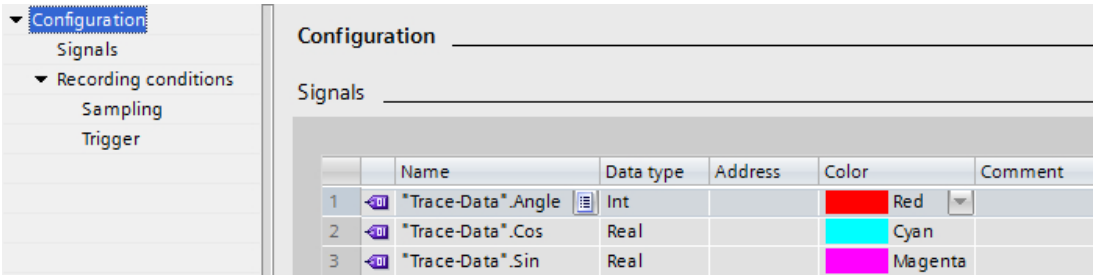
Software user interface of the configuration

Structure of the user interface

Display areas in the "Configuration" tab of the working area

The settings options differ depending on the configured device.

The following figure shows an example of the display in the TIA Portal:



The area navigation provides the following entries for selection:

- Configuration
  - Signals (Page 7651)
  - Recording conditions (Page 7652)

Displaying and changing properties of a trace configuration

A trace is selected in the project tree and displayed in the "Configuration" tab.

The trace configuration can only be changed offline and is displayed online with the write protection.

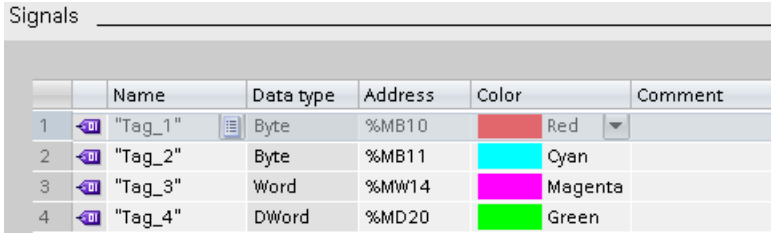
User interface - Signals

The "Signals" area shows a table in which the signals to be recorded are configured for the selected trace configuration.

Signals can also be inserted in the table using drag-and-drop.
The signals can be sorted using drag-and-drop.

Setting options and displays in "Signals"

The following figure shows an example of the display in the TIA Portal:



The following table shows the settings and displays:

Column	Icon	Description
		Display of the signal icon for a selected signal.
"Name"	-	Input field for the name or address of the signal. **Examples:** • M0.0 • DB1.DBW3 • "Data_block_1".pressure
-		Button to open the signal selection table. The button is displayed when the table line is selected. Clicking the icon opens a table which offers possible signals for selection. The selected signal is displayed in the input field.
"Data type"	-	Text field with display of the data type for the signal.
"Address"	-	Input field for the address of the signal. With purely symbolic signals the field remains empty.
"Color"	-	Text field for display and selection of the color. Click the signal color to open the color selection dialog.
"Comment"		Input field for a comment on the signal.

Shortcut menu commands

The following table shows the shortcut menu commands of the table:

Shortcut menu command	Description
"Cut"	Cannot be selected.
"Copy"	Copies the contents of the selected lines to the clipboard.
"Paste"	Pastes the contents of the clipboard to the selected line. The existing contents are overwritten.
"Delete"	Deletes the selected lines from the table or deletes the content of the selected cell.
"Rename"	Switches the selected cell to the editing mode.

Recording conditions

Supported data types

The following table shows the supported data types for the trigger tag:

Memory requirement and format of the number	Data type
1 byte	BOOL
8-bit integers	SINT, USINT, BYTE
16-bit integers	INT, UINT, WORD
32-bit integers	DINT, UDINT, DWORD
64-bit integers	LINT, ULINT, LWORD (device-dependent)

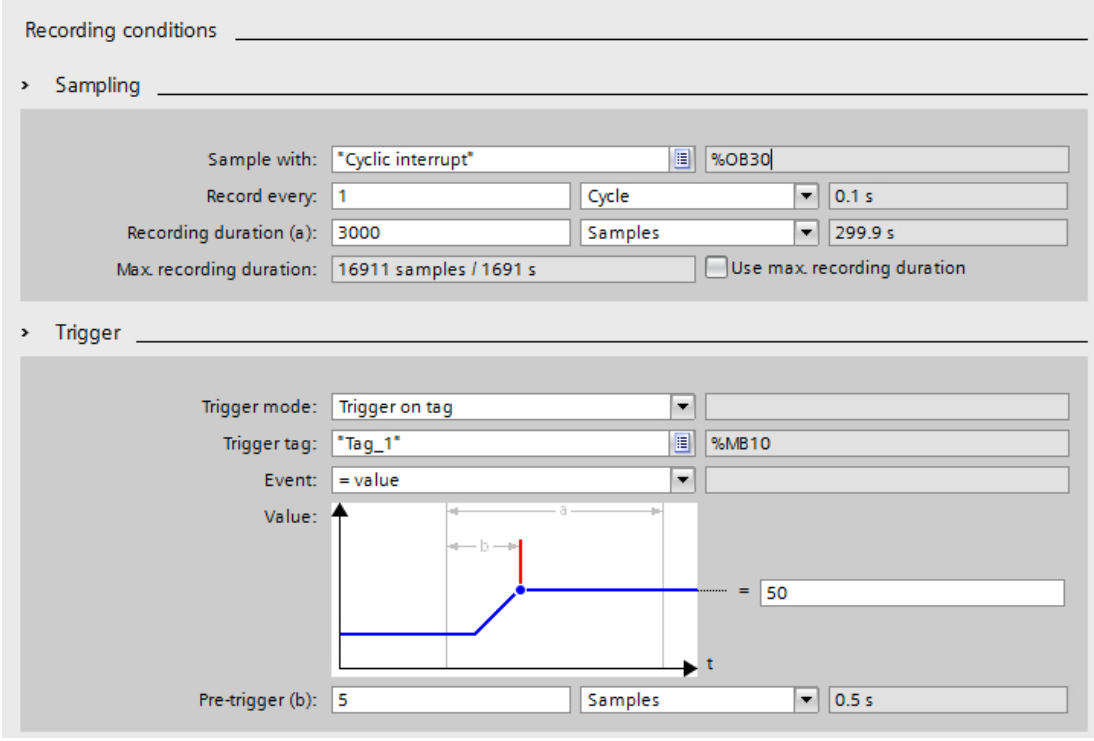
Memory requirement and format of the number	Data type
32-bit floating-point numbers	REAL
64-bit floating-point numbers	LREAL

User interface - Recording conditions

The "Recording conditions" area shows the trigger condition for the selected trace configuration and in which cycle, how fast and how long the recording is made. Configuration is possible when the trace configuration is displayed in offline mode or online mode with deactivated viewing .

Setting options and displays in "Recording conditions"

The following figure shows an example of the display in the TIA Portal:



Setting/display		Description
"Recording time"		
	Drop-down list	Selection of the recording time. See Recording levels (Page 7647)
	Text field	Detailed information on the selected recording time.
"Record every"		
	Input field	Input of the reduction in relation to the cycles.

Setting/display		Description
	Drop-down list	Selection of the reduction factor. The following settings are possible: • "Cycle"
	Text field	Display of the sampling time, taking into account the configured reduction (only for constant bus cycle time OBs).
"Recording duration"		
	Input field	Input of the recording duration in relation to the samples. If the "Recording duration = max. recording duration" checkbox is activated, entries are overwritten by the value displayed in "Max. recording duration".
	Drop-down list	Select the time unit for the recording duration. The following settings are possible: • "Samples"
	Text field	Display of the calculated recording duration (only for constant bus cycle time OBs)
"Max. recording duration"		
	Text field	Displays the calculated maximum recording duration. The "Max. recording duration" depends on how many signals are recorded and the data type of these signals.
	"Use max. recording duration"	Set the recording duration to the maximum value. When the checkbox is activated, the recording duration is set to the maximum possible recording duration. The set reduction in the "Record every" input field is taken into account. The recording duration is also adapted when additional signals are added.
"Trigger mode"		Selection of the trigger mode.
	Drop-down list	The following settings are possible: • "Record immediately" The recording is made immediately after the device is activated. • "Trigger on tag" The recording is made as soon as the installed trace is activated and the configured trigger condition is fulfilled.
	Text field	-
"Trigger tag"		The "Trigger tag" specifies a signal that triggers the recording.
	Input field	Enter a signal. **Examples:** • M0.0 • DB1.DBW3 • "DataBlock_1".Temperature
		Opens the signal selection table. Clicking the icon opens a table which offers possible signals for selection as trigger tag. The selected signal is displayed in the input field.
	Text field	Display of the trigger tag address.

Setting/display		Description
"Event"		The events that can be used on this trigger tag are offered for selection according to the data type of the trigger tag. The event can only be configured when a valid signal has been entered as trigger tag.
	Drop-down list	Event selection for which the trigger tag is checked. The entries in the drop-down list are described in Section Trigger event (Page 7655).
	Text field	-
"Value"		Configuration of the selected event. The configuration options differ depending on the format of the trigger tag and the selected event. See Trigger event (Page 7655).
"Pre-trigger"		"Pre-trigger" defines the number of samples that are already recorded before the actual trigger condition is fulfilled. If the trigger event occurs immediately or shortly after the activation of the recording, this may result in a shorter recording duration. Examples of "Recording duration (a)" = 20 samples and "Pre-trigger (b)" = 5 samples: • Case 1: Trigger event occurs 50 samples after activation of the recording Actual recording duration (a) = 20 samples • Case 2: Trigger event occurs 2 samples after activation of the recording Actual recording duration (a) = 17 samples
	Input field	Input of the duration in relation to the selection in the drop-down list.
	Drop-down list	Select the time unit The following settings are possible: • "Samples"
	Text field	Display of the calculated "Pre-trigger" interval (only for recording in constant bus cycle time OBs)

Trigger event

Depending on the selection in the drop-down list, the further settings differ for the "event".

The individual events are described below.

"=TRUE"

Supported data types: Bit (Page 7650)

The recording starts when the state of the trigger is TRUE.

"=FALSE"

Supported data types: Bit (Page 7650)

The recording starts when the state of the trigger is FALSE.

"Rising edge"

Supported data types: Bit (Page 7650)

The recording is started when the trigger state changes from FALSE to TRUE.
After activation of the installed trace, at least two cycles are required to identify the edge.

"Rising signal"

Supported data types: Integers and floating-point numbers (Page 7650)

The recording is started when the rising value of the trigger reaches or exceeds the value configured for this event.
After activation of the installed trace, at least two cycles are required to identify the edge.

"Falling edge"

Supported data types: Bit (Page 7650)

The recording is started when the trigger state changes from TRUE to FALSE.
After activation of the installed trace, at least two cycles are required to identify the edge.

"Falling signal"

Supported data types: Integers and floating-point numbers (Page 7650)

The recording is started when the falling value of the trigger reaches or falls below the value configured for this event.
After activation of the installed trace, at least two cycles are required to identify the edge.

"In the range"

Supported data types: Integers and floating-point numbers (Page 7650)

The recording starts as soon as the value of the trigger is in the value range configured for this event.

"Outside of the range"

Supported data types: Integers and floating-point numbers (Page 7650)

The recording starts as soon as the value of the trigger is outside the value range configured for this event.

"Value change"

All data types are supported.

The value is checked for change when the recording is activated. The recording starts when the value of the trigger changes.

This trigger event is supported as of V13 SP1. Older versions of the TIA Portal cannot interpret the trigger. Note that no explicit information is output in this case. This can occur, for example,

when the trace is transferred from a CPU to a TIA Portal less than V13 SP1 or a trace configuration is imported.

"= value"

Supported data types: Integers (Page 7650)

The recording starts when the value of the trigger is equal to the value configured for this event.

"<> "<> value"

Supported data types: Integers (Page 7650)

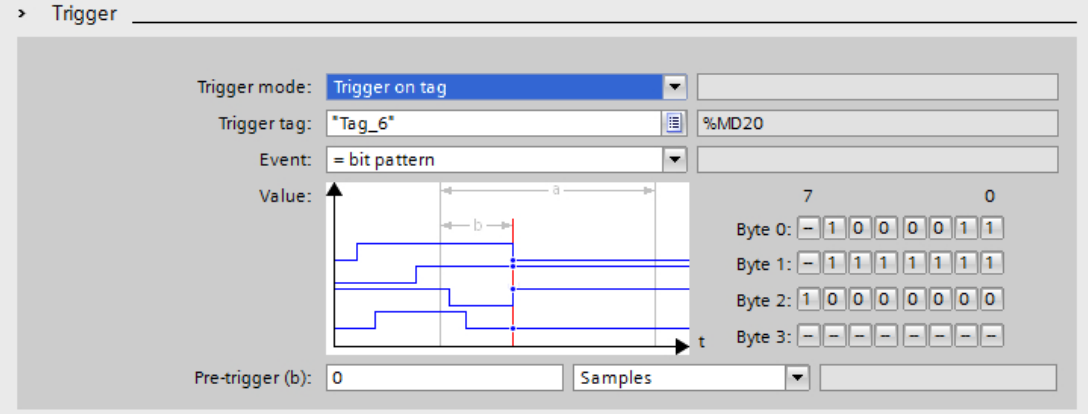
The recording starts when the value of the trigger is not equal to the value configured for this event.

"= bit pattern"

Supported data types: Integers (Page 7650)

The recording starts when the value of the trigger matches the bit pattern configured for this event.

The following figure shows the setting options for a "bit pattern":



It is possible to switch between the icons by clicking the respective button.

The following table shows the icons:

Icon	Description
–	Bit is not evaluated
0	Bit is checked for FALSE
1	Bit is checked for TRUE

"<> bit pattern"

Supported data types: Integers (Page 7650)

The recording starts when the value of the trigger does not match the bit pattern configured for this event.

See also

Configuring the trigger conditions (Page 7659)

Configuration

Trace configuration - overview

The configuration of the recording conditions and the signals to be recorded is device-specific.

Requirement

A trace configuration has been created and opened in the working area of the "Configuration" tab.

Procedure

The following table shows the procedure for configuring.

Step	Description
1	Documentation of the configuration (optional) Enter a comment and an author for the configuration in the Inspector window.
2	Selecting signals (Page 7658) Select the signals to be recorded in the "Signals" area.
3	Configuring the recording cycle and duration Select a recording time, a cycle and the duration in the "Recording conditions" area.
4	Configuring the trigger conditions (Page 7659) In the "Recording conditions" area, select whether the recording is to be performed immediately or depending on a trigger condition.

Selecting signals

Requirement

- A trace configuration has been created and opened.
- The "Signals" area is open in the "Configuration" tab.

Procedure

To configure the signals to be recorded, proceed as follows:

1. Select a signal. The following options are available:
   - In the "Name" column, click the button and select a tag.
   - Enter the symbolic tag name in the cell in the "Name" column.
   - Enter the address directly in the "Address" column.
   - Drag a signal to the table using drag-and-drop.
2. Click in the "Color" column and select a color for the display of the signal.
3. Click in the "Comment" column and enter a comment for the signal.
4. Repeat the procedure from step 1 until all the signals to be recorded have been entered in the table.

Configuring the recording cycle and duration

Requirement

- A trace configuration has been created and opened.
- The "Recording conditions" area is open in the "Configuration" tab.

Procedure

To configure the cycle and the duration of a recording, proceed as follows:

1. Click the button for the recording time.
2. Select an OB for the recording time (Page 7647).
3. Select a unit for the reduction factor in the drop-down list for "Record every".
4. Enter the factor for the reduction in the input field for "Record every".
5. Select a unit in the drop-down list for "Recording duration".
6. Specify the recording duration.
   The following options are available:
   - Enter a value for the duration in the input field for "Recording duration".
   - Activate the "Use max. recording duration" checkbox.

Configuring the trigger conditions

Requirement

- A trace configuration has been created and opened.
- The "Recording conditions" area is open in the "Configuration" tab.

"Record immediately" trigger condition

To start the recording immediately, proceed as follows:

1. Select the "Record immediately" entry in the drop-down list for "Trigger mode".
   The input fields for the trigger tag are hidden.

"Trigger on tag" trigger condition

To start the recording depending on a condition, proceed as follows:

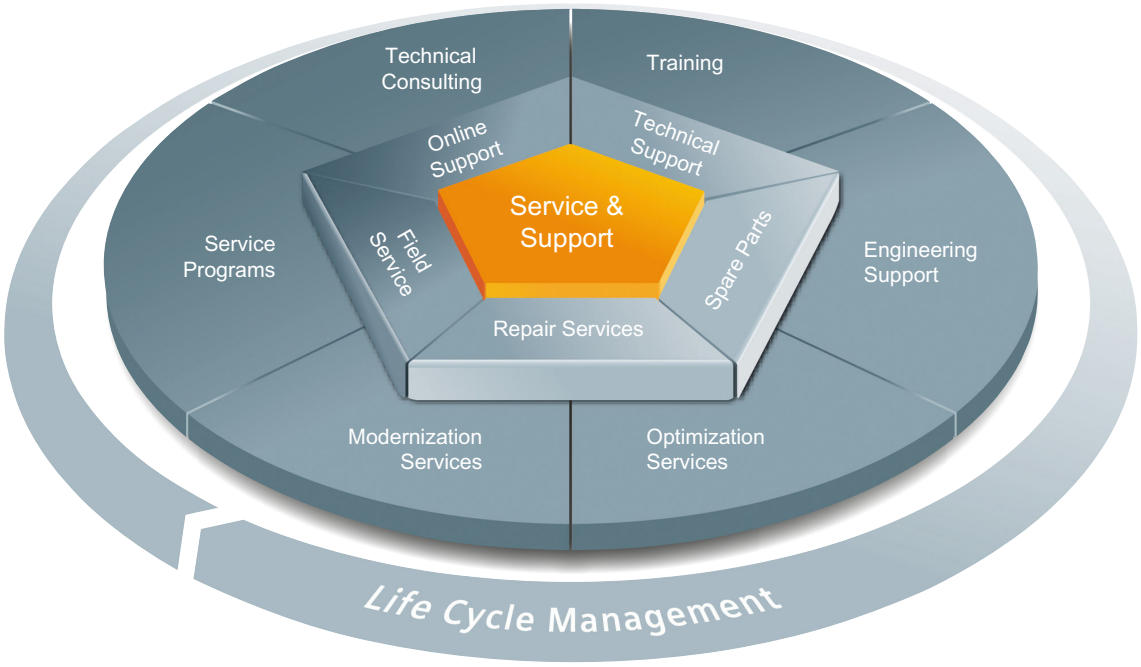
1. Select the "Trigger on tag" entry in the drop-down list for "Trigger mode".
2. Select a trigger tag. The following options are available:
   - Click the button for the trigger tag and select a tag.
   - Enter the address or the symbolic name of the tag directly in the input field for the trigger tag.

   A drop-down list with events and input fields is displayed. The display depends on the data type of the tag.
3. Configure the event.
4. Select a unit for the pre-trigger in the drop-down list for "Pre-trigger".
5. In order to record a period before the trigger event, enter a value greater than 0 in the input field for the pre-trigger.

Note

The trigger condition is checked in every cycle irrespective of the setting in "Record every". To reliably identify the trigger, the trigger signal must be present for at least one full cycle.

A Service & Support



The unmatched complete service for the entire life cycle

For machine constructors, solution providers and plant operators: The service offering from Siemens Industry Automation and Drive Technologies includes comprehensive services for a wide range of different users in all sectors of the manufacturing and process industry.

To accompany our products and systems, we offer integrated and structured services that provide valuable support in every phase of the life cycle of your machine or plant – from planning and implementation through commissioning as far as maintenance and modernization.

Our Service & Support accompanies you worldwide in all matters concerning automation and drive technology from Siemens. We provide direct on-site support in more than 100 countries through all phases of the life cycle of your machines and plants.

You have an experienced team of specialists at your side to provide active support and bundled know-how. Regular training courses and intensive contact among our employees – even across continents – ensure reliable service in the most diverse areas

Online Support

The comprehensive online information platform supports you in all aspects of our Service & Support at any time and from any location in the world.

You can find Online Support on the Internet at the following address: Internet (http://www.siemens.com/automation/service&support).

Technical Consulting

Support in planning and designing your project: From detailed actual-state analysis, definition of the goal and consultation on product and system questions right through to the creation of the automation solution.

Technical Support

Expert advice on technical questions with a wide range of demand-optimized services for all our products and systems.

You can find Technical Support on the Internet at the following address: Internet (http://www.siemens.com/automation/support-request).

Training

Extend your competitive edge – through practical know-how directly from the manufacturer.

You can find the training courses we offer on the Internet at the following address: Internet (http://www.siemens.com/sitrain).

Engineering Support

Support during project engineering and development with services fine-tuned to your requirements, from configuration through to implementation of an automation project.

Field Service

Our Field Service offers you services for commissioning and maintenance – to ensure that your machines and plants are always available.

Spare parts

In every sector worldwide, plants and systems are required to operate with constantly increasing reliability. We will provide you with the support you need to prevent a standstill from occurring in the first place: with a worldwide network and optimum logistics chains.

Repairs

Downtimes cause problems in the plant as well as unnecessary costs. We can help you to reduce both to a minimum – with our worldwide repair facilities.

Optimization

During the service life of machines and plants, there is often a great potential for increasing productivity or reducing costs.

To help you achieve this potential, we are offering a complete range of optimization services.

Modernization

You can also rely on our support when it comes to modernization – with comprehensive services from the planning phase all the way to commissioning.

Service programs

Our service programs are selected service packages for an automation and drives system or product group. The individual services are coordinated with each other to ensure smooth coverage of the entire life cycle and support optimum use of your products and systems.

The services of a Service Program can be flexibly adapted at any time and used separately.

Examples of service programs:

- Service contracts
- Plant IT Security Services
- Life Cycle Services for Drive Engineering
- SIMATIC PCS 7 Life Cycle Services
- SINUMERIK Manufacturing Excellence
- SIMATIC Remote Support Services

Advantages at a glance:

- Reduced downtimes for increased productivity
- Optimized maintenance costs due to a tailored scope of services
- Costs that can be calculated and therefore planned
- Service reliability due to guaranteed response times and spare part delivery times
- Customer service personnel will be supported and relieved of additional tasks
- Comprehensive service from a single source, fewer interfaces and greater expertise

Contact

At your service locally, around the globe: your partner for consultation, sales, training, service, support, spare parts... for the entire range of products supplied by Industry Automation and Drive Technologies.

You can find your personal contact in our contacts database at: Internet (http://www.siemens.com/automation/partner).

14.7 Establishing a remote connection with TeleService

14.7.1 Basics of working with TeleService

14.7.1.1 Introduction to TeleService

Introduction

TeleService gives your controller telecommunication capability. You can manage, control and monitor distributed plants centrally by means of remote connections.

Scope of functions

TeleService allows you to use the range of TIA portal functions via a telephone network or an Internet connection by establishing a remote connection to a remote system. The online connection allows you to edit a remote system in the usual way with the TIA Portal.

Advantages

Using TeleService offers the following advantages:

- You can easily access even remote sections of plants and include them in a complete system.
- You can offer rapid help and support in the event of faults in a remote system without having to go there yourself.
- You can employ your resources effectively.
- It significantly reduces costs.
- It can significantly reduce plant downtimes.
- It improves the efficiency of your plant.

See also

TeleService functionality (Page 7665)

14.7.1.2 TeleService functionality

TeleService fields of application

TeleService offers the following the fields of application:

- **Access to remote systems (remote maintenance):**
  You can manage, control, and monitor remote systems centrally by means of remote connections.
  This is possible with an S7-300/400 CPU, an S7-1200 CPU and an S7-1500 CPU and a TS Adapter MPI or a TS Adapter IE.
- **Establishing connections from and to remote systems (PG-AS remote link):**
  You can use PRODAVE MPI V5.0 and newer to establish a remote connection to a remote system, and the communications instruction "PG_DIAL" to establish a remote connection from a remote system.
  This is possible with an S7-300/400 CPU and a TS Adapter MPI.
- **Data exchange between systems (AS-AS remote link):**
  The communications instruction "AS_DIAL" allows two automation systems to exchange process data over the telephone network.
  This is possible with an S7-300/400 CPU and a TS Adapter MPI.
- **Sending an text message from a system:**
  An automation system can send a message (SMS) via a GSM wireless modem using the communications instruction "SMS_SEND" ".
  This is possible with an S7-300/400 CPU and a TS Adapter MPI.
- **Sending an email from a system**
  An automation system can also send an email with the following communications instructions and a TS Adapter IE .
  - S7-300/400 CPUs (CPU S7-31x-2PN/DP or CPU 41x-3PN/D) use the instruction "AS_MAIL"
  - S7-1200 CPUs use the instruction "TM_MAIL"
  - S7-1500 CPUs use the instruction "TMAIL_C"

14.7.1.3 Telephone book at TeleService

Introduction

By double clicking the "phone book" folder in the project tree you open the phone book editor displaying the TeleService phone book.

Each version of the TIA Portal has its own "global phone book". If a global phone book from an earlier version of the TIA Portal is found in a more recent version of the TIA Portal, you will be asked once if you want to import this phone book.

This gives you the advantage of being able to import the system data from the previous version into the more recent version of the TIA Portal.

Global phone book properties

The global phone book is used in TeleService to manage specific system data that are required to establish a remote connection.

When you open the phone book for the first time, you will see an empty phone book with all available columns; in all other cases, the last phone book edited is displayed.

You can enter any number of systems in a phone book. Systems contain the data required for establishing a remote connection, for example, the name and location of the device and the phone number to be dialed, along with all country-specific details. With VPN connections, you can enter an IP address or a DNS name instead of the phone number.

The TS adapters used for establishing the connection are distinguished by color, depending on whether a TS Adapter MPI or a TS Adapter IE is used for establishing the connection.

See also

Working with the phone book (Page 7666)

14.7.2 Working with the phone book

14.7.2.1 Basics on working with the phone book

Working with the phone book

You have the following options when working with a phone book:

- Open phone book
- Save phone book
- Import phone book data
- Export phone book data
- Printing the phone book
- Use phone book data to establish a remote connection

You can implement these functions simply and easily by using the buttons in the phone book toolbar.

Note

Access to phone books

The phonebook is user specific in TeleService. However, it is not possible to access the global phone book with more than one instance of the TIA portal at the same time.

See also

Open phone book (Page 7668)
Saving phone book (Page 7669)
Exporting phone book data (Page 7671)
Printing the phone book (Page 7672)
Structure of the phone book (Page 7667)

14.7.2.2 Structure of the phone book

Introduction

A global phone book in TeleService is used to manage the data you require for establishing a remote connection. Once you have created the connection data and saved it in the phone book, you can access it each time you want to establish a remote connection.

Structure of the phone book

The integrated global phone book in TeleService contains the following columns:

Column name	Meaning
System name	Enter a name for your system.
Adapter type	Select the TS Adapter type used in the drop-down list: TS Adapter MPI or TS Adapter IE.
Connection type	Select the required connection type: Dial-up connection or VPN connection.
Area code	Enter the required area code. This column is only active with dial-up connections. This column cannot be edited for VPN connections.
Phone number/remote address	Enter the required connection data for establishing the remote connection. For dial-up connections, this can be a phone number, and for VPN connections a DNS name or an IP address.
Fingerprint	Enter the corresponding fingerprint for establishing a VPN connection to the TS Adapter IE Advanced.
Country	Enter the country code. This column is only active with dial-up connections. This column cannot be edited for VPN connections.
User name	Enter the user name you have logged on under.
Password	Enter the password for this user name.
Group	Enter the group if you have carried out grouping.
Company	Enter the company to be contacted.
Department	Enter the relevant department.
Street	Enter the street.
Town/City	Enter the town or city to which the remote connection is to be established.
Comment	Enter a comment if required.

Showing or hiding columns

You can show or hide the individual columns. To do so, select the required column header and open the shortcut menu with the right mouse button.

14.7.2.3 Symbols in the phone book

Meaning of the TeleService icons

The following table shows the meaning of the TeleService icons:

Symbol	Meaning
	Open the global phone book
	Imports a phone book
	Exports a phone book
	Establishes a remote connection
	Terminates the active remote connection
	Establishes a remote connection or terminates it
	Displays the connection to a TS Adapter IE in the phone book
	Displays the connection to a TS Adapter MPI in the phone book
	Adds a new row to the phone book
	Inserts a new row in the phone book

14.7.2.4 Manage phone book

Open phone book

Opening phone books

To open the phone book, follow these steps:

1. In the project tree, double-click on the "Phone book" folder under "Online access" > "TeleService".
2. The phone book opens so that you can enter or edit the required system data.

Inserting rows in the phone book

Inserting rows in the phone book

To insert a new row in the phone book, follow these steps:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button in the toolbar.

Result

A new row is inserted in the phone book above the selected row.

Showing and hiding columns in the phone book

Showing and hiding columns

To show or hide columns in the phone book, follow these steps:

1. Click a column header.
2. In the shortcut menu, select the "Show/hide columns" command.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

Result

The respective columns are shown or hidden when displaying the phone book.

Saving phone book

Saving phone books

When you exit the phone book editor or the TIA Portal, you are asked whether you want to save the global phone book.

Click "Yes" to save the phone book.

Importing phone book data

Introduction

It is possible to import the phone book data from an external file or from an older version of the TIA Portal.

Requirement

You have already created an import-capable phone book file.

You have already created a phone book with an older version of the TIA Portal.

Importing phone book data from a phone book file

To import phone book data from a phone book file, follow these steps:

1. Open the "TeleService" folder under "Online access" in the project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Import" button in the toolbar.
4. Confirm the prompt asking if you want to save the current state of the phone book with "Yes", if applicable, and specify the location for storing the phone book in the dialog that follows.
5. If you do not want to save the current state of the phone book, answer the prompt with "No". In the subsequent dialog, select the phone book file to which the current phone book should be stored.
6. Close the dialog box with "OK".

Result

The imported phone book data is displayed in the global phone book.

Note

Defining dialing rules

Dialing rules must be defined before the "area code" and "country" columns in the imported phone book can be edited for dial-up connections.

To define dialing rules, follow the link in the history.

Importing phone book data from an older version of the TIA Portal

To import phone book data from an older version of the TIA portal, follow these steps:

1. Open the "TeleService" folder under "Online access" in the project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Import" button in the toolbar.
4. Confirm the prompt asking if you want to save the current state of the phone book with "Yes", if applicable, and specify the location for storing the phone book in the dialog that follows.
5. If you do not want to save the current state of the phone book, answer the prompt with "No".

6. Enter the following path in the dialog that follows: "%appdata%\siemens\automation \TeleService\GlobalTeleServicePhoneBook.tel".
7. Close the dialog box with "OK".

Result

The phone book data imported from the older version of the TIA Portal is displayed in the global phone book.

Note

Defining dialing rules

Dialing rules must be defined before the "area code" and "country" columns in the imported phone book can be edited for dial-up connections.

To define dialing rules, follow the link in the history.

See also

Defining dialing rules (Page 7672)

Exporting phone book data

Introduction

It is possible to export phone book data to an external file.

Requirement

You have already created a phone book under TeleService with the corresponding system data.

Procedure

Proceed as follows to export the phone book data:

1. Open the "TeleService" folder in project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Export" button in the toolbar.
4. In the next dialog box, select where the current phone book is to be exported.
5. Close the dialog box with "OK".

Result

The exported phone book data are saved in the specified export file.

Printing the phone book

Printing phone books

You can print out all or some of the data in a phone book.

Proceed as follows:

1. Open the phone book.
2. Select the **Phone book > Print** menu command or click on the appropriate button in the toolbar. The "Print" dialog will open.
3. Specify whether you wish to print the complete phone book or just part of it and set all other options.
4. Start the print job with "OK".

Result

The phone book data is printed on the default printer. If the printout is more than one page long, an identifier is printed after the page number at the bottom right corner of the page to indicate that there is another page. The last page does not have this symbol, indicating that no more pages are to follow.

Defining dialing rules

Procedure

To define location-specific dialing rules, follow these steps:

1. Open the "Online access" folder in the TIA Portal and select the "TeleService" folder.
2. Use the shortcut menu to access the "TeleService" properties.
3. In the dialog that follows, open the "Advanced settings" tab and activate the "Use dialing rules" option.
4. Click the "Adapt" button.
5. In the dialog that follows, enter the required location information for connection establishment.
   - For new locations, enter the country or region and the corresponding area code.
   - If required, set the correct destination code and pre-dial line for local/long-distance calls.
   - Select your chosen dialing mode for the location.

6. Exit the dialog by clicking "OK".
7. In the next dialog, select the location from which you wish to dial, and click "OK".
8. In the TeleService dialog that follows, check that the correct location appears under "Location".
9. Confirm your entries by clicking "OK".

Note

Modem operation on a local loop or extension

You do not need to specify a pre-dial line if you operate your modem on a local loop (main telephone line). The fields for the pre-dial lines for local calls and long-distance calls must be empty.
If you operate your modem on an extension, you should enter the pre-dial lines which need to be called to access a local loop.

Example of the use of dialing rules

The following example illustrates the use of location-specific dialing rules for the city of Karlsruhe in Germany.

- In the TeleService phone book, you enter the phone number "1234567", without the area code and without the country code.
  Result:
  The phone number "1234567" is always dialed, irrespective of the location of the caller.
- In addition to the phone number "1234567", you enter the location-specific dial parameters for the Karlsruhe area code "0721" and the country code "+49" for Germany in the TeleService phone book.
  Result:
  "1234567" is dialed from Karlsruhe.
  "07211234567" is dialed from other towns in Germany.
  "00497211234567" is dialed from other countries.

Note

Display in the phone book

The "area code" and "country" columns in the TeleService phone book can only be edited once you have defined the dialing rules as detailed above.

14.7.3 Remote connections as dial-up connections

14.7.3.1 Basics for establishing a dial-up connection

Using a TS Adapter for dial-up connections

A TS Adapter is needed for establishing a remote dial-up connection using TeleService.

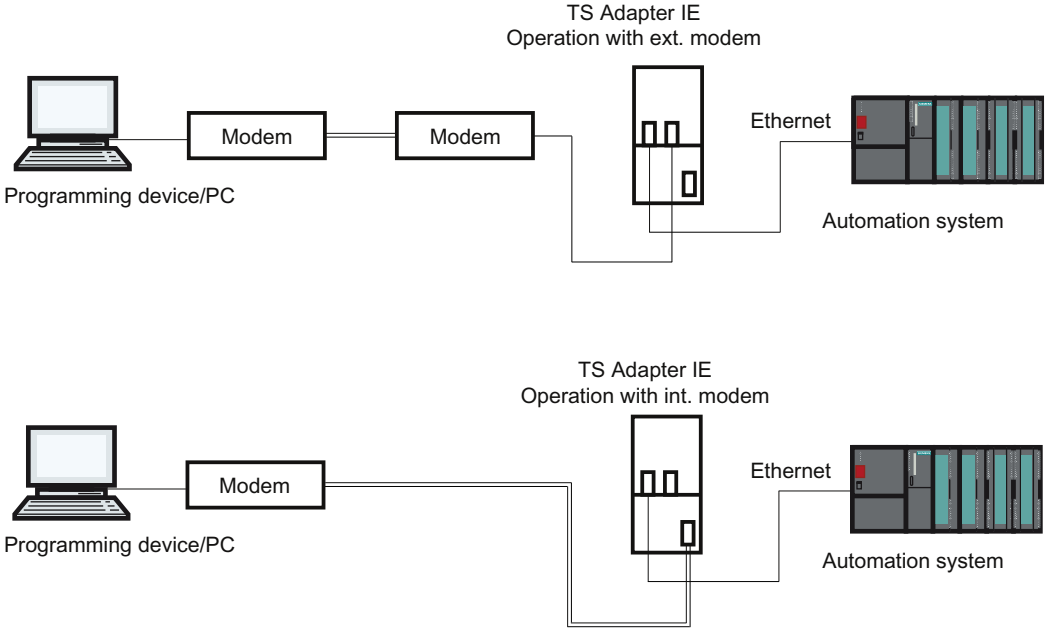
The TS Adapter is used to prepare an automation system for use with TeleService by connecting it to a telephone network via a modem. The TS Adapter has an integrated parameter memory in which a parameter set for TeleService operation is stored.

With the function "export adapter parameter", different parameter sets can be saved in external files, and reloaded in the TS Adapter via the function "import adapter parameter".

Establishing a remote dial-up connection

You can choose between a number of different TS Adapters, each of which offers a different functionality and different connection options.

The figure below shows two possible configurations for establishing a dial-up connection to a system with the TS Adapter IE.



Overview of possible TS Adapter:

The TS Adapter comes in the following versions:

- TS Adapter II (also designated "TS Adapter MPI")
- TS Adapter IE Standard (also designated "TS Adapter IE")
- TS Adapter IE Basic (also designated "TS Adapter IE")

Designation "TS Adapter"

In the following pages, the designation "TS Adapter" stands for all versions. The relevant product designation is listed beside information which only applies to a specific version, for example, "TS Adapter II" or "TS Adapter IE Standard".

Note

For more detailed information on your TS Adapter, please refer to the documentation supplied with the TS Adapter.

See also

Short description of the TS adapter MPI (Page 7683)

Short description of the TS adapter IE (Page 7690)

Exporting adapter parameters (Page 7689)

Importing adapter parameters (Page 7690)

14.7.3.2 Telephone networks and modems

Supported telephone networks and modems

Telephone networks which can be used

TeleService can be used with digital networks (ISDN), analog networks and wireless networks (with GSM technology). This version supports a remote connection to a TS Adapter.

Modem support

TeleService has been implemented to be independent of the modem. This means that all standard modems which can be installed in the Windows Control Panel and are visible as modems can also be used by TeleService.

The choice of modem type is determined primarily by the existing hardware of the programming device/PC and the telephone network to be used.

The following modem types/media are supported:

- Modems (external modems at the COM interface, internal modems and PCMCIA cards)
- External ISDN adapter at the COM interface or USB interface
- Internal ISDN adapter with virtual COM interface (for example AVM CAPI port)
- External ISDN modems (ISDN adapter with integrated analog modem functionality) at the COM interface or USB interface
- Radio network modems with GSM technology, PCMCIA adapter card or data cable and mobile phone

Gateways

Gateways between the various telephone networks are in principle possible. Remote connections from an ISDN adapter to an analog modem and vice versa only function with special ISDN telephone adapters.

Performance in telephone networks

The data throughput of a remote connection depends on the modem and telephone network used and the quality of the telephone line.

Installing the local modem

Introduction

If you have already installed a modem for data transfer in your operating system, this modem can also be used for TeleService.

If a modem has not yet been installed for your operating system, one must be installed before you can establish a remote connection with TeleService.

Procedure

Proceed as follows:

1. Make sure your programming device/PC and the modem are switched off.
2. Physically connect the external modem to a COM or USB interface on your programming device/PC. You can also install an internal modem or a PCMCIA card in accordance with the manufacturer's specifications.
3. Now switch on the modem and then the programming device or the PC.

Result

Plug-and-play modems are recognized and installed automatically by the operating system. Dialogs will take you through the installation procedure.

Note

Modems without plug-and-play

If your modem is not recognized automatically when switched on, you will have to install it yourself using the Control Panel.

Please refer to the information in the documentation supplied with your modem.

Connecting and configuring the remote modem

Introduction

A modem must also be connected to the remote system before you can work with TeleService. This modem is designated the "remote modem".

Configuring the remote modem

The modem receives all parameters required for operation from the TS Adapter connected. These include data for initializing the modem and settings for serial transmission between the TS Adapter and the modem.

The data required for the remote modem is specified during the configuration of the TS Adapter.

The modem may be internal or external depending on the TS Adapter used.

How to connect a TS Adapter with an internal modem

1. Switch off the TS Adapter.
2. Connect the TS Adapter to the automation system.
3. Connect the TS Adapter to the telephone line.
4. Switch on the TS Adapter.

How to connect a TS Adapter with an external modem

1. Switch off the modem.
2. Connect the TS Adapter to the automation system.
3. Connect the TS Adapter to the modem using a modem cable.
4. Connect the modem to the telephone line.
5. Switch on the modem.
6. Switch on the TS Adapter.

Note

Please note the following information on configuring the remote modem:

- The default parameters for the modem and the serial port set in the TS Adapter should in most cases ensure successful operation; changes in the parameter assignment will only be needed in rare cases.
- You need only change the parameter assignment of the TS Adapter if a modem connection is not established or if factory settings are to be adapted or optimized.
- TS Adapter parameter assignment can be changed via either a direct or a remote connection.

14.7.3.3 Access protection for dial-up connections

Access protection information

Introduction

When you assign the parameters for your TS adapter, you can restrict access to the parameters of the TS adapter and access to remote systems.

Scope of access protection

Access protection only exists for remote connections; TS adapter parameter assignment can be accessed at any time in direct connection mode.

Access protection also exists in direct connection for the TS adapter IE.

Access protection information

The TS Adapter MPI is not delivered with access protection activated. There is a default password for the TS adapter IE.

The first user who assigns the parameters for this adapter can therefore activate access protection by defining the password for a user and/or a callback number.

This is a multi-level access protection with several users, each with or without administrator rights. For the TS adapter MPI there is only one adminstrator and no more than two users.

For a modem connection, only an administrator can define the two users and change and, if necessary, delete their settings. Those logged in as users can only change their own passwords and their own callback numbers. However, with the TS adapter MP you can access the process of assigning parameters of the TS adapter in direct connection, without restriction.

Advantages

Access protection offers the following advantages:

- Unauthorized access by persons outside the system is almost impossible.
- The plant operator bears most of the telephone costs.

TeleService callback options

Callback variants

The costs of a telephone connection are normally borne by the caller who establishes the dial-up connection.

TeleService can, however, be used so that after a short initial connection the modem connection is established again in the opposite direction, in other words initiated by the TS Adapter (callback). In this case, the plant operator bears the costs of the callback.

There are two callback variants in TeleService:

1. Callback to a number specified during connection establishment.
2. Callback to a number stored on the TS Adapter.

Levels of protection

Introduction

You can set up one of two possible levels of access protection for TeleService access to the TS Adapter. Different options are available with each protection level.

Access protection options

Access protection level 1:

The TS Adapter is protected by the user name and password. You can access the TS Adapter via any telephone line and specify any callback number during connection establishment.

Access protection level 2:

The TS Adapter is protected by the user name, password, and the callback number. You can only access the TS Adapter from one telephone connection per user.

The table below sets out the above conditions for the various protection levels:

Level of access protection	Administrator/User password	Callback number
1	enter	do not enter
2	enter	enter

Logging on to TS Adapter

When you log on to the TS Adapter and after you have set up access protection, enter your user name, the corresponding password and, if desired, a callback number:

Level of access protection	Administrator/User password	Callback number
1	enter	do not enter or enter any callback number
2	enter	do not enter

If you have entered a callback number during connection establishment (access protection level 1) or stored a callback number in the TS Adapter (access protection level 2), the modem connection will be terminated and the TS Adapter will call back the given number.

Setting up access protection and callback number for the TS adapter

Introduction

During the parameter assignment for the TS adapter MPI in TeleService, you can set up access protection and a callback number for the parameter assignment of the adapter and connection to the remote system. The following describes the parameter assignment for a TS adapter MPI. The parameter assignment of a TS adapter IE is carried out in analog. The specific method is described in the web help of this adapter.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Accessible nodes".

Procedure

To set up access protection for the TS adapter, proceed as follows:

1. Click on the command "Assign TS Adapter MPI parameters" in the project tree.
2. Open the "Access security" tab.
3. Enter a password for your user name and/or number that you want the modem to call back following logon.
   - If you are an administrator, you can change all the settings for administrators and users, and delete or create users.
   - If you are logged on as a user, you can only change your own settings (password and callback number).
4. Confirm all entries before exiting the dialog with "OK".
5. Click the "Yes" button to confirm the following query.

Result

The parameter assignment for the access protection and the callback number is saved in the non-volatile memory of the TS adapter MPI.

Note

Important points to note when setting up access protection:

- The settings in the "Modem" tab must correspond to the conditions at the plant if callback functionality is to be guaranteed.
- Entering an incorrect callback number in the role of "ADMIN" user will mean you are no longer able to access the TS Adapter MPI over a remote connection!
- Test the callback number before you enter it as the "ADMIN" user by calling the given callback number during connection establishment (access protection level 1).

Complete a callback in TeleService

Callback options

Two different callback variants can be set up in TeleService.

The following callback options are available:

- Callback to a number specified during connection establishment.
- Callback to a number stored on the TS Adapter

Callback to a number specified during connection establishment

1. In the project tree of the TIA Portal, click the "Online access" folder.
2. Click on the "TeleService" folder it contains.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. In the "Adapter type" drop-down list, select the adapter type used.
5. Select the "dial-up connection" under "Connection type" if it is not already selected.
6. Select the modem you are using under "Local Settings".
7. Enter the phone number to be dialed in the appropriate box or open the phone book by clicking on the button behind it and take the desired phone number from the phone book.
8. Enter your user name and associated password of the TS adapter.
9. If you want a "Connection setup with callback", select the appropriate option button.

10. Click the "Connect" button to establish the required remote connection. This button only becomes active when you have entered all the parameters needed to establish a remote connection. Any remote connection is displayed under "Status".
11. Enter the desired callback number in the dialog that follows.

Result

The remote connection to the desired system is made with callback.

The connected system is shown with the corresponding icon in the project tree.

Note

This procedure is useful if the costs of the modem connection are to be borne by the plant operator and if the actual callback number is not fixed, i.e. callback is not always to the same receiver. It is particularly useful for mobile users.

Callback to a number stored on the TS Adapter

1. Assign the parameters for the desired callback number in the TS adapter.
2. Establish a connection to the TS adapter as described above, and observe the following features:
   - Enter the user name and password for which the callback number parameters are assigned in the TS adapter.
   - The check box "Establish a connection with callback" does not have to be selected, since the callback number is already known by the TS adapter.

Result

Callback to a number stored on the TS adapter has been established. If a remote connection is established, the callback occurs from the remote system.

Note

This procedure offers the highest level of access protection. However, it does pose a risk: if the callback number stored on the TS Adapter is not correct, it will no longer be possible to access the TS Adapter over a modem connection. The device can in such a case only be put back into operation by changing the parameter settings on site.

14.7.3.4 TS adapter MPI

Short description of the TS adapter MPI

TS Adapter MPI:

The designation "TS Adapter MPI" is a collective term for allTS Adapter with an MPI/DP interface.

The TS Adapter MPI comes in the following versions:

- As TS Adapter I (parameters cannot be assigned via the TIA Portal)
- as TS Adapter II

The table below provides a short description of the functionalities. For detailed information on your TS Adapter, please refer to the documentation supplied with your TS Adapter.

TS Adapter II:	
Direct connection	Connection via the Universal Serial Bus (USB). The firmware can be replaced. The modem is integrated or can also be connected as external modem. The TS Adapter II switches automatically between the modems. As long as no external modem is connected, the adapter will use the internal modem.
There are two variants	• With internal analog modem. An external modem can also be connected to the RS232 port. • With internal ISDN adapter. An external modem can also be connected to the RS232 port.

Use of the designation "TS Adapter"

For TeleService the designation "TS Adapter" is the generalization for all versions. The relevant product designation is listed beside information which only applies to a specific version of a TS Adapter, for example, "TS Adapter II", "TS Adapter IE Standard" or "TS Adapter IE Basic".

How the TS adapter MPI works

How the TS Adapter MPI Works

In line with the configuration, the TS Adapter MPI connects the serial port or USB port of your programming device/personal computer (direct connection) or the serial port of a modem (modem connection) to the MPI/PROFIBUS network of your automation system.

The TS Adapter MPI has a non-volatile memory. Parameters for the following functions are stored in this memory:

- The MPI/PROFIBUS network (network parameters)
- The mode of the modem used

- The serial port to the modem
- Access protection

Default parameter assignment

The TS Adapter comes with default parameter assignment. The parameters can be set and saved to the non-volatile memory of the TS adapter in a parameter assignment session.

When "Direct connection" is configured, the TS Adapter will only use the network parameters for access to the MPI/PROFIBUS network.

In the "Modem connection" configuration, all the parameters stored on the TS Adapter will be activated.

Note

For more detailed information on the configuration of your TS Adapter, please refer to the documentation supplied with it.

Operating a TS adapter MPI in direct connection mode

Direct connection with TS Adapter MPI

The direct connection is used to assign the parameters of the TS Adapter MPI. The same configuration also allows you to go online in the TIA Portal and thereby check the assigned MPI/PROFIBUS parameters for bus compatibility. This means that (as with a PC adapter) SIMATIC S7/C7 systems can be accessed via the MPI/DP interface without an MPI/PROFIBUS module occupying a slot for a programming device/PC.

Access protection for the TS Adapter is not active in direct connection configuration. This means that the parameter assignment of the TS Adapter can be changed without any problems, for example by importing adapter parameters.

Note

Display of the TS Adapter MPI in the TIA Portal

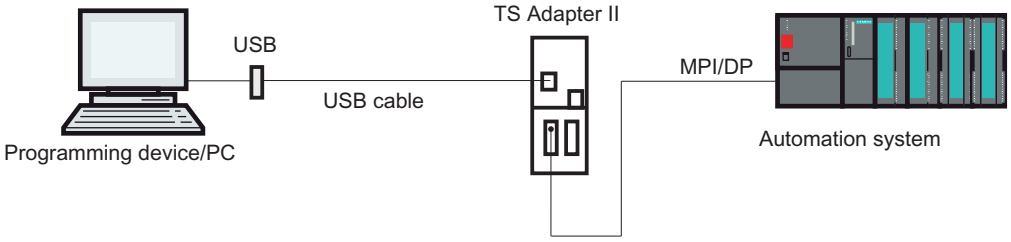
As soon as you have connected a TS Adapter MPI to the programming device/PC via the USB interface, the "TS Adapter" folder is displayed in the project tree in the TIA Portal.

When you open the folder, you can assign the parameters of the connected TS Adapter MPI via the following dialog.

Establishing the direct connection for TS Adapter MPI

Direct connection mode means there is a direct connection via the TS Adapter MPI between the programming device/personal computer on which TeleService is installed and the automation system. No modem is required.

The figure below shows the configuration of the TS Adapter MPI with a direct connection.



Operating a TS adapter MPI in modem connection mode

Introduction to the modem connection with TS Adapter MPI

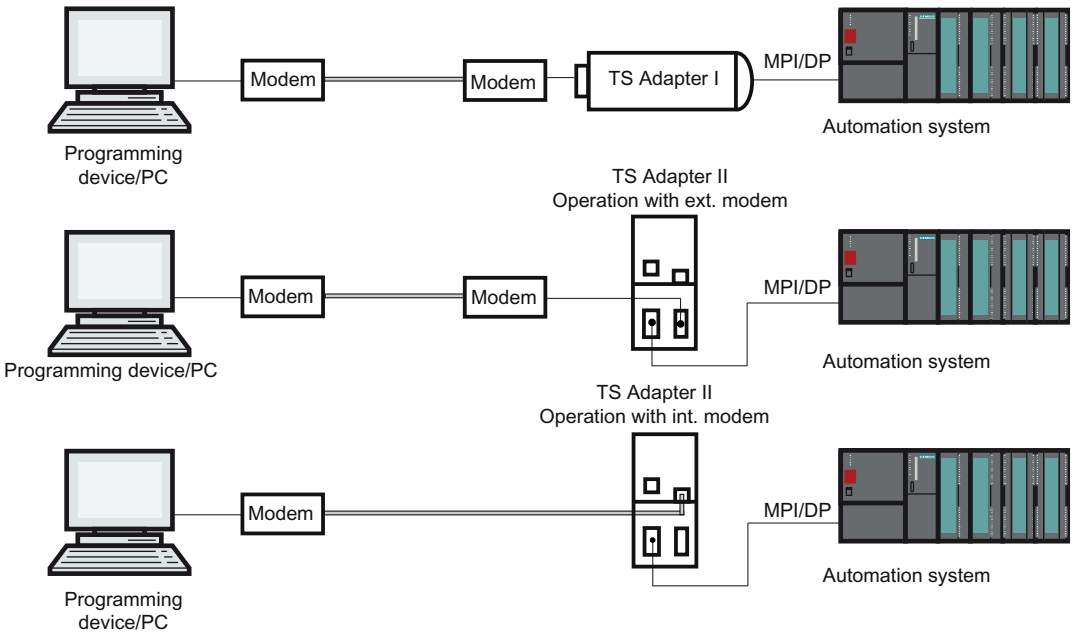
This configuration allows you to dial into a remote system. To do this, you establish a remote connection to a remote system using TeleService on a telephone network. You can then work with the selected system as usual, with the TIA Portal, over the established modem connection.

Establishing a modem connection with TS Adapter MPI

This connection between the programming device or PC on which TeleService is installed and the automation system in which the TS Adapter MPI is inserted in the MPI/DP interface is made through a modem route.

The configuration therefore includes the programming device or PC via the telephone network and the TS Adapter MPI on the MPI/DP interface of the automation system.

The figure below shows the structure of the modem connection.



Note

Parallel operation between direct and modem connection

The TS Adapter II has two connections for communication with PG/PC, both of which can be connected at the same time. At the same time, connect the USB interface with the PG/PC and the modem interface with the telephone network.

In this configuration you can either use the direct or the modem connection.

A parallel operation is **not** possible!

TS adapter MPI configuration options

Useful information on configuring the TS Adapter MPI

The TS Adapter MPI can be configured in both direct connection mode and via an existing remote connection.

The following parameter assignment options are available:

- Reconfiguration (Page 7687)
- Restoring default parameter assignment (Page 7688)
- Importing adapter parameters (Page 7690)
- Exporting adapter parameters (Page 7689)
- Setting up access protection (Page 7678)

Parameter assignment

Configure your TS Adapter in accordance with the documentation supplied with the TS Adapter. It will detail the exact procedure for parameter assignment.

Note

Please note the following when configuring the TS Adapter MPI

- If you change the current parameter settings when there is an established remote connection, there is a risk it will not subsequently be possible to establish a modem connection with the modified parameters. The TS Adapter MPI can in this case only be configured in direct connection mode.
- This means that either parameter assignment must be carried out with a programming device/personal computer at the plant location or the TS Adapter MPI must be brought to the location of the local programming device/personal computer in order to be configured.

Positive acknowledgement

During parameter assignment, the data is written to the non-volatile memory of the TS Adapter MPI. The parameter assignment process is not acknowledged positively until all precautions have been taken to ensure that parameter changes have been carried out correctly and will thus survive a power failure.

Changes become effective for the TS Adapter MPI as follows:

- The serial parameters, the modem parameters and the parameters for access protection are activated once the remote connection has been terminated.
- The modified network parameters are activated immediately.

Configuring TS adapter MPI

Introduction

The TS Adapter MPI can be configured in both direct connection mode and via an existing remote connection in modem connection mode.

The following describes the method for assigning parameters.

Requirement

A TS Adapter MPI is connected to your computer and the folder "TS Adapter" is displayed in the project tree under "Online access".

Procedure

To assign the parameters for the TS Adapter MPI im Direktanschluss please proceed as follows:

1. Double-click the "Online access" folder in the project tree.
2. Open the required folder:
   - The "TS Adapter" folder for a direct connection.
   - For an existing remote connection, the "TeleService" folder followed by the folder with the required system name.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Set the required parameters in the individual tabs of the dialog.
5. Confirm your entries with "OK".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter MPI. Parameter assignment is then complete.

Restoring default parameter assignment for TS adapter MPI

Introduction

You can restore the default, factory state parameters of the TS Adapter MPI.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

Proceed as follows to restore default parameters for the TS Adapter MPI:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Reset" button under "General".
5. Confirm your entries with "OK".

Result

The TS Adapter MPI default parameters set on delivery are restored.

See also

TS adapter MPI configuration options (Page 7684)

Exporting adapter parameters

Introduction

You can export the configuration of a TS Adapter MPI to an external file. The configuration saved in this file can be imported in turn into any number of TS Adapter MPI.

This can for example be useful if you want to assign identical parameters to multipleTS Adapter MPI or if you want save, document or distribute the parameter set.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

To export the adapter parameters of a TS Adapter MPI:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Export" button.
5. A window will open in which you can select the file to which you wish to export the configuration of the TS Adapter MPI.
6. Confirm with "Save".

Result

The parameters of the TS Adapter MPI are saved in the specified file (*.tap). The export of the adapter parameters is now complete.

Importing adapter parameters

Introduction

You can import the configuration of a TS Adapter MPI from a previously created export file (*.tap).

The configuration saved in this file can be imported into any number of TS Adapter. This can for example be useful if you want to assign identical parameters to multiple TS Adapter MPI.

You can import parameters locally in direct connection mode or via an existing remote connection in modem connection mode.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

To import the adapter parameters of a TS Adapter MPI:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Import" button.
5. A dialog will open in which you can select the file to which you wish to import the configuration of the TS Adapter MPI.
6. Confirm the next dialog with "Yes".

Result

The parameters selected are saved in the non-volatile memory of the TS Adapter MPI. Adapter parameter import is then complete.

14.7.3.5 TS adapter IE

Short description of the TS adapter IE

TS Adapter IE

The designation "TS Adapter IE" is a collective term for allTS Adapter with an Ethernet port.

The TS Adapter IE comes in the following versions:

- as TS Adapter IE Standard
- as TS Adapter IE Basic

The tables below provide a short description of the functionalities. For detailed information on your TS Adapter, please refer to the documentation supplied with it.

TS Adapter IE Standard:
Direct connection by means of Industrial Ethernet (IE). Firmware update possible. Modem integrated or external. The TS Adapter IE cannot automatically switch between modems like the TS Adapter II. Parameters are assigned via a Web interface.
There are 2 variants:
• With internal analog modem. An external modem can also be connected to the RS232 port.
• With internal ISDN adapter. An external modem can also be connected to the RS232 port.

TS Adapter IE Basic:
Direct connection by means of Industrial Ethernet (IE). Firmware update possible. Plug-in modules. Parameters are assigned via a Web interface.
There are 4 variants:
• TS Adapter IE Basic MODEM: Basic device TS Adapter IE Basic with TS Module MODEM for operation on the analog telephone network.
• TS Adapter IE Basic ISDN: Basic device TS Adapter IE Basic with TS Module ISDN for operation on ISDN telephone systems.
• TS Adapter IE Basic GSM: Basic device TS Adapter IE Basic with TS Module GSM for operation on the GSM radio network.
• TS Adapter IE Basic RS232: Basic device TS Adapter IE Basic with TS Module RS232 for connecting an external modem.

Use of the designation "TS Adapter"

"TS Adapter" is used in the TeleService online help as a general designation for all versions. The relevant product designation is listed beside information which only applies to a specific version of a TS Adapter, e.g. "TS Adapter I", "TS Adapter II", "TS Adapter IE Standard" or "TS Adapter IE Basic".

How the TS adapter IE works

How the TS Adapter IE works

The TS Adapter IE connects the telephone network or the serial port of a modem with the Industrial Ethernet of your automation system.

The TS Adapter IE has a non-volatile memory. Parameters for the following functions are stored in this memory:

- The mode of the modem used
- The serial port to the modem
- Access protection

Default parameter assignment

The TS Adapter IE comes with default parameter assignment. The parameters can be set and saved to the non-volatile memory of the TS adapter in a parameter assignment session.

Note

For more detailed information on the configuration of your TS Adapter, please refer to the documentation supplied with it.

Connection Types

Connection types of the TS Adapter IE Basic

The following diagrams show the connection types possible with the TS Adapter IE Basic.

Direct connection

In the direct connection to the PG/PC, you can set the TS Adapter IE Basic through Ethernet.

Note

The operation of the TS Adapter IE Basic without a TS module is not permitted.

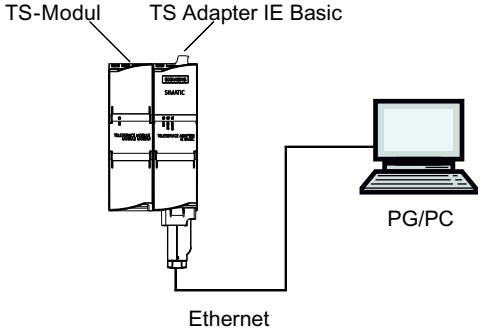


Figure 14-1 Direct connection

Connection to the telephone network

In order to have a direct connection to the telephone network, you must connect the TS Adapter IE Basic together with one of the following TS modules:

- TS Module Modem
- TS Module ISDN

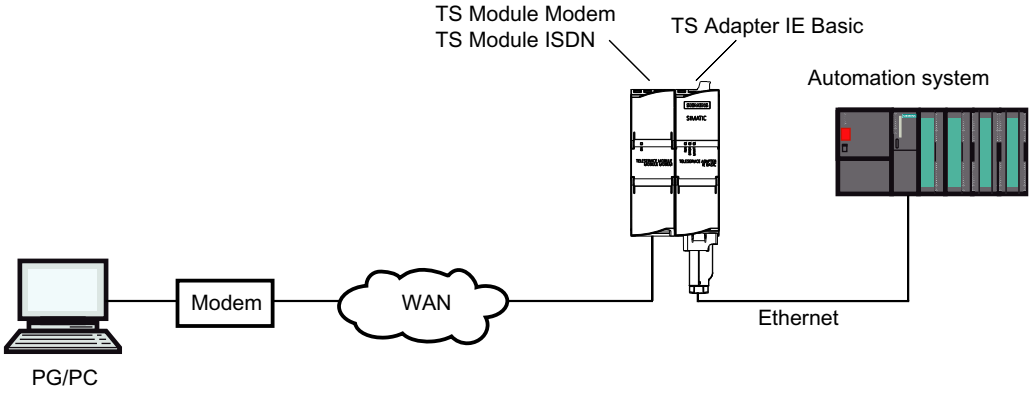


Figure 14-2 Direct connection to the telephone network

More information about the TS modules can be found in the *TS Adapter modular* manual.

Connection to the GSM network

In order to connect to the GSM network, you must operate the TS Adapter IE Basic together with this TS module:

- TS Module GSM

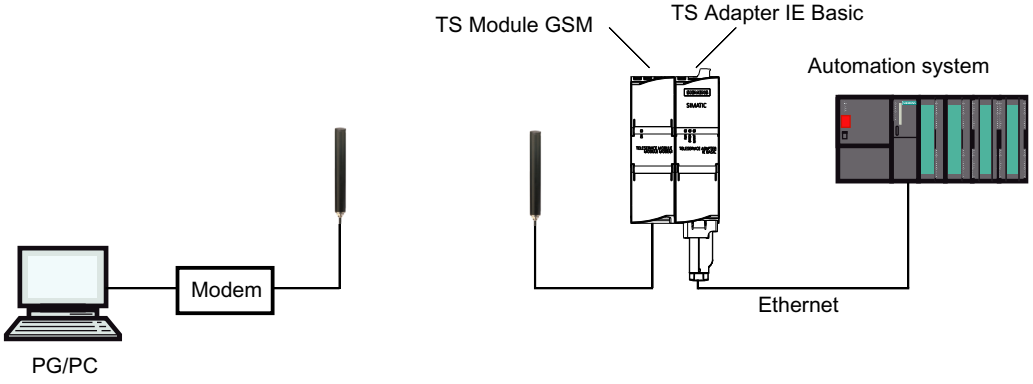


Figure 14-3 Connection to the GSM network

More information about the TS modules can be found in the *TS Adapter modular* manual.

Connection to the telephone network through an external modem

For the connection to an external modem, you must operate the TS Adapter IE Basic together with this module:

- TS Module RS232

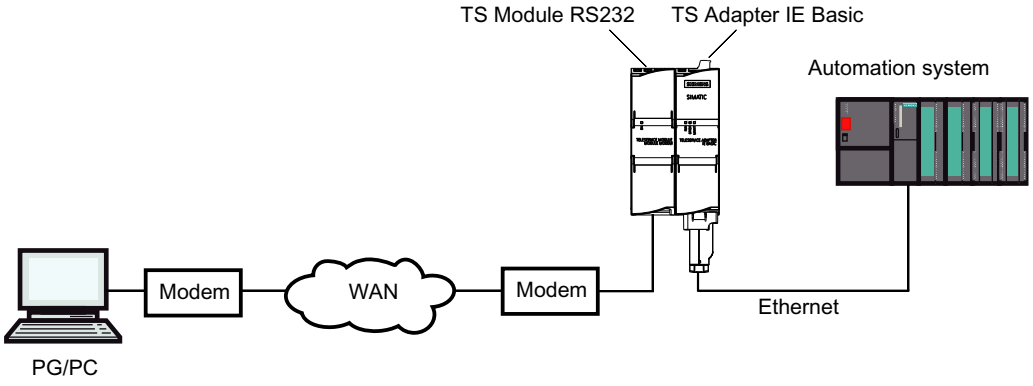


Figure 14-4 Connection to an external modem

More information about the TS modules can be found in the *TS Adapter modular* manual.

TS Adapter IE parameter assignment options

Useful information for configuring the TS Adapter IE

The TS Adapter IE is configured via a Web interface.

Web help associated with the parameter assignment interface is made available for the configuring the TS Adapter IE.

The following parameter assignment options are available:

- Reconfiguration
- Restoring default parameter assignment
- Importing adapter parameters
- Exporting adapter parameters

Note

Parameter assignment

Configure your TS Adapter in accordance with the documentation supplied with the TS Adapter. It will detail the exact procedure for parameter assignments.

Parameter assignment for TS Adapter IE

Introduction

The TS Adapter IE can be configured in both direct connection mode and via an existing remote connection in modem connection mode.

Both parameter assignment options are described below.

Specific details for assigning parameters of the TS Adapter IE can be obtained from the TS Adapter IE documentation.

Parameter assignment of the TS Adapter IE in direct connection

Requirement

There is a LAN connection to your TS Adapter IE .

The TS Adapter IE Basic is connected to the power supply.

Procedure

To assign the parameters for the TS Adapter IE , proceed as follows:

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Double-click on the Ethernet port of your computer.
3. Double-click on the "Display accessible nodes" command. The TS Adapter IE is then displayed.
4. Double-click on the <TS Adapter IE> folder and then on "Online and diagnostic", and assign the desired IP address to the TS Adapter IE in the following dialogs. Please note that the IP address of the PG/PC interface card is located in the same subnet as the IP address that you issue for the TS Adapter IE.
5. Update the view in the project tree for the "Accessible nodes", so that the TS Adapter IE is displayed with the newly allocated IP address.
6. Open the folder <TS Adapter IE> in the device list.
7. Double-click the command "Assign TS Adapter IE parameters". The allocated web interface opens for assigning the parameters of the TS Adapter IE.
8. Complete the "logon" for the web interface.
9. Set the required parameters in the individual tabs of the dialog.
10. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE. Parameter assignment is then complete.

Parameter assignment of the TS Adapter IE using a remote connection

Requirement

There is an established remote connection to a TS Adapter IE .

Procedure

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Open the "TeleService" folder followed by the required system folder.
3. Double-click the command "Assign TS Adapter IE parameters". The associated web interface for assignment of the TS Adapter IE parameters opens. The "logon" for the web interface takes place automatically with the login data of the remote connection.
4. Set the required parameters in the individual tabs of the dialog.
5. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE. Parameter assignment is then complete.

14.7.3.6 Establishing a dial-up connection to a remote system

Establishing a dial-up connection

Introduction to establishing a remote dial-up connection

A dial-up connection is established when you use TeleService to dial into a remote system via a telephone network. The programming device/personal computer is connected to the telephone network with TeleService via a modem. At the other end, the automation system is connected to the telephone line via a configured TS Adapter and a modem.

Requirements

A local modem is installed and configured.

The TS Adapter is located in the remote system.

A remote modem is installed and parameters have been assigned.

Proceed as follows:

1. In the project tree of the TIA Portal, click the "Online access" folder.
2. Click on the "TeleService" folder it contains.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. In the "Adapter type" drop-down list, select the adapter type used.
5. Under "Connection type", select "Dial-up connection".
6. Select the modem you are using under "Local settings".
7. Enter the phone number to be dialed in the appropriate box or open the phone book by clicking on the button behind it and take the desired phone number from the phone book.
8. Enter your user name and the corresponding password.
9. If you want a "Connection setup with callback", select the appropriate option button.
10. Click the "Connect" button to establish the required remote connection. This button only becomes active when you have entered all the parameters needed to establish a remote connection. Any remote connection is displayed under "Status".

Result

The dial-up connection to the desired system is established.

The dialog closes as soon as the dial-up connection is established. The following message appears in the status bar of the TIA Portal: "Remote connection is up". You can now use the remote connection with the TIA Portal and communicate with the automation system.

Connection cannot be established

If the connection cannot be established, try to find the cause using the "Notes on troubleshooting".

Terminating the connection

Once you have finished editing the remote system, exit the remote connection in the project tree by double-clicking on the entry "Establish/disconnect remote connection".

By exiting the TIA Portal, you are also terminating the remote connection.

Terminating a dial-up connection

Terminating an active dial-up connection

Note

Terminating the dial-up connection

You should go offline in the TIA Portal before you terminate the remote connection.

Proceed as follows:

1. Double-click the "Set up/close remote connection" entry in the TIA Portal.
2. Confirm the prompt in the dialog that follows with "Yes".

Result

The connection is terminated.

14.7.4 Remote VPN connections

14.7.4.1 Basics for establishing a VPN connection

Using a TS Adapter IE Advanced for VPN connections

A TS Adapter IE Advanced is required to establish a VPN connection using TeleService.

VPN definition

VPN stands for "Virtual Private Network". This is a private network of computers based on a public network infrastructure.

VPN is used to create a connection that is as secure as possible between devices in a private network at different locations and the gateway of another network.

A VPN device has direct access to the gateway in the other network over an encrypted connection.

VPN connections

When a VPN connection is established with TeleService, a CA certificate and a unique fingerprint generated from it are used for the unique identification of the TS Adapter IE Advanced. The VPN connection allows a communication that is secure from tapping and manipulation between the remote PC and the TS Adapter.

If you want to establish a VPN connection to multiple remote networks, you will require a separate TS Adapter IE Advanced as an interface for each network. The networks connected by the TS Adapter can used identical IP addresses internally. Only the external IP addresses (WAN) of the TS Adapters need to be different.

Only one VPN connection to a TS Adapter is possible at any one time.

Note

Authentication

Please not that the CA certificate exclusively authenticates the TS Adapter IE Advanced.

As in the case of dial-up connections, users are authenticated via the login (user name and password).

Therefore, use only a secure password for the login.

See also

- Establishing VPN connections (Page 7704)
- Terminating VPN connections (Page 7706)

14.7.4.2 Basics of CA certificates

Introduction

To establish a secure VPN connection with TeleService, you need to generate a CA certificate with a unique fingerprint when you configure the TS Adapter IE Advanced.

You can install this CA certificate as follows on each PC which is to access the TS Adapter IE Advanced:

- Using the automatic download in the TeleService connection dialog, by entering the fingerprint for the CA certificate
- Manually by using the Microsoft® Management Console.

Definition of CA certificate

A CA certificate is a digital certificate issued by a "certificate authority" or "certification authority", hereinafter referred to as "CA". In the case of the TS Adapter IE Advanced, self-signed certificates are used; the certification authority in this case is the TS Adapter IE Advanced itself.

Certificates for "SSTP" (Secure Socket Tunneling Protocol) and "HTTPS" (Hypertext Transfer Protocol Secure) are derived from the CA certificate.

CA certificates contain a "key" and additional information used for authentication and for encrypting and decrypting confidential data. Additional information includes the validity period, references to certificate revocation lists, etc. which are included in the certificate by the CA.

Using CA certificates with TeleService

A CA certificate with a unique fingerprint is generated by the TS Adapter to uniquely identify the TS Adapter IE Advanced as connection partner for the remote PC.

This CA certificate must be saved in the Windows certificate store of your programming device/ PC before you can establish a VPN connection. When you access the Web server using a direct connection, a security warning will appear if no CA certificate exists. In such cases, however, you can choose to ignore the warning and allow the connection.

Note

Handling CA certificates

Specialist knowledge of the operating system is required for handling CA certificates. CA certificates should only be managed by trained personnel!

You require administrator rights to manage CA certificates.

Definition of fingerprint

The fingerprint is a 20-byte hexadecimal expression. It provides a unique value for a CA certificate and is used to identify a specific CA certificate.

A fingerprint is calculated dynamically using the SHA-1 algorithm and is not physically contained in the CA certificate.

Use of fingerprints with TeleService

The CA certificate is used to uniquely identify the TS Adapter IE Advanced as connection partner. A unique 20-byte fingerprint of this certificate is generated each time a CA certificate is generated by the TS Adapter IE Advanced. It is calculated automatically by the TS Adapter IE Advanced when the certificate is generated. Each certificate has a specific, unique fingerprint. This fingerprint must be transferred securely to your computer, for example by phone or in an encrypted e-mail. You need to enter this fingerprint in the connection dialog when establishing a VPN connection with TeleService, unless the CA certificate is already saved on your PC in the Windows certificate store.

You will find the fingerprint for your TS Adapter IE Advanced in the web interface for the TS Adapter IE Advanced. To open the web interface, double-click the "Configure TS Adapter IE Advanced" command in the device list in the TIA Portal. Perform the Web login to view the fingerprint in the "Security > Certificates" tab.

CA certificate download during connection establishment

When the connection is established, TeleService checks whether a suitable CA certificate is installed in the Windows certificate store of your programming device/PC. If an appropriate certificate is found, the VPN connection is established as an SSTP connection (Secure Socket Tunneling Protocol).

If no appropriate CA certificate is found, the CA certificate is first loaded by the corresponding TS Adapter IE Advanced. The TS Adapter IE Advanced is called by the remote address entered in the connection dialog. If this download of the CA certificate is successful, the fingerprint of the downloaded CA certificate is calculated and compared with the fingerprint entered in the connection dialog. If the two fingerprints match, a dialog opens and you are prompted to save the CA certificate in the Windows certificate store of your programming device/PC. You require administrator rights to save the CA certificate.

The VPN connection is then established.

See also

- Installing CA certificates for VPN connections (Page 7701)
- Deleting CA certificates for VPN connections (Page 7704)

14.7.4.3 Installing CA certificates for VPN connections

Installing CA certificates

To establish a secure VPN connection between your programming device/PC and a remote system with TeleService, you require a valid CA certificate generated by the TS Adapter IE Advanced. This must be saved in the Windows certificate store of your programming device/PC.

A CA certificate can be installed manually or using an automatic download.

CA certificates are managed with the Microsoft® Management Console.

Note

Handling CA certificates

Specialist knowledge of the operating system is required for handling CA certificates. CA certificates should only be managed by trained personnel!

You require administrator rights to manage CA certificates!

Requirement

A CA certificate has yet to be installed in the Windows certificate store on your computer.

Installing CA certificates using the automatic download

Proceed as follows:

1. Log on to the system as administrator.
2. Transfer the fingerprint for the CA certificate from the TS Adapter IE Advanced to your computer using a "secure route", for example by phone or in an encrypted e-mail.
   You will find the fingerprint for your TS Adapter IE Advanced in the web interface for the TS Adapter IE Advanced. To open the web interface, double-click the command "Assign TS Adapter IE parameters" in the device list in the TIA Portal. Perform the Web login to view the fingerprint in the "Security > Certificates" tab.
3. In the project tree of the TIA Portal, click the "Online access" folder.
4. Click on the "TeleService" folder it contains.
5. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
6. Select the "TS Adapter IE Advanced" as Adapter type from the drop-down list.
7. Under "Connection type", select "VPN".
8. Enter the IP address / DNS name for the TS Adapter IE Advanced to be contacted in the relevant field. Alternatively, you can apply any existing data from the phone book by clicking the corresponding button.
9. Enter your user name and the corresponding password.
10. Copy the fingerprint displayed in the web interface for the TS Adapter IE Advanced to the "Fingerprint" column.
11. Click the "Connect" button to establish the required remote connection. This button only becomes active when you have entered all the parameters needed to establish a remote connection.
12. No valid CA certificate is found, because no CA certificate has been installed on your PC yet.
    A "normal" connection (not a VPN connection) is therefore established and the certificate required is downloaded from the TS Adapter IE Advanced to the working memory of your computer. The fingerprint is then calculated (SHA-1 algorithm) and compared with the fingerprint entered in the connection dialog. If the two fingerprints match, a dialog opens and you are prompted to save the CA certificate in the Windows certificate store of your programming device/PC.
13. Confirm that you want to save the CA certificate.

Result

The VPN connection to the required TS Adapter IE Advanced is established. The dialog closes once the connection is established.

Manual installation of CA certificates

Proceed as follows:

1. Log on to the system as administrator.
2. Open the Windows Certificate Manager on your programming device/PC with the Microsoft® Management Console.
   Click "Start", enter "mmc" in the search field and press ENTER.
   The console will open.
3. Open the "File" menu and click "Add/remove snap-in...".
   The "Add/remove snap-in..." dialog will open.
4. Double-click "Certificates" in the "Snap-in" list and select "Computer account" in the dialog that opens.
5. Select "Local computer" in the next dialog and click "Finish" and "OK".
   The console root opens and the "Certificates (local computer)" folder appears.
6. Open the displayed "Certificates (local computer)" folder and click "Trusted certification authorities".
7. Click the "Certificates " folder and use the shortcut menu to access the command "All tasks" > "Import...".
8. Note the information that appears in the "Certificate - Import wizard" dialog and click "Next".
9. In the dialog that follows, click "Browse ..." and select the required CA certificate.
10. Then click "Next" twice and subsequently "Finish" to install the CA certificate.

Result

The selected CA certificate is installed at the specified location in the Windows certificate memory.

Note

Additional information ...

... on installing CA certificates can be found using the "F1" button in the online help of your operating system.

14.7.4.4 Deleting CA certificates for VPN connections

Deleting CA certificates

Proceed as follows:

1. Log on to the system as administrator.
2. Open the Windows Certificate Manager on your programming device/PC with the Microsoft® Management Console.
   Click "Start", enter "mmc" in the search field and press the ENTER KEY.
   The console will open.
3. Open the "File" menu and click "Add/remove snap-in...".
   The snap-in selection dialog will open.
4. Double-click "Certificates" in the "Snap-in" list and select "Computer account" in the dialog that opens.
5. Select "Local computer" in the next dialog and click "Finish" and "OK".
   The console root opens and the "Certificates (local computer)" folder appears.
6. Open the displayed "Certificates (local computer)" folder and click "Trusted certification authorities".
7. Open the "Certificates" folder, select the required CA certificate and click "Delete" in the shortcut menu.
8. Confirm the next prompt with "Yes".

Result

The selected CA certificate is deleted from the list of available certificates.

14.7.4.5 Establishing a VPN connection to a remote system

Establishing VPN connections

Introduction to establishing VPN connections

A VPN connection is established when you use TeleService to connect to a remove system over the Internet.

Your programming device/PC with installed TIA Portal is connected to the Internet at one end for this. At the other end, the automation system is connected to the Internet via the WAN (Wide Area Network) interface of the configured TS Adapter IE Advanced.

Requirement

Your programming device/PC is connected to the Internet.

There is a TS Adapter IE Advanced in the remote system.

The TS Adapter IE Advanced has been configured and is connected to the Internet.

The CA certificate required for identifying the TS Adapter has been generated and installed in the Windows certificate store of your programming device/PC.

Proceed as follows:

1. In the project tree of the TIA Portal, click the "Online access" folder.
2. Click on the "TeleService" folder it contains.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. Select the "TS Adapter IE Advanced" as "Adapter type" from the drop-down list.
5. Enter "VPN" as "connection type".
6. Enter the IP address / DNS name for the TS Adapter IE Advanced to be contacted in the relevant field. Alternatively, you can apply any existing data from the phone book by clicking the corresponding button.
7. Enter your user name and the corresponding password.
8. Click the "Establish" button to establish the required VPN connection. This button only becomes active once you have entered all the parameters needed for establishing the remote connection.

Result

The VPN connection to the required system is established. "Status" indicates the progress in establishing the connection. The dialog closes once the VPN connection is established. The following message appears in the status bar of the TIA Portal: "Remote connection is up". You can now use the remote connection with the TIA Portal and communicate with the automation system.

Connection cannot be established

If the connection cannot be established, try to find the cause using the "Notes on troubleshooting".

Note

Rules for IP addresses

- Only use IP addresses which have not yet been assigned in the system network.
- If the IP address configured for the TS Adapter IE Advanced has already been assigned in the system network, the TS Adapter IE Advanced can only be addressed by its MAC address.

See also

Installing CA certificates for VPN connections (Page 7699)

Terminating VPN connections

Terminating an active VPN connection

Note
Terminating the VPN connection
You should go offline in the TIA Portal before you terminate the VPN connection.

Proceed as follows:

1. Double-click the "Set up/close remote connection" entry in the TIA Portal.
2. Confirm the prompt in the dialog that follows with "Yes".

Result

The VPN connection is terminated.

14.7.4.6 TS Adapter IE Advanced

Short description of the TS Adapter IE Advanced

TS Adapter IE Advanced

The TS Adapter IE Advanced has the following properties:

- Direct connection over Industrial Ethernet (IE), 2 ports
- WAN (Wide Area Network) interface for VPN connections
- Firmware update supported
- Plug-in modules
- Parameters are assigned via a Web interface.

Note

Further information on the TS Adapter IE Advanced

For detailed information on your TS Adapter, please refer to the documentation supplied with your TS Adapter.

Connection types

TS Adapter IE Advanced connection types

The following diagrams show the types of connection possible with the TS Adapter IE Advanced.

Direct connection

In the direct connection to the programming device/PC, you can set the TS Adapter IE Advanced parameters over the Ethernet.

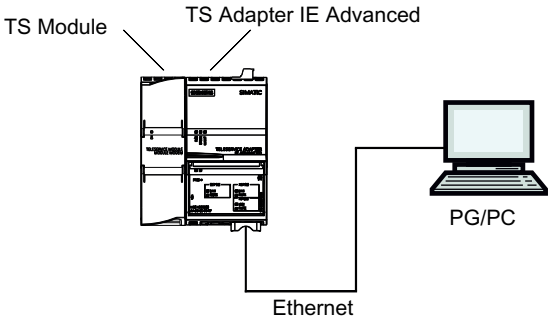


Figure 14-5 TS Adapter IE Advanced - direct connection

Connection to the Internet (DSL modem/router)

In order to connect to the Internet, you must operate the DSL modem/router at the WAN connection of the TS Adapter IE Advanced.

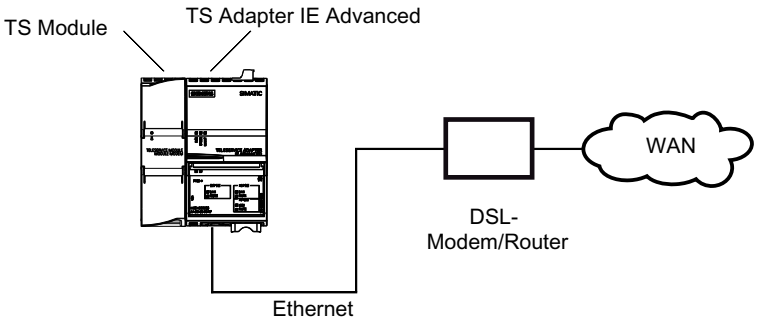


Figure 14-6 TS Adapter IE Advanced - connection to the Internet

Connection to the company network (Intranet)

In order to connect to the Intranet, you must operate system network (Ethernet) at one of the two LAN connections of the TS Adapter IE Advanced.

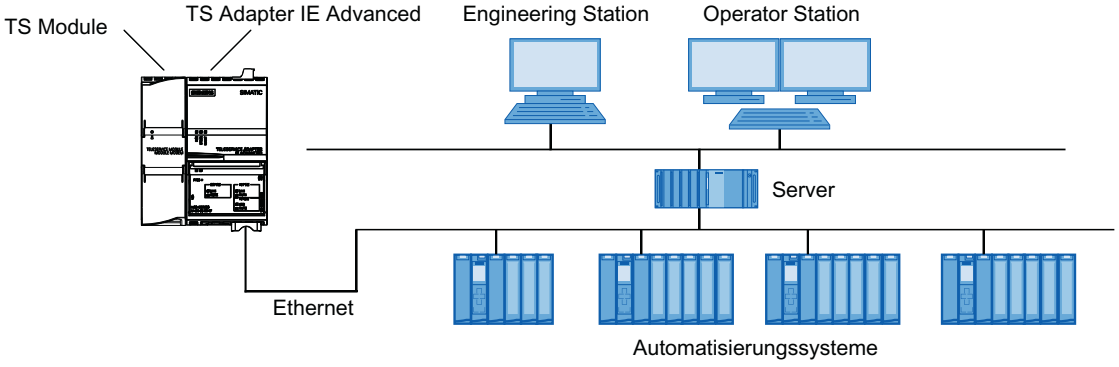


Figure 14-7 TS Adapter IE Advanced - connection to company network (Intranet)

Additional information

For more detailed information on the TS modules, please refer to the documentation supplied with your TS Adapter IE Advanced.

TS Adapter IE Advanced parameter assignment options

Basics on configuring the TS Adapter IE

The TS Adapter IE Advanced is configured via a Web interface.

Web help in the Web interface is available for assigning TS Adapter IE Advanced parameters.

Parameter assignment options include:

- Reconfiguration
- Restoring default parameter assignment
- Importing adapter parameters
- Exporting adapter parameters

Note

Parameter assignment

Configure your TS Adapter IE Advanced in accordance with the documentation supplied. This details the exact procedure for parameter assignment.

Parameter assignment for TS Adapter IE Advanced

Introduction

The TS Adapter IE Advanced can be configured in both direct connection mode and via an existing remote connection.

Both parameter assignment options are described below.

Specific details on assigning parameters for the TS Adapter IE Advanced are available in the TS Adapter IE Advanced documentation.

Assigning parameters for the TS Adapter IE Advanced in direct connection

Requirement

There is a LAN connection to your TS Adapter IE Advanced.

Procedure

To assign the parameters for the TS Adapter IE Advanced, proceed as follows:

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Double-click on the Ethernet port of your computer.
3. Double-click on the "Display accessible nodes" command. The TS Adapter IE Advanced is then displayed.
4. Double-click on the "TS Adapter IE Advanced" folder and then on "Online and diagnostics", and assign the desired IP address to the TS Adapter in the dialogs that follow. Please note that the IP address of the programming device/PC interface card must be located in the same subnet as the IP address that you assign for the TS Adapter IE Advanced.

5. Update the view in the project tree for the "Accessible nodes", so that the TS Adapter IE Advanced is displayed with the newly assigned IP address.
6. Open the folder "TS Adapter IE Advanced" in the device list.
7. Double-click the command "Assign TS Adapter IE parameters". The corresponding web interface opens for assigning TS Adapter parameters.
8. Complete the "logon" for the web interface.
9. Set the required parameters in the individual tabs of the dialog.
10. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE Advanced. Parameter assignment is then complete.

Assigning parameters for the TS Adapter IE Advanced using a remote connection

Requirement

There is an established remote connection to a TS Adapter IE Advanced..

Procedure

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Open the "TeleService" folder followed by the required plant folder.
3. Double-click the command "Assign TS Adapter IE parameters". The associated web interface for assignment of the TS Adapter IE parameters opens. The "logon" for the web interface takes place automatically with the login data of the remote connection.
4. Set the required parameters in the individual tabs of the dialog.
5. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE Advanced. Parameter assignment is then complete.

14.7.5 CPU controlled TeleService remote connections

14.7.5.1 Overview of CPU controlled remote connections

Introduction

TeleService offers a range of options for establishing remote connections; these differ according to the CPU used. The initiative for establishing a connection starts from the CPU. The communication instructions given below are used for the individual connection options.

Connection establishment with S7-300/400 CPUs

The following communication instructions are available:

- Communication instruction "PG_DIAL": Establish remote connection to programming device/PC
- Communication instruction "SMS_SEND": Send text message (SMS)
- Communication instruction "AS_DIAL": Establish remote connection to AS
- Communication instruction "AS_MAIL": Transfer email

Connection establishment with S7-1200 CPUs

The following communication instruction is available:

- Communication instruction "TM_MAIL": Transfer email

Connection establishment with S7-1500 CPUs

The following communication instruction is available:

- Communication instruction "TMAIL_C": Transfer email

Note

Description of individual communication instructions

More detailed information on the available communication instructions can be found in the information system of the TIA Portal in the "References > Communication > TeleService" directory.

See also

TM_MAIL: Transfer email (Page 3854)

TS Adapter IE parameter assignment options (Page 7692)

14.7.5.2 Establishing a connection from and to remote systens (PG-AS-remote coupling)

Remote plant access to a programming device/personal computer

Introduction

You can establish a remote connection to and communicate with a remote system using the application TeleService and a TS Adapter MPI. The initiative for establishing the remote connection comes from the programming device/personal computer.

However, events which require rapid intervention often occur at a remote system. In such cases, the automation system can initiate a remote connection to a programming device/ personal computer if an asynchronous event occurs.

The graphic below shows the components which are required for establishing a connection from a plant to a programming device/personal computer.

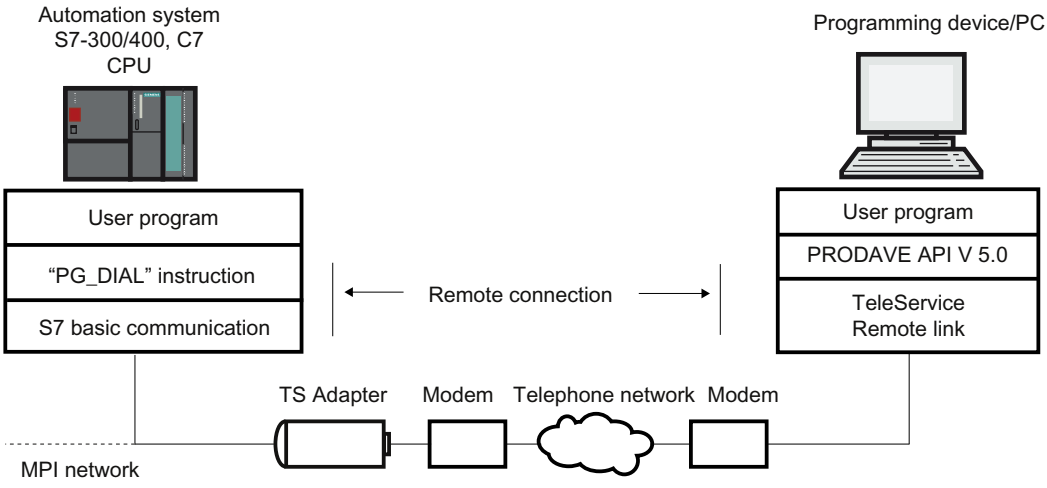


Figure 14-8 How the "PG_DIAL" communications instruction works

Requirements for establishing a connection

Introduction

Certain hardware and software requirements must be fulfilled if a remote system is to establish a remote connection to a programming device/personal computer. These requirements are detailed below.

Hardware requirements:

The only hardware required for establishing a remote connection from a remote system to a programming device/personal computer is that needed for accessing the remote system from the programming device/personal computer.

Your user program calls the "PG_DIAL" communication instruction to establish the connection. This can only be executed on an S7-300 or S7-400 CPU on which S7 basic communication is implemented.

A TS Adapter I , version 5.0 or later, or a TS Adapter II must be used.

Software requirements on the system side:

The "PG_DIAL" communication instruction is included in the scope of delivery of TeleService and is installed when the TIA Portal is installed. You will find the installed communication instructions in the "Communication > TeleService" folder in the task card of the block editor.

If a remote system is to establish a remote connection to a programming device/PC, the user program of the system must call the "PG_DIAL" communication instruction.

Software requirements on the programming device/PC side:

You require a software product on the programming device/PC which, with TeleService, waits for a call from a remote system, recognizes this call and informs your user program.

14.7.5.3 Data exchange between remote systems (AS-AS-remote coupling)

AS-AS remote link basics

Introduction

The AS-AS remote link allows two automation systems to exchange process data via the telephone network.

Requirement

Communication instruction "AS_DIAL" is available if you use a CPU from the S7-300/400 family.

Definition: Local and remote automation system

- The automation system from which the initiative to establish the remote connection originates is described as **local**.
- The automation system to which the remote connection is to be established is described as **remote**.

Data exchange over the AS-AS remote link

Data exchange is carried out using specific communication instructions for non-configured S7 connections. Use the communication instruction "AS_DIAL" to establish a remote connection to the automation system.

More detailed information on establishing the connection can be found in the information system in the directory "References > Communication > TeleService".

The following graphic shows the components required for establishing a connection from a local to a remote automation system.

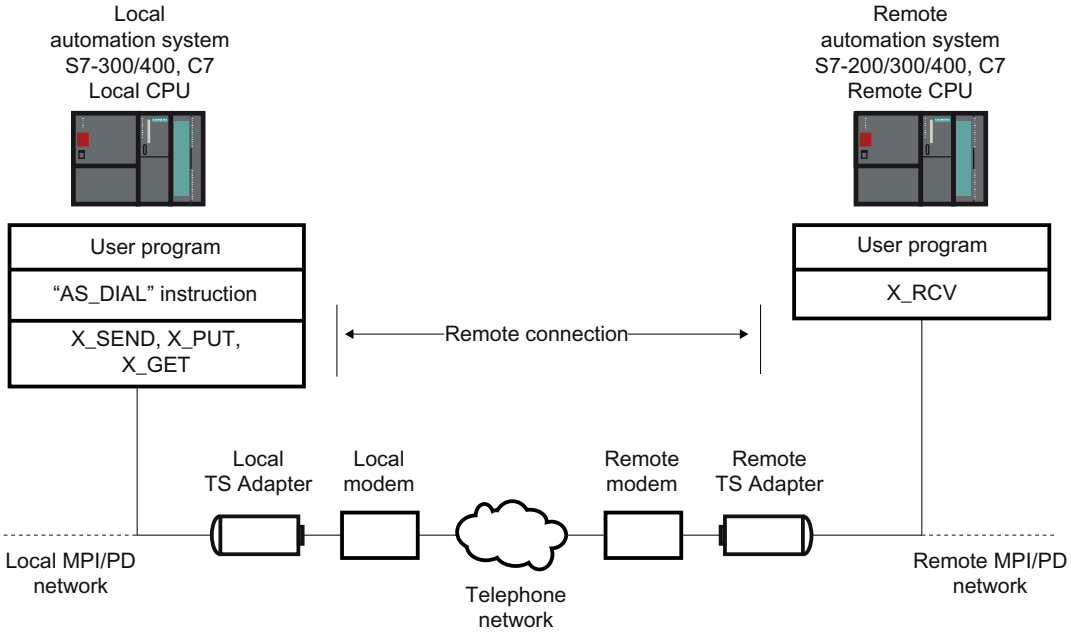


Figure 14-9 Data exchange over the AS-AS remote link

Hardware and software requirements for AS-AS remote link

Introduction

Certain hardware and software requirements must be fulfilled before a local automation system can establish a remote connection to a remote automation system. These requirements are detailed below.

Hardware requirements

The only hardware you need for transferring process data from a local to a remote automation system is that also needed for accessing the respective automation system from the programming device/personal computer.

To establish and terminate the remote connection, the user program of the TIA Portal of the local CPU calls a communication instruction. This communication instruction can be executed on an S7-300/400 CPU or a C7 CPU. The communication instruction requires S7 basic communication to be implemented on the CPU. The remote CPU must also support S7 basic communication.

A TS Adapter I, version V5.1 or later, or a TS Adapter II must be used.

Software requirements

The "AS_DIAL" communication instruction is included in the scope of delivery of TeleService. When installed, the instruction is integrated in the library of the TIA Portal in the task card of the Communication instructions folder under TeleService. In order to establish and terminate a remote connection to a remote automation system from a local automation system, the "AS_DIAL" communication instruction must be called in the user program of the TIA Portal on the local CPU

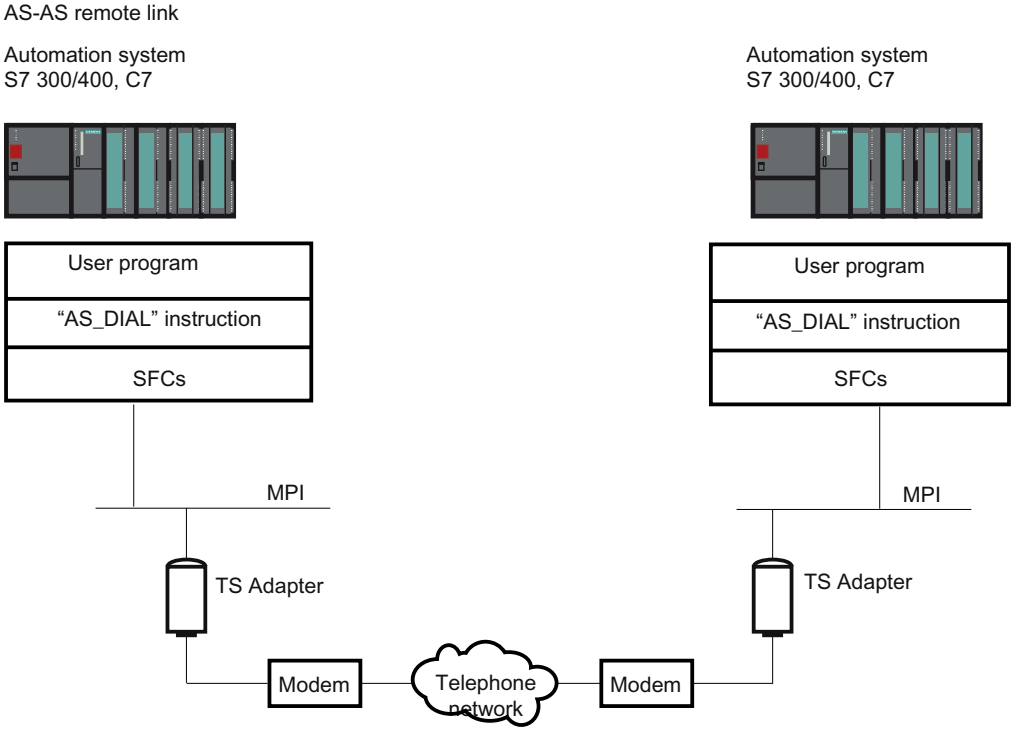


Figure 14-10 Hardware and software requirements for AS-AS remote link

14.7.5.4 Send SMS from a system

Requirements for sending an SMS

Introduction

Certain hardware and software requirements must be fulfilled before a system can send an SMS. These requirements are detailed below.

Hardware requirements

To send an SMS from a system, you will require a GSM wireless modem and a TS Adapter MPI.

A TS Adapter I, version V5.2 or later, or a TS Adapter II must be used.

Software requirements on the system side

The "SMS_SEND" communication instruction is included in the scope of delivery of TeleService. When installed, the instruction is integrated in the library of the TIA Portal in the task card of the Communication instructions folder under TeleService. If a system is to send an SMS, the user program of the system must call the "SMS_SEND" communication instruction.

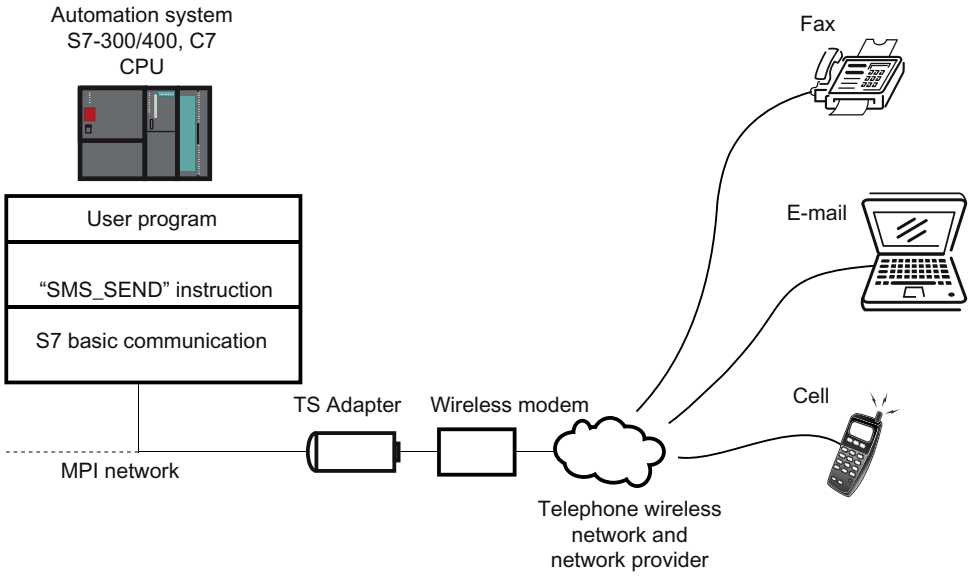


Figure 14-11 How the "SMS_SEND" communication instruction works

Note

You may be able to send an SMS not only to a cell phone but also to an email address or a fax device by using additional services offered by the cell phone service provider.

14.7.5.5 Send an email from a system

Requirements for sending e-mails

Introduction

The following hardware and software requirements must be fulfilled if a system is to send an email:

Hardware requirements

You need a TS Adapter IE to send an email from a system and one of the CPUs listed below:

- a CPU 31x2 PN/DP as of firmware version V2.5
- a CPU 41x-3 PN/DP
- a CPU of the S7-1200 series
- a CPU of the S7-1500 series

Software requirements on the system side

Various communication instructions are included in the scope of delivery of TeleService depending on the CPU. When installed, they are integrated in the library of the TIA Portal in the task card of the Communication instructions folder under TeleService.

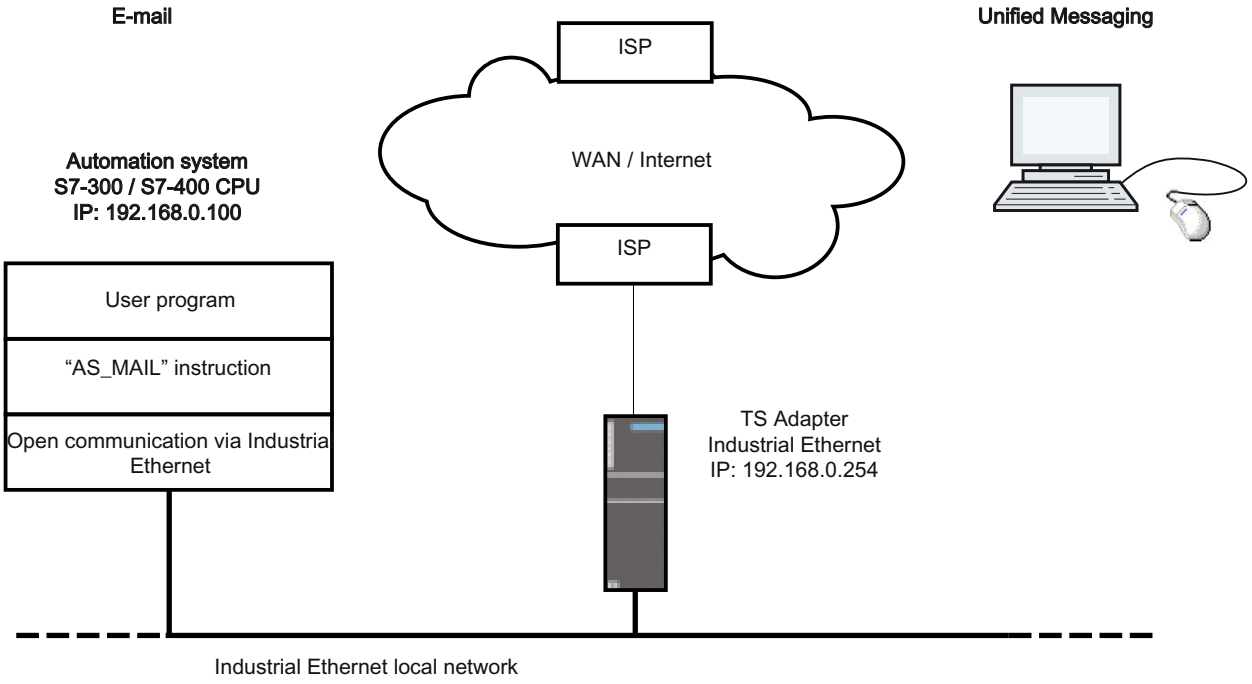
If a system is to send an email, the user program of the system must call the corresponding CPU-dependent communication instruction.

The following communication instructions are available for sending an email:

- S7-300/400 CPUs: use the communication instruction "AS_MAIL": Transfer email
- S7-1200 CPU V2.x and V3.x: use the communication instruction "TM_MAIL": Transfer email
- S7-1200 CPU >V4.0: use the communication instruction "TMAIL_C": Transfer email
- S7-1500 CPUs: use the communication instruction "TMAIL_C": Transfer email

The respective communication instruction transfers an email from a CPU to a mail server by means of the Simple Mail Transfer Protocol (SMTP) with the "LOGIN" authentication method. The data is transferred unencrypted with this SMTP process.

The figure below shows an example with the "AS_MAIL" communication instruction:



The "Use gateway/router" property must also be set for the Ethernet interface in the configuration of the CPU on which the "AS_MAIL" communication instruction runs. (available in the device configuration under Ethernet addresses and there under IP protocol)The IP address of the Ethernet interface of the TS Adapter IE is to be specified as "Address".

Note

You can find further information in the task card in the "Communication instructions" folder under TeleService.

14.7.6 Notes on troubleshooting

14.7.6.1 General information on troubleshooting for modem problems

Introduction

The information below should help you establish and eliminate the causes of any modem problems.

1. Enable "Record log file" for data traffic between PG/PC and modem. The entries in this file can provide valuable information for determining the cause of errors.
2. Switch on the loudspeaker on your local modem. Select a volume loud enough to be clearly heard.
   You can then hear whether:
   - there is a dial tone at the connection
   - the modem called is busy
   - the modem called accepts the call.

Common modem problems

Modem connection problems are among the most common modem problems:

- Modem connection is not established
- Modem connection is interrupted

The topics below contain tables detailing possible causes and providing information on eliminating the error in question.

See also

Dial-up connection to the TS Adapter is not established (Page 7720)

Dial-up connection from the TS Adapter is not established (Page 7722)

Modem connection is interrupted (Page 7723)

Modem alarms (Page 7724)

Recording a log file for the modem (Page 7719)

14.7.6.2 Recording a log file for the modem

Introduction

It is advisable to record a log file as this makes it easier to find the causes of faults in a modem.

Procedure:

Proceed as follows:

1. Activate the properties dialog of the modem used via the "Phone and modem options" option in the Control Panel.
2. Check the settings of the "Log" option in the "Diagnostics" tab and if necessary change the log file settings so that the file is recorded.

Result:

Activity between the programming device/personal computer and the modem are entered in the log file. If there are problems establishing the connection, you can evaluate the data recorded in the log file to determine the cause of the error.

14.7.6.3 Dial-up connection to the TS Adapter is not established

Dial-up connection to TS Adapter is not established

The table below sets out possible causes and how to eliminate them if no remote connection to the TS Adapter can be established.

Possible cause	Check/Remedy
Cabling faulty	• Are all the connecting cables connected correctly? • Are the connectors loose?
Dial parameters for main telephone line and extension incorrectly set	• Do the properties and dial parameters of your modem match the phone connection to main phone line or extension? • Do not specify a dial-out code in the "Dial parameters" dialog if you operate your modem on a local loop (main telephone line). The fields for the dial-out code for local calls and long-distance calls must be empty.
Dialing mode incorrectly set	• Is the correct dialing mode (tone/pulse) set in the dialog for the dial parameters of your modem? • Use a connected telephone to check the connection on which you want to operate the modem. You should hear crackling noises on the telephone during pulse dialing and tones of varying pitches during tone dialing. Set the corresponding dialing mode in the modem dial parameters.

Possible cause	Check/Remedy
Dial disable active	• The dial disable function is a country-specific modem property which, depending on the modem, becomes effective after one or more unsuccessful attempts to establish a connection. • If your modem still does not respond after several attempts to dial, the dial disable function may be active. Characters are still sent to the modem after the dial command but the modem does not start the dialing process. The driver receives a general error message. • Refer to the modem documentation for information on how the dial disable function is implemented for your modem. • Create a log file (Page 7717) (modemlog.txt) in which the activities between the programming device/personal computer and modem are recorded. Then check whether the file contains an entry caused by dial disable (e.g. DELAYED).
Phone connection defective or busy	• Connect a phone and check whether a dial tone can be heard on this connection. • Any analog phone connected on the same connection must be hung up. You cannot establish an additional modem connection on this connection if there is an existing phone connection.
Serial parameters set incorrectly	• Are the correct values entered in the "Settings" tab for modem properties (8 data bits, no parity, 1 stop bit)? Is the correct COM interface set in the "General" tab for the modem properties?
Initialization string of the TS Adapter is not suitable for the modem.	• Familiarize yourself with the modem initialization string requirements and set the string accordingly. Procedure for configuring the TS Adapter IE (Page 7689)
Settings for error correction between the modem at the TS Adapter and the modem at the PC/programming device are not compatible.	• Adapt the modem settings. Useful information on configuring the TS Adapter MPI (Page 7684) Restoring the default parameter assignment of a TS Adapter MPI (Page 7686) Procedure for configuring the TS Adapter IE (Page 7685)

14.7.6.4 Dial-up connection from the TS Adapter is not established

No callback from TS Adapter

The table below sets out possible causes and how to eliminate them if there is no callback from the TS Adapter.

Possible cause	Check/Remedy
Errors in the location or call settings in the TS Adapter	Check the TS Adapter parameter assignment: • Are the dialing mode and dial-out code set correctly for your phone connection? • Does the modem at the TS Adapter support the characters configured for the dial-out code? • Is "Wait for dial tone before dialing" deactivated for an extension?
Initialization of modem insufficient	Check the string for modem initialization: • The modem may require a further initialization in order to establish a remote connection. • Properties of the modem initialization string for the TS Adapter MPI
Callback number is incorrect	Check the configuration of the callback number you assigned.

No call from TS Adapter MPI

The table below sets out possible causes and and how to remedy them if there is no call from the TS Adapter MPI.

Possible cause	Check/Remedy
Phone number is incorrect	Is the required number being transferred to the communication instruction "PG_DIAL" ?
TS Adapter MPI parameter assignment incorrect	Check the TS Adapter MPI parameter assignment: • Are the dialing mode and dial-out code set correctly for your phone connection? • Does the modem at the TS Adapter MPI support the characters configured for the dial-out code? • Is "Wait for dial tone before dialing" deactivated for an extension?

14.7.6.5 Modem connection is interrupted

Modem connection is interrupted

The table below sets out possible causes and and how to remedy them if the modem connection is interrupted.

Possible cause:	Check / Remedy:
Metering pulses in the line	Metering pulses will be generated if you have applied to the phone company for a metering clock. This may mean that the modem no longer recognizes the carrier signal and switches off. • Set a longer waiting or switch-off time at the modem. • Have the metering pulse deactivated by the phone company.
Shielding	• Are the connection cables used shielded sufficiently? • Make sure that the modem cables do not run next to power cables and that they are as far as possible from power supply units and monitors.
Protocol timeout	• Set fixed monitoring times.
Automatic connection termination	• Deactivate the option that terminates an existing connection automatically after a specified time without data transfer ("Terminate after idle of ...").
Data flow control deactivated	• Click on the "Extended" button in the "Settings" tab of the modem properties and activate the following options in the dialog displayed (if available and not yet set): – Data flow control – Hardware (RTS/CTS) – Data compression – Error control
Initialization string of the TS Adapter is not suitable for the modem	• Set the modem initialization string in accordance with the following requirements. See also: TS Adapter IE parameter assignment options (Page 7692)

See also

TS adapter MPI configuration options (Page 7684)

14.7.6.6 Checklist for troubleshooting the modem

Introduction

The following list should help you establish the potential cause of any problems with the modem. The help topics below set out how and in which dialogs you define the relevant settings.

Modem connection cannot be established:

- Check the cabling and the connections.
- Check whether the correct dialing mode (tone/pulse) is set.

- If your modem does not react after several attempts to dial, a dial disable function may be active. Familiarize yourself with dial disable on your modem.
- Are you operating your modem on a main telephone line or on an extension line? Configure the properties and dialing parameters of the modem accordingly.
- Enable the log file option in the advanced properties. The next attempt to establish a connection will then be recorded in a file in the Windows directory.
- Ensure that the ISDN TAs used work with the same B and D channel protocol.

The modem connection is terminated:

- Metering pulses can have a negative affect on a connection. Have the pulses deactivated by your telephone company.
- Set fixed monitoring times.
- Deactivate the option that terminates an existing connection automatically after a specified time without data transfer (idle).
- Make sure that you have activated RTS/CTS for data flow control.

14.7.6.7 Modem alarms

Information in the log file

The modem alarms are entered in a log file if you have activated the recording function.

The log file contains the following information:

Alarm:	Possible cause:	Remedy:
NO DIALTONE	A phone call may currently be being carried out on this line.	• Repeat the process once the phone call is over.
NO CARRIER	The device dialed is not ready, is not a modem or cannot establish a connection in the set operating mode.	• Check the numbers and the settings.
BUSY	The device dialed is busy.	• Try again later.
DELAYED: ...	Dial disable	• Refer to the modem documentation for information on how the dial disable function is implemented for your modem and if necessary remove it.

14.7.6.8 Possible error messages with VPN connections

VPN connection to TS Adapter IE Advanced is not established

The table below sets out possible errors and how to eliminate them if no VPN connection to the TS Adapter IE Advanced can be established.

Possible errors	Check/Remedy
The error message "The webpage cannot be displayed" appears when the Web interface is accessed.	• Note: This problem occurs when the Windows certificate store contains a CA certificate with the same name as the CA certificate which has just been generated. • If a CA certificate for this TS Adapter is already installed in the Windows certificate store and you have generated a new CA certificate on the TS Adapter, you need to remove the old CA certificate from the Windows certificate store and install the newly generated CA certificate. • See also: Installing CA certificates for VPN connections (Page 7699) • See also: Deleting CA certificates for VPN connections (Page 7702)
Error message: "Unexpected error in application".	• Check whether other users have been saved in the TS Adapter IE Advanced user database. • The user "Administrator" cannot establish a remote connection.
Error message: "No corresponding CA certificate was found in the Windows certificate store".	• Use the automatic certificate download. Enter the fingerprint in the corresponding field in the connection dialog. • See also: Establishing VPN connections (Page 7702) Or • Export the CA certificate from the Web interface of the TS Adapter IE Advanced and install it manually in the Windows certificate store. • See also: Installing CA certificates for VPN connections (Page 7699)
Error message: "The remote address specified does not correspond to the remote address of the TS Adapter".	• You must use the remote address which you specified in the TS Adapter IE Advanced to establish the connection. • You cannot use the IP address if you have entered the DNS name (and vice versa).
Error message: "The signature of the certificate could not be verified".	• If you have generated a new CA certificate on the TS Adapter IE Advanced, you need to delete the old CA certificate from the Windows certificate store and install the new one. • See also: Deleting CA certificates for VPN connections (Page 7702) • See also: Installing CA certificates for VPN connections (Page 7699)
Error message: "Protocol error"	• Check to make sure that the IP address of the service PC is set in the configuration of the TS Adapter IE Advanced (Parameter > Plant Network)
Error message: "Connection setup error."	• Check to make sure that the LAN IP-address of the TS Adapter IE Advanced and the IP address of your network cards are in the same subnet. If this is the case, deactivate this network card before you establish the remote connection.
An e-mail cannot be sent.	• Check whether there is a network/Internet connection at the WAN port. • Check whether the mail server is accessible. • Check whether the outgoing connections in the TS Adapter IE Advanced are valid. • Check whether the SMTP port in the firewall is open for outgoing connections.

Using Team Engineering 15

15.1 Shared commissioning of projects

15.1.1 Basics for shared commissioning

Introduction

As part of Team Engineering , you have the option to perform shared commissioning of projects. Several editors can access one CPU at the same time with up to five Engineering Systems (ES) in this process.

A special advantage is that parts of a master project can be edited independently and offline at the same time during the commissioning phase. The changes of the other editors are displayed in the dialog "Software synchronization" during loading and synchronized automatically, if possible.

Depending on the firmware version of the CPU in use, certain online functions can also be executed at the same time from several Engineering Systems on the shared CPU, for example:

- Monitoring blocks on the CPU
- Modifying and forcing blocks on the CPU
- Trace functions

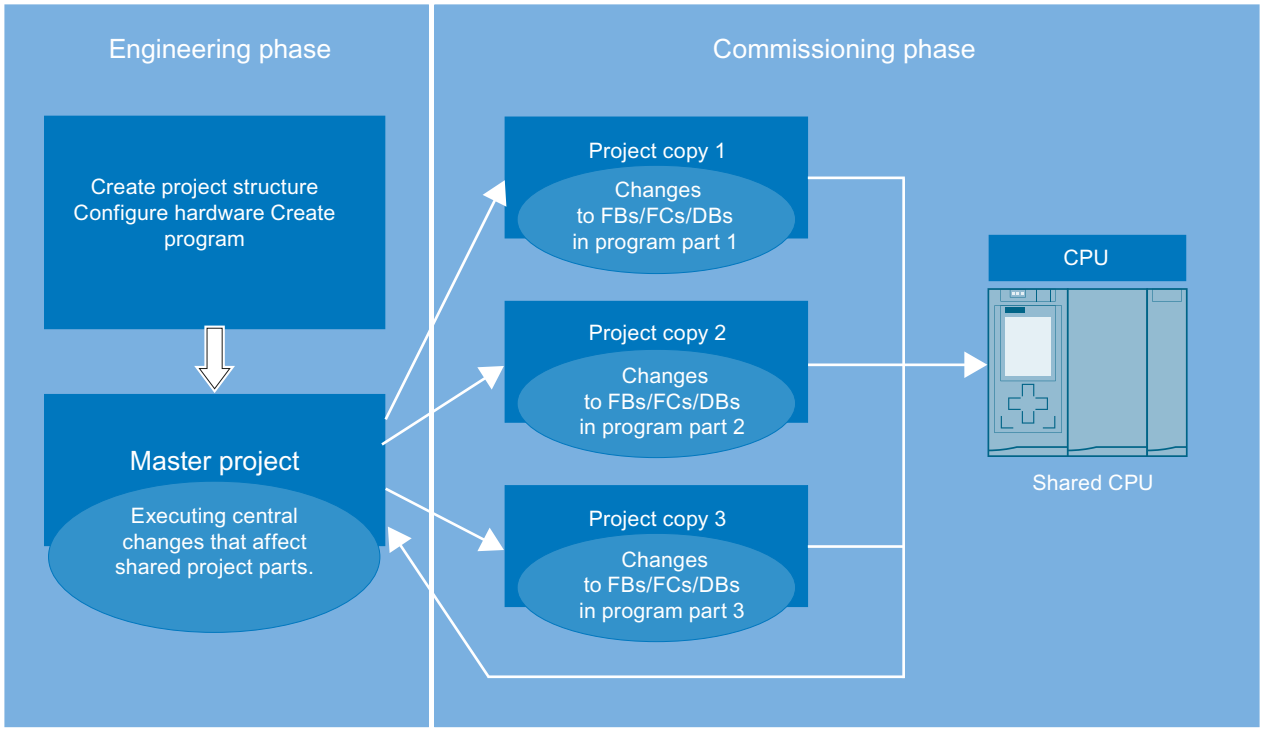
The following online functions cannot be executed at the same time:

- Loading: Only one ES can load to the CPU.

You can also find additional information on the topic of Team Engineering at Service and support in Siemens Industry online support (http://support.automation.siemens.com/WW/view/en/82142829).

Creating a master project

A "master project" structured according to the specified rules (Page 7734), which already contains the complete configured hardware configuration with all required tags and blocks, is used as the basis for the shared commissioning. This project is loaded in the jointly used CPU and then distributed as "master project" by means of project copies to up to five participating Engineering Systems.



Working with project copies

Each Engineering System only processes the parts it was assigned within the specific project copy. This is especially important to avoid conflicts during loading and unintentional overwriting of blocks that were already modified by other Engineering Systems.

Each ES loads its parts to the shared CPU after editing. The TIA Portal helps you during download to the CPU. The changes made since the last download are displayed in the dialog "Software synchronization before loading to a device" (hereafter referred to as "synchronization dialog"), by means of an online-offline comparison. You are provided recommendations during download to the CPU about how you can synchronize the offline data you have edited with the existing online data.

If there are "competing changes", you may have to synchronize them manually.

Competing changes are:

- when different editors edit the same block in different project copies at the same time.
- when PLC tags (I, Q, M, T, C) are added, changed or deleted in the project copies.

- when blocks that have relations to the hardware configuration are edited in the project copies.
- when F blocks are edited in the project copies.

Each editor loads their changes only after all conflicts have been removed.

The procedure for editing and loading of project copies can be repeated as often as necessary by the individual Engineering Systems until the shared commissioning is complete.

Integrating project copies into the master project once again

The individual project copies must once again be integrated into the master project at the end of commissioning at the same time. This way you also back up the data that only exist in the offline projects and have not been downloaded to the CPU. These include, for example, the project languages and text lists created in the project copies.

To do this, open the master project and one project copy as reference object. You determine the differences between the two projects by performing an offline/offline comparison using the comparison editor. Copy the edited objects from the respective project copy to the master project. Repeat this step for all project copies.

When the shared commissioning is completed, you have an executable master project with all created project data.

The integration of the project data into the master project is required even if you determine during the commissioning phase that the shared central elements need to be expanded or corrected, for example, the PLC tag table needs to be expanded, or a new hardware component needs to be added.

Notes on compatibility

The following compatibility rules apply to shared commissioning:

- If an Engineering System with TIA Portal V13 or later is online on the CPU and performs a download, a second ES cannot establish an online connection, regardless of the version.
- If an Engineering System with TIA Portal < V13 is online on the CPU, a second ES cannot establish an online connection, regardless of the version.

Team Engineering

As part of Team Engineering , you have the option to perform shared commissioning of projects. Several editors can access one CPU in parallel and at the same time with multiple Engineering Systems (ES) in this process.

Shared commissioning

In shared commissioning, multiple Engineering Systems work together and in parallel on one CPU as part of team engineering.

Master project

The master project is a basic project for shared commissioning. It is a project structured according to the specified rules which already contains the fully configured hardware configuration with all required tags and blocks. This project is loaded into the jointly used CPU and then distributed as "master project" by means of project copies to the participating Engineering Systems. Each Engineering System only processes the parts it was assigned within the specific project copy. The project copies are then integrated into the master project once again.

Project copy

Project copies are created from structured master projects when working with team engineering and distributed to the participating Engineering Systems for editing. Each Engineering System only processes the parts it was assigned within the specific project copy as part of shared commissioning. The individual project copies are once again integrated into the master project at the end of shared commissioning.

See also

Requirements for shared commissioning (Page 7730)

Procedure for shared commissioning (Page 7731)

15.1.2 Requirements for shared commissioning

Software and hardware requirements

For shared commissioning of projects, the minimum software and hardware requirements for the installation of TIA Portal V13 or later must be met.

The following requirements also apply:

Software:

- You have installed TIA Portal V13 or later and the "SIMATIC STEP 7 Professional" software package on the involved Engineering Systems.
- The same software version must be installed on all Engineering Systems.
- The master project must be one created with V13 or later.

Hardware:

- You have access to a fully configured CPU S7-1500 as of firmware version V1.5.
- The involved Engineering Systems can establish an online connection to this CPU.

Requirements for shared, parallel working on the project

The following requirements are in effect for shared commissioning of projects:

- You have created a project with complete hardware configuration and fully programmed user program that can be commissioned.
- The project was downloaded to the CPU and defined as "master project".
- Multiple Engineering Systems have access to this CPU and can establish an online connection to this CPU.
- You are familiar with the defined rules and procedures for working together on one CPU.

Note on compatibility mode

The functions for shared commissioning are not available in compatibility mode.

See also

Basics for shared commissioning (Page 7725)

Procedure for shared commissioning (Page 7731)

Rules for shared commissioning (Page 7734)

15.1.3 Procedure for shared commissioning

Introduction

If you are performing shared commissioning a project as part of team engineering, it is very important that all editors of the project observe a defined procedure.

Only when the specified procedure is observed are changes and corrections in the project automatically synchronized and applied, and changes of individual editors not unintentionally overwritten during loading or even lost in case of competing changes.

Procedure for creating the master project

The master project is created in the following steps:

1. Create a master project that includes the entire project structure.
2. Fully configure the hardware for the master project.
3. Define a language as project language, which must also be used exclusively by all participating engineering systems.
4. Create all tags and blocks that are required in the master project.
5. Create your own folders and groups for the blocks to be edited by the individual engineering systems.
6. Create a fully programmed and executable user program.

7. Download the master project to the shared CPU.
8. Save the master project following every download.

For this, also observe the Rules for shared commissioning (Page 7734).

Procedure for shared commissioning

Shared commissioning takes place with the following steps:

1. Download the master project to the shared CPU.
2. Create copies of the master project and distribute them to the editors involved in the project on the associated engineering systems.
3. Inform all editors who is to edit the specific parts of the project copies and who may download to the CPU.
4. Have the project copies edited in the individual engineering systems.
5. After editing, the individual engineering systems download the changes one after the other to the CPU.
6. All changes are automatically detected during loading by means of an online-offline comparison. The displayed synchronization dialog offer you recommendations for synchronizing the modified data when possible. You may first have to download blocks that were edited by other editors to your ES before you can download your changes to the CPU. This may be needed to update the corrections of the other editors that have already been downloaded to the CPU in your project copy.
   The following synchronization options are available:
   - Download block: There are edited blocks on the CPU that must be updated in your project copy.
   - Download block: There are new blocks on the CPU that must be downloaded to your project copy.
   - Blocks with competing changes: System-supported synchronization is not possible in this case; the conflict must be resolved manually.
7. Manually resolve any conflicts that may occur due to competing changes or changes to central objects.
8. Download your project copy to the CPU when all conflicts have been removed.
9. Save your project copy following every download.
10. Repeat editing of project copies and download to the CPU as often as necessary until shared commissioning is complete.
11. Integrate the completely edited project copies into the master project once again in order to also back up the project data that only exist offline.

You should also observe the Rules for shared commissioning (Page 7734).

Procedure for manual synchronization of competing changes

Manual synchronization of competing changes is performed in the following steps:

1. Start the comparison editor to manually resolve a conflict displayed in the synchronization dialog during download to the CPU. Select the shared CPU in the project tree and select the "Compare > Offline/Online" command in the shortcut menu. You can define certain actions depending on the status of the objects. Note, however, that you can only perform actions in one direction during synchronizing.
2. First, select the action "Upload from device" for all blocks that were changed by other editors on the CPU and which you want to apply.
3. If necessary, perform a detailed comparison of the blocks to identify differences between your offline blocks and the online blocks loaded on the CPU.
4. Manually correct the competing changes to the blocks.
5. Then download the affected blocks to the CPU using the "Continue without synchronization" command.
6. Save your project or project copy following every download.

For this, also observe the Rules for shared commissioning (Page 7734).

Note

To prevent online changes occurring again while an engineering system is being manually synchronized, no downloading to the shared CPU should be performed by another participating engineering system during the manual synchronizing.

Procedure for integrating project copies into the master project

You integrate project copies into the master project with the following steps:

1. Open the master project and the project copy to be integrated as a reference object.
2. Copy the program parts you have edited from the respective project copy to the master project and confirm overwriting the existing objects. You can also use the comparison editor to apply your program parts to the master project.
3. Save the master project and download it to the shared CPU with the "Continue without synchronization" command.
4. Save the master project following every download.

For this, also observe the Rules for shared commissioning (Page 7734).

Procedure for modifying central objects within the master project

Modification of central objects that have an effect on all program parts is performed with the following steps:

1. Stop further processing of the project copies.
2. Integrate all existing project copies one after the other into the master project as described above.
3. Download the master project to the shared CPU with the "Continue without synchronization" command.
4. Make the required modification on the shared objects such as in the hardware configuration, for example, or in the PLC tag table. The procedure is the same for modification of other central objects such as technology objects, F blocks, OBs, etc.
5. Download the master project into the CPU once again when modification is complete.
6. Save the master project following every download.
7. Distribute the updated project copies to the respective engineering systems for further processing.

For this, also observe the Rules for shared commissioning (Page 7734).

See also

Basics for shared commissioning (Page 7725)

Requirements for shared commissioning (Page 7728)

Rules for shared commissioning (Page 7734)

15.1.4 Rules for shared commissioning

Introduction

If you are performing shared commissioning a project as part of team engineering, it is very important that all editors of the project observe predefined rules for successful cooperation.

Rules for the master project

The following rules are to be observed:

- Create a master project that is suitable for shared editing.
- Divide the user program into program parts that are independent of each other.
- Use "Groups" to visually separate the program parts from each other.
- Use a main OB and a central FC for each program part which calls the functions of the part program.

- If possible, create a separate PLC tag table for each part.
- In the master project, specify a project language which cannot be changed in the project copies.
- If you want to exchange data between the program parts, use the FC and FB interface parameters (IN, OUT, INOUT) or global data blocks.
- Use global data blocks to save the data for the individual program parts, no bit memories.
- Do not assign different names for blocks with identical addresses.
- Do not assign identical names for blocks with different addresses.

Example of the program structure in the master project

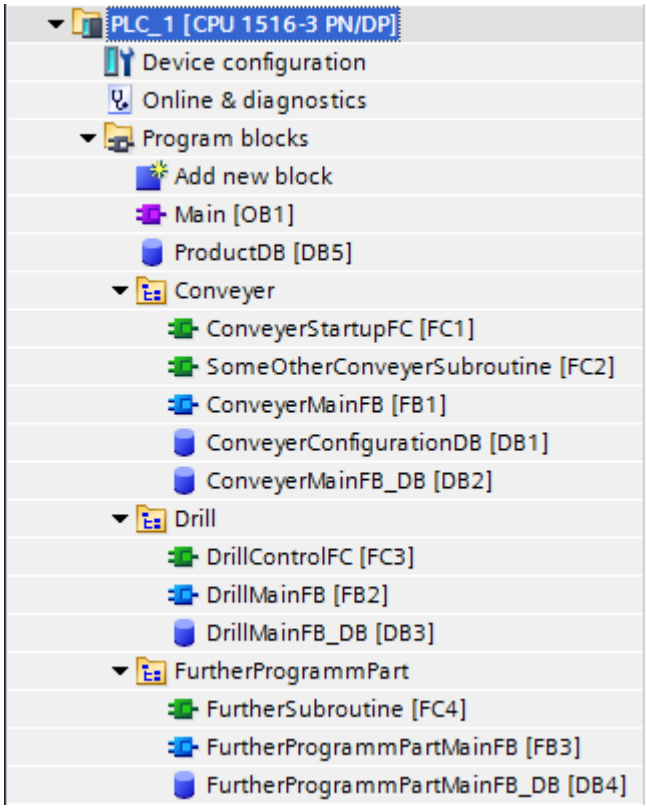
Here is an example of a team-capable, structured master project that is suitable for parallel editing during shared commissioning.

The program parts edited by the individual Engineering Systems are:

- Program part 1: "Conveyer"
- Program part 2: "Drill"
- Program part 3: "FurtherProgramPart"

Each program part contains a "main FB", which calls the specific lower-lever functions for this program part.

Example: in the "Conveyer" program part, the "ConveyerMainFB" calls the "ConveyerStartup" function.



In team engineering, this subdivision within the project structure allows the parallel editing of individual program parts and the automatic synchronization of changes during the shared commissioning phase.

Rules for shared working on one CPU

Observe the following rules for shared working:

- Editor only edits the blocks they have been assigned in the project copy in the assigned groups.
- Only FBs, DBs, FCs and UDTs may be changed in the project copies, i.e. the following program elements should **not** be edited in the project copies:
  - Hardware configurations
  - PLC tag table
  - Organization blocks
  - Technological objects
  - Text lists and project language
  - F blocks
- Global data blocks should be used in programming instead of bit memories.

- IEC timers and counters should be used during programming instead of SIMATIC timers and counters.
- The project language specified in the master project may not be changed in the project copies.

Rules for editing shared, central objects

Observe the following rules for shared central objects:

- Changes to shared central objects always have to be made by means of the master project. These objects include:
  - Hardware configurations
  - PLC tag table
  - Organization blocks
  - Technological objects
  - Text lists and project language
  - F blocks
- Before the shared central objects can be modified in the master project, the various working versions within the project copies must first be integrated into the master project once again.
- The modification can then be made within the master project, and the master project can then be downloaded to the shared CPU.
- Once modification is complete, new project copies can be created and distributed to the Engineering Systems for further processing.

Rules for downloading to the CPU

Observe the following rules for downloading:

- Editors can download the project copies one after the other to the CPU; only one Engineering System can download at any given time.
- The online connection of the involved Engineering Systems can be maintained when downloading the blocks (hardware may only be downloaded in the master project).
- Perform the synchronization that is offered automatically in the synchronization dialog during download. This ensures that changes from other editors are not unintentionally overwritten.
- Correct the name conflicts that are shown in the synchronization dialog, which you rename in your offline block before downloading to the CPU. A name conflict is indicated if you want to download to the CPU a block that already exists with identical name on the CPU.

- So-called "competing changes" must be corrected manually. Competing changes come about when two editors work on the same block on two Engineering Systems at the same time. The first editor can still download his or her changes to the CPU without any problems. However, if the second editor wants to download the same block with the changes the editor has made, the synchronization dialog displays a conflict that cannot be removed automatically. The previous changes made to this block would be undone by overwriting during download. The editors have to decide in this case which change is to be applied and which one is to be discarded. Alternatively, you can manually merge the changes with the help of the detailed comparison of the blocks in the comparison editor.
  Such changes should always be avoided by implementing a good project structure and corresponding agreements between the editors.
- Changes to central objects are displayed in the synchronization dialog but cannot be synchronized automatically. Changes to central objects and to the hardware configuration always have to be made by means of the master project.
- Save your project or project copy following every download.

Note

Creating and subsequent deleting of objects in the project copies

Please note that a software synchronization is also required before downloading when you create a new object in a project copy and then delete this again immediately afterwards.

The creation of an object causes a change in the internal data management which cannot be undone by the subsequent deleting of the object. As a result it is necessary to perform a synchronization before the next loading if the newly created object was deleted immediately thereafter and no change in the existing object is visible within the project copy.

Rules for online functions

The following rules are to be observed when using online functions:

Monitoring and controlling:

- Up to five Engineering Systems can be monitored and controlled simultaneously on the CPU blocks.
- A specific code block can only be monitored and controlled by one Engineering System at a time.
- However, additional Engineering Systems can simultaneously monitor and control other code blocks.

Notice

Danger due to modifying an identical operand in parallel with different modify values in more than one watch table

When working with more than one watch table, avoid modifying identical operands permanently multiple times with different modify values.

If an identical operand is modified permanently with different modify values at the same time in different watch tables, all watch tables will display the last modified value, because the modify value assigned last will be used in this case.

Force

- Forcing can only be implemented on the CPU from one Engineering System at a time.
- The Engineering Systems that starts the forcing has taken over the force job exclusively. Although the other Engineering Systems receive the information that a force job is running, they cannot access this job to make changes. Use the command "Update forced operands" to update all operands and values currently being forced on the CPU in the open force table. All forced operands in the open force table are updated with the relevant values. A red "F" is displayed in the first column indicating which operands are being forced.
- As soon as the Engineering System to which the force job belongs ends the online connection, the force job can be applied by another ES that establishes an online connection to the CPU and has executed the "Update forced operands" command. This enables the "Force all" and "Stop forcing" buttons and you can select these functions.

Trace functions:

- Up to four Engineering Systems can perform trace functions simultaneously.
- The Engineering Systems that starts the tracing has taken over the trace job exclusively. Although the other Engineering Systems can see this job in the trace editor, they cannot access it.
- As soon as the Engineering System to which the trace job belongs closes the trace editor, the trace job can be taken over by another ES that re-opens the trace editor.

See also

Basics for shared commissioning (Page 7725)

Requirements for shared commissioning (Page 7728)

Procedure for shared commissioning (Page 7729)

15.2 Exchanging data with Inter Project Engineering (IPE)

15.2.1 Basics of Inter Project Engineering (IPE)

Introduction to Inter Project Engineering (IPE)

The functionality of Inter Project Engineering, hereinafter referred to as IPE, is used to exchange the controller data in a source project between different projects with the help of a device proxy.

You can then transfer this data to other projects, for visualization in HMI, for example, and use it there for further configuration.

Inter Project Engineering gives you a simple way of providing controller data from a PLC programming across multiple projects for an HMI configuration.

Using the object "Device Proxy Data", you exchange controller data across multiple projects consistently and easily without redundant configuration overheads.

This makes it possible for HMI project engineers to work on a project without the hardware configuration having to be present in their project. This means they can work on the PLC project and on the HMI project at the same time.

You can exchange the following controller data between projects via the device proxy:

- Program blocks
- Technological objects
- PLC tags
- PLC alarms

The following data is automatically included in the exchange:

- Controller interfaces
- Configured communication processors and communication modules

This automatic data exchange ensures that the interfaces and the configured communication processors and modules are transferred along with the data you selected. This data is required to allow you to continue working consistently and without problems in your target project.

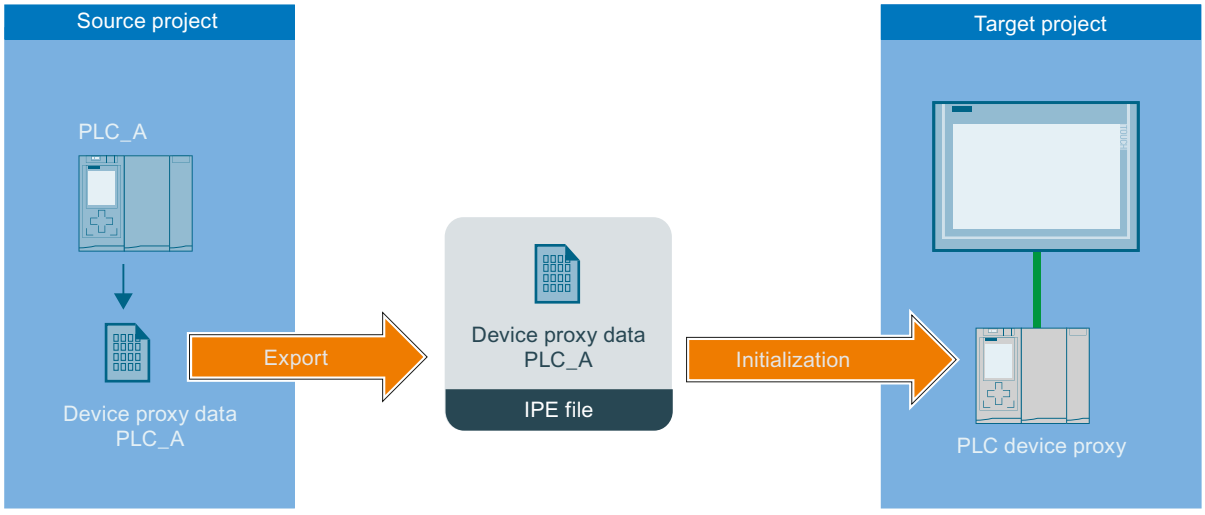
Cross-project data exchange

The following options are available for cross-project data exchange with Inter Project Engineering:

- Exchanging controller data via IPE file
- Exchanging controller data via project file

Exchanging controller data via IPE file

The figure below shows cross-project data exchange via an IPE file.

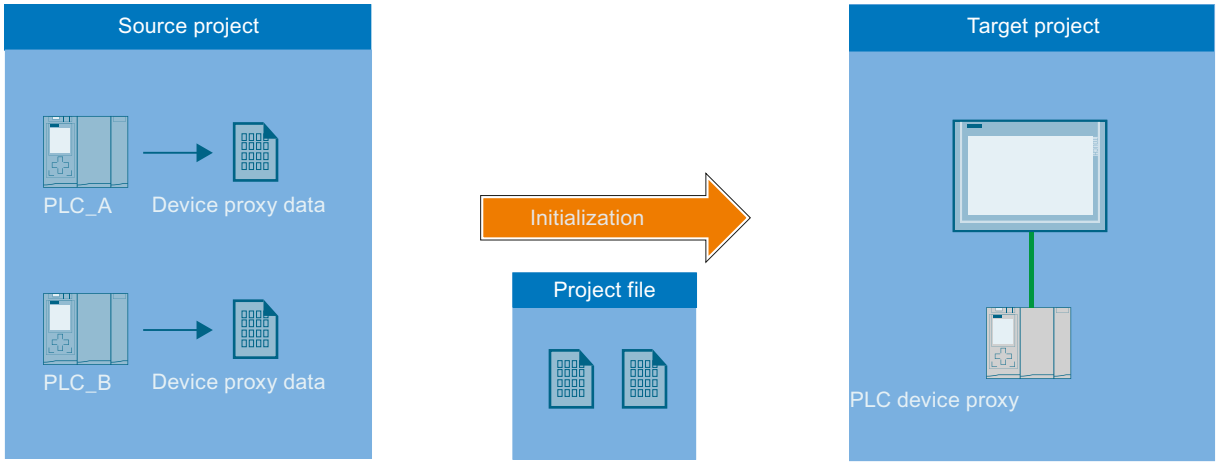


The PLC controller data from the source project is transferred to the PLC in the target project via the object "Device Proxy Data".

PLC data that is exchanged via an IPE file must originate from TIA Portal projects of V13 or higher.

Exchanging controller data via a project file

The figure below shows cross-project data exchange via a project file.



The PLC controller data from the source project is transferred to the PLC in the target project via a project file.

You can easily transfer and continue to use controller data from older TIA Portal projects as well as STEP 7 Classic projects that are not integrated in TIA Portal in your current TIA Portal projects.

Inter Project Engineering (IPE)

The functionality of Inter Project Engineering, also referred to as "IPE", is used to exchange the controller data in a source project between different projects with the help of the "Device proxy data" object.

IPE

The functionality of Inter Project Engineering, also referred to as "IPE", is used to exchange the controller data in a source project between different projects with the help of the "Device proxy data" object.

IPE file

The IPE file includes CPU controller data from a source project which is transferred with the "Device proxy data" object to the CPU in the target project.

Project file

The project file transfers CPU controller data from a source project to the CPU in the target project. You can conveniently transfer and continue to use controller data from older TIA Portal projects as well as STEP 7 Classic projects that are not integrated in TIA Portal in your current TIA Portal projects.

Device proxy data

The "Device proxy data" object supports consistent data exchange of CPU controller data across projects between a source project and a target project without redundant configuration.

Data exchange across projects with IPE

The Inter Project Engineering (IPE) functionality gives you the following options for consistent data exchange across projects:

- Exchanging PLC data via an IPE file
- Exchanging controller data via a project file

Both files transfer CPU controller data from a source project to a CPU in the target project.

See also

Requirements for Inter Project Engineering (IPE) (Page 7743)

Overview for working with Inter Project Engineering (IPE) (Page 7743)

15.2.2 Requirements for Inter Project Engineering (IPE)

Software and hardware requirements

To use the functionality of Inter Project Engineering, the minimum software and hardware requirements for the installation of TIA Portal V13 or higher must be met.

The following requirements also apply:

- You have installed TIA Portal V13 or higher and the "SIMATIC STEP 7 Professional" software package.
- You are using a CPU of the S7-1200/1500 or S7-300/400 series.
- The same software version must be installed on all involved controllers.

Requirements for IPE

The requirements for accepting controller data from projects are as follows:

- You have created a project with hardware configuration and PLC data that can be transferred to another project.
- You have made sure that the project is consistent before you transfer the controller data by means of the "Device proxy data" object.
- You are familiar with the defined procedures for Inter Project Engineering.

Notes on compatibility mode

The functions for Inter Project Engineering are not available in compatibility mode.

See also

Basics of Inter Project Engineering (IPE) (Page 7738)

Overview for working with Inter Project Engineering (IPE) (Page 7743)

15.2.3 Overview for working with Inter Project Engineering (IPE)

Exchanging controller data across projects

You can use the functionality of Inter Project Engineering to exchange existing controller data between different projects with the help of a "Device proxy" and to continue using this data in another project. The "Device proxy" provides you with a consistent and convenient way of exchanging data between projects without redundant configuration overheads.

Procedure for exchanging controller data via an IPE file

To exchange data via an IPE file, follow these steps:

1. Using TIA Portal V13 or higher, create a project with all required controller data and an executable user program.
2. Configure the hardware required for the project.
3. Compile the project to ensure consistency in your project.
4. In your project, create the object "Device Proxy Data" below the required CPU by clicking the corresponding command "Add device proxy data" in the "Device Proxy Data" folder.
   Result: The "Device Proxy Data" object has been created.
5. Select the required device proxy data in the project tree and double-click the object to open it in the editor.
6. Enter the required data for the device proxy data in the editor under "General" or apply the default setting. You can change the name and enter a comment for export of the controller data.
   In the "Define content" area of the editor select which of the existing controller data you wish to export as IPE file and click "Export device proxy data".
7. In the subsequent dialog, enter the name and storage location for the IPE file to be created and click "Save". You will be notified as soon as the export has been completed successfully.
   Result: The selected controller data are exported as IPE file.
8. In your target project, create a device proxy to be able to import the controller data contained in the IPE file. To do so, click the command "Add new device" and select the required device proxy in the dialog that opens.
9. Click the newly created device proxy and select the "Initialize device proxy" command from the shortcut menu.
10. Select the previously created IPE file to be used for the initialization, and confirm the selection in the following dialog with "OK". This starts the initialization.
    Result: The controller data from the source project contained in the IPE file is transferred to the device proxy in the target project.

After the successful exchange of controller data, you can continue with the configuration in your target project and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

Procedure for exchanging controller data via a project file

When exchanging controller data via a project file, device proxy data can either be predefined in the source project or it can be defined and selected from the target project.

Follow these steps to exchange data via a project file:

1. Using TIA Portal V13 or higher, create a project with all required controller data and an executable user program.
2. Configure the hardware required for the project.
3. Compile the project to ensure consistency in your project.

4. Open the TIA Portal project from which you want to import the controller data into the target project.
5. In your project, create the object "Device Proxy Data" below the required CPU by clicking the corresponding command "Add device proxy data" in the "Device Proxy Data" folder.
   Result: The "Device Proxy Data" object has been created.
6. Enter the required data for the device proxy data in the editor under "General" or apply the default setting. You can change the name and enter a comment for the device proxy data.
7. In the "Define content" area of the editor select which of the existing controller data you want to provide by means of the device proxy data.
8. Click "Save" to save the changes in the project.
   Result: You have successfully saved the selected controller data in your device proxy data.
9. In your target project, create a device proxy to import the controller data contained in the project file. To do so, click the command "Add new device" and select the required device proxy in the dialog that opens.
10. Click the newly created device proxy and select the "Initialize device proxy" command from the shortcut menu.
11. Select the previously created project file with the prepared device proxy data to be used for the initialization, and confirm the selection in the following dialog with "OK". This starts the initialization.
    Result: The controller data from the source project contained in the project file is transferred to the device proxy in the target project.

After the successful exchange of controller data, you can continue with the configuration in your target project and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

Procedure for updating already transferred controller data

System requirements for the update:

The following requirements must be met for updating already transferred controller data with the help of the already created "Device Proxy Data" object:

- You are using the same project for the update as for the previous transfer of the controller data.
- You are using the already created "Device Proxy Data" for transfer of the controller data.
- You have not made any changes to the hardware configuration or the communication interfaces in the meantime. You can add new hardware components.

Updating procedure:

1. Open the TIA Portal project from which you want to update the controller data in the target project.
2. Double-click the existing "Device Proxy Data" object and select under content which controller data you want to make available for the update.
3. Save the project or export the IPE file once again.
4. Open the required target project in the TIA Portal.

5. Click the existing device proxy in the target project and select the "Update device proxy" command from the shortcut menu.
6. Select the source which is to be used for the update, and confirm the selection.
7. In the following dialog, select under "Device" the CPU that was previously used as source, from which the controller data is to be re-imported.
8. Under "Defined device proxy data", select the required object and exit the dialog with "OK".
9. The update is then started and the selected device proxy data is transferred from the source project to the device proxy in the target project.

After the successful update of the controller data, you can continue with the configuration in your target object and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

See also

Basics of Inter Project Engineering (IPE) (Page 7738)

Requirements for Inter Project Engineering (IPE) (Page 7741)

15.2.4 Creating "Device proxy data" in the source project

Introduction

In order to exchange controller data across projects you require the "Device proxy data" object that can transfer the controller data of the associated CPU as an IPE file in the source project.

Creating a "Device proxy data" object in the source project

Proceed as follows to create a new "Device proxy data" object:

1. Open the "Device proxy data" folder below the CPU to which you want to create a "Device proxy data".
2. Click the command "Add new device proxy data".

Result

The "Device proxy data" object has been created and is displayed in the project navigation below the associated CPU.

The created "Device proxy data" contains the controller data of the associated CPU.

15.2.5 Creating an IPE file by means of the "Device proxy data"

Opening the "Device proxy data" and defining the content for an IPE file

Proceed as follows to create an IPE file by means of a "Device proxy data":

1. Open the required "Device Proxy Data" with a double-click in the "Device Proxy Data" folder in the project tree below the CPU.
2. Enter the required settings for the "Device proxy data" object in the editor under "General" or apply the default setting. You can change the name and enter a comment for export of the controller data.
3. In the "Define content" area of the editor select which of the existing controller data of the CU you wish to export as an IPE file and click "Export device proxy data".
4. In the subsequent dialog, enter the name and storage location for the IPE file to be created and click "Save". You will be notified as soon as the export has been completed successfully.

Result

The selected controller data have been exported as an IPE file.

15.2.6 Using controller data from other projects with IPE

15.2.6.1 Using controller data from other projects in an HMI device

Basics on controller data

Introduction

In the source project of the PLC, you can select which controller data is to be stored as device-proxy data.

To this end, you create a device proxy data from a configured PLC.

Which controller data can be exchanged?

The following controller data can be exchanged via IPE:

- Data blocks
- Technology objects
- PLC tags
- PLC alarms

The following information is exchanged automatically:

- Interfaces of the PLC
- Configured communication processors and communication modules at the PLC

Initializing a device proxy via an IPE file

Initializing device proxy via an IPE file

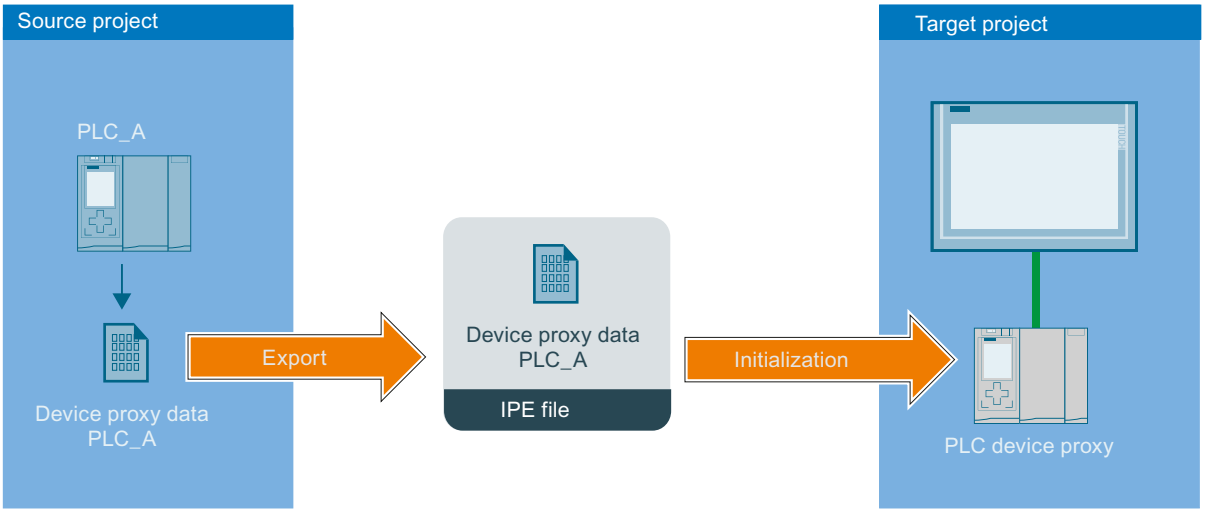
How does initialization of a device proxy via an IPE file work?

You can exchange controller data between two or more projects using IPE files.

Due to their small data volume, you can send IPE files by e-mail, for example.

You can create multiple IPE files with different device proxy data for each PLC.

A device proxy can only be filled with one device proxy data record in the target project.



- An IPE file is exported from the source project.
  The IPE file contains device proxy data of the PLC "PLC_A".
- A device proxy is created in the target project.
- In the target project, the device proxy is then initialized with the device proxy data from the IPE file.
- Following initialization, all device proxy data from the IPE file is contained in the device proxy.
- If changes are made in the source project, the device proxy of PLC_A in the target project can be updated via the new IPE file.

Initializing a device proxy via an IPE file

Introduction

You initialize device proxy data using a dialog on the device proxy in the TIA Portal.

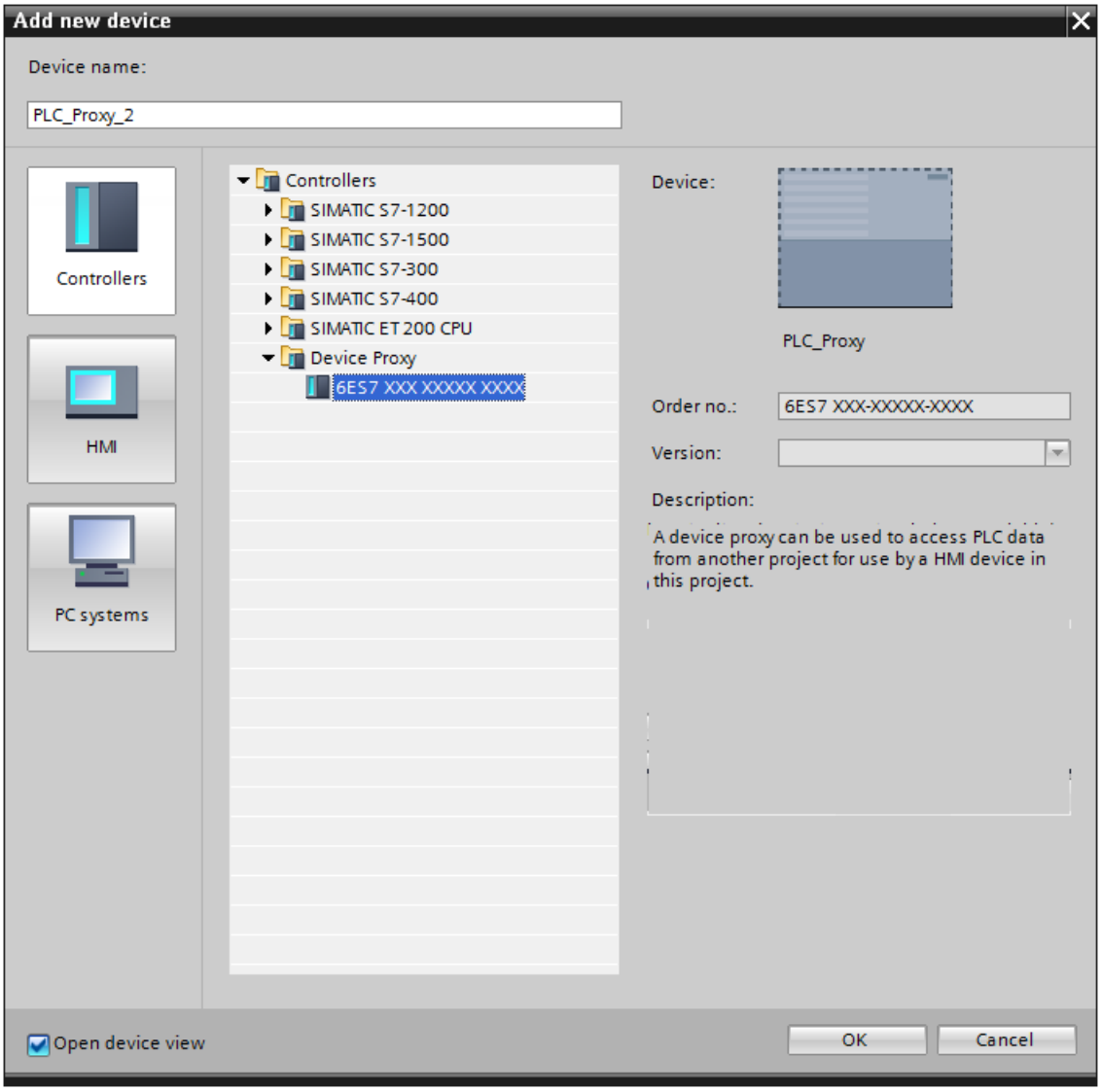
The IPE file contains the device proxy data from the source project.

Requirement

IPE file is available.

Procedure

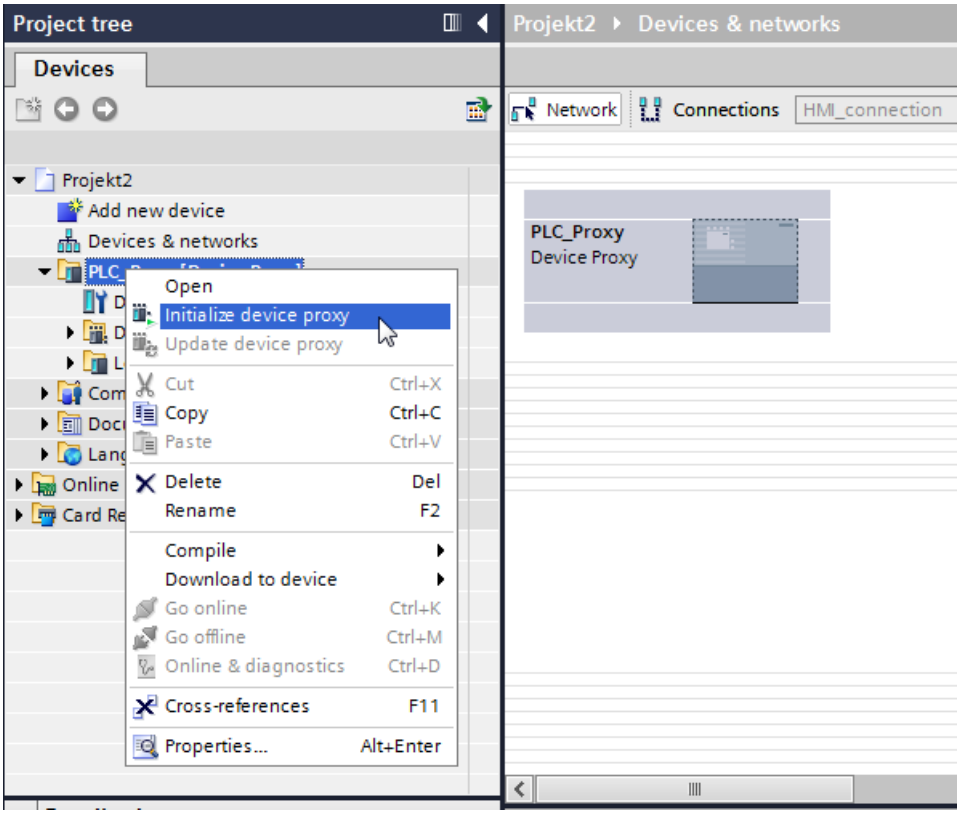
1. Double-click "Add new device" in the project tree.
2. Select the device proxy under "Controller".



A new device is created in the "Devices & Networks" editor.

3. Select the device proxy in the project tree.

4. Click "Initialize device proxy" in the shortcut menu.



5. Select the IPE file and click "Open".
   The "Initialize device proxy" dialog opens.

Note

You cannot select the device proxy data before initialization in the target project.

All device proxy data included in the IPE file is transferred.

Initialize the device proxy via project file if you want to select specific device proxy data before initialization: Initializing a device proxy via a project file (Page 7754)

6. Click "OK".
   Initialization of the device proxy is started.

Result

Following initialization, all device proxy data from the IPE file is contained in the device proxy.

You can now configure an HMI connection with the device proxy and connect PLC tags from the device proxy with HMI tags, for example.

Updating a device proxy via an IPE file

Introduction

If changes were made to the device proxy data in the device proxy source project, the device proxy data can be updated in the target project.

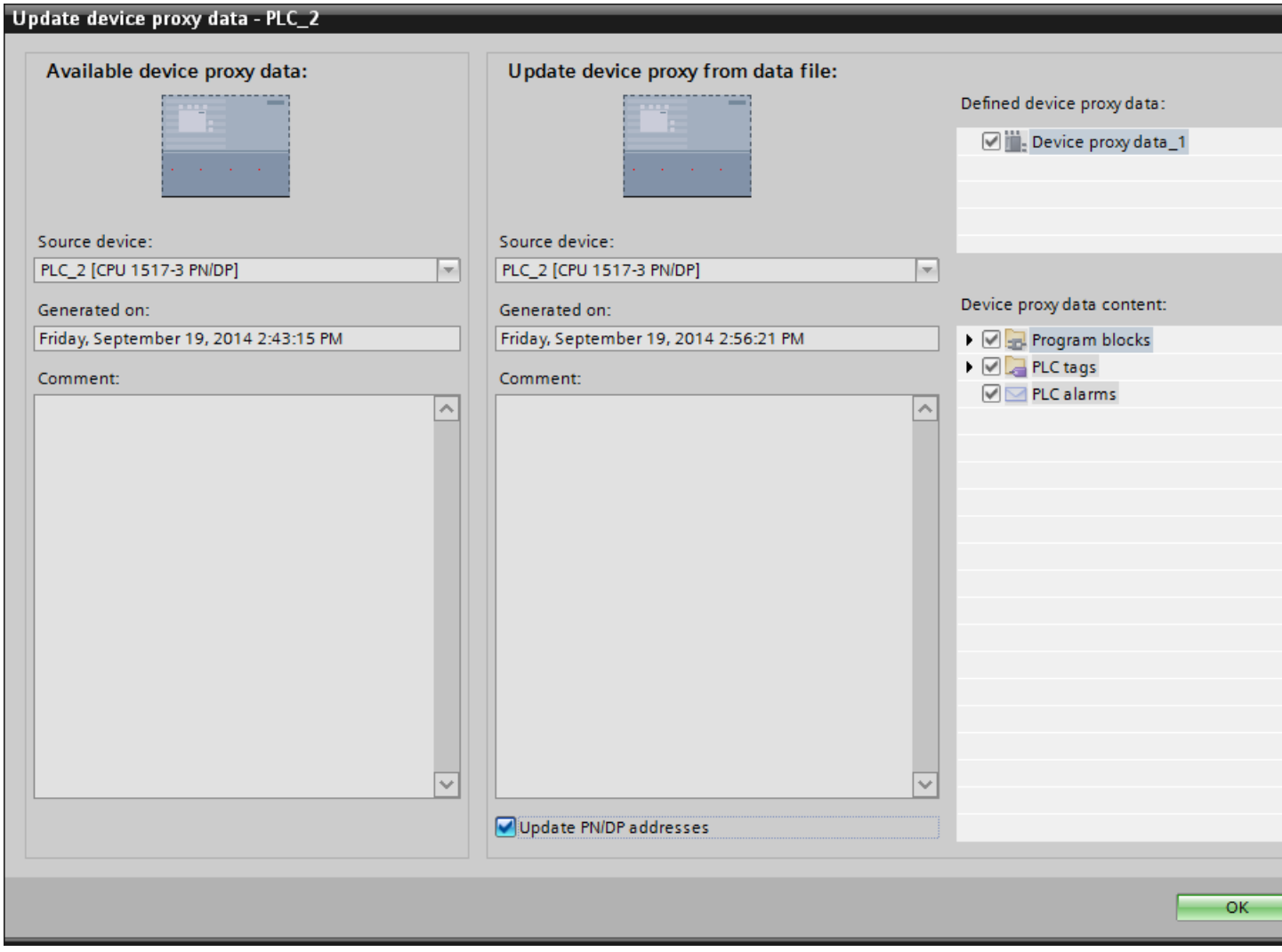
Requirement

- New IPE file contains the device proxy data from the source project.
- A device proxy has already been initialized using an IPE file in the target project.

Procedure

1. Select the device proxy in the project tree.
2. Click "Update device proxy" in the shortcut menu.

3. Select the IPE file.



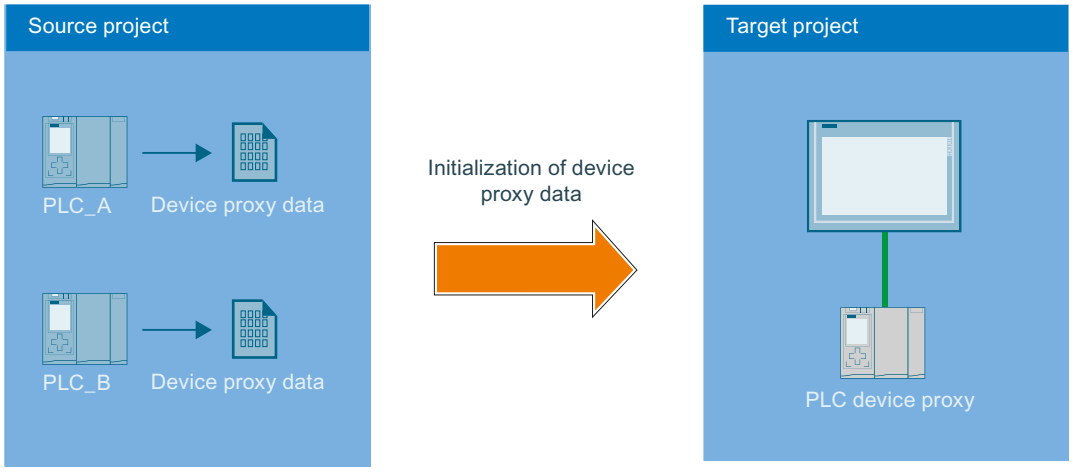
4. Select whether you wish to update the PROFINET or PROFIBUS addresses.
5. Click "OK".

Initializing a device proxy via a project file

Initializing a device proxy via a project file

How does an initialization via project file work?

You can transfer controller data using a project file in your TIA Portal project:



- There are two PLCs in the source project.
- Device proxy data objects of the two PLCs are created in the source project.
- The source project is saved.
- A device proxy is created in the target project.
- The source project (*.ap13) is first selected using the command for the initialization of device proxy in the device shortcut menu.
- The "PLC_A" and "PLC_B" controllers are available for selection and their device proxy data have already been created.
- After selection of device proxy data of "PLC_A", the data is stored in the PLC_Proxy of the target project.
- Changes from the source project can be transferred by an update via the project file.

Initializing a device proxy via a project file

Introduction

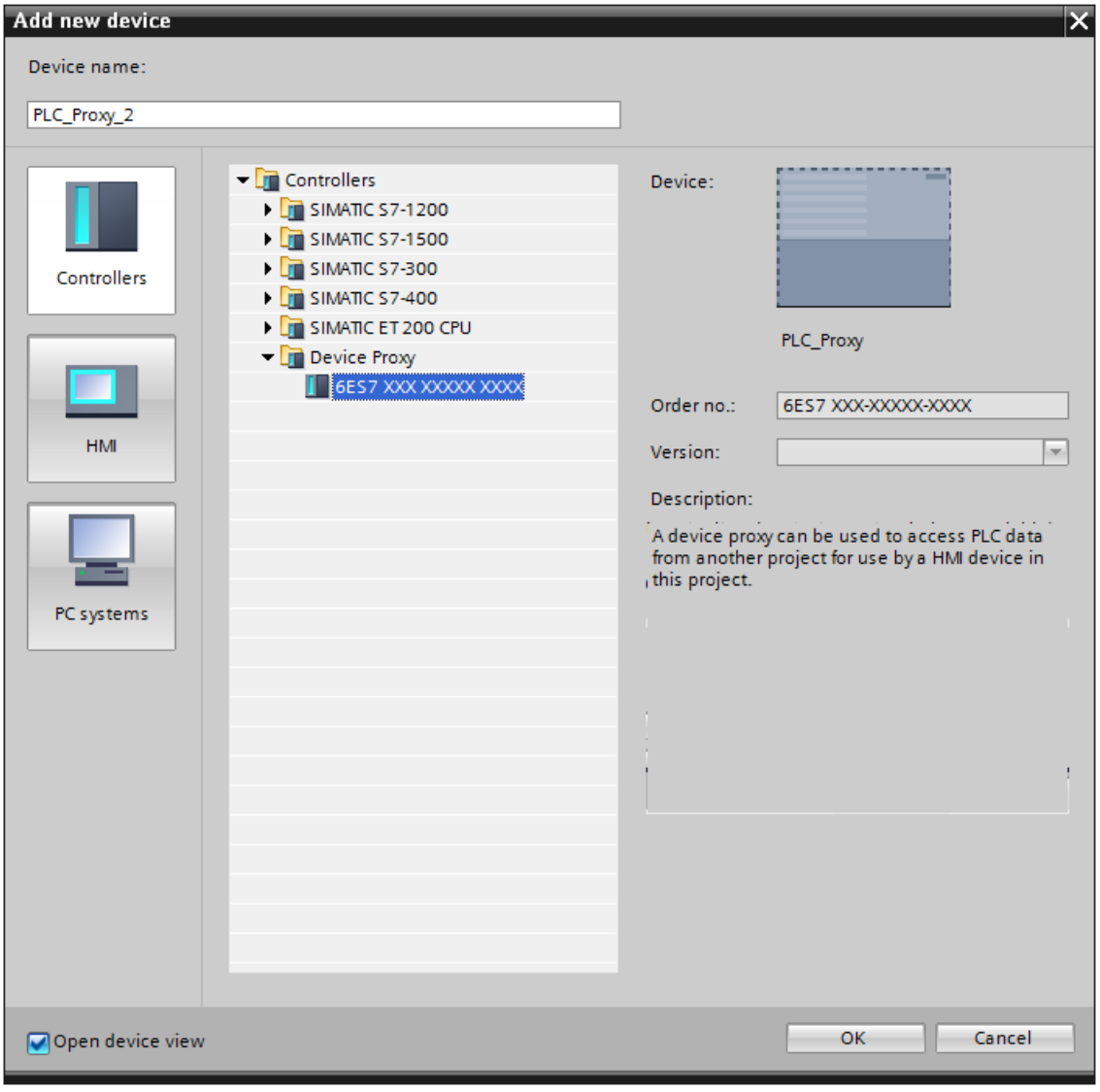
You initialize the device proxy using a project file.

Requirement

Project file (*.ap13) exists.

Procedure

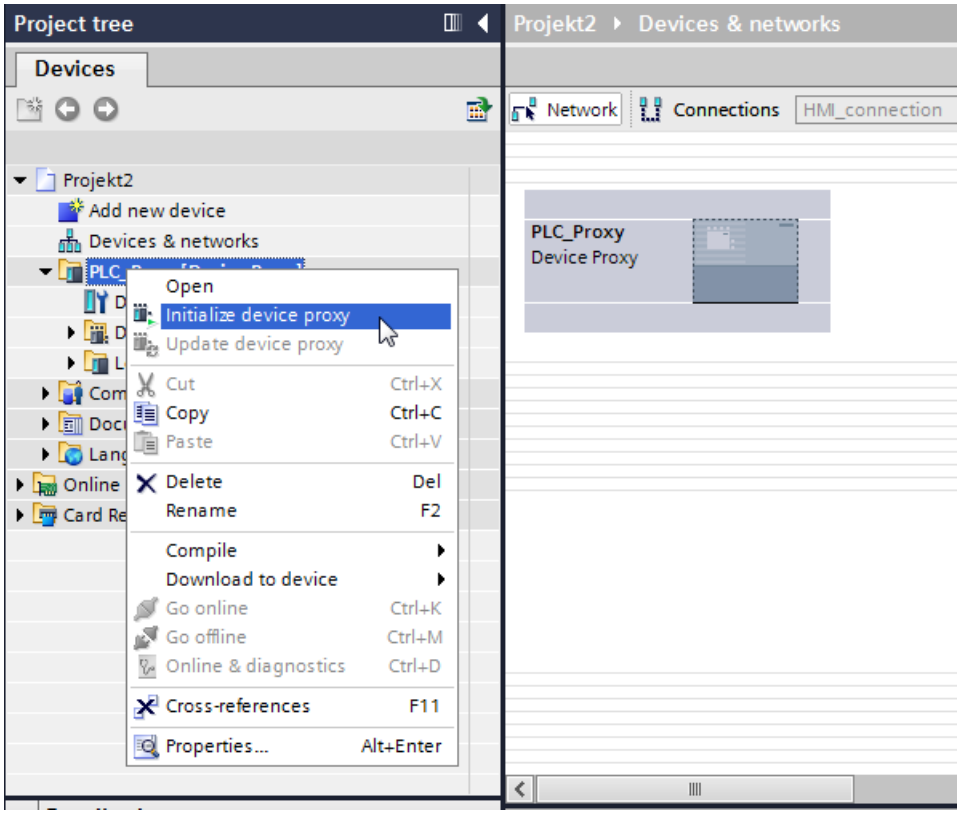
1. Double-click "Add new device" in the project tree.
2. Select the device proxy under "Controller".



A new device is created in the "Devices & Networks" editor.

3. Select the device proxy in the project tree.

4. Click "Initialize device proxy" in the shortcut menu.



5. Select the following entry in the "Open device proxy data source" dialog:
   "TIA Portal projects (*.ap13)"
6. Select a project file and click "Open".
   The "Initialize device proxy" dialog opens.
7. Select a device proxy data object already created from an available PLC to initialize the device proxy.
8. Click "OK".

Note

During initialization of the device proxy a check is automatically carried out whether the selected data are consistent and can be imported into the target project. If the source project contains inconsistent data, compile your source project.

Result

Following initialization, the controller data from the project file in the selected device proxy data object is stored in the device proxy.

You can now configure an HMI connection with the device proxy and connect PLC tags from the device proxy with HMI tags, for example.

Initializing via STEP 7 V5.5 projects

You can find more information about importing control data from projects prior to TIA Portal V11 in the section Integrated configuration with WinCC and Simatic Manager (Page 7766).

You can find additional information in the FAQ with entry ID: 73502293 (http://support.automation.siemens.com/WW/view/de/73502293)

Updating a device proxy via a project file

Introduction

If changes have been made to the controller data in the source project of the device proxy, you can update the device proxy in your TIA Portal project.

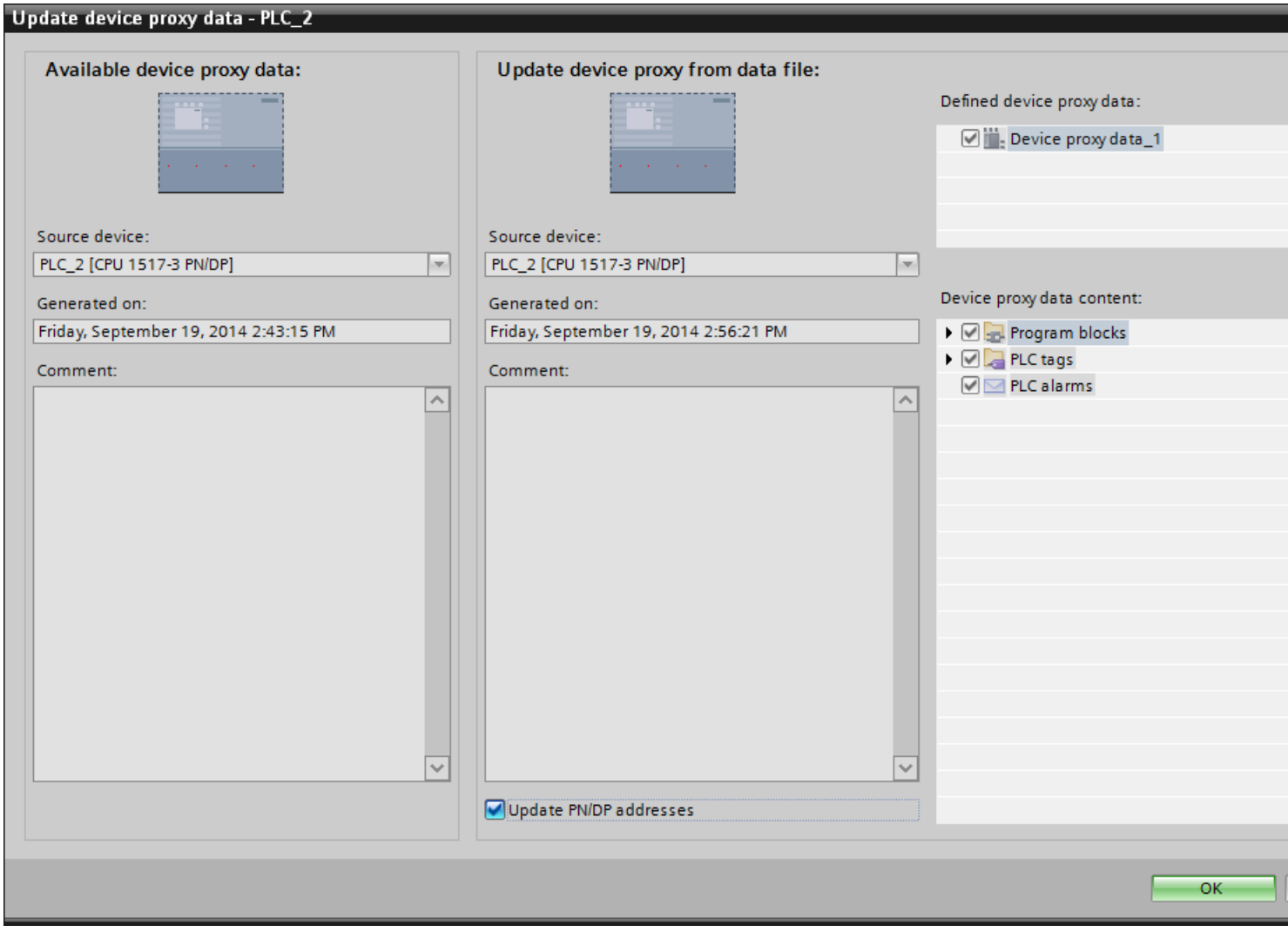
Requirement

- Project file has been generated from the source project of the device proxy.
- A device proxy that has already been initialized is available in the target project.

Procedure

1. Select the device proxy in the project tree.
2. Click "Update device proxy" in the shortcut menu.

3. Select the project file.
   The "Update device proxy" dialog opens.



4. Select a device.

   Note

   If you have created several device proxy data objects in the source project, you need to select one device proxy data object.

   Note

   Already initialized device proxies cannot be overwritten with controller data from a different PLC.

5. Select whether you wish to update the PROFINET or PROFIBUS addresses.
6. Click "OK".

15.2.6.2 Communication with device proxies

Basics of communication with device proxies

Introduction

Data exchange between the HMI device in your target project and existing controller from the source project is enabled in the TIA Portal using device proxies. You initialize a device proxy with device proxy data from the source project and link the HMI device to the device proxy PLC in the target project.

The HMI device and the device proxy PLCs can communicate with each other directly via a subnet or across multiple subnets. For communication across multiple subnets, you need to configure a device proxy-PLC as a router that connects the HMI device and the target PLC.

The communication between connection partners is possible via the following connections:

- PROFINET
- PROFIBUS
- MPI

Setting network parameters

In the TIA Portal, you connect the HMI device and the device proxy PLC in the "Devices & Networks" editor.

Once you have connected an HMI device with the device proxy PLC, adapt the properties of the network parameters. Note the
network parameters from the source project and enter them in the properties of the network connection between the device proxy PLC and the HMI device.

Note

IPE does not transfer data to the network configuration. You therefore configure the required networks in the target project manually so that the network configuration in the target project corresponds to the configuration in the source project. The subnet IDs of the devices in the target project in particular must correspond to each other and to the IDs of the devices in the source project. Otherwise, communication will not be possible between the source device and the HMI devices in runtime

Depending on the connection, provide the following parameters:

Connection	Parameters
Ethernet	S7 subnet ID
MPI PROFIBUS	S7 subnet ID Highest address Transmission rate Bus parameters

Communication via S7 routing

To establish a routed connection to these connection partners, a router must be interposed. Configure both the router and the target PLC as device proxies.

The router and the target PLC can be initialized with an IPE file, TIA Portal project file or a STEP 7 V5.5 project file.

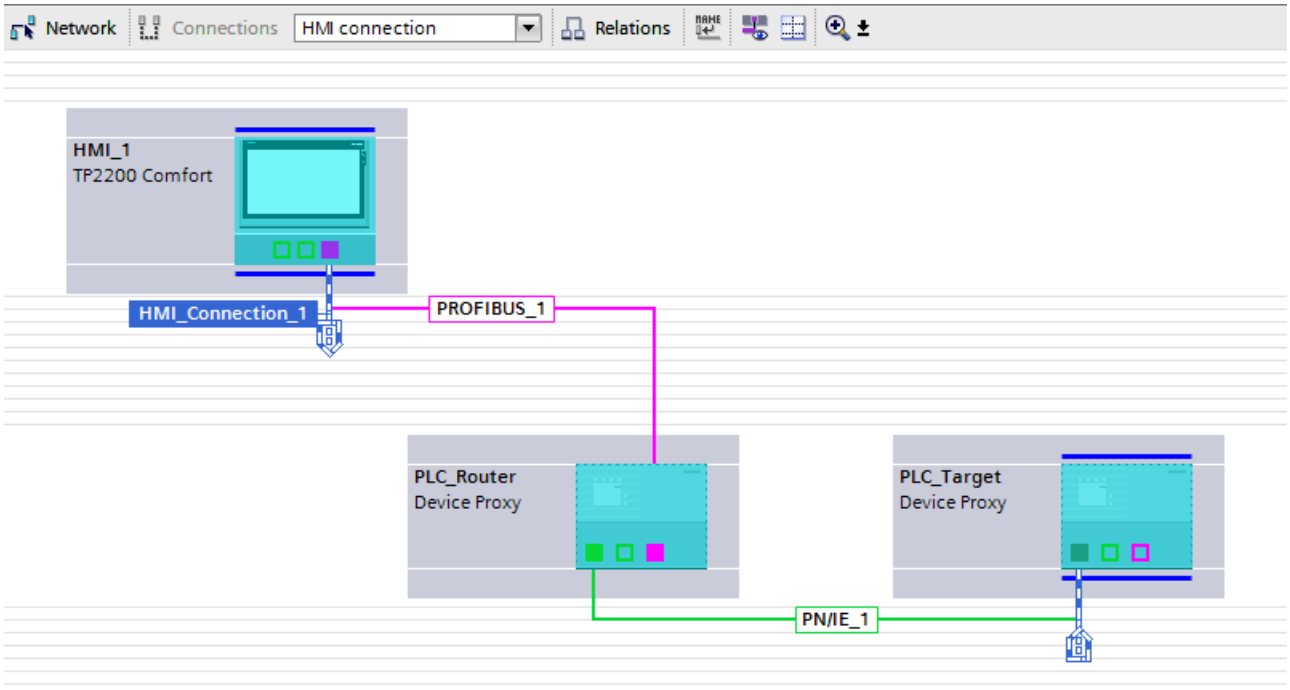
Modules with communication capability (CPUs or CPs) used to establish gateways between the subnetworks must have routing capability.

The S7 routing settings can be edited in the relevant interface properties.

You can configure the S7 routing connection in the "Devices & Networks" editor.

Note

A routed connection must be configured in the source project for a routed connection to a connection partner that was loaded with the source project data.



The routing path is determined in Runtime by the system and cannot be influenced by the user. During configuration, it is not possible to generate information about a faulty connection. All relevant routing paths have to be mapped in the TIA Portal exactly as they were configured in STEP 7 source project.

Configuring a direct connection

Introduction

In the target project, configure a direct connection between a device proxy PLC and an HMI device in "Devices & Networks" editor.
The following configuration describes a network for the following communication partners:

- HMI device
- Device proxy PLC

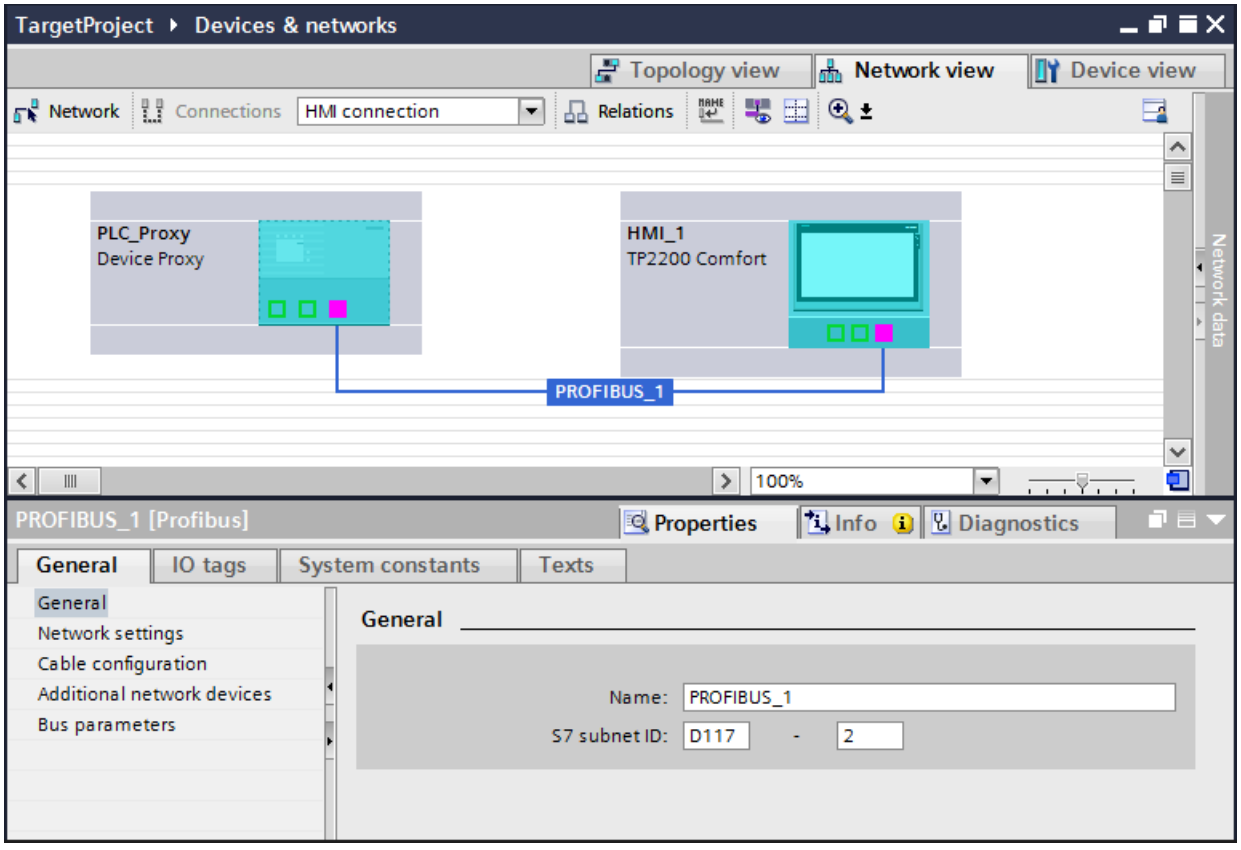
Requirements

- Device proxy data were exported from the source project
- An HMI device has been created in the target project
- A proxy device has been created

Procedure

1. Note the subnet ID of the connection in the source project.
   For an MPI or PROFIBUS connection, also note also the highest address, the transmission speed and the bus parameters.
2. Initialize the device proxy in the target project with the device proxy data from your source project.

3. Connect the communication interfaces of the HMI device and of the device proxy PLC. The corresponding S7 subnet is automatically generated.



4. Transfer the parameters that you noted from the source project into the parameters of the TIA project:
   - Enter the S7 subnet ID under "Properties > General".
   - Enter the highest address and the transmission rate under "Properties > Network Settings".
   - Enter the bus parameters under "Properties > Bus parameters".

Configuring a routed connection

Introduction

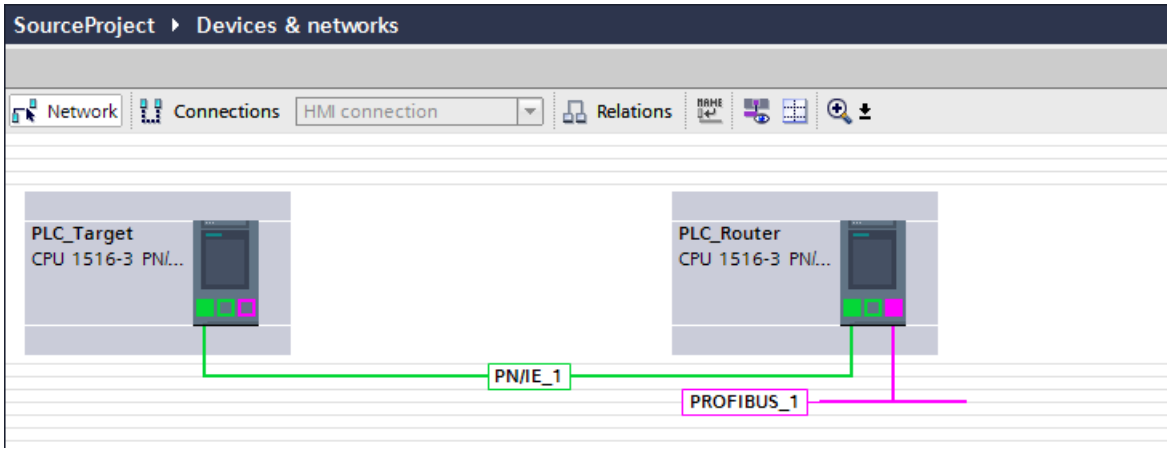
The HMI device and the device proxy PLCs can communicate with each other across multiple subnets. For a routed connection, configure a device proxy PLC as a router between the HMI device and the target PLC.

Note

For a routed connection to the connection partner from the source project, a routed connection must be configured in the source project.

The following configuration describes a network for the following communication partners:

- HMI device
- Device proxy PLC as router
- Device proxy PLC as target PLC
  All subnets that are involved in the routing between the HMI device and the target PLC must be configured in the source project. They must be identically configured in the target project.



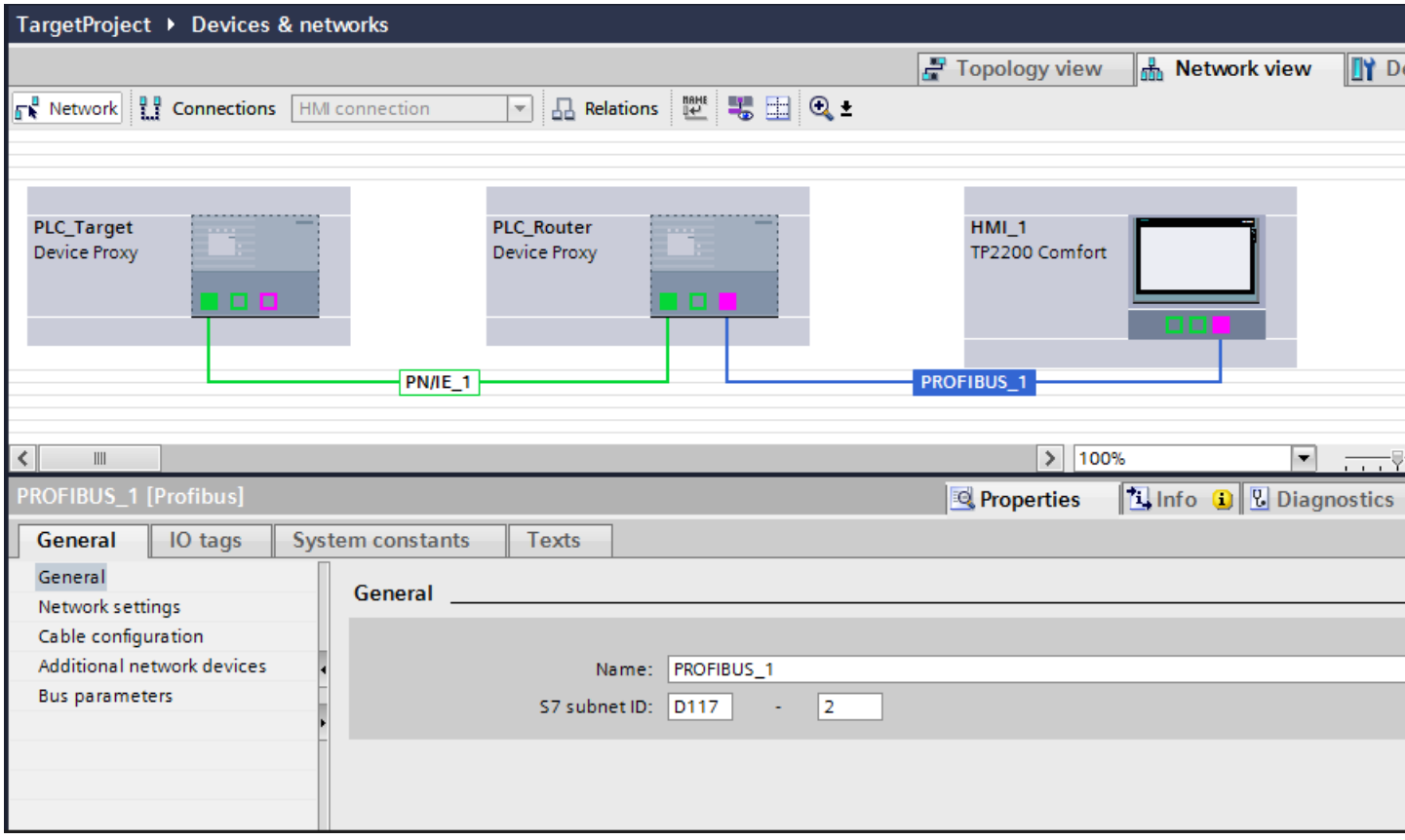
Requirements

- Device proxy data were exported from the source project
- An HMI device has been created
- Two device proxies have been added

Procedure

1. Note the subnet IDs of the connections in the source project.
   For a PROFIBUS connection, also note also the highest address, the transmission speed and the bus parameters.
2. Initialize the source proxy in the target project with the device proxy data from your source project.
3. Create the network connection between the HMI device and the router as well as between the router and the target PLC as it was configured in the source project.
4. Click the "Connections" button.
5. Connect the communication interfaces of the HMI device and of the router.
   Router and target PLC are connected automatically.
6. Create an HMI connection between the HMI device and the target PLC using drag-and-drop operation.
   The "Connect with subnet" dialog opens.

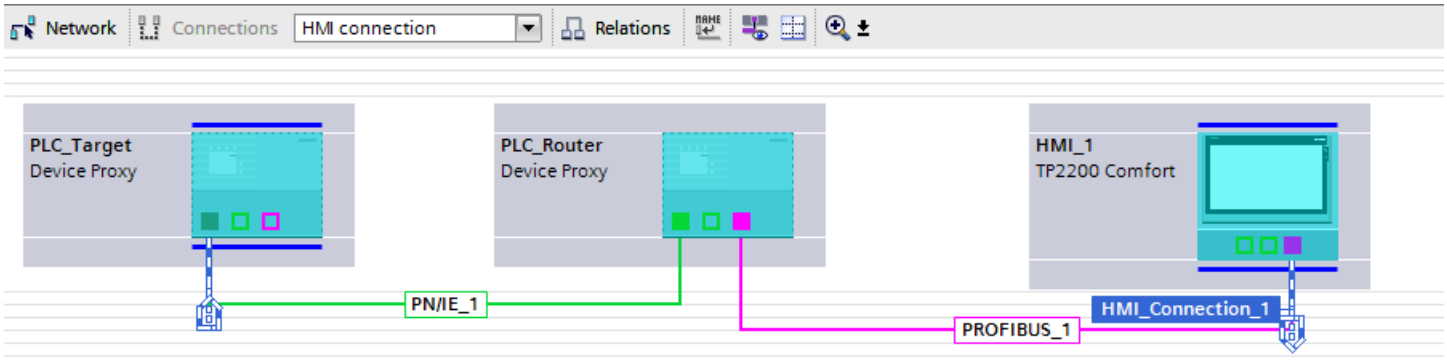
7. Select "Add S7 routed connection".
   The routing connection is created.



8. Transfer the parameters that you noted from the source project into the parameters of the TIA project:
   - Enter the S7 subnet ID under "Properties > General".
   - Enter the highest address and the transmission rate under "Properties > Network Settings".
   - Enter the bus parameters under "Properties > Bus parameters".

Result

The created connection is displayed in the "HMI connections" editor.



15.2.6.3 Integrated configuring with WinCC and SIMATIC Manager

Basic principles for configuring with WinCC and SIMATIC Manager

Integrated configuring with WinCC and SIMATIC Manager

The TIA Portal provides the option to create PLC programs and the HMI configurations using a uniform user interface.

In some cases it is still necessary, for technical or customer-specific reasons, to create the PLC program with the "STEP 7 V5.x" software and the HMI configuration with the WinCC software (TIA Portal).

You can directly access the PLCs configured in the SIMATIC Manager and their current data via the device proxy PLC and integrate this into WinCC (TIA Portal).

Requirement

- WinCC with version V13 or higher
- SIMATIC Manager STEP 7 with version V5.4 SP3 or higher
- In SIMATIC Manager all required software option packages that are used in the STEP 7 project must be installed.
- The project must be consistent in the SIMATIC Manager.

Supported PLCs

- S7-300 PLC
- S7-300F PLC
- S7-400 PLC
- S7-400F PLC

- S7-300 T-PLC
- ET200 PLC (IM 151-x CPU)

Supported alarm procedures

- Discrete alarm types
- Analog alarm types
- Signal system fault (SFM)
- Alarm_S, Alarm_SQ, Alarm_D, Alarm_DQ

Initializing a device proxy via a STEP 7 project file

Introduction

A connection to a CPU within the SIMATIC Manager is established through the initialization. After successful initialization the tags and messages contained in the SIMATIC Manager are also available in the WinCC project.

Initializing a device proxy PLC

1. Initialize the device proxy via the project file (*.s7p) of your STEP 7 project.
   In the "Initialize device proxy" dialog, select the CPU, program blocks, symbols and PLC messages for the device proxy.
2. After successful initialization the selected data are displayed in the program folders in the project tree.

Note

The data are not processed in WinCC (TIA Portal). Only reading access to the tags is available. Changes to the STEP 7 program are always carried out in the SIMATIC Manager.

Note

Support of H-PLCs

IPE functionality in the TIA Portal also supports the use of H-PLCs from STEP 7 as of V5.5. Only the master H-PLC is relevant during the initialization of the device proxy.

Display of the original name of the source device

1. Open the device configuration of the device proxy PLC.
2. Select "Device view"".
3. Open "Properties > General > General > Device proxy information".
   The original name is displayed under "Proxy source".

Creating a network connection between HMI and PLC

As when using a "Standard PLC", you configure a network connection between the used HMI device and the device proxy PLC.

1. In the device configuration select "Network view".
2. Mark the device proxy PLC and enter the same connection properties under "Properties" that the PLC program has in the SIMATIC Manager.
3. Network and connect the two devices.

Note

Network connection and addresses

Irrespective of whether you have created an Ethernet connection or a PROFIBUS connection, check the respective addresses and bus parameters of the modules used and if necessary adapt these. If you use PROFIBUS, observe the additional information under Adapting network parameters (Page 7768).

Adapting network parameters

Introduction

If you have networked an HMI device with the device proxy PLC, you have to adapt the properties of the parameters in the STEP 7 5.x project in the following cases:

- You have a STEP 7 V5.x project. The HMI devices in this project created with WinCC flexible were migrated to WinCC (TIA Portal) and networked with the device proxy PLC.
- You have a STEP 7 V5.x project. The HMI devices were configured only with WinCC (TIA Portal) and are networked with the device proxy PLC.

Depending on the connection, provide the following parameters:

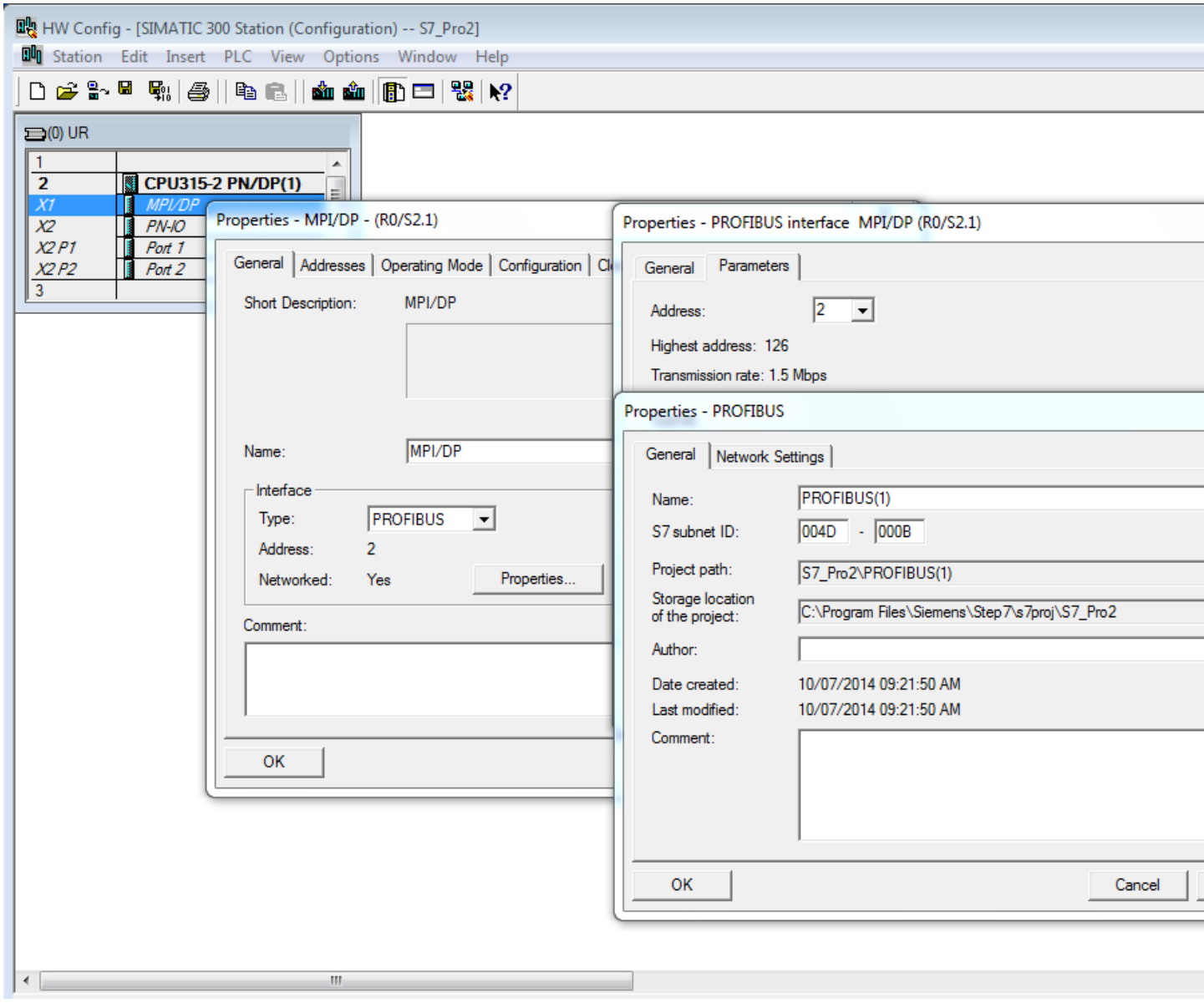
Connection	Parameters
Ethernet	S7 subnet ID
MPI PROFIBUS	S7 subnet ID Highest address Transmission rate Bus parameters

Record the parameters in the STEP 7 project and enter these in the TIA Portal at the properties of the network connection.

Adapting network parameters

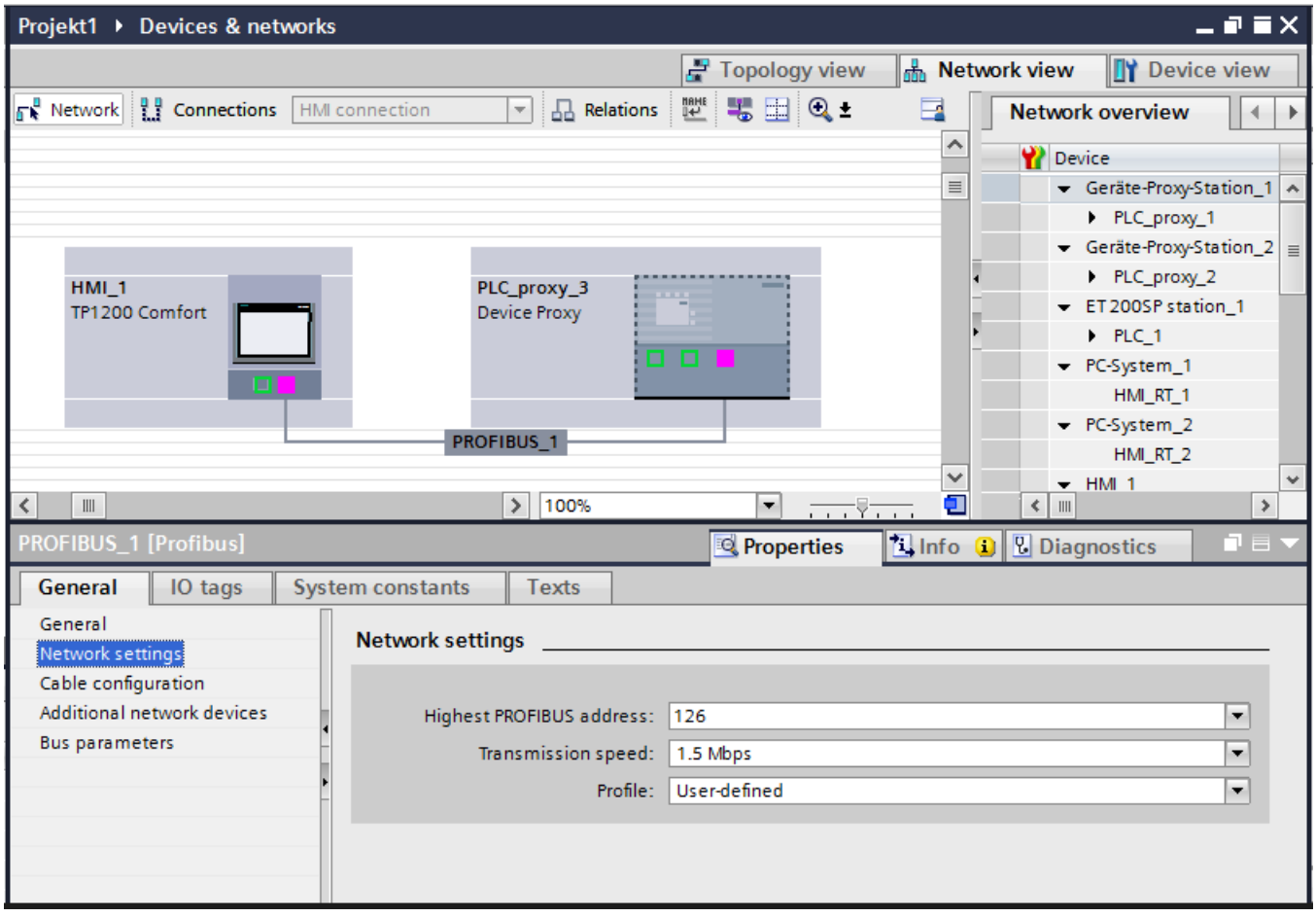
1. Open the hardware configuration of the CPU in the SIMATIC Manager.
2. Open the object properties of the PROFIBUS interface of the CPU.
3. Select the "Properties..." button under "General" in the "MPI/DP properties" dialog. The "Properties – PROFIBUS interface" dialog is opens.

4. Select the "Properties..." button under "Parameters" in the "Properties - PROFIBUS interface" dialog.
   The "Properties - PROFIBUS" dialog is opens.
5. Note the S7 subnet ID.

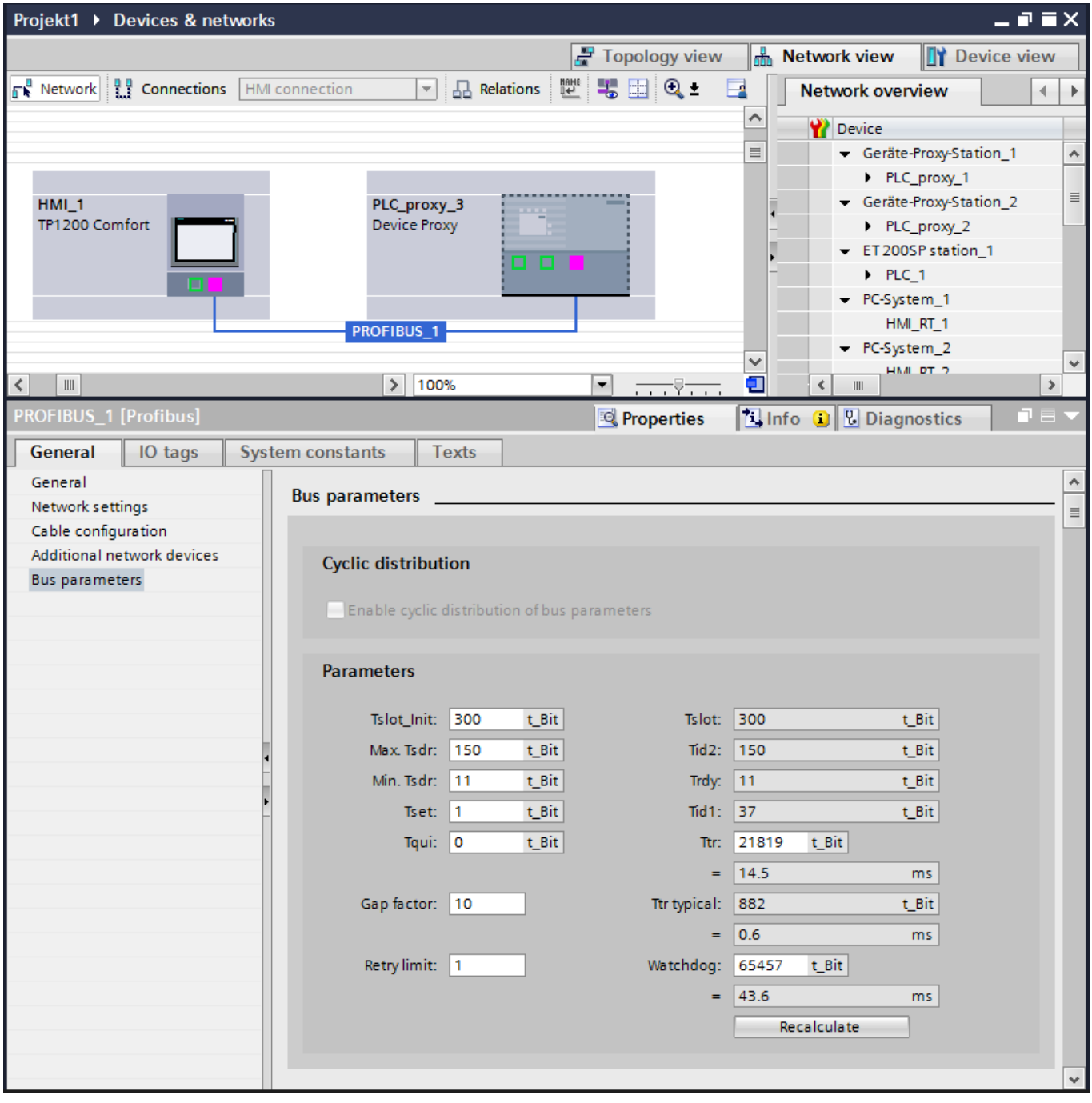


6. If you adapt the PROFIBUS connection, record the "Highest address" and "Transmission rate" parameters in the "Network settings" dialog box.
7. If you adapt the PROFIBUS connection, select the "Bus Parameters" button in the "Properties - PROFIBUS" window under "Network settings".
   Note the bus parameters listed there.
8. In "Devices & Networks" editor of the TIA Portal, select the network connection between the device proxy PLC and the HMI device.

9. Select the "User-defined" profile under "Properties > General> Network Settings".



10. Transfer the parameters that you noted from the STEP 7 configuration into the parameters of the TIA project:
   - Enter the S7 subnet ID under "Properties > General".
   - Enter the highest address and the transmission rate under "Properties > Network Settings".
   - Enter the bus parameters under "Properties > Bus parameters".


Projekt1 ▸ Devices & networks
Topology view
Network view
Device view
Network
Connections
HMI connection
Relations
Network overview
Device
Geräte-Proxy-Station_1
PLC_proxy_1
Geräte-Proxy-Station_2
PLC_proxy_2
ET 200SP station_1
PLC_1
PC-System_1
HMI_RT_1
PC-System_2
HMI_1
TP1200 Comfort
PLC_proxy_3
Device Proxy
PROFIBUS_1
100%
PROFIBUS_1 [Profibus]
Properties
Info
Diagnostics
General
IO tags
System constants
Texts
General
Network settings
Cable configuration
Additional network devices
Bus parameters
Bus parameters
Cyclic distribution
Enable cyclic distribution of bus parameters
Parameters
Tslot_Init: 300 t_Bit
Max. Tsdr: 150 t_Bit
Min. Tsdr: 11 t_Bit
Tset: 1 t_Bit
Tqui: 0 t_Bit
Gap factor: 10
Retry limit: 1
Tslot: 300 t_Bit
Tid2: 150 t_Bit
Trdy: 11 t_Bit
Tid1: 37 t_Bit
Ttr: 21819 t_Bit
= 14.5 ms
Ttr typical: 882 t_Bit
= 0.6 ms
Watchdog: 65457 t_Bit
= 43.6 ms
Recalculate

Checking inconsistent data

Introduction

During initialization of a device proxy a check is automatically carried out whether the selected data are consistent in the source project and can be imported into the target project. If the source project contains inconsistent data, repair them in the source project in the SIMATIC Manager.

The following tools are available for determining and repairing the inconsistent data in STEP 7 V5.x:

- "Check block consistency" for checking data blocks and alarms
- "Symbol Editor" for checking the symbols

Note

You can find additional information on "Check block consistency" and "Symbol editor" in the STEP 7 V5.x documentation.

Checking data blocks and messages

1. Open the source project in the S7 SIMATIC Manager.
2. Mark the block folder in the project window.
3. Select the "Check block consistency..." entry from the shortcut menu.
   The "Check block consistency" editor opens.
4. Select the "Program > Compile" command from the menu.
   Inconsistencies in the blocks are corrected automatically and objects are compiled.
5. If an inconsistency cannot be corrected automatically, an error message is output. By double-clicking the error message you jump to the faulty object. Correct the inconsistencies manually and save the project afterwards.

Checking symbols

1. Open the source project in the S7 SIMATIC Manager.
2. Start the Symbol Editor by double-clicking the symbol table.
   The Symbol Editor is opened.
3. Invalid global symbols in the table are marked in red.
4. Repair or delete the invalid symbols.

Adapting tags and connections

Introduction

Adapt the connection names and synchronize the tags for establishing the connection after the migration.

When the HMI device is networked with the device proxy PLC the system creates a connection and assigns a connection name. If an HMI connection already existed, adapt the new connection name.

Adapting the connection name

1. Copy the previous connection name.
2. Delete the existing connection.
3. Replace the new connection name by the previous name.

Symbolic connection of the tags

1. Call up the tag editor.
2. Select all the tags.
3. Click "Synchronize to compare with PLC tag".
   The "Options for synchronizing WinCC tags" window opens.
4. Select the options "Data type and absolute address agree" and "Replace WinCC tag name by the PLC tag name" in the window.
5. Click "Synchronize".
   Synchronization of the tags is carried out.

Updating a device proxy via a STEP 7 project file

Introduction

The data of the device proxy PLC have to be updated whenever changes are made in the SIMATIC Manager that affect the HMI device. This is, for example, the case when address area in the data blocks have been changed or extended.

Updating device proxy data

1. Update the data of the device proxy via the project file (*.s7p) of your STEP 7 project.
2. In the update dialog. select the same CPU that you also used during the initialization of the device proxy PLC as the data source.
   An update with a different hardware configuration is not allowed.

3. Through the dialog you have the possibility to select newly created data blocks from the SIMATIC Manager additionally.
   Only selected objects are imported in the TIA Portal project.

   Note

   All the data that were selected during the previous initialization are automatically selected again as well. You should only change this default setting if you consciously want to remove data from the device proxy PLC. If, for example, the DB1 was included in the TIA Portal project through the initialization, and you now deactivate the DB1, the DB1 is deleted from the TIA Portal project during updating.

4. Confirm the selection using the "OK" button.
   Updating of the tags is started.
5. The addresses of the tags were adapted automatically through the symbolic connection of the tags.
   This completes updating of the tags.

Including tags from the SIMATIC Manager

Introduction

Through the initialization of the device proxy PLC you have included the tags from the SIMATIC Manager into the WinCC project.

In WinCC, you insert tags of the device proxy PLC directly into an HMI screen or a screen object.

Inserting a tag into the HMI configuration

1. Open the folder with the program blocks of the device proxy PLC in the project tree.
2. Select the block that contains the tags.
3. All the tags of the selected block are displayed in the detail view.
4. Drag&drop the tag from the detail view into the working area.
   An I/O field with the selected tag is created. Carry out further settings in the Inspector window.

Alternatively configure an I/O field in the HMI screen and drag&drop the tag into the I/O field.

Result

The tag from a block of the device proxy PLC is inserted into the HMI project and connected with the I/O field.

Configuring the "Direct keys" system function

Introduction

If you want to integrate HMI devices that can only be configured with WinCC (TIA Portal), for example SIMATIC HMI Comfort Panels, into the hardware configuration of the SIMATIC Manager, you require the corresponding matching GSD/GSDML files.

The required GSD/GSDM files for WinCC are located in FAQs mutual configuration with WinCC (TIA Portal) and STEP 7 V5.x (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=4&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid4=20229806&objaction=csopen)

Installing GSD files

1. Open the hardware configuration in the SIMATIC Manager.
2. Select the "Options > Install New GSD Files" menu command.
   The "Install GSD files" dialog is displayed.
3. Use the "Browse" button to select the file folder that contains the GSD files.

   Note

   PROFINET / PROFIBUS storage paths in the SIMATIC Manager

   The GSDML files for the HMI devices are located in the hardware catalog under „PROFINET IO > HMI > SIMATIC HMI > GSD > KP/x“.

   The GSD files for the HMI devices are located in the hardware catalog under „PROFIBUS DP > Weitere FELDGERÄTE > MMI > SIMATIC_HMI > HMI CP_x“.

4. Mark all the files and click the "Install" command button.

PROFINET connection: PROFINET IO direct keys

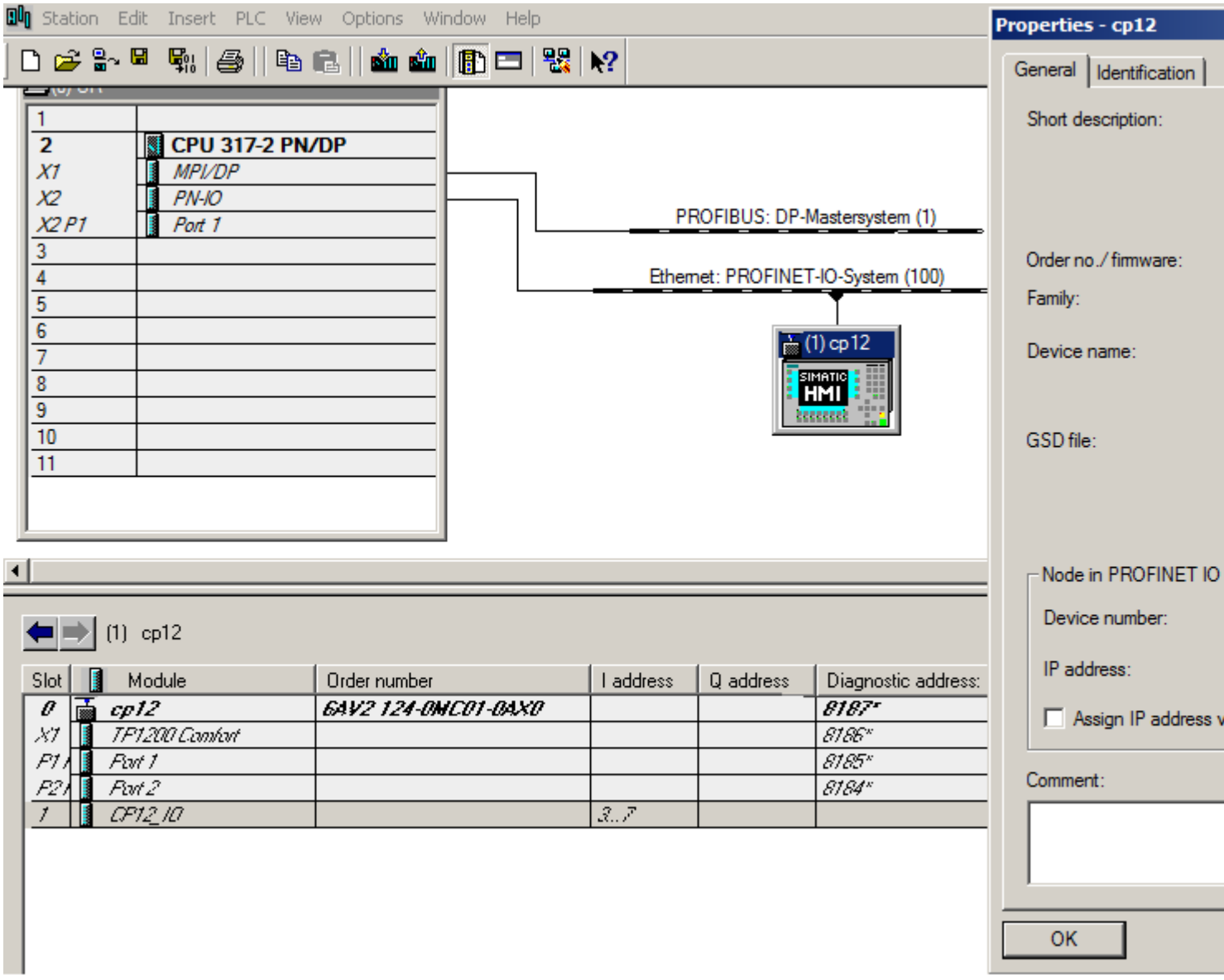
1. Note the properties used for the HMI device to be replaced.
   Take particular note of I/O addresses, device name, device number and diagnostic addresses if these are evaluated.
2. Remove the existing GSD file.
3. Drag&Drop the configured HMI device type from the hardware catalog onto the displayed PROFINET IO bus.
4. Adapt the properties as noted.

   Note

   Take the rules for the names of PROFINET IO devices into consideration for the device name. Use only lower-case characters and no special characters for the device name.

5. Confirm the entries with OK.

6. Save and compile the configuration and transfer the hardware configuration to the controller.



7. Open the device configuration of the TP1200 Comfort and change to the "Device view".
8. Enter the PROFINET device name in the properties of the configured HMI device in WinCC under "Properties > General > PROFINET interface X1 > Ethernet addresses".
   To assign the PROFINET device name, deselect the option "Generate PROFINET device name automatically".

Note

The PROFINET name in the WinCC configuration must correspond to the PROFINET name stored in the SIMATIC Manager.

PROFIBUS connection: PROFIBUS DP direct keys

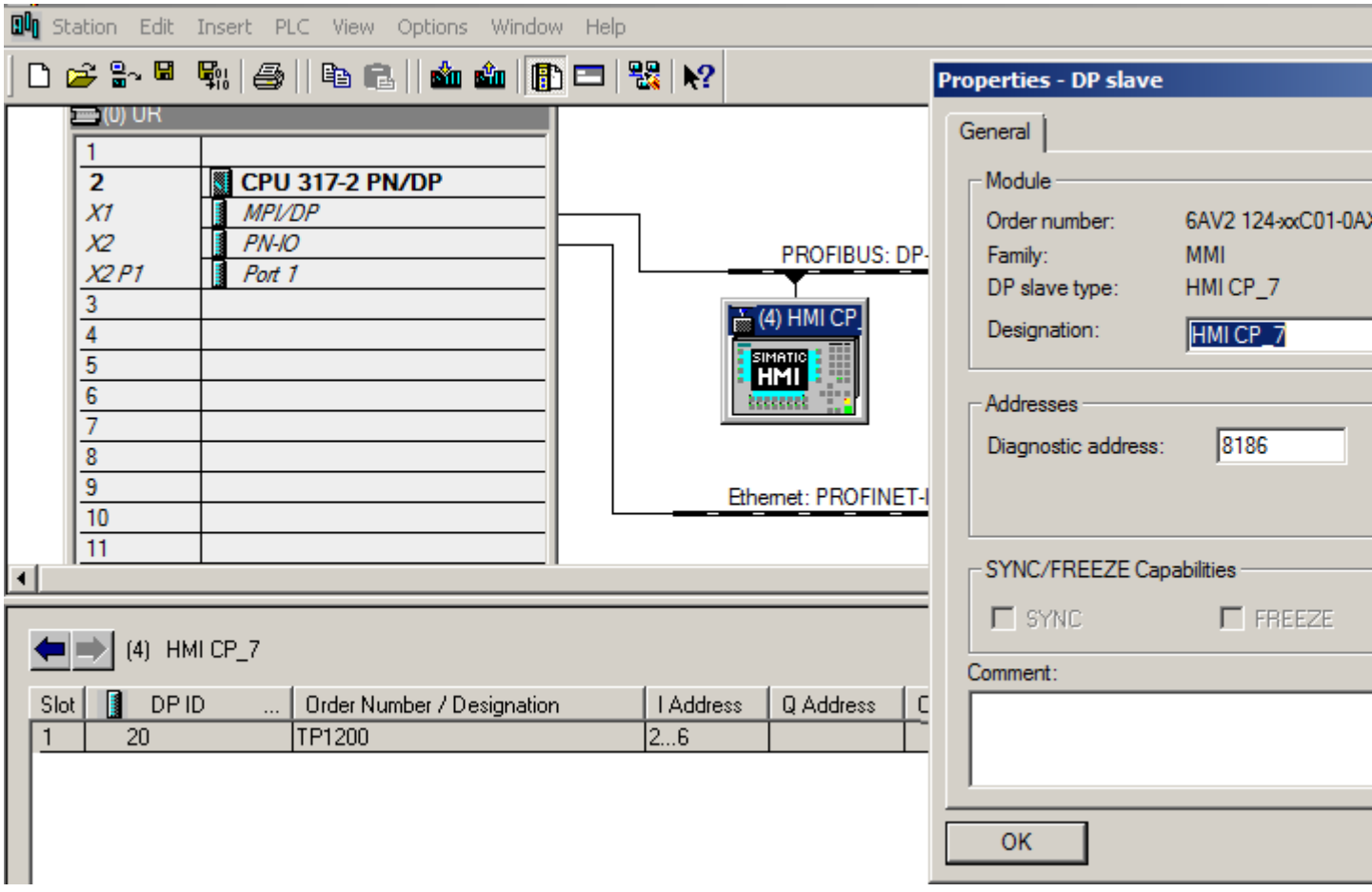
Please note that the direct keys are configured in the SIMATIC Manager.

Note

If direct keys to several CPUs are configured from one panel, the direct keys will only function on one CPU.

The further CPUs display a bus / group error.

1. Note the properties used for the HMI device to be replaced.
   Take particular note of I/O addresses, device name and diagnostic addresses if these are evaluated.
2. Remove the existing GSD file.
3. Drag-and-drop the GSD module that contains the configured HMI device type from the hardware catalog to the PROFIBUS DP bus displayed.
4. Adapt the properties I/O address, device name, PROFIBUS address and diagnostic address as previously noted.
5. Confirm the entries with OK.
6. Save and compile the configuration and transfer the hardware configuration to the controller.



7. Open the device configuration of the HMI device that you want to use and change to the "Device view".
8. Network the PROFIBUS interface in the properties of the configured HMI device under "Properties > General > MPI/DP interface X2 > PROFIBUS address > Network interface with".
9. Specify the PROFIBUS address at "Parameters".

Note

The PROFIBUS address in the WinCC configuration must correspond to the PROFIBUS address stored in the SIMATIC Manager.

Hardware documentation 16

16.1 General information on the hardware documentation

Additional information on the available hardware

The TIA Portal can be used to configure a wide variety of hardware, depending on the installed products. You can find the available hardware in the hardware catalog. You can find all the current manuals, operating instructions and FAQs as well as updates for your devices in the Siemens Industry Online Support (https://support.automation.siemens.com/).

To help you locate the appropriate documents for your hardware in the Siemens Industry Online Support, the following sections list all modules and module families that are available in the current scope of installation of the TIA Portal. You will find a link for each module that takes you directly to the relevant manuals and operating instructions in the Siemens Industry Online Support.

16.2 HMI

16.2.1 Basic Panels

16.2.1.1 Basic Panels

Information about Basic Panels is available here (http://support.automation.siemens.com/WW/view/en/28426379/133300).

16.2.2 Panels

16.2.2.1 Panels of the 70 series

Information about Basic Panels of the 70 series is available here (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid0=15271786&objaction=csopen).

16.2.2.2 Panels of the 170 series

Information about panels of the 170 series is available here (http://support.automation.siemens.com/WW/view/en/10805566/133300).

16.2.2.3 Panels of the 270 series

Information about panels of the 270 series is available here (http://support.automation.siemens.com/WW/view/en/10805567/133300).

16.2.3 Comfort Panels

16.2.3.1 Comfort Panels

Information about Comfort Panels is available here (http://support.automation.siemens.com/WW/view/en/47182890/133300).

16.2.4 Multi Panels

16.2.4.1 170 series

Information about Multi Panels of the 170 series is available here. (http://support.automation.siemens.com/WW/view/en/28421795/133300)

16.2.4.2 270 series

Information about Multi Panels of the 270 series is available here (http://support.automation.siemens.com/WW/view/en/10805569/133300).

16.2.4.3 370 series

Information about Multi Panels of the 370 series is available here (http://support.automation.siemens.com/WW/view/en/10805570/133300).

16.2.5 Mobile Panels

16.2.5.1 170 series

Information about Mobile Panels of the 170 series is available here (http://support.automation.siemens.com/WW/view/en/26268543/133300).

16.2.5.2 270 series

Information about Mobile Panels of the 270 series is available here (http://support.automation.siemens.com/WW/view/en/22584001/133300).

16.2.6 Key Panels

16.2.6.1 Key Panels

Information about Key Panels is available here (http://support.automation.siemens.com/WW/view/de/47416561/0/en).

16.2.6.2 Push Button Panels

Information about Push Button Panels is available here (http://support.automation.siemens.com/WW/view/en/19860219/133300).

16.2.7 WinAC for Multi Panels

16.2.7.1 WinAC for Multi Panel

Information about WinAC MP is available here (http://support.automation.siemens.com/WW/view/en/10997567/130000).

16.2.8 PC-based Automation

Information about PC-based automation is available here (http://support.automation.siemens.com/WW/view/en/42728754/130000).

16.3 Controller

16.3.1 SIMATIC S7-1200

16.3.1.1 CPU

CPU 1211C (6ES7 211-1xxx-0XB0)

Information on the CPUs is available here:

- CPU 1211C AC/DC/Rly (6ES7 211-1BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C AC/DC/Rly (6ES7 211-1BE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111BE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C AC/DC/Rly (6ES7 211-1BE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111BE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C DC/DC/DC (6ES7 211-1AD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111AD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C DC/DC/DC (6ES7 211-1AE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111AE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C DC/DC/DC (6ES7 211-1AE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111AE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C DC/DC/Rly (6ES7 211-1HD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111HD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- CPU 1211C DC/DC/Rly (6ES7 211-1HE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111HE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C DC/DC/Rly (6ES7 211-1HE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111HE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

CPU 1212C (6ES7 212-1xxx-0XB0)

Information on the CPUs is available here:

- CPU 1212C AC/DC/Rly (6ES7 212-1BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C AC/DC/Rly (6ES7 212-1BE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121BE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C AC/DC/Rly (6ES7 212-1BE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121BE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C DC/DC/DC (6ES7 212-1AD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121AD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C DC/DC/DC (6ES7 212-1AE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121AE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C DC/DC/DC (6ES7 212-1AE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121AE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C DC/DC/Rly (6ES7 212-1HD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121HD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- CPU 1212C DC/DC/Rly (6ES7 212-1HE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121HE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C DC/DC/Rly (6ES7 212-1HE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121HE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

CPU 1214C (6xxx 214-1xxx-xXB0)

Information on the CPUs is available here:

- CPU 1214C AC/DC/Rly (6ES7 214-1BE30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141BE300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C AC/DC/Rly (6ES7 214-1BG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141BG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C AC/DC/Rly (6ES7 214-1BG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141BG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C DC/DC/DC (6ES7 214-1AE30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141AE300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C DC/DC/DC (6ES7 214-1AG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141AG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C DC/DC/DC (6ES7 214-1AG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141AG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C DC/DC/Rly (6ES7 214-1HE30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141HE300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- CPU 1214C DC/DC/Rly (6ES7 214-1HG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141HG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C DC/DC/Rly (6ES7 214-1HG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141HG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS CPU 1214C DC/DC/RLY (6AG1 214-1HG40-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12141HG405XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

CPU 1215C (6xxx 215-1xxx-xXB0)

Information on the CPUs is available here:

- CPU 1215C AC/DC/Rly (6ES7 215-1BG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151BG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1215C AC/DC/Rly (6ES7 215-1BG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151BG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1215C DC/DC/DC (6ES7 215-1AG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151AG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1215C DC/DC/DC (6ES7 215-1AG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151AG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1215C DC/DC/Rly (6ES7 215-1HG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151HG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1215C DC/DC/Rly (6ES7 215-1HG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151HG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- SIPLUS CPU 1215C DC/DC/DC (6AG1 215-1AG40-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12151AG402XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS CPU 1215C DC/DC/DC (6AG1 215-1AG40-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12151AG404XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS CPU 1215C DC/DC/DC (6AG1 215-1AG40-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12151AG405XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS CPU 1215C AC/DC/RLY (6AG1 215-1BG40-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12151BG402XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS CPU 1215C AC/DC/RLY (6AG1 215-1BG40-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12151BG404XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS CPU 1215C AC/DC/RLY (6AG1 215-1BG40-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12151BG405XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS CPU 1215C DC/DC/RLY (6AG1 215-1HG40-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12151HG402XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS CPU 1215C DC/DC/RLY (6AG1 215-1HG40-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12151HG404XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS CPU 1215C DC/DC/RLY (6AG1 215-1HG40-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12151HG405XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

CPU 1217C (6ES7 217-1xxx-0XB0)

Information on the 1217C DC/DC/DC CPU is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72171AG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1214FC (6ES7 214-1xF40-0XB0)

Information on the CPUs is available here:

- CPU 1214FC DC/DC/DC (6ES7 214-1AF40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141AF400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214FC DC/DC/Rly (6ES7 214-1HF40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141HF400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

CPU 1215FC (6ES7 215-1xF40-0XB0)

Information on the CPUs is available here:

- CPU 1215FC DC/DC/DC (6ES7 215-1AF40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151AF400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1215FC DC/DC/Rly (6ES7 215-1HF40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151HF400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

16.3.1.2 Signal boards

Signal boards (6xxx 2xx-xxx30-xXB0)

Information on signal boards for S7-1200 is available here:

- DI 4x24VDC (6ES7 221-3BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72233BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- DI 4x5VDC (6ES7 221-3AD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72213AD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DQ 4x24VDC (6ES7 222-1BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- DQ 4x5VDC (6ES7 222-1AD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221AD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- DI 2/DQ 2x24VDC (6ES7 223-0BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72230BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- DI 2/DQ 2x24VDC (6ES7 223-3BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72213BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI 2/DQ 2x5VDC (6ES7 223-3AD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72233AD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- AI 1x12BIT (6ES7 231-4HA30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72314HA300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- AI 1xRTD (6ES7 231-5PA30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315PA300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- AI 1xTC (6ES7 231-5QA30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315QA300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))

- AQ 1x12BIT (6ES7 232-4HA30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72324HA300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- SIPLUS SB 1221, 4DI, 5VDC (6AG1 221-3AD30-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12213AD305XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- SIPLUS SB 1221 4DI (6AG1 221-3BD30-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12213BD305XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- SIPLUS SB1222 4DQ 5VDC (6AG1 222-1AD30-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221AD305XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- SIPLUS SB 1222 4DQ (6AG1 222-1BD30-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221BD305XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- SIPLUS SB 1223 2DI/ 2DO (6AG1 223-0BD30-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12230BD304XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- SIPLUS SB 1223 2DI/ 2DO (6AG1 223-0BD30-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12230BD305XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- SIPLUS SB 1223 2DI/2DQ (6AG1 223-3AD30-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12233AD305XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- SIPLUS SB 1223, 2DI/2DQ, 24VDC (6AG1 223-3BD30-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12233BD305XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))

16.3.1.3 Communication boards

Point-to-point

CB 1241 (6ES7 241-1CH30-1XB0)

Information on the CB 1241 communication board is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411CH301XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.3.1.4 Battery boards

BB 1297 (6ES7 297-0AX30-0XA0)

Information on the BB 1297 communication board is available here (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=100&lang=en&referer=%2fWW%2f&func=cslib.cssearch&nodeid0=41886045&viewreg=WW&siteid=csius&extranet=standard&groupid=4000002&objaction=cssearch&content=adsearch%2Fadsearch%2Easpx).

16.3.1.5 Digital input modules

Digital input modules (6xxx 221-1Bx3x-xXB0)

Information on the digital input modules is available here:

- SM 1221 DI8 x 24VDC (6ES7 221-1BF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72211BF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1221 DI8 x 24VDC (6ES7 221-1BF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72211BF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- SM 1221 DI16 x 24VDC (6ES7 221-1BH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72211BH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1221 DI16 x 24VDC (6ES7 221-1BH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72211BH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1221 8DI (6AG1 221-1BF32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12211BF322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1221 8DI (6AG1 221-1BF32-2XY0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12211BF322XY0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1221 8DI (6AG1 221-1BF32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12211BF324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1221 16DI (6AG1 221-1BH32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12211BH322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1221 16DI (6AG1 221-1BH32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12211BH324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

SM 1226 F-DI 8/16 x 24 VDC (6ES7 226-6BA32-0XB0)

You will find information on the SM 1226 F-DI 8/16 x 24 VDC digital input module here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72266BA320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.3.1.6 Digital output modules

Digital input modules (6xxx 222-1xx3x-xXB0)

Information on the digital output modules is available here:

- SM 1222 DQ 8x24VDC (6ES7 222-1BF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221BF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1222 DQ 8x24VDC (6ES7 222-1BF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221BF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1222 DQ 16x24VDC (6ES7 222-1BH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221BH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1222 DQ 16x24VDC (6ES7 222-1BH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221BH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1222 DQ 16xRelay (6ES7 222-1HH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221HH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1222 DQ 16xRelay (6ES7 222-1HH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221HH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1222 DQ 8xRelay (6ES7 222-1HF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221HF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1222 DQ 8xRelay (6ES7 222-1HF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221HF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1222 DQ 8xNO/NC Relay (6ES7 222-1XF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221XF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1222 DQ 8xNO/NC Relay (6ES7 222-1XF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221XF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- SIPLUS SM 1222 8DQ (6AG1 222-1BF32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221BF322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1222 8DQ (6AG1 222-1BF32-2XY0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221BF322XY0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1222 8DQ (6AG1 222-1BF32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221BF324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1222 16DQ (6AG1 222-1BH32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221BH322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1222 16DQ (6AG1 222-1BH32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221BH324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1222 8DQ RLY (6AG1 222-1HF32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221HF322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1222 8DQ RLY (6AG1 222-1HF32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221HF324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1222 16DQ RLY (6AG1 222-1HH32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221HH322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1222 16DQ RLY (6AG1 222-1HH32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221HH324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1222 8DQ RLY (6AG1 222-1XF32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221XF322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1222 8DQ RLY (6AG1 222-1XF32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12221XF324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

SM 1226 F-DQ 4 x 24 VDC (6ES7 226-6DA32-0XB0)

Information on the SM 1226 F-DQ 4 x 24 VDC digital input module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72266DA320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view):

SM 1226 F-DQ 2 x Relay (6ES7 226-6RA32-0XB0)

Information on the SM 1226 F-DQ 2 x Relay digital output module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72266RA320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view):

16.3.1.7 Digital input and digital output modules

Digital input and output modules (6xxx 223-1xx3x-xXB0)

Information on the digital input and output modules is available here:

- SM 1223 DI 8/DQ 8x24VDC (6ES7 223-1BH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231BH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1223 DI 8/DQ 8x24VDC (6ES7 223-1BH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231BH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1223 DI 16/DQ 16x24VDC (6ES7 223-1BL30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231BL300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1223 DI 16/DQ 16x24VDC (6ES7 223-1BL32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231BL320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- SM 1223 DI 8x24VDC/DQ 8xRelay (6ES7 223-1PH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231PH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1223 DI 8x24VDC/DQ 8xRelay (6ES7 223-1PH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231PH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1223 DI 16x24VDC/DQ 16xRelay (6ES7 223-1PL30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231PL300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1223 DI 16x24VDC/DQ 16xRelay (6ES7 223-1PL32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231PL320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1223 DI/DO 8x120VAC/DQ 8xRelay (6ES7 223-1QH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231QH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1223 DI/DO 8x120VAC/DQ 8xRelay (6ES7 223-1QH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231QH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 8DI/8DQ (6AG1 223-1BH32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231BH322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 8DI/8DQ (6AG1 223-1BH32-2XY0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231BH322XY0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 8DI/8DQ (6AG1 223-1BH32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231BH324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 16DI/16DQ (6AG1 223-1BL32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231BL322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 16DI/16DQ (6AG1 223-1BL32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231BL324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- SIPLUS SM 1223 8DI/8DQ RLY (6AG1 223-1PH32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231PH322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 8DI/8DQ RLY (6AG1 223-1PH32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231PH324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 16DI/16DQ RLY (6AG1 223-1PL32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231PL322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 16DI/16DQ (6AG1 223-1PL32-2XY0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231PL322XY0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 16DI/16DQ RLY (6AG1 223-1PL32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231PL324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 8DI AC/8DQ RLY (6AG1 223-1QH32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231QH322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1223 8DI AC/8DQ RLY (6AG1 223-1QH32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12231QH324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

16.3.1.8 Analog input modules

Analog input modules (6xxx 231-xxx3x-xXB0)

Information on the analog input modules is available here:

- SM 1231 AI 4x13BIT (6ES7 231-4HD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72314HD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 4x13BIT (6ES7 231-4HD32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72314HD320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 4x16BIT (6ES7 231-5ND30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315ND300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 4x16BIT (6ES7 231-5ND32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315ND320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 8x13BIT (6ES7 231-4HF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72314HF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 8x13BIT (6ES7 231-4HF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72314HF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 4xRTD (6ES7 231-5PD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315PD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- SM 1231 AI 4xRTD (6ES7 231-5PD32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315PD320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 8xRTD (6ES7 231-5PF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES723145PF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 8xRTD (6ES7 231-5PF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315PF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 4xTC (6ES7 231-5QD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315QD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 4xTC (6ES7 231-5QD32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315QD320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 8xTC (6ES7 231-5QF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315QF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1231 AI 8xTC (6ES7 231-5QF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315QF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 4AI 13Bit (6AG1 231-4HD32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12314HD324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 8AI 13Bit (6AG1 231-4HF32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12314HF324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 4AI 16Bit (6AG1 231-5ND32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12315ND322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 4AI 16Bit (6AG1 231-5ND32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12315ND324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- SIPLUS SM 1231 4AI RTD 16Bit (6AG1 231-5ND32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12315PD322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 4AI RTD 16Bit (6AG1 231-5PD32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12315PD324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 8AI RTD 16Bit (6AG1 231-5PF32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12315PF322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 8AI RTD 16Bit (6AG1 231-5PF32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12315PF324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 4AI TC 16Bit (6AG1 231-5QD32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12315QD322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 4AI TC 16Bit (6AG1 231-5QD32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12315QD324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 8AI TC 16Bit (6AG1 231-5QF32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12315QF322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1231 8AI TC 16Bit (6AG1 231-5QF32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12315QF324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

16.3.1.9 Analog output modules

Analog input modules (6xxx 234-4Hx3x-xXB0)

Information on the analog output modules is available here:

- SM 1232 AQ 2x14BIT (6ES7 232-4HB30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72344HB300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1232 AQ 2x14BIT (6ES7 232-4HB32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72344HB320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1232 AQ 4x14BIT (6ES7 232-4HD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72344HD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1232 AQ 4x14BIT (6ES7 232-4HD32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72344HD320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SB 1232 1AQ (6AG1 232-4HA30-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12324HA304XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SB 1232 1AQ (6AG1 232-4HA30-5XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12324HA305XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1232 2AQ 13Bit (6AG1 232-4HB32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12324HB324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- SIPLUS SM 1232 4AQ 14Bit (6AG1 232-4HD32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12324HD322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1232 4AQ 14Bit (6AG1 232-4HD32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12324HD324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

16.3.1.10 Analog input and analog output modules

Analog input and output module (6xxx 234-4HE3x-xXB0)

Information on the analog input and output modules is available here:

- SM 1234 AI 4x13BIT/AQ 2x14BIT (6ES7 234-4HE30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12344HE324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SM 1234 AI 4x13BIT/AQ 2x14BIT (6ES7 234-4HE32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72324HB320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1234 4AI/2AQ (6AG1 234-4HE32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12344HE322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1234 4AI/2AQ (6AG1 234-4HE32-2XY0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12344HE322XY0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- SIPLUS SM 1234 4AI/2AQ 13Bit (6AG1 234-4HE32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72324HB300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

16.3.1.11 Communications modules

Industrial Remote Communication

CP 1242-7 (6xxx 242-7KX30-xXE0)

You will find information on the telecontrol communications modules here:

- CP 1242-7 6AG1 242-7KX30-4XE0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK72427KX300XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CP 1242-7 6AG1 242-7KX30-4XE0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12427KX304XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

CP 1243-1 (6GK7 243-1JX30-0XE0)

You will find information on the Ethernet communications modules here:

- CP 1243-1 6GK7 243-1JX30-0XE0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK72431JX300XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CP 1243-1 6AG1 243-1JX30-7XE0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12431JX307XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

CM 1243-7 (6GK7 243-7xX30-0XE0)

Information on the CM 1243-7 LTE communication module for LTE connections is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK72437KX300XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PROFIBUS

CM 1242-5 (6xxx 242-5DX3x-xXE0)

You will find information on the PROFIBUS communications modules here:

- CM 1242-5 6GK7 242-5DX30-0XE0 and 6GK7 242-5DX31-0XE0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK72425DX300XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CM 1242-5 6AG1 242-5DX30-2XE0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12425DX302XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CM 1242-5 6AG1 242-5DX30-2XY0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12425DX302XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

CM 1243-5 (6xxx 243-5DX3x-xXE0)

You will find information on the PROFIBUS communications modules here:

- CM 1243-5 6GK7 243-5DX30-0XE0 and 6GK7 243-5DX31-0XE0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK72435DX300XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CM 1243-5 6AG1 243-5DX30-2XE0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12435DX302XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CM 1243-5 6AG1 243-5DX30-2XY0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12435DX302XY0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

Point-to-point

CM 1241 RS232 (6xxx 241-1AH30-xXB0)

You will find information on the communications modules for point-to-point connections here:

- CM 1241 RS232 6ES7 241-1AH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411AH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CM 1241 RS232 6AG1 241-1AH30-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12411AH302XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

CM 1241 RS232 (6xxx 241-1AH32-xXB0)

You will find information on the communications modules for point-to-point connections here:

- CM 1241 RS232 6ES7 241-1AH32-0XB0 (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411AH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CM 1241 RS232 6AG1 241-1AH32-2XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12411AH322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CM 1241 RS232 6AG1 241-1AH32-4XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12411AH324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

CM 1241 RS485 (6xxx 241-1CH30-xXBx)

You will find information on the communications modules for point-to-point connections here:

CM 1241 RS485 6ES7 241-1CH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411CH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

SIPLUS CB 1241 RS485 6AG1 241-1CH30-5XB1 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12411CH305XB1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

CM 1241 RS422/485 (6ES7 241-1CH31-0XB0)

Information on the CM 1241 (RS422/485) communication module for point-to-point connections is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411CH310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM 1241 RS422/485 (6xxx 241-1CH32-xXB0)

You will find information on the communications modules for point-to-point connections here:

- CM 1241 RS422/485 6ES7 241-1CH32-0XB0 (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411CH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CM 1241 RS422/485 6AG1 241-1CH32-2XB0 (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12411CH322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS CM 1241 RS422/485 6AG1 241-1CH32-4XB0 (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12411CH324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

Identification systems

RF120C (6GT2 002-0LA00)

Information on the communication module for Ident technology RFC120C is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GT20020LA00&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AS-Interface

AS-Interface CM 1243-2 (3RK7 243-2AA30-0XB0)

Information on the CM 1243-2 AS-i communication module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=3RK72432AA300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.3.1.12 Technology modules

SIWAREX WP231 (7MH4 960-2AA01)

Information on the SIWAREX WP231 weighing module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=7MH4960-2AA01&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

SIWAREX WP241 (7MH4 960-4AA01)

Information on the SIWAREX WP241 weighing module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=7MH4960-4AA01&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

4SI IO-Link (6ES7 278-4BD32-0XB0)

You will find information on the IO-Link Master technology modules here:

- 4SI IO-Link 6ES7 278-4BD32-0XB0 (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72784BD320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS SM 1278 IO-Link Master 6AG1 278-4BD32-2XB0 (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12784BD322XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)
- SIPLUS SM 1278 IO-Link Master 6AG1 278-4BD32-4XB0 (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6AG12784BD324XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

Power signal booster segment module (6ES7 228-1RC52-0AA0)

You will find information on the power signal booster segment module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72281RC520AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

Power signal booster carrier module (6ES7 228-1RC51-0AA0)

You will find information on the power signal booster carrier module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72281RC510AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4 Distributed I/O

16.4.1 ET 200SP

16.4.1.1 Interface modules

PROFINET

IM 155-6 PN ST (6ES7 155-6AU00-0BN0)

You can find information on the interface module IM 155-6 PN ST here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71556AU000BN0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

IM 155-6 PN HF (6ES7 155-6AU00-0CN0)

You can find information on the interface module IM 155-6 PN HF here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71556AU000CN0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

IM 155-6 PN BA (6ES7 155-6AR00-0AN0)

You will find information on the IM 155-6 PN BA interface module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71556AR000AN0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

IM 155-6 PN HS (6ES7 155-6AU00-0DN0)

You will find information on the IM 155-6 PN HS interface module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71556AU000DN0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PROFIBUS

IM 155-6 DP HF (6ES7 155-6BU00-0CN0)

Information on the IM 155-6 DP HF interface module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71556BU000CN0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

Other fieldbuses

IM 155-6 Receive (6ES7 155-6DU00-0BN0)

Information on the IM 155-6 Receive interface module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71556DU000BN0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.1.2 Digital input modules

DI 4x120..230VAC ST (6ES7 131-6FD00-0BB1)

Information on the DI 4x120..230VAC ST digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316FD000BB1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 8x24VDC ST (6ES7 131-6BF00-0BA0)

Information on the digital input module DI 8x24VDC ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316BF000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 8x24VDC HF (6ES7 131-6BF00-0CA0)

Information on the digital input module DI 8x24VDC HF is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316BF000CA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 8xNAMUR HF (6ES7 131-6TF00-0CA0)

Information on the DI 8xNAMUR HF digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316TF000CA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 8x24VDC SRC BA (6ES7 131-6BF60-0AA0)

Information on the DI 8x24VDC SRC BA digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316BF6000A0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 16x24VDC ST (6ES7 131-6BH00-0BA0)

Information on the digital input module DI 16x24VDC ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316BH000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 8x24VDC BA (6ES7 131-6BF00-0AA0)

You will find information on the DI 8x24VDC BA digital input module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316BF000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 8x24VDC HS (6ES7 131-6BF00-0DA0)

You will find information on the DI 8x24VDC HS digital input module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316BF000DA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.1.3 Digital output modules

DQ 4x24VDC/2A ST (6ES7 132-6BD20-0BA0)

Information on the digital output module DQ 4x24VDC/2A ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BD200BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 4x24..230VAC/2A ST (6ES7 132-6FD00-0BB1)

Information on the DQ 4x24..230VAC/2A ST digital output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326FD000BB1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 4x24VDC/2A HF (6ES7 132-6BD20-0CA0)

Information on the DQ 4x24VDC/2A HF digital output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BD200CA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

RQ 4x120..230VUC/5A NO ST (6ES7 132-6HD00-0BB0)

Information on the RQ 4x120..230VUC/5A NO ST relay output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326HD000BB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

RQ 4x24VUC/2A CO ST (6ES7 132-6GD50-0BA0)

Information on the RQ 4x24VUC/2A CO ST relay output module is available here.

DQ 8x24VDC/0.5A ST (6ES7 132-6BF00-0BA0)

Information on the digital output module DQ 8x24VDC/0.5A ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BF000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 8x24VDC/0.5A SNK BA (6ES7 132-6BF60-0AA0)

Information on the DQ 8x24VDC/0.5A SNK BA digital output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BF600AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 8x24VDC/0.5A HF (6ES7 132-6BF00-0CA0)

Information on the digital output module DQ 8x24VDC/0.5A HF is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BF000CA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 16x24VDC/0.5A ST (6ES7 132-6BH00-0BA0)

Information on the digital output module DQ 16x24VDC/0.5A ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BH000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 4x24VDC/2A HS (6ES7 132-6BD20-0DA0)

Information on the DQ 4x24VDC/2A HS digital output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BD200DA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.1.4 Analog input modules

AI 2xU/I 2-/4-wire HF (6ES7 134-6HB00-0CA1)

Information on the AI 2xU/I 2-/4-wire HF analog input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346HB000CA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 2xU/I 2-/4-wire HS (6ES7 134-6HB00-0DA1)

Information on the AI 2xU/I 2-/4-wire HS analog input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346HB000DA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 4xI 2-/4-wire ST (6ES7 134-6GD00-0BA1)

Information on the analog input module AI 4xI 2-/4-wire ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346GD000BA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 4xU/I 2-wire ST (6ES7 134-6HD00-0BA1)

Information on the analog input module AI 4xU/I 2-wire ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346HD000BA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 4xRTD/TC 2-/3-/4-wire HF (6ES7 134-6JD00-0CA1)

Information on the analog input module AI 4xRTD/TC 2-/3-/4-wire HF is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346JD000CA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 8xRTD/TC 2-wire HF (6ES7 134-6JF00-0CA1)

Information on the AI 8xRTD/TC 2-wire HF analog input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346JF000CA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI EnergyMeter ST (6ES7 134-6PA00-0BD0)

Information on the AI EnergyMeter ST relay output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346PA000BD0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 8xI 2-/4-wire BA (6ES7 134-6GF00-0AA1)

You will find information on the AI 8xI 2-/4-wire BA analog input module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346GF000AA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 8xU BA (6ES7 134-6FF00-0AA1)

You will find information on the AI 8xU BA analog input module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346FF000AA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.1.5 Analog output modules

AQ 2xU/I HF (6ES7 135-6HB00-0CA1)

Information on the AQ 2xU/I HF analog output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71356HB000CA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AQ 2xU/I HS (6ES7 135-6HB00-0DA1)

Information on the analog output module AQ 2xU/I HS is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71356HB000DA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AQ 4xU/I ST (6ES7 135-6HD00-0BA1)

Information on the analog output module AQ 4xU/I ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71356HD000BA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.1.6 Communication modules

CM DP (6ES7 545-5DA00-0AB0)

Information on the CM DP communication module is available here (func=cslib.csinfo&lang=en&objid=6ES75455DA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM AS-i Master ST (3RK7 137-6SA00-0BC1)

Information on the CM AS-i Master ST communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=3RK71376SA000BC1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM PtP (6ES7 137-6AA00-0BA0)

Information on the CM PtP communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71376AA000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

F-CM AS-i (3RK7 136-6SC00-0BC1)

Information on the F-CM AS-i module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=3RK71366SC000BC1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM 4xIO link (6ES7 137-6BD00-0BA0)

Information on the CM 4xIO-Link communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71376BD000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.1.7 Power modules

PM 120..230VAC/10A ST (6ES7 133-6AA00-0BC0)

Information on the PM 120..230VAC/10A ST power module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71336AA000BC0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.1.8 Special modules

Server module (6ES7 193-6PA00-0AA0)

Information on the server module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936PA000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

BusAdapter BA 2xSCRJ (6ES7 193-6AP00-0AA0)

Information on the BA 2xSCRJ BusAdapter is available here.

See also

Link to the device manual (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AP000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

BusAdapter BA 2xRJ45 (6ES7 193-6AR00-0AA0)

Information on the BusAdapter BA 2xRJ45 is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AR000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

BusAdapter FC (6ES7 193-6AF00-0AA0)

Information on the BusAdapter FC is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AF000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

BusAdapter Send (6ES7 193-6AS00-0AA0)

Information on the BusAdapter Send is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AS000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

BusAdapter Receive (6ES7 193-6AE00-0AA0)

Information on the BusAdapter Receive is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AE000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.1.9 Technology modules

TM Count 1x24V (6ES7 138-6AA00-0BA0)

Information on the relay output module TMCount 1x24V is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71386AA000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

TM PosInput 1 (6ES7 138-6BA00-0BA0)

Information on the positioning module TM PosInput 1 is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71386BA000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

TM Timer DIDQ 10x24V (6ES7 138-6CG00-0BA0)

You will find information on the TM Timer DIDQ 10x24V timer module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71386CG000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

SIWAREX WP321 (7MH4 138-6AA00-0BA0)

You will find information on the Siwarex WP321 weighing module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=7MH41386AA000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.1.10 BusAdapter

BusAdapter PROFINET BA SCRJ/RJ45 (6ES7 193-6AP20-0AA0)

You will find information on the PROFINET BusAdapter BA SCRJ/RJ45 for the ET 200SP here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AP200AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

BusAdapter PROFINET (6ES7 193-6AP40-0AA0)

You will find information on the PROFINET BusAdapter for the ET 200SP here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AP400AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.2 ET 200MP

16.4.2.1 Interface modules

PROFINET

IM 155-5 PN ST (6ES7 155-5AA00-0AB0)

Information on the distributed I/O module IM 155-5 PN ST is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71555AA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

IM 155-5 PN HF (6ES7 155-5AA00-0AC0)

Information on the IM 155-5 PN HF distributed I/O module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71555AA000AC0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PROFIBUS

IM 155-5 DP ST (6ES7 155-5BA00-0AB0)

Information on the IM 155-5 DP ST distributed I/O module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71555BA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.2.2 Digital input modules

DI 16x24VDC BA (6ES7 521-1BH10-0AA0)

Information on the DI 16x24VDC BA digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211BH100AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 16x24VDC HF (6ES7 521-1BH00-0AB0)

Information on the digital input module DI 16x24VDC HF is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211BH000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 16x24VDC SRC BA (6xxx 521-1BH50-xAA0)

Information on the DI 16x24VDC HF digital input module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211BH500AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 16x230VAC BA (6xxx 521-1FH00-xAA0)

Information on the DI 16x24VDC HF digital input module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211FH000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 32x24VDC HF (6ES7 521-1BL00-0AB0)

Information on the digital input module DI 16x24VDC HF is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211BL000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 32x24VDC BA (6ES7 521-1BL10-0AA0)

Information on the DI 16x24VDC BA digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211BL100AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.2.3 Digital output modules

DQ 8x24VDC/2A HF (6xxx 522-1BF00-xAB0)

Information on the DQ 8x24VDC/2A HF digital output module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75221BF000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 8x230VAC/2A ST (6xxx 522-5FF00-xAB0)

Information on the DQ 8x230VAC/2A ST digital output module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75225FF000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 8x230VAC/5A ST (6xxx 522-5HF00-xAB0)

Information on the 8x230VAC/5A ST digital output module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75225HF000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 16x24VDC/0.5A BA (6ES7 522-1BH10-0AA0)

Information on the DQ 16x24VDC/0.5A BA digital output module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75221BH100AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 16x24VDC/0.5A ST (6ES7 522-1BH00-0AB0)

Information on the digital output module DQ 16x24VDC/0.5A ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75221BH000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 32x24VDC/0.5A BA (6ES7 522-1BL10-0AA0)

Information on the DQ 32x 24VDC/0.5A BA digital output module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75221BL100AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 32x24VDC/0.5A ST (6ES7 522-1BL00-0AB0)

Information on the digital output module DQ 32x 24VDC/0.5A is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75221BL000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.2.4 Digital input and digital output modules

DI16/DO 16x24VDC (6ES7 523-1BL00-0AA0)

Information on the DI 16/DO 16x24VDC digital input and output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75231BL000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.2.5 Analog input modules

AI 4xU/I/RTD/TC ST (6ES7 531-7QD00-0AB0)

Information on the AI 4xU/I/RTD/TC ST analog input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75317QD00AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 8xU/I/RTD/TC ST (6ES7 531-7KF00-0AB0)

Information on the analog input module AI 8xU/I/RTD/TC ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75317KF000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 8xU/I HS (6xxx 531-7NF10-xAB0)

Information on the AI 8xU/I HS analog input module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75317NF100AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.2.6 Analog output modules

AQ 2xU/I ST (6ES7 532-5NB00-0AB0)

Information on the AQ 2xU/I ST analog output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75325NB000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AQ 4xU/I ST (6ES7 532-5HD00-0AB0)

Information on the analog output module AQ 4xU/I ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75325HD000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AQ 8xU/I HS (6xxx 532-5HF00-xAB0)

Information on the AQ 8xU/I HS analog output module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75325HF000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.2.7 Analog input and analog output modules

AI/AQ 4xU/I/RTD/TC / 2xU/I ST (6ES7 534-7QE00-0AB0)

Information on the AI/AQ 4xU/I/RTD/TC / 2xU/I ST analog input and output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75347QE000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.2.8 Communications modules

PROFINET/Ethernet

CM 1542-1 (6GK7 542-1AX00-0XE)

Information on the CM 1542-1 PROFINET/Ethernet communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK75421AX000XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CP 1543-1 (6GK7 543-1AX00-0XE0)

Information on the CP 1543-1 PROFINET/Ethernet communications processor is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK75431AX000XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PROFIBUS

CM 1542-5 (6GK7 542-5DX00-0XE0)

Information on the CM 1542-5 PROFIBUS communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK75425DX000XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

Point-to-point

RS232

CM PtP RS232 BA (6xxx 540-1AD00-xAA0)

Information on the CM PtP RS232 BA point-to-point communication module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75401AD000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM PtP RS232 HF (6ES7 541-1AD00-0AB0)

Information on the CM PtP RS232 HF point-to-point communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75411AD000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

RS422/485

CM PtP RS422/485 BA (6xxx 540-1AB00-xAA0)

Information on the CM PtP RS422/485 BA point-to-point communication module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75401AB000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM PtP RS422/485 HF (6xxx 541-1AB00-xAB0)

Information on the CM PtP RS422/485 HF point-to-point communication module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75411AB000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.2.9 Power supply module

PS 25W 24VDC (6ES7 505-0KA00-0AB0)

Information on the PS 25W 24VDC power supply module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75050KA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PS 60W 24/48/60VDC (6xxx 505-0RA00-xAB0)

Information on the PS 60W 24/48/60VDC power supply module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75050RA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PS 60W 120/230VAC/DC (6xxx 507-0RA00-xAB0)

Information on the power supply module PS 60W 120/230VAC/DC (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75070RA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.2.10 Technology modules

TM Count 2x24V (6ES7 550-1AA00-0AB0)

Information on the counter module TM Count 2x24V is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75501AA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

TM PosInput 2 (6ES7 551-1AB00-0AB0)

Information on the positioning module TM PosInput 2 is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75511AB000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

TM Timer DIDQ 16x24V (6ES7 552-1AA00-0AB0)

You will find information on the TM Timer DIDQ 16x24V timer module here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75521AA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.3 ET 200AL

16.4.3.1 Interface modules

PROFINET

IM 157-1 PN (6ES7 157-1AB00-0AB0)

Information on the IM 157-1 PN distributed I/O module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71571AB000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PROFIBUS

IM 157-1 DP (6ES7 157-1AA00-0AB0)

Information on the IM 157-1 DP distributed I/O module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71571AA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.3.2 Digital input modules

DI 8x24VDC 8xM8 (6ES7 141-5BF00-0BA0)

Information on the DI 8x24VDC 8xM8 digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71415BF000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.3.3 Digital input modules

DIQ 4+DQ 4x24VDC/0.5A 8xM8 (6ES7 143-5BF00-0BA0)

Information on the DIQ 4+DQ 4x24VDC/0.5A 8xM8 digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71435BF000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.3.4 Analog input modules

AI 4xU/I/RTD 4xM12 (6ES7 144-5KD00-0BA0)

Information on the AI 4xU/I/RTD 4xM12 analog input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71445KD000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

16.4.3.5 Communications modules

CM 4xIO link 4xM12 (6ES7 147-5JD00-0BA0)

Information on the CM 4xIO-Link 4xM12 communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71475JD000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

Index

add external graphic, 6838
add graphic to graphics library, 6837

-

-, 1660

"

"Change color reference" dialog, 6889
"WWW" instruction, 875

#

#, 1558

&

&, 1665

(

(), 1665

*****

*, 1660
**, 1660
*.bmp, 3973
*.cer, 770
*.dat, 770
*.emf, 3973
*.gif, 3973
*.ico, 3973
*.jpeg, 3973
*.jpg, 3973
*.p12, 690, 770
*.tif, 3973
*.wmf, 3973

.

.Net control
Add, 3970
Remove, 3971
.Net service packs, 43
.Net versions, 43

/

/, 1660

:

:=, 1665, 1666
:P, 1177
:PROFIBUS DP
Direct key, 6538

{

{"ColumnDateFormat property (VBS)"}{"Properties (VBS)";"ColumnDateFormat"}, 5297
{"ColumnTimeFormat property (VBS)"}{"Properties (VBS)";"ColumnTimeFormat"}, 5300
{"Restore operation";"License key"}, 133

+

+, 1660

<

<, 1663
<=, 1663
<>, 1663, 1665

=

=, 1663
==, 1665

>

>, 1663
>=, 1663

3

32-bit down counter, 1255
3DES, 746
3RK7 136-6SC00-0BC1, 7812
3RK7 137-6SA00-0BC1, 7812
3RK7 243-2AA30-0XB0, 7804

6

6AG1 214-1HG40-5XB0, 7783
6AG1 215-1AG40-2XB0, 7785
6AG1 215-1AG40-4XB0, 7785
6AG1 215-1AG40-5XB0, 7785
6AG1 215-1BG40-2XB0, 7785
6AG1 215-1BG40-4XB0, 7785
6AG1 215-1BG40-5XB0, 7785
6AG1 215-1HG40-2XB0, 7785
6AG1 215-1HG40-4XB0, 7785
6AG1 215-1HG40-5XB0, 7785
6AG1 221-1BF32-2XB0, 7789
6AG1 221-1BF32-2XY0, 7789
6AG1 221-1BF32-4XB0, 7789
6AG1 221-1BH32-2XB0, 7789
6AG1 221-1BH32-4XB0, 7789
6AG1 222-1BF32-2XB0, 7792
6AG1 222-1BF32-2XY0, 7792
6AG1 222-1BF32-4XB0, 7792
6AG1 222-1BH32-2XB0, 7792
6AG1 222-1BH32-4XB0, 7792
6AG1 222-1HF32-2XB0, 7792
6AG1 222-1HF32-4XB0, 7792
6AG1 222-1HH32-2XB0, 7792
6AG1 222-1HH32-4XB0, 7792
6AG1 222-1XF32-2XB0, 7792
6AG1 223-0BD30-4XB0, 7788
6AG1 223-1BH32-2XB0, 7795
6AG1 223-1BH32-2XY0, 7795
6AG1 223-1BH32-4XB0, 7795
6AG1 223-1BL32-2XB0, 7795
6AG1 223-1BL32-4XB0, 7795
6AG1 223-1PH32-2XB0, 7795
6AG1 223-1PH32-4XB0, 7795
6AG1 223-1PL32-2XB0, 7795
6AG1 223-1PL32-2XY0, 7795
6AG1 223-1PL32-4XB0, 7795
6AG1 223-1QH32-2XB0, 7795
6AG1 223-1QH32-4XB0, 7795
6AG1 223-3AD30-5XB0, 7788
6AG1 231-4HD32-4XB0, 7798
6AG1 231-4HF32-4XB0, 7798
6AG1 231-5ND32-2XB0, 7798
6AG1 231-5ND32-4XB0, 7798
6AG1 231-5PD32-2XB0, 7798
6AG1 231-5PD32-4XB0, 7798
6AG1 231-5PF32-2XB0, 7798
6AG1 231-5PF32-4XB0, 7798
6AG1 231-5QD32-2XB0, 7798
6AG1 231-5QD32-4XB0, 7798
6AG1 231-5QF32-2XB0, 7798
6AG1 231-5QF32-4XB0, 7798
6AG1 232-4HA30-4XB0, 7799
6AG1 232-4HA30-5XB0, 7799
6AG1 232-4HB32-4XB0, 7799
6AG1 232-4HD32-2XB0, 7799
6AG1 232-4HD32-4XB0, 7799
6AG1 234-4HE32-2XB0, 7800
6AG1 234-4HE32-2XY0, 7800
6AG1 241-1AH30-2XB0, 7802
6AG1 241-1AH32-2XB0, 7802
6AG1 241-1AH32-4XB0, 7802
6AG1 241-1CH30-5XB1, 7803
6AG1 241-1CH32-2XB0, 7803
6AG1 241-1CH32-4XB0, 7803
6AG1 242-5DX30-2XE0, 7801
6AG1 242-5DX30-2XY0, 7801
6AG1 242-7KX30-4XE0, 7800
6AG1 243-1JX30-7XE0, 7800
6AG1 243-5DX30-2XE0, 7802
6AG1 243-5DX30-2XY0, 7802
6AG1 278-4BD32-2XB0, 7804
6AG1 278-4BD32-4XB0, 7804
6AG1 505-0RA00-0AB0, 7822
6AG1 507-0RA00-0AB0, 7822
6AG1 521-1BH50-7AA0, 7817
6AG1 521-1FH00-7AA0, 7817
6AG1 522-1BF00-7AB0, 7817
6AG1 522-5FF00-7AB0, 7818
6AG1 522-5HF00-2AB0, 7818
6AG1 531-7NF10-7AB0, 7819
6AG1 532-5HF00-7AB0, 7820
6AG1 540-1AB00-7AA0, 7822
6AG1 540-1AD00-7AA0, 7821
6AG1 541-1AB00-7AB0, 7822
6AV2124-0GC01-0AX0, 7778
6AV2124-0JC01-0AX0, 7778
6AV2124-0MC01-0AX0, 7778

6AV2124-0QC02-0AX0, 7778
6AV2124-0UC02-0AX0, 7778
6AV2124-0XC02-0AX0, 7778
6AV2124-1DC01-0AX0, 7778
6AV2124-1GC01-0AX0, 7778
6AV2124-1JC01-0AX0, 7778
6AV2124-1MC01-0AX0, 7778
6AV3688-3AF37-0AX0, 7779
6AV3688-3AY36-0AX0, 7779
6AV3688-3EH47-0AX0, 7779
6AV3688-4CX07-0AA0, 7779
6AV3688-4EY06-0AA0, 7779
6AV3688-4EY07-0AA0, 7779
6AV6545-0CC10-0AX0, 7778
6AV6591-1DC20-0AB0, 7778
6AV6641-0AA11-0AX0, 7777
6AV6641-0BA11-0AX0, 7777
6AV6641-0CA01-0AX0, 7777
6AV6642-0AA11-0AX0, 7778
6AV6642-0AA11-0AX1, 7778
6AV6642-0BA01-1AX0, 7778
6AV6642-0BA01-1AX1, 7778
6AV6642-0BC01-1AX0, 7778
6AV6642-0BC01-1AX1, 7778
6AV6642-0BD01-3AX0, 7778
6AV6642-0DA01-1AX0, 7778
6AV6642-0DA01-1AX1, 7778
6AV6642-0DC01-1AX0, 7778
6AV6642-0DC01-1AX1, 7778
6AV6642-0EA01-3AX0, 7778
6AV6642-8BA10-0AA0, 7778
6AV6643-0AA01-1AX0, 7778
6AV6643-0CB01-1AX0, 7778
6AV6643-0CB01-1AX1, 7778
6AV6643-0CD01-1AX0, 7778
6AV6643-0CD01-1AX1, 7778
6AV6643-0DB01-1AX0, 7778
6AV6643-0DB01-1AX1, 7778
6AV6643-0DD01-1AX0, 7778
6AV6643-0DD01-1AX1, 7778
6AV6643-0ED01-2AX0, 7778
6AV6643-8AD10-0AA0, 7778
6AV6644-0AA01-2AX0, 7778
6AV6644-0AC01-2AX0, 7778
6AV6644-0AC01-2AX1, 7778
6AV6644-0BA01-2AX0, 7778
6AV6644-0BA01-2AX1, 7778
6AV6644-2AB01-2AX0, 7778
6AV6645-0AA01-0AX0, 7779
6AV6645-0AB01-0AX0, 7779
6AV6645-0AC01-0AX0, 7779
6AV6645-0BA01-0AX0, 7779
6AV6645-0BB01-0AX0, 7779
6AV6645-0BC01-0AX0, 7779
6AV6645-0BE02-0AX0, 7779
6AV6645-0CA01-0AX0, 7779
6AV6645-0CB01-0AX0, 7779
6AV6645-0CC01-0AX0, 7779
6AV6645-0DD01-0AX1, 7779
6AV6645-0DE01-0AX1, 7779
6AV6645-0EB01-0AX1, 7779
6AV6645-0EC01-0AX1, 7779
6AV6645-0EF01-0AX1, 7779
6AV6645-0FD01-0AX1, 7779
6AV6645-0FE01-0AX1, 7779
6AV6645-0GB01-0AX1, 7779
6AV6645-0GC01-0AX1, 7779
6AV6645-0GF01-0AX1, 7779
6AV6647-0AA11-3AX0, 7777, 7780
6AV6647-0AB11-3AX0, 7777, 7780
6AV6647-0AC11-3AX0, 7777, 7780
6AV6647-0AD11-3AX0, 7777, 7780
6AV6647-0AE11-3AX0, 7777, 7780
6AV6647-0AF11-3AX0, 7777, 7780
6AV6647-0AG11-3AX0, 7777, 7780
6AV6647-0AH11-3AX0, 7777, 7780
6AV6647-0AJ11-3AX0, 7777, 7780
6AV6647-0AK11-3AX0, 7777, 7780
6AV6651-1AA01-0AA0, 7777
6AV6651-1BA01-0AA0, 7777
6AV6651-2AA01-0AA0, 7778
6AV6691-1DA01-0AA1, 7777
6AV6691-1DG01-0AA1, 7778
6AV6691-1DJ01-0AA0, 7778
6AV6691-1DJ01-0AB0, 7778
6AV6691-1DJ01-0AC0, 7778
6AV6691-1DJ01-0AD0, 7778
6AV6691-1DJ01-0AE0, 7778
6AV6691-1DR01-0AB0, 7778
6ES7 131-6BF00-0AA0, 7807
6ES7 131-6BF00-0BA0, 7807
6ES7 131-6BF00-0CA0, 7807
6ES7 131-6BF00-0DA0, 7808
6ES7 131-6BF60-0AA0, 7807
6ES7 131-6BH00-0BA0, 7807
6ES7 131-6FD00-0BB1, 7806
6ES7 131-6TF00-0CA0, 7807
6ES7 132-6BD20-0BA0, 7808
6ES7 132-6BD20-0CA0, 7808
6ES7 132-6BD20-0DA0, 7809
6ES7 132-6BF00-0BA0, 7809
6ES7 132-6BF00-0CA0, 7809
6ES7 132-6BF60-0AA0, 7809
6ES7 132-6BH00-0BA0, 7809

6ES7 132-6FD00-0BB1, 7808
6ES7 132-6GD50-0BA0, 7808
6ES7 132-6HD00-0BB0, 7808
6ES7 133-6AA00-0BC0, 7813
6ES7 134-6FF00-0AA1, 7811
6ES7 134-6GD00-0BA1, 7810
6ES7 134-6GF00-0AA1, 7811
6ES7 134-6HB00-0CA1, 7809
6ES7 134-6HB00-0DA1, 7810
6ES7 134-6HD00-0BA1, 7810
6ES7 134-6JD00-0CA1, 7810
6ES7 134-6JF00-0CA1, 7810
6ES7 134-6PA00-0BD0, 7810
6ES7 135-6HB00-0CA1, 7811
6ES7 135-6HB00-0DA1, 7811
6ES7 135-6HD00-0BA1, 7811
6ES7 137-6AA00-0BA0, 7812
6ES7 137-6BD00-0BA0, 7812
6ES7 138-6AA00-0BA0, 7814
6ES7 138-6BA00-0BA0, 7814
6ES7 138-6CG00-0BA0, 7814
6ES7 141-5BF00-0BA0, 7824
6ES7 143-5BF00-0BA0, 7824
6ES7 144-5KD00-0BA0, 7824
6ES7 147-5JD00-0BA0, 7825
6ES7 155-5AA00-0AB0, 7815
6ES7 155-5AA00-0AC0, 7816
6ES7 155-5BA00-0AB0, 7816
6ES7 155-6AR00-0AN0, 7806
6ES7 155-6AU00-0BN0, 7805
6ES7 155-6AU00-0CN0, 7805
6ES7 155-6AU00-0DN0, 7806
6ES7 155-6BU00-0CN0, 7806
6ES7 155-6DU00-0BN0, 7806
6ES7 157-1AA00-0AB0, 7824
6ES7 157-1AB00-0AB0, 7823
6ES7 193-6AE00-0AA0, 7814
6ES7 193-6AF00-0AA0, 7814
6ES7 193-6AP00-0AA0, 7813
6ES7 193-6AP20-0AA0, 7815
6ES7 193-6AR00-0AA0, 7813
6ES7 193-6AS00-0AA0, 7814
6ES7 193-6PA00-0AA0, 7813
6ES7 211-1AD30-0XB0, 7781
6ES7 211-1AE31-0XB0, 7781
6ES7 211-1AE40-0XB0, 7781
6ES7 211-1BD30-0XB0, 7781
6ES7 211-1BE31-0XB0, 7781
6ES7 211-1BE40-0XB0, 7781
6ES7 211-1HD30-0XB0, 7781
6ES7 211-1HE31-0XB0, 7781
6ES7 211-1HE40-0XB0, 7781
6ES7 212-1AD30-0XB0, 7782
6ES7 212-1AE31-0XB0, 7782
6ES7 212-1AE40-0XB0, 7782
6ES7 212-1BD30-0XB0, 7782
6ES7 212-1BE31-0XB0, 7782
6ES7 212-1BE40-0XB0, 7782
6ES7 212-1HD30-0XB0, 7782
6ES7 212-1HE31-0XB0, 7782
6ES7 212-1HE40-0XB0, 7782
6ES7 214-1AE30-0XB0, 7783
6ES7 214-1AF40-0XB0, 7785
6ES7 214-1AG31-0XB0, 7783
6ES7 214-1AG40-0XB0, 7783
6ES7 214-1BE30-0XB0, 7783
6ES7 214-1BG31-0XB0, 7783
6ES7 214-1BG40-0XB0, 7783
6ES7 214-1HE30-0XB0, 7783
6ES7 214-1HF40-0XB0, 7785
6ES7 214-1HG31-0XB0, 7783
6ES7 214-1HG40-0XB0, 7783
6ES7 215-1AF40-0XB0, 7786
6ES7 215-1AG31-0XB0, 7785
6ES7 215-1AG40-0XB0, 7785
6ES7 215-1BG31-0XB0, 7785
6ES7 215-1BG40-0XB0, 7785
6ES7 215-1HF40-0XB0, 7786
6ES7 215-1HG31-0XB0, 7785
6ES7 215-1HG40-0XB0, 7785
6ES7 217-1AG40-0XB0, 7785
6ES7 221-1BF30-0XB0, 7789
6ES7 221-1BF32-0XB0, 7789
6ES7 221-1BH30-0XB0, 7789
6ES7 221-1BH32-0XB0, 7789
6ES7 221-3AD30-0XB0, 7788
6ES7 221-3BD30-0XB0, 7788
6ES7 222-1AD30-0XB0, 7788
6ES7 222-1BD30-0XB0, 7788
6ES7 222-1BF30-0XB0, 7792
6ES7 222-1BF32-0XB0, 7792
6ES7 222-1BH30-0XB0, 7792
6ES7 222-1BH32-0XB0, 7792
6ES7 222-1HF30-0XB0, 7792
6ES7 222-1HF32-0XB0, 7792
6ES7 222-1HH30-0XB0, 7792
6ES7 222-1HH32-0XB0, 7792
6ES7 222-1XF30-0XB0, 7792
6ES7 222-1XF32-0XB0, 7792
6ES7 223-0BD30-0XB0, 7788
6ES7 223-1BH30-0XB0, 7795
6ES7 223-1BH32-0XB0, 7795
6ES7 223-1BL30-0XB0, 7795
6ES7 223-1BL32-0XB0, 7795

6ES7 223-1PH30-0XB0, 7795
6ES7 223-1PH32-0XB0, 7795
6ES7 223-1PL30-0XB0, 7795
6ES7 223-1PL32-0XB0, 7795
6ES7 223-1QH30-0XB0, 7795
6ES7 223-1QH32-0XB0, 7795
6ES7 223-3AD30-0XB0, 7788
6ES7 223-3BD30-0XB0, 7788
6ES7 226-6BA32-0XB0, 7790
6ES7 226-6DA32-0XB0, 7792
6ES7 228-1RC51-0AA0, 7805
6ES7 228-1RC52-0AA0, 7805
6ES7 231-4HA30-0XB0, 7788
6ES7 231-4HD30-0XB0, 7798
6ES7 231-4HD32-0XB0, 7798
6ES7 231-4HF30-0XB0, 7798
6ES7 231-4HF32-0XB0, 7798
6ES7 231-5ND30-0XB0, 7798
6ES7 231-5ND32-0XB0, 7798
6ES7 231-5PA30-0XB0, 7788
6ES7 231-5PD30-0XB0, 7798
6ES7 231-5PD32-0XB0, 7798
6ES7 231-5PF30-0XB0, 7798
6ES7 231-5PF32-0XB0, 7798
6ES7 231-5QA30-0XB0, 7788
6ES7 231-5QD30-0XB0, 7798
6ES7 231-5QD32-0XB0, 7798
6ES7 231-5QF30-0XB0, 7798
6ES7 231-5QF32-0XB0, 7798
6ES7 232-4HA30-0XB0, 7788
6ES7 232-4HB30-0XB0, 7799
6ES7 232-4HB32-0XB0, 7799
6ES7 232-4HD30-0XB0, 7799
6ES7 232-4HD32-0XB0, 7799
6ES7 234-4HE30-0XB0, 7800
6ES7 234-4HE32-0XB0, 7800
6ES7 241-1AH30-0XB0, 7802
6ES7 241-1AH32-0XB0, 7802
6ES7 241-1CH30-0XB0, 7803
6ES7 241-1CH30-1XB0, 7788
6ES7 241-1CH31-0XB0, 7803
6ES7 241-1CH32-0XB0, 7803
6ES7 278-4BD32-0XB0, 7804
6ES7 505-0KA00-0AB0, 7822
6ES7 505-0RA00-0AB0, 7822
6ES7 507-0RA00-0AB0, 7822
6ES7 521-1BH00-0AB0, 7816, 7817
6ES7 521-1BH10-0AA0, 7816
6ES7 521-1BL10-0AA0, 7817
6ES7 522-1BF00-0AB0, 7817
6ES7 522-1BH00-0AB0, 7818
6ES7 522-1BH10-0AA0, 7818
6ES7 522-1BL00-0AB0, 7818
6ES7 522-1BL10-0AA0, 7818
6ES7 522-5FF00-0AB0, 7817
6ES7 522-5HF00-0AB0, 7818
6ES7 523-1BL00-0AA0, 7819
6ES7 531-7KF00-0AB0, 7819
6ES7 531-7NF10-0AB0, 7819
6ES7 531-7QD00-0AB0, 7819
6ES7 532-5HD00-0AB0, 7820
6ES7 532-5HF00-0AB0, 7820
6ES7 532-5NB00-0AB0, 7820
6ES7 534-7QE00-0AB0, 7820
6ES7 540-1AB00-0AA0, 7822
6ES7 540-1AD00-0AA0, 7821
6ES7 541-1AB00-0AB0, 7822
6ES7 541-1AD00-0AB0, 7821
6ES7 545-5DAD00-0BA0, 7812
6ES7 550-1AA00-0AB0, 7823
6ES7 552-1AA00-0AB0, 7823
6ES7671-4EE00-0YA0, 7779
6ES7671-5EF01-0YA0, 7779
6GK7 242-5DX30-0XE0, 7801
6GK7 242-5DX31-0XE0, 7801
6GK7 242-7KX30-0XE0, 7800
6GK7 243-1JX30-0XE0, 7800
6GK7 243-5DX30-0XE0, 7802
6GK7 243-5DX31-0XE0, 7802
6GK7 243-7KX30-0XE0, 7801
6GK7 243-7SX30-0XE0, 7801
6GK7 542-1AX00-0XE0, 7820
6GK7 542-5DX00-0XE0, 7821
6GK7 543-1AX00-0XE0, 7821
6GT2 002-0LA00, 7803

7

7MH4 138-6AA00-0BA0, 7815
7MH4 960-2AA01, 7804
7MH4 960-4AA01, 7804

A

AB, 1482
Ability to read back connection parameters, 647
AboveUpperLimitColor property (VBS), 5223
ABS, 2312, 2588, 2801
Absolute Addressing
    of a tag, 4202, 6036
Absolute value, 2312, 2588, 2801
AcceptOnExit property (VBS), 5223
AcceptOnFull property (VBS), 5224

Access
HMI tag, 4580
Local tag, 4580
Access control, 1041
Automatic learning, 1041
Access level, 6097
Access point, 5976
Access Point, 5942, 5976
Access protection, 4559
Access protection level 1, 7677
Access protection level 2, 7677
Advantages, 7677
Alarm view, 4140
Configuring, 4564
Factory state, 7676
User administration, 4558
Validity, 7676
Access to operands, 1454, 1455, 1456, 1458, 1459, 1461, 1462
Accessible devices, 1137, 6103, 6188, 6269, 6406
Accuracy, 290
ACK
Key, 6915, 6974
Acknowledge, 4285, 4853
Key, 6915, 6974
Acknowledge alarm
Alarm group, 4291
AcknowledgeAlarm, 4743, 4857
Acknowledgement
Audit Trail, 7069
Acknowledgment, 4290
Configuring, 4333, 4334, 4336
Acknowledgment concept
Alarm with simple acknowledgment, 4292
Alarm without acknowledgment, 4292
Acknowledgment model, 4292
ACL, 1041
ACOS, 2328, 2605, 2816
Acquisition cycle
Area pointer, 6042
Tag, 4221, 4235, 4247
Acquisition mode
Tag, 4221
ACT_TINT, 3227
Actions
Basics of redoing actions, 446
Basics of undoing, 446
Redoing, 449
Undoing, 448
activate, 4846
Activate
Project language, 6826
Activate method, 5823
Activate property (VBS), 5737
ActivateCleanScreen, 4659
ActivateDynamic, 5826
ActivatePreviousScreen, 4660, 4857
ActivateScreen, 4657, 4858
ActivateScreenByNumber, 4658, 4859
ActivateSystemDiagnosticsView, 4795, 4860
Activating
Font size adjustment, 3923
Active
Area pointer, 6042
Active bus module (ET 200M), 1336
Active nodes, 750
Active property (VBS), 5760
ActiveScreen property (VBS), 5226
ActiveScreenItem object, 5228
ActiveX control
Add, 3970
Remove, 3971
ActualPointIndex property (VBS), 5228
ActualPointLeft property (VBS), 5229
ActualPointTop property (VBS), 5229
Acyclic triggers, 5983
AD, 1482
AdaptBorder property (VBS), 5230
AdaptFontSizeToLineHeight property (VBS), 5761
Adapting
Connection name, 7771
Adapting a project
For a different HMI device, 6906, 6954
AdaptPicture property (VBS), 5231
AdaptScreenToWindow property (VBS), 5231
AdaptWindowtoScreen property (VBS), 5231
Add, 2303, 2579
.Net control, 3970
ActiveX control, 3970
Objects to the group, 3980
ADD, 2303, 2579
Add empty line, 7421
Add separator line, 7421
Adding
Adding timers with T_ADD, 2974
Additional field devices (PROFIBUS and PROFINET), 1109, 1141, 1142
Add-on
Installing, 128
Removing, 128
Address
Area pointer, 6042
Mitsubishi, 6656
Omron Host Link, 6718

address conversion
Instructions for, 3358
Address multiplexing
with absolute addresses, 4225
with symbolic addresses, 4226
Address overview, 576
Address packing, 1100
Address property (VBS), 5232
Address range, 717
Changing, 850
Address register, 1470, 1471
AddressEnabled property (VBS), 5232
Addresses
Assigning, 851
Determining a module with GEO_LOG, 3366
Interrupt with packed, 1106
Pack, 1100
Read station address with GetStationInfo, 3286
Reading out the MAC address with GetStationInfo, 3286
Unpack, 1100
Addressing, 1177
Addressing tags indirectly, 4232, 4233
Allen-Bradley, 6615, 6634
Allen-Bradley Ethernet IP, 6610
Changing, 850
Ethernet/IP, 6615, 6634
General, 848
MPI, 6298
Multiplexing, 4232
Valid, 6616
Addressing operands, 1454, 1455, 1456, 1458, 1459, 1461, 1462, 1464, 1465, 1466, 1467, 1469, 1470, 1471
AdjustBorder3DWithStyle property (VBS), 5737
AdjustContrast, 4661
AdjustRulerWindow property (VBS), 5761
Administrator, 693
AdressPreview property (VBS), 5232
Advanced Encryption Standard (AES), 746
Advanced mode
Global firewall rules, 702
Advanced recipe view, 4444, 4448
AES, 746
AES-128, 741
Aggressive mode, 746
Aging time, 1045
Alarm
acknowledge, 4358, 4360
Calling the infotext, 4361
Components, 4293
Configuring, 4303, 4313
Displaying, 4132, 4134
Editing, 4358
Event, 4365
Exporting, 6798
Importing, 6799
In Runtime, 4351, 4353
Inspector window, 61
Sequence is not adhered to, 4602
System function, 4364
Alarm buffer, 4336
In Runtime, 4352, 4354
Memory size, 4337
Alarm buffer overflow, 4337, 4853
Alarm class, 4293
Diagnostics, 4323
Identifying, 4137
In Runtime, 4352, 4354
Layout, 4356, 4359
Alarm classes, 4287, 4288
Common, 4288
Custom, 4287, 4288
Name change through migration, 182
Predefined, 4287, 4288
Use, 4287, 4288
Alarm display, 1157
"Active alarms" view, 1160
Acknowledging an alarm, 1161
Archive view, 1158
Clear archive, 1160
Export archive, 1159
Ignoring alarms, 1161
Layout of the alarms in the "Active alarms" view, 1160
Layout of the alarms in the archive view, 1158
Receiving alarms, 1159
Using the keyboard, 1162
Alarm event
Acknowledge, 4285
Incoming, 4285
Outgoing, 4285
Alarm group, 4292, 4294
Acknowledge alarm, 4291
Configuring, 4301
Creating, 4301
Migration, 181
Alarm indicator, 60, 4144, 4317, 4361
Alarm classes, 4145
Application, 4361
Configuring, 4326
Events, 4145
In Runtime, 4353, 4355
Layout, 4145, 4361

Operation, 4361
Operation using the mouse, 4362
Alarm line, 4134, 4139
Alarm log, 4337, 4418
Configuring, 4340
Configuring an alarm view, 4330
Creating, 4338, 4340
Displaying contents, 4330
In Runtime, 4354
Migrating, 188
Naming conventions, 4422
Alarm logging, 4279
Alarm message
Acknowledgment by the PLC, 6137, 6227, 6316, 6373, 6441, 6494, 6510, 6643, 6673, 6705, 6724
Acknowledgment on the HMI device, 6137, 6227, 6316, 6373, 6441, 6494, 6511, 6644, 6674, 6705, 6725
Configure acknowledgment, 6136, 6227, 6316, 6372, 6440, 6494, 6510, 6643, 6673, 6705, 6724
Alarm number, 4293, 4294
Alarm object, 4983
Alarm property (VBS), 5232
Alarm report
Configuring, 7057
Layout in reports, 4522
Use in reports, 4521
Alarm status, 4294
Acknowledged, 4285
Incoming, 4285
Outgoing, 4285
Alarm system, 4278
Alarm text, 4294
Formatting, 4306
Output fields, 68
Removing format settings, 4306
Special characters, 68
Alarm types, 4281
Alarm view, 60, 4132, 4134, 4142, 4316, 4355, 4360
~ configuring for logged alarms, 4325, 4342
Access protection, 4140
Alarm line, 4134
Alarm text window, 4360
Application, 4355
Column, 4133, 4136
Column headers, 4141
Column sequence, 4140
Configuring, 4319
Configuring the display of S7 diagnostic alarms, 4323
Configuring the layout, 4322
Define columns, 4140
Define display area, 4141
Enable sorting, 4137
Filter alarms, 4141
Filter display, 4328
Identify alarm classes, 4141
Identifying an alarm class, 4137
Layout, 4132, 4356, 4359
Operation, 4356, 4360
Operation using the keyboard, 4357
Operation using the mouse, 4357
Operator control, 4133, 4136, 4356
Operator controls, 4139
Select alarm classes, 4140
Sorting, 4140
Alarm window, 60, 4316, 4355
Application, 4355
Configuring, 4325
In Runtime, 4353, 4355
Operation, 4356
Operation using the keyboard, 4357
Operation using the mouse, 4357
Operator control, 4356
Alarm with simple acknowledgment, 4292
Alarm without acknowledgment, 4292
AlarmClasses property (VBS), 5233
AlarmColor property (VBS), 5233
AlarmControl object, 5018
AlarmFilter property (VBS), 5738
AlarmHigh property (VBS), 5762
AlarmID property, 5234
AlarmLog property (VBS), 5234
AlarmLogs object, 4986
AlarmLow property (VBS), 5677
AlarmLowerLimit property (VBS), 5234
AlarmLowerLimitColor property (VBS), 5235
AlarmLowerLimitEnabled property (VBS), 5236
AlarmLowerLimitRelative property (VBS), 5236
Alarms
Configuring, 6135, 6226, 6315, 6371, 6439, 6493, 6509
Data types, 6135, 6226, 6315, 6371, 6439, 6493, 6509
Modicon Modbus, 6704
Non-integrated connection, 6704
Output of a tag value, 4307
Output of texts from a text list, 4308
Restriction, 6673
Alarms object (list), 4984
AlarmSource property (VBS), 5237
AlarmUpperLimit property (VBS), 5237
AlarmUpperLimitColor property (VBS), 5238
AlarmUpperLimitEnabled property (VBS), 5238

AlarmUpperLimitRelative property (VBS), 5239
AlarmView object, 5029
AlarmViewAcknowledgeAlarm, 4725
AlarmViewEditAlarm, 4724
AlarmViewShowOperatorNotes, 4726
AlarmViewUpdate, 4724
Align
  Object flush, 3946, 4512
Alignment property (VBS), 5762
AlignmentLeft property (VBS), 5763
AlignmentTop property (VBS), 5678
Allen-Bradley, 6599, 6624, 6625
  Allen-Bradley DF1 communication driver, 6624, 6625
  Analog alarm, 6642
  Area pointer, 6637
  Basic Panel, 6049
  Communication drivers, 6602
  Data type, 6641, 6642
  DF1, 6602
  EtherNet/IP, 6602, 6616
  Mitsubishi, 6672
  Panel, 6054
Allen-Bradley DF1
  Configuring a connection, 6620
  Connection, 6620, 6624
  Connection parameters, 6621
  CPU type, 6634
  KF2 module, 6624, 6625
  KF3 module, 6624, 6625
  Migrating data types, 197
  Valid data type, 6632
Allen-Bradley DH485
  Migrating data types, 197
Allen-Bradley Ethernet IP
  Address multiplexing, 6614
  Addressing, 6610
  Addressing type, 6612
  Migrating data types, 198
Allen-Bradley EtherNet/IP
  Configuring a connection, 6603
  Connection, 6603, 6607
  Connection parameters, 6605
  Data type, 6608
AllFilters property (VBS), 5239
AllFiltersForHitlist property (VBS), 5239
AllowEdit property (VBS), 5240
AllowMenu property (VBS), 5240
AllowPersistence property (VBS), 5763
Alphanumeric key assignment, 6915, 6974
ALT
  Key, 6915, 6974
Analog alarm, 4281, 4282
  Allen-Bradley, 6642
  Configuring, 4303, 4314
  Mitsubishi, 6672
  Omron, 6723
Analog alarm types, 4281, 4282
Analog alarms
  Configuring, 4295, 4296
Analog module
  Resetting to factory settings, 1397
Analog property (VBS), 5240
Analog value, 2390, 2393, 2670, 2672, 2885, 2888
Analog value processing, 2390, 2393, 2670, 2672, 2885, 2888
AND, 1665, 2424, 2471, 2472, 2704
AngleMax property (VBS), 5240
AngleMin property (VBS), 5241
Animation, 4016
  Configuring, 4018, 4443
  Diagonal movement, 4022
  Direct movement, 4022
  Green arrow in Overview, 4018
  Horizontal movement, 4021
  Multiple selection, 4030
  Object group, 4028, 4029
  Overview, 4018
  Tag binding, 4026
  Vertical movement, 4022
Antivirus programs, 111, 120
ANY, 1948
Appearance
  Dynamization of an object, 4019
Appearance property (VBS), 5242
Applet, 695
Application, 1143
  Alarm indicator, 4361
  Alarm view, 4355
  Alarm window, 4355
  Date/time field, 4517
  Key switch, 4164
  Recipe view, 4153, 4477
  Simple alarm view, 4358
  simple alarm window, 4358
  Simple user view, 4099
  User view, 4102
Application example
  Entering recipe data offline, 4489
  Recipe with manual production sequence, 4491
ApplicationWindow object, 5157
ApplyProjectSettings property (VBS), 5242
ApplyProjectSettingsForDesignMode property (VBS), 5242

Arccosine, 2328, 2605, 2816
ArchiveLogFile, 4663, 4861
Archiving projects, 386, 387, 388
Arcsine, 2327, 2603, 2815
Arctangent, 2817
Arctangent value, 2329, 2606
Area of unplugged modules, 563
Area pointer, 6039, 6077, 6119, 6206, 6299, 6354, 6420, 6476, 6728
    Acquisition cycle, 6042
    Active, 6042
    Address, 6042
    Allen-Bradley, 6637
    Availability, 6077
    Basic Panel, 6052
    Comment, 6042
    Connection, 6040, 6046
    Connections editor, 6041
    Coordination, 6733
    Data record, 4436, 6128, 6216, 6306, 6363, 6430, 6484, 6737
    Date/time, 6207, 6208, 6421, 6422
    Job mailbox, 6126, 6214, 6304, 6360, 6428, 6482, 6734
    Length, 6042
    Migration, 179
    Mobile Panel, 6072
    PLC tag, 6042
    Pointer name, 6042
    Project ID, 6125, 6213, 6303, 6359, 6427, 6481, 6734
    Screen number, 6120, 6212, 6299, 6355, 6426, 6477, 6730
    Tab, 6042
    Trends, 6133, 6223, 6313, 6369, 6437, 6491, 6507, 6638, 6669, 6701, 6721
Area pointers
    Configuring, 6044
    Coordination, 6124, 6210, 6302, 6358, 6424, 6480
    Creating, 6044
    Date/time, 6121, 6300, 6356, 6478, 6731
    Date/time PLC, 6122, 6301, 6357, 6479, 6732
    Length, 6045
    Mitsubishi, 6668
    Modicon Modbus, 6700
    Omron, 6720
ARP, 820
    Diagnostics, 820
    Table, 820
Arrange
    Object in the screen, 3939
Arrangement of byte sequence, 2634, 2837
Array, 67, 1941, 1942, 4236, 4238
    Creating, 4238
    Indirect addressing, 4232
    see ARRAY, 1559, 1711
ARRAY, 251, 1467
    Addressing, 1459
    Declaration in global data blocks, 1711
    Declaration in PLC data types, 1738
    Declaration in the block interface, 1559
    Example, 1943, 1944
    Format, 1941, 1942
ARRAY component, 99
Array data block, 2357, 2359, 2361, 2364, 2635, 2637, 2639, 2642, 2838, 2840, 2842, 2845
ARRAY data block, 239, 1416, 1419, 1456, 1459, 1508, 1704, 1706, 1730
ARRAY DB, 239
Array element
    Location of use of HMI tag, 66
    Name, 66
ARRAY limits, 1668
Array tag, 4236
    Char, 67
Article number, 549
AS interface, 1111
ASCII code table, 650
ASCII TSAP, 650
Asian, 43
Asian characters
    Input on the HMI device, 6847
    Interpretation, 6847
    Memory requirements, 6847
Asian languages
    Configuration, 6847
    Font size, 6847
    Text field length, 6847
Asian operating system, 6825
ASIN, 2327, 2603, 2815
AskOperationMotive property (VBS), 5242
Assembling manuals, 360
Assign
    Object of a layer, 4052
Assigning
    a function to a function key, 4041
    a graphic to a function key, 4045
    Function key, 4036, 4039, 4040
Assigning an IP address
    Basics, 1389
    from the project context, 1390
    Using "Accessible devices", 1389
Assigning global data blocks, 100

Assigning symbol, 851
Assigning tag, 851
Assignment, 1666, 2201, 2477
  Negate, 2202, 2478
Assignment list
  Enabling the display of retentive bit memories, 1818
Assignment list
  Defining filter, 1815
  Delete filter, 1815
  Displaying, 1813
  Example for displaying bit memory, 1811
  Example for displaying inputs and outputs, 1811
  Filter options, 1815
  Filtering, 1816
  Introduction, 1810
  Meaning of symbols, 1812
  Setting view options, 1814
  Structure, 1811
Assignments property (VBS), 5243
AssociatedS7GraphDBTag property (VBS), 5243
AssumeOnExit property (VBS), 5679
AssumeOnFull property (VBS), 5679
Asynchronous
  Transferring data, 6128, 6217, 6307, 6363, 6431, 6489, 6737
Asynchronous error event
  Delaying with DIS_AIRT, 3245
  Disabling with DIS_IRT, 3242
  Enabling with EN_AIRT, 3245
  Enabling with EN_IRT, 3243
AT, 1462
ATAN, 2329, 2606, 2817
ATH, 3035
ATTACH, 1182, 3216
AttachDB, 5830
ATTR_DB, 3355
Audit
  Configuration, 7080
  Enhancements in the ES, 7042
  Forcing, 7065
  Functional scope, 7042
  Logging concept, 7043
  Scope of logging, 7043
  Screen object, 7081
  Supported HMI devices, 7080
Audit log, 765, 767
Audit Trail
  Acknowledgement, 7069
  Checksum, 4262, 4346, 4423, 7063
  Comments, 7069
  CSV file, 7062
  Editor, 7046
  Effects in runtime, 7066
  Electronic signature, 7069
  File format, 7062
  Log tag value change, 7066
  Logging recipe data changes, 7069
  Logging system functions, 7075
  Logging user actions, 7072
  Memory medium, 7064
  Printing, 7054
  Protection against change, 7065
  Reporting, 7054
  Storage location, 7064
  Troubleshooting, 7064
Audit trail editor, 7046
Authentication, 983, 1075
Authentication methods, 744, 745
Authorization
  Assignment, 4539, 4565
  Changing the name, 4543
  Configuring, 4557
  Creating, 4537, 4563
  Deleting, 4543
  Managing, 4542
Authorization property (VBS), 5244
AutoCompleteColumns property (VBS), 5245
AutoCompleteRows property (VBS), 5246
Autocompletion, 4578
  Function, 1547
  Insert tag, 1547, 1548
  Inserting an instruction, 1548
Automatic
  Reporting, 4305
Automatic commissioning, 1137
Automation License Manager, 6766
Automation system, 5995, 6018
  Local, 7711
  Remote, 7711
  Setting up, 5995
Autonegotiation, 1136
  Disabled, 6102, 6187, 6268, 6405
AutoScroll property (VBS), 5246
AutoSelectionColors property (VBS), 5247
AutoSelectionRectColors property (VBS), 5247
AutoSizing property (VBS), 5248
Availability
  Object for Basic Panels, 4088
  Object for Comfort Panel, 4090
  Object for Mobile Panels, 4094
  Object for Panels, 4089
  Object for WinCC Runtime Advanced, 4096
  Objects for Multi Panels, 4092

Availability for specific devices
Screen, 3891
Available system functions
Basic Panels, 4612, 4618
Comfort Panels, 4636
Mobile Panels, 4643
Multi Panels, 4630
Panels, 4624
WinCC Runtime, 4651
AverageLast15Values property (VBS), 5248
AW, 1482
AWP command, 862, 863
AWP_In_Variable, 867
AWP_Out_Variable, 864
AX, 1482
Axis and command table technology object
List of ErrorIDs and ErrorInfo, 7475
AxisXBunchCount property (VBS), 5249
AxisXMarkCount property (VBS), 5249
AxisXShowBunchValues property (VBS), 5249
AxisY1BunchCount property (VBS), 5249
AxisY1MarkCount property (VBS), 5249
AxisY1ShowBunchValues property (VBS), 5249
AxisY2BunchCount property (VBS), 5249
AxisY2MarkCount property (VBS), 5250
AxisY2ShowBunchValues property (VBS), 5250

B

Back page
Report, 4495
Back up RAM file system, 4779, 4863
BackColor property (VBS), 5250
BackColorBottom property (VBS), 5252
BackColorTop property (VBS), 5252
BackFillStyle property (VBS), 5253
BackFillStyleReadOnlySpecial property (VBS), 5764
BackFlashingColorOff property (VBS), 5254
BackFlashingColorOn property (VBS), 5255
BackFlashingEnabled property (VBS), 5256
BackFlashingRate property (VBS), 5257
Background color
Dynamization, 4019
BackgroundColor property (VBS), 5258
Backing up the table layout, 335
Backing up the user interface, 335
Backing up the layout in editors, 335
BACKSPACE key, 6914, 6973
Backup
Deleting, 6883, 6958, 6983
Rename, 6883, 6958, 6983
Backup from online device, 7598, 7600
BackupRAMFileSystem, 4779, 4863
Bar, 4097
Color transition, 4098
Display limit lines, 4098
Bar object, 5033
Bar segment
Defining, 4098
BarBackColor property (VBS), 5258
BarBackFillStyle property (VBS), 5258
BarBackFlashingColorOff property (VBS), 5260
BarBackFlashingColorOn property (VBS), 5260
BarBackFlashingEnabled property (VBS), 5260
BarBackFlashingRate property (VBS), 5261
BarColor property (VBS), 5261
BarEdgeStyle property (VBS), 5261
BarStartValue property (VBS), 5765
BA-Send, 1269
BA-Send 1xFC, 1269
BaseScreenName property (VBS), 5261
Basic mode, 1855
Basic Panel
Area pointer, 6052
Communication drivers, 6049
Display and operating element, 4088
ETHERNET, 6051
IF1B, 6051
Interface, 6051
Mitsubishi, 6049
Modicon Modbus, 6049
Omron, 6049
OPC, 6049
Runtime start, 6903
Basic Panels
Available system functions, 4612, 4618
Basics
HMI HTTP protocol, 6555
Migration, 162
Battery object, 5038
BCDCPL, 2469, 2750, 2968
BeginTime(i) property (VBS), 5765
Behavior
Simple recipe view, 4471
BelowLowerLimitColor property (VBS), 5262
Benefits of using TeleService, 7662
Bit (0, 1)
Graphics list, 4007
Text list, 3996
Bit field
Reset, 2482
Resetting, 2206
Set, 2481
Setting, 2205

Bit logic operation
AND, 2471, 2472
EXCLUSIVE OR, 2475
Insert input, 2476
OR, 2473, 2474
Bit mask, 2460, 2741, 2959
Bit memories
Enabling the display of retentive bit memories, 1818
Bit number (0 - 31)
Graphics list, 4009
Text list, 3997
Bit string, 1913, 1914, 1915
64-bit, 1915
BitNumber property (VBS), 5262
Bits
Count, 2470, 2969
Counting, 2751
BITSUM, 2470, 2751, 2969
Bit-triggered trends, 6133, 6223, 6313, 6369, 6437, 6491, 6507, 6638, 6669, 6701, 6721
BlinkColor property (VBS), 5263
BlinkMode property (VBS), 5263
BlinkSpeed property (VBS), 5264
BLKMOV, 2375, 2653, 2870
Block
Block, 2653
Changing passwords for know-how protected blocks, 1808
Closing, 1526
Comparing, 1754
Comparing code blocks, 1752
Comparing data blocks, 1753
Comparison, 1750
Compiling, 1782
Compiling in the program editor, 1785
Compiling in the project tree, 1784
Consistency check, 1782, 1823, 1828, 1829
Consistency check in the call structure, 1824
Copying, 1511, 1514
Deleting offline, 1527
Deleting online, 1527
Displaying properties, 1523
Download to device, 1786, 1794
Downloading to a memory card, 1800
Downloading to device in RUN operating mode, 1790
Editing properties, 1523
Entering a comment, 1516
Entering a title, 1515
exporting to an external source file, 1746
Fill, 2352, 2380, 2630, 2659, 2833, 2875
Fill uninterruptible, 2353, 2632, 2835
Find and open, 1524
Inserting, 1511
Know-how protection, 1802
Leave, 2910
Move, 2621, 2825, 2870
Move uninterruptible, 2350, 2378, 2627, 2656, 2831, 2873
Moving, 2344, 2375
Opening, 1524
Opening know-how protected blocks, 1806
Optimized access, 1419, 1421
Pasting, 1515
Printing know-how protected blocks, 1807
Properties, 1518
Removing copy protection, 1804
Renaming, 1526
Saving, 1525
Setting up copy protection, 1804
Time stamp, 1520
Types, 1413
Upload from device, 1786
Uploading blocks from a memory card, 1801
Uploading blocks from device, 1799
Using a library, 1510
Write to memory card, 1800
BLOCK, 1953
Block access
Data block, 1716
Block call
"Call by reference" or "Call by value", 1434
Basics, 1424
Calling as single instance or multiple instance, 1426
Changing, 1598, 1640
Correcting the call type, 1598, 1640
Inserting, 1595, 1637, 1684, 1685, 1687, 1688, 1689, 1690
Multi-instance, 1427
Nesting depth, 1425
Single instance, 1427
Transfer parameter as copy or as pointer, 1434
Updating, 1596, 1638, 1692
Block comment
Hiding, 1538
Showing, 1538
Block comparison
Basics, 1750
Comparing code blocks, 1752
Comparing data blocks, 1753
Detailed comparison, 1756, 1758, 1759, 1763, 1766, 1769

Execute action, 416
Navigation, 1775
Synchronize scrolling, 1776
Updating comparison results, 413, 1778
Block consistency
Checking, 1824
Checking in the dependency structure, 1829
Block folder, 1506
Block interface
Declaring ARRAY, 1559
Declaring PLC data type, 1562
Declaring STRUCT, 1560
Declaring tags, 1557, 1558, 1562
Hiding, 1538
Importing and exporting tags, 1575
Multi-instance, 1563
Purpose of tag declaration, 1551
Retentivity, 1568
Showing, 1538
Structure, 1551
Tag properties, 1567, 1569
Updating, 1564
Valid data types, 1554, 1556
Block parameters, 1429, 1431, 1433, 1437, 1438, 1439, 1440
Basics, 1227
Block interface, 1551
Block property
Displaying, 1523
Editing, 1523
Function, 1517
Overview, 1518
BLOCK_DB_TO_WORD, 2193
Blocks property (VBS), 5264
Bookmarks
Deleting, 1676
Function, 1674
Navigating, 1675
Setting, 1674
BOOL, 1912, 1962, 2010, 2094, 2113, 2160, 2171
BOOL_TO_, 2010, 2113, 2171
BorderBackColor property (VBS), 5265
BorderBrightColor3D property (VBS), 5266
BorderColor property (VBS), 5267
BorderEnabled property (VBS), 5268
BorderEndStyle property (VBS), 5269
BorderFlashingColorOff property (VBS), 5269
BorderFlashingColorOn property (VBS), 5271
BorderFlashingEnabled property (VBS), 5272
BorderFlashingRate property (VBS), 5273
BorderInnerStyle3D property (VBS), 5274
BorderInnerWidth3D property (VBS), 5274
BorderOuterStyle3D property (VBS), 5275
BorderOuterWidth3D property (VBS), 5275
Borders
Placeholder for document information, 432
Specifying the print area, 431
BorderShadeColor3D property (VBS), 5276
BorderStyle property (VBS), 5277
BorderStyle3D property (VBS), 5278
BorderWidth property (VBS), 5278
BorderWidth3D property (VBS), 5280
BottomMargin property (VBS), 5280
Boundaries, 1137, 6103, 6188, 6269, 6406
Boundary reached, 4853
Box ID, 5959
Branch, 2898, 2900
Closing, 1614
Definition, 1612, 1654
Deleting, 1615, 1656
Inserting, 1614, 1655
Rules, 1613, 1655
BRCV, 3623
Broadcast, 717, 725, 1047
BrowserView object, 5096
BSEND, 3621
BusyText property (VBS), 5280
Button, 4161
Adding a system diagnostics indicator, 4413
as release button, 6971
Configuring, 4564
Configuring access protection, 4559
Define hotkey, 4162
for release of operator control, 6969
Graphic, 4162
HTML Browser, 4117
Lock/unlock, 6970
Mode, 4161
PDF view, 4147
Status/Force, 4172
Text, 4162
Button object, 5040
ButtonBackColor property (VBS), 5281
ButtonBackFillStyle property (VBS), 5281
ButtonBarElements property (VBS), 5281
ButtonBarStyle property (VBS), 5281
ButtonBorderBackColor property (VBS), 5281
ButtonBorderColor property (VBS), 5281
ButtonBorderWidth property (VBS), 5281
ButtonCommand property (VBS), 5682
ButtonCornerRadius property (VBS), 5282
ButtonEdgeStyle property (VBS), 5282
ButtonFirstGradientColor property (VBS), 5282
ButtonFirstGradientOffset property (VBS), 5282

ButtonMiddleGradientColor property (VBS), 5282
Buttons and Switches
Library, 6774
ButtonSecondGradientColor property (VBS), 5282
ButtonSecondGradientOffset property (VBS), 5282
BV_ColumnWidth_Date property (VBS), 5283
BV_ColumnWidth_Event property (VBS), 5283
BV_ColumnWidth_EventSeverity property (VBS), 5283
BV_ColumnWidth_EventState property (VBS), 5283
BV_ColumnWidth_Number property (VBS), 5283
BV_ColumnWidth_Time property (VBS), 5283
BV_ItemText_Date property (VBS), 5283
BV_ItemText_Event property (VBS), 5284
BV_ItemText_EventSeverity property (VBS), 5284
BV_ItemText_EventState property (VBS), 5284
BV_ItemText_Number property (VBS), 5284
BV_ItemText_Time property (VBS), 5284
BV_ShowItem_Date property (VBS), 5284
BV_ShowItem_Event property (VBS), 5284
BV_ShowItem_EventSeverity property (VBS), 5285
BV_ShowItem_EventState property (VBS), 5285
BV_ShowItem_Number property (VBS), 5285
BV_ShowItem_Time property (VBS), 5285
BY, 2902
BYTE, 1913, 1963, 2012, 2095, 2114, 2161, 2172
Byte assignment
Direct key, 6532
BYTE_TO_, 2012, 2114, 2172
Bytes
Swap, 2355, 2634, 2837

C

CA certificate, 686, 689, 7697
CA group certificate, 689
Cabling rules (PROFINET), 1136, 6102, 6187, 6268, 6405
CALC, 2300, 2575
Calculate, 1599, 1641, 2300, 2575
CALCULATE, 1599, 1641
CalculateStatistic, 5830
CalibrateTouchScreen, 4702, 4863
Calibrating, 1395
Overview, 1394
Call environment
Basics, 1841
Setting for blocks, 1842
Call hierarchy, 1425
Call structure, 1818
Displaying, 1822
Introduction, 1818
Meaning of symbols, 1820
Setting view options, 1822
Structure, 1821
Callback to a number specified during connection establishment, 7679
Calling FAQs for a module, 551
Calling manuals for a module, 551
Calling user-defined documentation, 370
Camera URL, 4119
Camera view
Configuring, 4119
CAN_DINT, 3231
CAN_TINT, 3226
Cancel
Connection to the faceplate type, 4079
Canceling a calibration, 1397
Canceling printing, 437
Caption property (VBS), 5285
CaptionBackColor property (VBS), 5286
CaptionColor property (VBS), 5286
CaptionText property (VBS), 5287
CaptionTop property (VBS), 5288
Card type, (See Memory card)
Cascaded counting function, 1255
CASE, 2900
Catalog, (see Hardware catalog)
CD, 2275, 2553
CEIL, 2386, 2666, 2881
CellCut property (VBS), 5288
CellSpaceBottom property (VBS), 5289
CellSpaceLeft property (VBS), 5289
CellSpaceRight property (VBS), 5290
CellSpaceTop property (VBS), 5290
CenterColor property (VBS), 5291
CenterSize property (VBS), 5291
CentrePoint property (VBS), 5766
CentrePointLeft property (VBS), 5767
CentrePointTop property (VBS), 5767
Certificate, 686, 745, 6568, 7146
Exporting, 686
Importing, 686
Importing on HTTP client, 6568, 7146
Installing on devices, 6568, 7146
Installing under Windows XP, 6568, 7146
Renewing, 688
Replace, 689
Replacing, 689
self-signed, 688
signed by certificate authority, 688
Certificate authority, 686, 687
Certificate manager, 686

change
Recipe data record in Runtime, 4481
Change, 4846
Change word order
Modicon MODBUS TCP/IP, 6681
ChangeConnection, 4802, 4864
ChangeMouseCursor property (VBS), 5292
Changeover contact, 1337
Changeover contact sensor type, 1337
Changing
Colors, 6889
Displayed name of user group, 4542
Logoff time in runtime, 4552
Name of the user, 4552
Object property, 4604
Object size, 3944, 4508
Password, 4541
User group in runtime, 4553
Changing a port interconnection
Graphic view, 670
Changing and displaying operating mode (example), 4608
Changing the configuration online, 7587
Changing the device configuration online, 7587
Changing the name
Authorization, 4543
User group, 4542
Users, 4541
Changing the PROFIBUS connection
Operating mode, 6517
Changing the PROFINET connection
Operating mode, 6517
ChannelDiagnose object, 5044
Char
Array tag, 67
CHAR, 1936, 2004, 2082, 2109, 2110, 2152, 2153, 2170, 2191
CHAR_TO_, 2082, 2152, 2153, 2191
Character, 1936, 1937
Character string, 101, 1937, 1939
Combining character strings with CONCAT, 3040
Comparing string tags with S_COMP, 3003
Converting from a hexadecimal number with HTA, 3037
Converting number string with STRG_VAL, 3008
Converting number to character string with VAL_STRG, 3011
Converting to a hexadecimal number with ATH, 3035
Converting with S_CONV, 3005
Copying characters to character string with Chars_TO_Strg, 3018
Deleting characters with DELETE, 3047
Determine maximum length with MAX_LEN, 3021
Determining length with LEN, 3039
Finding a character with FIND, 3053
Inserting characters with INSERT, 3049
Moving with S_MOVE, 3001
Reading out left character with LEFT, 3042
Reading out middle character with MID, 3045
Reading out right character with RIGHT, 3043
Replacing character with REPLACE, 3051
Characters, 43
Charging condition, 4125
Operation, 4125
Chars_TO_Strg, 3018
CheckBox object, 5046
Checking
Device version, 6880
Checking the connection, 3694
Checklist for troubleshooting the modem, 7721
CheckMarkAlignment property (VBS), 5292
CheckMarkCount property (VBS), 5293
Checksum, 4348, 7065
Audit Trail, 4262, 4346, 7063
Log, 4262, 4346, 4423, 7063
Updating WinCC, 4262, 4346
Circle, 4122
Radius, 4122
Circle object, 5049
CircleSegment object, 5051
Circular log, 4339
Select size, 4255, 4350
CircularArc object, 5054
CJ1, 6717
CJ2, 6717
Class of Service, 1011
ClearAlarmBuffer, 4722, 4866
ClearAlarmBufferProTool, 4723, 4867
ClearAlarmBufferProtoolLegacy, 4723, 4867
ClearDataRecordMemory, 4721, 4869
cleared, 4846
ClearLog, 4719, 4870
ClearOnError property (VBS), 5293
ClearOnFocus property (VBS), 5294
Click, 4851
Click when flashing, 4852
Client
Configuring, 6559
Parameters, 6560
Tags, 6564
Clipping, 1303
Clock, 4183
Display, 4183

Display dial, 4183
Length of the pointer, 4183
Width of hands, 4183
Clock memory, 1185, 1207
Clock object, 5056
Closeable property (VBS), 5294
CloseAllLogs, 4756, 4871
Closed loop control, 7395
CMP <, 2286, 2562
CMP <=, 2283, 2559
CMP <>, 2279, 2556
CMP ==, 2276, 2554
CMP >, 2284, 2561
CMP >=, 2281, 2558
Code templates
inserting in user-defined functions, 4579
Coil, 2201
Negate, 2202
Color
Diagnostics of Ethernet cables, 1362
Diagnostics of ports, 1362
Color of individual ranges
Gauge, 4192
Color property (VBS), 5295
Color transition
Bar, 4098
ColorChangeHysteresis property (VBS), 5295
ColorChangeHysteresisEnabled property (VBS), 5296
Colors
Changing, 6889
Find and replace, 6888
Column
Alarm view, 4133, 4136
Column headers
Alarm view, 4141
Recipe view, 4160
Column sequence
Alarm view, 4140
Column width
User view, 4103
ColumnAlignment property (VBS), 5296
ColumnColor(i) property (VBS), 5769
ColumnDisplayName(i) property (VBS), 5769
ColumnResize property (VBS), 5297
Columns property (VBS), 5298
ColumnScrollbar property (VBS), 5298
ColumnSettings property (VBS), 5299
ColumnSettingsBufferView property (VBS), 5299
ColumnSizingEnable property (VBS), 5740
ColumnsMoveable property (VBS), 5299
ColumnTextAckGroup property (VBS), 5299
ColumnTextAlarmState property (VBS), 5299
ColumnTextAlarmText property (VBS), 5299
ColumnTextClassName property (VBS), 5300
ColumnTextDevice property (VBS), 5300
ColumnTextDiagnosable property (VBS), 5300
ColumnTextNumber property (VBS), 5300
ColumnTextTime property (VBS), 5300
ColumnTitleAlignment property (VBS), 5301
ColumnTitles property (VBS), 5302
ColumnUpdateEnabled(i) property (VBS), 5770
ComboboxFont property (VBS), 5302
Comfort Panel
Area pointer, 6063
Deleting HMI device images, 6885, 6960, 6985
Display and operating element, 4090
Interface, 6062
S7-1200, 80
Comfort Panels
Available system functions, 4636
Command property (VBS), 5770
Command table technology object
Add new object, 7415
Basic parameters, 7417
Command table configuration, 7417
Command.Command[1...32] tags, 7530
Configuration window icons, 7416
Configuring activate warnings, 7417
Configuring duration, 7420
Configuring position / travel path, 7419
Configuring the command type, 7418
Configuring the next step, 7420
Configuring the step code, 7420
Configuring use axis parameters of, 7418
Configuring velocity, 7419
Extended parameters, 7429
General configuration, 7417
Tools, 7414
Usage, 7413
Comment
Area pointer, 6042
Comments
Audit Trail, 7069
Electronic signature, 7069
Inserting in SCL program, 1694
Common alarm classes, 4288
CommonTimeAxisColor property (VBS), 5771
Communication, 1212, 5994, 6752
Basics, 5995
between HMI devices, 6557
Configuring, 6580
Definition, 5994
ET 200 CPU, 6390

OPC, 6571
Routing, 6576, 6578, 6580, 7757
Runtime settings, 6558
S7 1200, 6162
S7 1500, 6079
S7 200, 6467
S7 300, 6249
S7 400, 6249
S7 routing, 7761
SIMATIC LOGO!, 6501
SIMATIC S7-1500 Software Controller, 6336
Third-party drivers, 6599
using area pointers, 5999
WinAC MP, 6749
Communication drivers, 5999
Allen-Bradley, 6602
Basic Panel, 6049
Mitsubishi, 6645
Modicon Modbus, 6675
Omron, 6707
Panel, 6054
Communication in the security mode
Configuring, 7165
Communication instruction
"AS_DIAL", 7712
"AS_MAIL", 7714
"PG_DIAL", 7711
"SMS_SEND", 7713
Communication load, 1185
Communication network, 6018
PROFINET, 6019
Communication partners
HMI device, 5994
Networking, 6022
PLC, 5994
SIMATIC S7, 6151, 6244, 6334, 6455, 6473, 6476
Communication types
Allen-Bradley, 6607
Connectable PLCs, 6683, 6694
Enabled, 6650, 6664, 6682, 6693, 6713
Ethernet/IP, 6607
Restriction, 6682, 6693
Communication via PUT/GET instructions
Basic information on PUT/GET instructions, 652
Creating and assigning parameters to a connection, 657
Deleting an interconnection, 658
Overview of connection configuration, 653
Requirements, 653
Starting connection parameter assignment, 656
Communications
Cycle load, 1190
Communications instruction
"AS_DIAL", 7711
"PG_DIAL", 7710
Communications load, 1190
Communications module (CM), 3744
Communications modules, 1198
Properties, 1198
Communications parameters
Ethernet, 6755, 6756
PROFIBUS DP, 6758, 6759
ProSave, 6755, 6758
WinCC, 6756, 6759
Communications port
Configuring, 1199
Communications protocol
Area pointer, 6077
Defining, 1202
Overview, 1201
Compact, 6685, 6698
CompactLogix, 6610
Comparator operations
Time tags, 2971
Compare
Bit mask, 2460, 2741, 2959
Character strings with S_COMP, 3003
Equal, 2276, 2554
Greater or equal, 2281, 2558
Greater than, 2284, 2561
Less or equal, 2283, 2559
Less than, 2286, 2562
Not equal, 2279, 2556
Compare editor
Changing the view, 417
Device comparison, 579
Filtering the view, 412
Overview, 405
Showing and hiding columns, 411
Specifying actions, 415
Compare offline/online
Automatic device assignment, 674
Compare scan matrix, 2462, 2743, 2961
Comparing library elements, 45
Comparing library objects, 471
Compatibility, 92, 100, 7154
Backward compatibility of projects, 378
Global libraries from older product versions, 485
Projects from older program versions, 377
Projects with add-on software, 378
S7-1200 CPU firmware versions, 1235
WinCC version, 6874

CompatibilityMode property (VBS), 5302
Compile
Block, 1784
Compiling
Address parameters, 79
Blocks, 1782
Consistency check, 1782
Correcting compilation errors, 1786
Migrated project, 166
Project, 6894, 6927
Complex user view
Configuring, 4100
Components
Alarm, 4293
Commissioning, 6618, 6636, 6658, 6667, 6687, 6699, 6719
CONCAT, 3040
Configuration
DP slave, 1096
DP slave, simple, 1087
Hardware, 554
Loading to device, 1147
Loading to PG/PC, 1148
Tag, 4213
Configuration control, 885, 894, 1274, 1278, 1316, 1326, 1346
Configuration control with ET 200pro, 1346
Configuration control with ET 200AL, 1316
Configuration control with ET 200MP, 1326
Configuration control with ET 200SP, 1274
Configuration mode, 967
Configuration PC
Ethernet, 6753
HMI device, 6753, 6757
PROFIBUS DP, 6757
Starting Runtime, 6950
WinCC Runtime, 6085, 6109, 6343, 6394
Configuration rights, 695
Configure a PROFIBUS subnet
Matching parameters to one another, 591
Setting bus parameters, 591
Configure acknowledgment
Alarm message, 6136, 6227, 6316, 6372, 6440, 6494, 6510, 6643, 6673, 6705, 6724
Configure alarms
Allen-Bradley, 6642
Data types, 6672
Non-integrated connection, 6672
Special considerations, 6672
Configured
Loop-in-alarm, 4311
ConfigureTimeAxis(i) property (VBS), 5771
Configuring, 4303, 4315, 4466
Access protection, 4564
Alarm acknowledgment, 4333, 4334, 4336
Alarm group, 4301
Alarm log, 4340
Alarm view, 4319
Alarm view for logged alarms, 4325, 4342
Alarms, 6135, 6226, 6371, 6439, 6493, 6509
Analog alarm, 4303, 4314
Analog alarms, 4295, 4296
Animation, 4443
Authorization, 4557
Camera view, 4119
Connections, 6146, 6147, 6153, 6324, 6327, 6328, 6381, 6383, 6449, 6451, 6458, 6467, 6502
Controller alarms, 4295, 4296
Direct connection, 7759
Discrete alarm, 4301, 4313
Discrete alarms, 4295, 4296
Display of S7 diagnostic alarms, 4323
Event, 4442, 4447, 4450
Event-driven tasks, 4305
Function list, 4575
Handle for slide-in screen, 3913
HART variable, 1247
HMI connection, 6085, 6088, 6168, 6171, 6254, 6257, 6343, 6346, 6394, 6397
HMI device as OPC DA server, 7157
HMI device as OPC DA Server, 7158
Integrated, 7764
I-slave, 1097
Layout of the alarm view, 4322
Logging tag, 4266
Multiple tags, 4213
OPC XML DA server in the OPC XML Manager, 7163
Project-wide alarm class, 4298
Properties for slide-in screen, 3912
Recipe screen, 4466
Recipe view, 4463, 4465
Rectangle, 3987
Server, 6558
Slide-in screen, 3912
Slide-in screen for devices without multi-touch function, 3914
System events, 4295, 4296
Trend view for logging, 4275
Trend view for values from the PLC, 4268, 4273
User-defined VB function, 4585
WinCC OPC Client, 7162
Configuring a connection
Allen-Bradley DF1, 6620

Allen-Bradley EtherNet/IP, 6603
Mitsubishi FX, 6659
Mitsubishi MC TCP/IP, 6646
Modicon Modbus RTU, 6688
Modicon Modbus TCP, 6676
Omron Host Link, 6708
OPC, 6572
Configuring a filter
for fixed character string, 4328
Configuring a PROFIBUS subnet
Meaning of the bus parameters, 592
Configuring a tag
HTTP client, 6562
HTTP server, 6559
Configuring internal subnets manually, 811
Configuring IP network nodes manually, 811
Configuring MAC network nodes manually, 811
Configuring the Control Panel
PROFIBUS DP, 6753
Configuring the network with Ethernet, 596, 6179, 6265
Creating private subnets, 598, 6180, 6266
Linking networks, 598, 6180, 6266
Relationship between IP address and subnet mask, 597, 6180, 6266
Setting the IP address, 597, 6179, 6265
Setting the subnet mask, 597, 6179, 6265
Configuring the remote modem, 7675
Connecting
PLC, 6607, 6649, 6663, 6681, 6712
Connecting a TS Adapter with an external modem, 7675
Connecting a TS Adapter with an internal modem, 7675
Connecting cable
6XV1440 - 2P for Mitsubishi PG protocol, 6665
Allen-Bradley cable 1784-CP10, 6629
Mitsubishi FX, 6663
Connecting cable 9-pole
Sub D, RS 422, 6628
Connecting cables
Point-to-point cable 1,
Point-to-point cable 2, 6696
Connection, 603, 616, 5998
Address details, 1410
Allen-Bradley DF1, 6620, 6624
Allen-Bradley EtherNet/IP, 6603
Area pointer, 6040, 6046
Change after migration, 178
Configuring, 6023, 6026, 6029, 6603, 6620, 6646, 6659, 6676, 6688, 6708
Configuring a connection when there is no or no clear network assignment, 608
Creating, 634, 6023, 6026, 6029
Deleting, 637
Highlighting, 6025
Integrated, 6023
integrated connection, 4202, 6036
Mitsubishi MC TCP/IP, 6646
Modicon, 6692
Modicon Modbus RTU, 6688
Modicon Modbus TCP, 6676
offline, 6125, 6213, 6304, 6360, 6427, 6481
Omron communication drivers, 6720
Omron Host Link, 6708
Parameters, 6605, 6621, 6647, 6661, 6678, 6690, 6710
Password, 6100, 6119, 6185, 6205, 6353
Routed, 6029
S7 200, 6467, 6502
Table, 6022
Connection boxes
For IRT communication, 6590
For media redundancy, 6594
For Mobile Panels, 6594
Connection cable
Allen-Bradley cable 1747-CP3, 6630
Allen-Bradley cable 1761-CBL-PM02, 6631
Modicon, 6693
Multipoint cable 1:MP/TP/PC, 6714
Multipoint cable 2:RS422, MP/TP/PC, 6715
Omron Host Link, 6712
Point-to-point cable PP2 for Omron, 6715, 6716
Connection configuration
Connection parameters, 630
General, 626
Overview, 628
Starting, 633
Connection description
Changing parameter values, 646
Data block, 640, 643, 644
Structure, 640, 643, 644
Connection details, 1409
Connection diagnostics
Detailed, 1407
Overview, 1406
Connection information, 1408
Diagnostics, 6006
Connection mechanisms, 1212
Connection name
Adapting, 7771
Connection parameter assignment of PUT/GET instructions, 656

Connection parameters
Allen-Bradley DF1, 6621
Allen-Bradley EtherNet/IP, 6605
Mitsubishi FX, 6661
Mitsubishi MC TCP/IP, 6647
Modicon Modbus RTU, 6690
Modicon Modbus TCP, 6678
Omron Host Link, 6710
Connection parameters of PUT/GET instructions, 655
Connection resource, 603, 616
Connection resources
Online, 1408, 1409
Connection rules, 833, 845
Connection status
Displaying using icons, 1407
Connections
Allen-Bradley DF1, 6624, 6625
Configuring, 6146, 6147, 6153, 6324, 6327, 6328, 6381, 6383, 6449, 6451, 6458, 6467, 6502
Editor, 6146, 6147, 6153, 6324, 6327, 6328, 6381, 6383, 6449, 6451, 6458
ConnectionType property (VBS), 5303
ConnectOnStart property (VBS), 5303
Connector object, 5058
ConnectTrendWindows property (VBS), 5303
Consistency
Slot rules, 556
Consistency check, 807, 1823, 1824, 1828, 1829
Introduction, 1823, 1828
local, 685
project-wide, 685
Constant
Data types, 1451
Definition, 1449
Global constant, 1450, 1451, 1452, 1493, 1494, 1495, 1497, 1498
Local constant, 1450, 1451, 1452, 1551, 1557, 1567
Non-typed constant, 1449
Symbolic constant, 1450
Typed constant, 1449
Constants, 295
Context filter, 557
ContinousChange property (VBS), 5304
CONTINUE, 2907
Control data record, 885, 1278, 1316, 1346
Control data record 196, 1274, 1278, 1316, 1326, 1346
Control element
Simple alarm view, 4359
Status/Force, 4171
Control panel
- assigning parameters, 6757
Transferring data, 6753, 6754
ControlDesignMode property (VBS), 5505
Controller alarm, 4282, 4283
System-defined, 4285
Controller alarms
Configuring, 4295, 4296
System-defined, 4282
Controller data, 7745
Initializing, 6816, 7747
ControlLogix, 6610
Controls
Camera view, 4119
PDF view, 4146
WinCC MediaControl, 5107
ControlSmartServer, 4791, 4872
Control-timer alarm, 2453, 2456, 2734, 2738, 2952, 2955
ControlWebServer, 4792, 4873
Conventions for user-defined documentation, 367
Conversion, 1959, 2091, 2157
Explicit, 2010, 2012, 2015, 2018, 2022, 2026, 2029, 2032, 2036, 2039, 2043, 2046, 2049, 2052, 2055, 2058, 2060, 2063, 2066, 2069, 2072, 2075, 2077, 2079, 2082, 2084, 2086, 2089, 2113, 2114, 2116, 2119, 2123, 2126, 2129, 2132, 2135, 2138, 2141, 2144, 2147, 2149, 2150, 2151, 2152, 2153, 2154, 2156, 2171, 2172, 2174, 2176, 2179, 2182, 2185, 2186, 2187, 2189, 2190, 2191, 2192, 2193
Implicit, 1962, 1963, 1966, 1967, 1969, 1970, 1972, 1973, 1975, 1977, 1979, 1981, 1983, 1985, 1986, 1988, 1990, 1992, 1994, 1996, 1998, 2000, 2002, 2004, 2005, 2007, 2008, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2160, 2161, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170
Conversion operations
Converting timers with T_CONV, 2972
Conversions, 102
Convert, 2383, 2662, 2878
HW identifier to slot, 3358
IO address to HW identifier, 3358
Slot to HW identifier, 3358
CONVERT, 2383, 2662, 2878
Convert character string, 101
Converted name (PROFINET), 1121
Converting
A character string to a hexadecimal number with ATH, 3035

Character string in array with Strg_TO_Chars, 3015
Character strings with S_CONV, 3005
Converting number string to number with STRG_VAL, 3008
Converting number to number string with VAL_STRG, 3011
Converting timers with T_CONV, 2972
Copying characters to numerical character string with Chars_TO_Strg, 3018
Hexadecimal number to character string with HTA, 3037
Converting an unspecified CPU, 213
Cookie, 868
Coordinated transfer, 4436
With PLC, 4436
Coordination, 5999
Copy
Recipe data record in runtime, 4480
Screen, 3897
Tag, 4212
Template, 3903
Copy protection, 1804
Copying
Adjusting screen size, 6868
Alarm indicator, 60
Alarm view, 60
Alarm window, 60
Color, 6867
Excel format, 6794
Font, 6867
Function key, 6868
Hardware component, 572
HMI device, 54
Invalid object, 6867
Linked objects, 6869, 6870
Pop-up screen, 3909
Principle, 6865, 6866
Screen, 6868
Style sheet, 3930
User-defined folders, 6866
CopyRows, 5831
Corner points, 4152
CornerRadius property (VBS), 5304
Corners, 4150
CornerStyle property (VBS), 5304
Corporate libraries, 471
CoS, 1011
Queue, 1011
COS, 2324, 2601, 2813
CoS (Class of Service), 927
Cosine, 2324, 2601, 2813
Count
Down, 2262, 2269, 2275, 2539, 2547, 2553, 2786, 2794
Up, 2259, 2268, 2274, 2536, 2545, 2783, 2792
Up and down, 2264, 2271, 2541, 2788, 2796
Up and Down, 2549
Count (8 DI NAMUR), 1253
Count property (VBS), 5306
CountDivisions property (VBS), 5306
COUNTER, 1953, 2271, 2796
Counter input, 1194
Counter mode, 1194
Counter, high-speed, 1193
Counters
Control high-speed counter (extended), 3441
Control high-speed counters, 3439
High-speed counters, 3439, 3441
Counting
Up, 2552
CountOfElements, 2369, 2648, 2868
CountOfLinesPerAlarms property (VBS), 5307
CountOfVisibleAlarms property (VBS), 5307
CountSubDivisions property (VBS), 5307
CountValueColumns property (VBS), 5688
CountVisibleItems property (VBS), 5307
Cover page
Placeholder for document information, 432
CP 1613, 46
CP 1623, 46
CP 343-2, 94
CP 5512, 46
CP1, 6717
C-PLUG
Formatting, 1000
CPM, 6717
CPU
Displaying the current LED status, 1372
Fill level of all types of memory, 1373, 1374
Inserting a signal board, 853
Properties, 1184
Reading date and time-of-day with RD_SYS_T, 2983
Reading out a diagnostics buffer, 1376
Selecting from the hardware catalog, 564
Setting time-of-day with WR_SYS_T, 2981
Switching operating mode, 1382
CPU control panel
Display area, 1373
CPU data block
Definition, 1418
Deleting, 1528
CPU firmware, 1235

CPU load through trace, 7648
CPU memory area
Displaying, 1833
CPU properties, 1185
CPU type
Allen-Bradley DF1, 6634
Allen-Bradley EtherNet/IP, 6610
FX3 series, 6655
Mitsubishi FX, 6666
Mitsubishi MC TCP/IP, 6655
Modicon Modbus RTU, 6698
Modicon Modbus TCP, 6685
Omron Host Link, 6717
Create
Connection, 6163
Faceplate type, 4062
Global library, 6779
Group, 3978
Recipe data record on the HMI device, 4480
Report, 4499
Report (overview), 4497
Template, 3904
Users, 4539
Users in runtime, 4550
Create script
Faceplate type, 4083
CREATE_DB, 3348
CreateTagSet-Methode, 5832
Creating, 4456
Alarm group, 4301
Alarm log, 4338, 4340
Array, 4238
Connection, 6023, 6026, 6029, 6081, 6250, 6340, 6391
Connection to an OPC UA Server, 7161
Connection to OPC DA server, 7160
Cycle, 4248
Data log, 4253
External tag, 4207
Infotext, 4302, 4305
Internal tag, 4209
Network connection, 7766
Pop-up screen, 3908
Project-wide alarm class, 4298
Recipe, 4456
Recipe data record on the HMI device, 4474, 4475
Report (overview), 4497
Screen, 3896
Style sheet, 3927
User group, 4538, 4565
User-defined VB function, 4585
Users, 4567
Creating a CHM file, 373
Creating a homepage for user-defined documentation, 366
Creating a new type
Style, 6791
Style sheet, 6792
Creating a print preview, 433
Creating a route, 802
Creating a type
Screen, 6790
Script, 6790
Creating a watch table, 1858
Creating custom documentation, 360
Creating labeling strips, 438
Creating user-defined documentation, 371
Crossing
Definition, 1615
Deleting, 1617
Inserting, 1616
rearranging, 1616
Cross-reference
Inspector window, 60
Cross-reference list, (See cross-references), (See cross-references)
Displaying, 1837, 6818
Overview, 1837, 6818
Settings, 1835, 6818
Sorting columns, 1835, 6818
Structure, 1835, 6818
Views, 1835, 6818
Cross-references
Displaying, 1837, 6818, 6820
Introduction, 1834, 6817
Linking, 6821
Replace, 6821
Uses, 1834, 6817
CS1, 6717
csv file
Example, 4276, 4366
Layout, 4276, 4366
CSV file, 6794, 6796, 6805
Audit Trail, 7062
CTD, 2262, 2539, 2786
CTRL
Key, 6915, 6974
CTRL_HSC, 3439
CTRL_HSC_EXT, 3441
CTRL_PWM, 3310
CTU, 2259, 2536, 2783
CTUD, 2264, 2541, 2788

CU, 2274, 2552
Cu10 sensor, 1300
CurrentColumnIndex property (VBS), 5688
CurrentCurveIndex property (VBS), 5772
Cursor key, 6913, 6973
CursorControl property (VBS), 5308
CurveColor(i) property (VBS), 5772
CurveLineWidth(i) property (VBS), 5773
Curves property (VBS), 5309
CurvesCount property (VBS), 5774
CurveUpdateEnabled(i) property (VBS), 5774
Custom alarm classes, 4287, 4288
Customized VB function
  Rename, 4588
CutRows, 5832
Cycle
  Creating, 4248
Cycle load, 1190
Cycle monitoring time, 2918
Cycle time, 1185, 1189
  display configured, 1366
  display measured, 1371
Cyclic
  Continuous, 4222
  In operation, 4222
Cyclic interrupt, 1220
Cyclic interrupt OB
  Assigning parameters, 1229
  Description, 1220
  Querying parameters with QRY_CINT, 3220
  Setting parameters with SET_CINT, 3219
Cyclic operation, 6336, 6476, 6506
Cyclic program execution
  Options for interrupting, 1213
  Programming, 1213
Cyclic triggers, 5982, 5991

D

D_ACT_DP, 3108
DangerRangeColor property (VBS), 5309
DangerRangeStart property (VBS), 5310
DangerRangeVisible property (VBS), 5310
Data areas
  Area pointer, 6044, 6730
  Reading, 6044, 6730
  Writing, 6044, 6730
Data backup, 133
  HMI device, 6991
Data bit (DBX), 1172
Data block
  Adjusting data values during commissioning, 1715, 1730, 1731, 1732, 1733
  ARRAY data block, 1416, 1419, 1456, 1459, 1508, 1704, 1706, 1730
  Based on a PLC data type, 1713
  CPU data block, 1418
  Creating, 1508, 1706
  Creating with CREATE_DB, 3348
  Declaration table, 1705
  Declaring ARRAY, 1711
  Declaring STRUCT, 1712
  Default value, 1713
  Deleting with DELETE_DB, 3357
  Downloading changes without reinitialization, 1709
  Global data block, 1416, 1704
  Importing and exporting tags, 1725
  Instance data block, 1417, 1704
  Monitoring and modifying tags, 1725, 1726, 1728
  Optimized access, 1419, 1421
  Programming, 1704, 1710
  Reading attributes with ATTR_DB, 3355
  Reading from load memory with READ_DBL, 3351
  Retentive behavior, 1716, 1717
  Snapshot, 1727, 1728
  Start value, 1713, 1714
  Tag properties, 1718, 1720, 1721
  Updating, 1708
  Using setting values, 1715, 1730, 1731, 1732, 1733
  Writing to load memory with WRIT_DBL, 3353
Data block 196, 1274, 1316, 1326, 1346
Data block 197, 1274, 1316, 1326, 1346
Data byte (DBB), 1172
Data consistency, 3608
Data double word (DBD), 1172
Data Encryption Standard (DES), 747
Data exchange, 5994
  DP slave, 1087
  I-slave - DP master, 1088
  Tags, 5998
  Trends, 6133, 6223, 6313, 6369, 6437, 6491, 6507, 6638, 6669, 6701, 6721
  using area pointers, 5999
Data exchange over the AS-AS remote link
  Communication instruction "AS_DIAL", 7711
Data Flow, 4429
Data flow control, 1200
Data log, 858, 4418
  Acquisition cycle, 4250

Closing with DataLogClose, 3337
Creating, 4253
Creating with DataLogCreate, 3323
Creating with DataLogNewFile, 3340
Creating with DataLogTypedNewFile, 3341
Deleting with DataLogDelete, 3338
Empty with DataLogClear, 3333
Logging cycle, 4250
Naming conventions, 4422
Opening with DataLogOpen, 3329
Opening with DataLogTypedOpen, 3331
Output of the tag value, 4272
Tags, 4251, 4256
Tolerance band, 4256
Writing with DataLogWrite, 3334
Data logging, 4249, 4250
Application, 4249
Data mailbox
For recipes, 4436
Data point, 904
Data record, 5999
Asynchronously reading data record of a module with RD_DPARA, 3210
Exporting, 4483
Importing, 4483
Making available on I-device with PRVREC, 3130
Reading, 4475, 4482
Reading from configured system data with RD_DPARM, 3213
Reading of a module with RD_DPAR, 3208
Reading with RD_REC, 3118
Reading with RDREC, 3069
Receiving on I-device with RCVREC, 3128
Transfer, 4476, 4483
Transferring with WRREC, 3071
Writing and reading data records, 3206
Writing predefined parameters with WR_DPARM, 3214
Writing with WR_REC, 3123
Data record 196, 1278
Data record 197, 1278
Data record list, 4440
Data transfer settings, 6757
Data type
Allen-Bradley, 6641, 6642
Allen-Bradley EtherNet/IP, 6608
ANY, 1948
ARRAY, 1941, 1942
BOOL, 1912, 1962, 2010, 2094, 2113, 2160, 2171
BYTE, 1913, 1963, 2012, 2095, 2114, 2161, 2172
CHAR, 1936, 2004, 2082, 2109, 2110, 2152, 2153, 2170, 2191
Conversion, 1959, 1966, 2091, 2097, 2106, 2107, 2108, 2157, 2163, 2166, 2168, 2169
Convert explicit, 2878
DATE, 1931, 1998, 2075, 2107, 2149, 2168, 2188
DINT, 1920, 1975, 2039, 2102, 2135, 2165, 2182
Discrete alarm, 6642, 6672, 6723
DT, 1933, 1992, 2066, 2169, 2190
DTL, 1935, 1996, 2072, 2109, 2151
DWORD, 1915, 1966, 2018, 2097, 2119, 2163, 2176
Explicit conversion, 2010, 2012, 2015, 2018, 2022, 2026, 2029, 2032, 2036, 2039, 2043, 2046, 2049, 2052, 2055, 2058, 2060, 2063, 2066, 2069, 2072, 2075, 2077, 2079, 2082, 2084, 2086, 2089, 2113, 2114, 2116, 2119, 2123, 2126, 2129, 2132, 2135, 2138, 2141, 2144, 2147, 2149, 2150, 2151, 2152, 2153, 2154, 2156, 2171, 2172, 2174, 2176, 2179, 2182, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2383, 2662
Implicit conversion, 1962, 1963, 1964, 1967, 1969, 1970, 1972, 1973, 1975, 1977, 1979, 1981, 1983, 1985, 1986, 1988, 1990, 1992, 1994, 1996, 1998, 2000, 2002, 2004, 2005, 2007, 2008, 2094, 2095, 2096, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2109, 2110, 2111, 2112, 2160, 2161, 2162, 2164, 2165, 2167, 2169, 2170
INT, 1918, 1972, 2032, 2100, 2129, 2164, 2179
Internal tag, 4205, 6039
LDT, 1934, 1994, 2069
LINT, 1922, 1979, 2046
LREAL, 1926, 1985, 2055, 2105, 2144
LTIME, 1931, 1990, 2063
LTIME_OF_DAY, 1932
LTOD, 2002, 2079
LWORD, 1915, 1967, 2022
Mitsubishi, 6672
Mitsubishi FX, 6665
Mitsubishi MC TCP/IP, 6654
Modicon Modbus RTU, 6697
Modicon Modbus TCP, 6684
Omron, 6722, 6723
Omron Host Link, 6716
PLC data type, 1734, 1954
POINTER, 1946
REAL, 1925, 1983, 2052, 2104, 2141, 2165, 2185
S5TIME, 1929, 1986, 2058, 2167, 2187
SCL instruction, 1679, 1680, 1681
SINT, 1917, 1969, 2026, 2098, 2123

STRING, 1937, 2007, 2086, 2111, 2112, 2154, 2156, 2170, 2192
STRUCT, 1945
TIME, 1930, 1988, 2060, 2106, 2147, 2166, 2186
TIME_OF_DAY, 1932
TOD, 2000, 2077, 2108, 2150, 2169, 2189
Trend, 6641
UDINT, 1921, 1977, 2043, 2103, 2138
UINT, 1919, 1973, 2036, 2101, 2132
ULINT, 1923, 1981, 2049
USINT, 1918, 1970, 2029, 2099, 2126
Valid, 6608, 6654, 6665, 6684, 6697, 6716
Validity, 1908
VARIANT, 1951
WCHAR, 1937, 2005, 2084
WORD, 1914, 1964, 2015, 2096, 2116, 2162, 2174
WSTRING, 1939, 2008, 2089
Data type conversion, 99
Data types, 225
ET 200 CPU, 6449
Migration, 196
Mitsubishi FX, 6671
Mitsubishi MC TCP/IP, 6671
Modicon Modbus RTU, 6703
Modicon Modbus TCP/IP, 6703
OPC DA, 6574
OPC UA, 6575
OPC XML DA, 6574
S7 1500, 6145, 6380
S7 200, 6500
S7 300/400, 6324
S7-1200 V1, 6235
Trends, 6135, 6225, 6314, 6371, 6439, 6492, 6509
Valid, 6145, 6235, 6324, 6380, 6449, 6500, 6574, 6575
Data word (DBW), 1172
DataFormat property (VBS), 5311
DataIndex(i) property (VBS), 5775
DataItem object, 4987
DataLogClear, 3333
DataLogClose, 3337
DataLogCreate, 3323
DataLogDelete, 3338
DataLogNewFile, 3340
DataLogOpen, 3329
DataLogs object, 4989
DataLogTag property (VBS), 5775
DataLogTypedNewFile, 3341
DataLogTypedOpen, 3331
DataLogWrite, 3334
DataRecordNameCaption property (VBS), 5312
DataRecordNrCaption property (VBS), 5312
DataSet object (list), 4990
DataX(i) property (VBS), 5776
DataXY(i) property (VBS), 5776
DataY(i) property (VBS), 5777
Date, 1931, 1933, 1934, 1935
DATE, 1931, 1998, 2075, 2107, 2149, 2168, 2188
Date/time, 5999
Area pointer, 6207, 6421
Date/time field, 4103
Application, 4517
Display system time, 4104
Format, 4104
Layout in reports, 4518
Using tags, 4104
Date/time PLC, 5999
DATE_AND_LTIME, 1934
DATE_AND_TIME, 1933
DATE_TO_, 2075, 2149, 2188
DateTimeField object, 5061
DB, 1508, 1706
DB variable
Definition, 1448
DB_ANY_TO_VARIANT, 2892
DCAT, 2453, 2734, 2952
DCP server, 967
Deactivate, 102, 4849
Dead peer detection (DPD), 754
Deadband
Setting, 4305
Debugger
Closing, 4600
Error types, 4593
Starting, 4599, 6934
Windows, 4592
Windows CE, 4592
DEC, 2311, 2587
Decimal places, 44, 2330, 2607, 2818
DECO, 2429, 2710, 2928
Decode, 2429, 2710, 2928
DecreaseFocusedValue, 4800
DecreaseTag, 4801, 4875
Decrement, 2311, 2587
Default font, 6844
Default style
Defining, 3922
Defective devices, 1355
Defective license, 117, 134
Define
Reference object, 3966, 4510

Define columns
Alarm view, 4140
Define display area
Alarm view, 4141
Define hotkey, 4162
Defining, 4349
For the assignment list, 1815
DEL key, 6914, 6973
Delay time
Setting, 4305
Delete
Template, 3903
DELETE, 3047
DELETE_DB, 3357
DeleteDataRecord, 4720, 4868
DeleteDataRecordMemory, 4721, 4869
DeleteDate(i) property (VBS), 5777
DeleteEnable property (VBS), 5689
DeleteRows, 5835
Deleting
Authorization, 4543
Filter in the assignment list, 1815
Hardware component, 571
Object, 3941, 4503
OPC XML DA server in the OPC XML Manager, 7164
Pop-up screen, 3910
Recipe data record in runtime, 4481
Report page, 4500
Screen, 3897
Style sheet, 3929
Tag, 4212
User group, 4543
Users, 4542
Deleting a category
Faceplate type, 4067
Deleting a connection, 217
Deleting a property
Faceplate type, 4067
Deleting CA certificates, 7702
Demultiplex, 2436, 2717, 2935
DEMUX, 101, 2436, 2717, 2935
Dependencies of rights, 697
Dependency on HMI device
Logging, 4420
Dependency structure, 1824
Introduction, 1824
Meaning of symbols, 1826
Setting view options, 1828
Structure, 1826
Dependency structure
Displaying, 1827
DES, 741, 747
Deserialize, 2338, 2615, 2819
Designation
TS Adapter, 7673, 7681, 7689
Designing a background, 3952
Designing a border, 3952
Destruction of license keys, 116
DETACH, 3218
DetachDB, 5835
Detail page
Adding to report, 4500
remove from report, 4500
Report, 4496
Sorting, 4500
Detailed comparison
Performing, 413
Starting, 1756, 1758
Visualization of the comparison result for GRAPH, 1769
Visualization of the comparison result for LAD/FBD, 1759
Visualization of the comparison result for SCL, 1766
Visualization of the comparison result for STL, 1763
Details view, 330
Determine diagnostics status, 1375
Determining time of day, 1383
Device
Adding to a hardware configuration, 565
Copying, 54, 572
Defective devices, 1355
Deleting, 571
Inserting, 5996
Moving, 573
Renaming, 673
Device address, 848
Device comparison, 579
Device dependency
S7 200, 6496
S7-300/400, 6318
SIMATIC LOGO!, 6511
SIMATIC S7-1500 V1.0, 6139
SIMATIC S7-1500 V1.0, 6375
Device information, 423, 1355
Diagnostics, 6006
Device name, 1115, 1121
Device name, automatic commissioning (PROFINET), 1117
Device number, 1116
Device overview
Address range, 848

Device proxy
Initializing, 7752
Updating, 7750, 7755
Device replacement, 574
Device Tool, 1145, 1146
Device type
Changing, 6751
Device version, 6901, 6995
Checking, 6880
Switching, 6879
Device view
Area of unplugged modules, 563
Edit parameters, 575
Edit properties, 575
Hardware and network editor, 540, 6009
Inserting a signal board, 852
Inserting module, 568
Racks, 558
Device wizard, 6766, 6768
Device-dependent
Font size, 3924
Devices
Connecting, 5998
Networking, 5998, 6022
Devices & Networks, 5998, 6022
Devices and networks, 5998, 6022
Connections, 6023, 6029
HMI connections, 6023, 6029
DeviceStates, 3292
DHCP
Client, 980
Server configuration, 805
DHCP server, 806
Diagnose repeater, 1403
Diagnostic data
Reading out with GET_DIAG, 3304
Diagnostic error interrupt, 1224
Diagnostic interrupt OB, 1224
Diagnostics, 757, 1264
Alarm class, 4323
Connection, 6006
Connection information, 6006
Device, 6006
Device information, 6006
DP slave, reading diagnostic data with DPNRM_DG, 3139
Diagnostics buffer
Basics, 1176, 1398
Organization, 1398
Diagnostics status
Determining and displaying online, 1356
Diagnostics user, 693
Diagonal movement
Animation, 4022
DialColor property (VBS), 5312
DialFillStyle property (VBS), 5313
Dialing rules in phone books, 7670
DialSize property (VBS), 5314
Dial-up connection
Establishing, 7694
Terminating, 7696
Diffie-Hellman key exchange, 746
DINT, 1920, 1975, 2039, 2102, 2135, 2165, 2182
DINT_TO_, 2039, 2135, 2182
Direct access to the I/O, 1177
Direct connection
Configuring, 7759
Direct key, 6516, 6523, 6541, 6976
Assigning the inputs/outputs, 6523, 6541
Bit coding, 6533
Byte assignment, 6532
configuring, 6518
KP 1500 Comfort, 6537, 6554
KP1200 Comfort, 6537, 6554
KP700 Comfort, 6535, 6552
KP900 Comfort, 6536, 6553
KTP400 Comfort, 6534, 6551
Mobile Panel 177, 6529, 6548
Mobile Panel 277, 6530, 6549
Mobile Panel 277 IWLAN V2, 6531, 6550
Multi Panel 277, 6528, 6547
Multi Panel 377, 6529, 6548
Multi Panel 177, 6526, 6545
OP 277, 6525, 6544
OP 177B, 6524, 6543
OP 77B, 6542
PROFIBUS DP, 6538, 6539, 6540
PROFINET IO, 6519, 6521, 6522
TP 177, 6525, 6544
TP 277, 6526, 6545
TP1200 Comfort, 6537, 6554
TP1500 Comfort, 6538, 6555
TP1900 Comfort, 6538, 6555
TP2200 Comfort, 6538, 6555
TP700 Comfort, 6535, 6552
TP900 Comfort, 6536, 6553
Direction output and travel direction relation, 7335
Direction property (VBS), 5778
DirectKey, 4672
Configuring, 4047
DirectKeyScreenNumber, 4674
Directories for user-defined documentation, 368
DIS_AIRT, 3245

DIS_IRT, 3242
Disabling
Project language, 6826
Disconnecting
PROFIsafe, 5956
Discrete alarm, 4281, 4282, 4283
Allen-Bradley, 6642
Configuring, 4301, 4313
Mitsubishi, 6672
Omron, 6723
Discrete alarm types, 4281, 4282
Discrete alarms
Configuring, 4295, 4296
DiskSpaceView object, 5063
display
Infotext, 6915, 6919, 6975, 6979
Display
HMI backup, 6881, 6956, 6981
Program name, 7765
Display classes, 4285
Display dial, 4183
Display number, 4156
Recipe view, 4159
Display object
Availability for Basic Panels, 4088
Availability for Comfort Panels, 4090
Availability for Mobile Panels, 4094
Availability for Multi Panels, 4092
Availability for Panels, 4089
Availability for WinCC Runtime Advanced, 4096
Display PDF files, 113
Display peak value
Gauge, 4191
Display system time, 4104
Display Welcome Tour, 113
DisplayButton2Plc property (VBS), 5314
DisplayButtonComparison property (VBS), 5314
DisplayButtonDelete property (VBS), 5314
DisplayButtonFromPlc property (VBS), 5314
DisplayButtonHelp property (VBS), 5315
DisplayButtonNew property (VBS), 5315
DisplayButtonSave property (VBS), 5315
DisplayButtonSaveAs property (VBS), 5315
DisplayComboBox property (VBS), 5315
DisplayGridLines property (VBS), 5315
Displaying
Assignment list , 1813
Call structure, 1822
CPU memory area, 1833
Cross-reference, 6820
Cross-references, 1837, 6818
Dependency structure , 1827
Load memory, 1834
Log contents, 4339
Maximum load memory available, 1834
Program information, 1809
Displaying a force table, 1886
Displaying alarms, 1157
Displaying memory types of a CPU, 1373, 1374
Displaying or hiding tag information, 1611, 1653, 1681
Displaying support for a module, 551
Displaying the call log, 371
Displaying the online status, 7592
Displaying types for an instance, 511
Displaying values
As a trend, 4268
DisplayLabeling property (VBS), 5315
DisplayName(i) property (VBS), 5778
DisplayNumbers property (VBS), 5316
DisplayOptions property (VBS), 5316
DisplayStatusBar property (VBS), 5316
DisplayTable property (VBS), 5316
Distribute
Objects evenly, 3946, 4512
Distributed I/O, 1085, 1262, 1269, 1313
Distributed operator stations
Configure SmartClient, 7121
Configure SmartServer, 7121
configuring, 7121
DIV, 2307, 2583
Divide, 2307, 2583
DMSK_FLT, 3240
DNS client, 971
DO, 2902, 2905
Documentation editor
Layout, 428
Documentation settings
Using frames and cover pages, 426
Double word, 1915
double-click, 4849
DoubleClickAction property (VBS), 5316
Download, (See Loading to device)
"Pack&Go" file to HMI device, 6945
Error messages, 6904, 6952
Project, 6902, 6940
via USB, 6946
Downloading
Blocks to the device, 1790
Downloading blocks to a memory card, 1799
Downloading blocks to device, 1795, 1796
Downloading to a memory card, 397
in RUN operating mode, 1790

Uploading blocks from a memory card, 1801
Writing to a memory card, 397
Downloading data to the PLC
Error message, 79
DP, 6204, 6284
DP interface, 1090
DP master, 848, 1249
Add DP master system, 1093
Devices and modules, 1090
Disconnection from DP master system, 1093
Display on DP slave, 1092
DP interface, 1090
DP master system
Add DP slave, 1095
Creating, 1086, 1089, 1090
Determining topology with DP_TOPOL, 3142
Disconnect DP slave, 1095
Disconnection from subnet, 1093
Edit properties, 1094
Highlight, 1092
Node disconnection, 1093
DP slave, 848
Activating and deactivating with D_ACT_DP, 3108
Add to DP master system, 1095
Assign DP master system, 1095
configuring, 1096
Configuring, 1087
Data exchange, 1087
Disconnect from DP master system, 1095
DP master display, 1092
Hardware catalog, 1086
Intelligent, (See I-slave)
Networking, 1089
Reading diagnostic data with DPNRM_DG, 3139
Synchronizing groups with DP_SYC_FR, 3133
Types, 1094
With preprocessing, (See I-slave)
DP standard slave, 1108
DP standard slaves
Reading a portion of the inputs with GETIO_PART, 3074
Reading all inputs with GETIO, 3072
Reading consistent data with DPRD_DAT, 3124
Writing a part of the outputs with SETIO_PART, 3075
Writing all outputs with SETIO, 3073
Writing consistent data with DPWR_DAT, 3126
DP/DP coupler, 1086
DP_TOPOL, 3142
DPNRM_DG, 3139
DPRD_DAT, 3124
DPSYC_FR, 3133
DPV1
Configuring an ET 200S, 1105
DPWR_DAT, 3126
DrawAlwaysInsideFrame property (VBS), 5689
DrawEnhancedHTMLBrowser property (VBS), 5779
DrawInsideFrame property (VBS), 5317
DrawStylishButton property (VBS), 5690
Drive property (VBS), 5318
DRUM, 2447, 2728, 2945
DSCP, 1011
DST
Daylight saving time, 985, 986
DT, 1933, 1992, 2066, 2169, 2190
DT_TO_, 2066, 2190
DTL, 1935, 1996, 2072, 2109, 2151
DTL_TO_, 2072, 2151
Duration, 1929
DWORD, 1915, 1966, 2018, 2097, 2119, 2163, 2176
DWORD_TO_, 2018, 2119, 2176
Dynamic control
Object property, 4012
Dynamic reference temperature, 1333
Dynamization
Color of an object, 4019
Control enable state of an object, 4024
Flashing, 4019
Object, 4012
Dynamize
Appearance of an object, 4019
Direct movement, 4022
Faceplate, 4079, 4080
Faceplate type, 4079
Green arrow in Overview, 4018
Movement of an object, 4021
Object appearance, 4413
Property list, 4015
Tag binding, 4026

E

Eastern characters
Input on the HMI device, 6847
EB, 1482
EC31-RTX, 94
ED, 1482
Edge
Negative, 2210, 2213, 2215, 2486, 2489, 2491, 2753
Positive, 2209, 2212, 2214, 2485, 2488, 2490, 2752
EdgeStyle property (VBS), 5319

Edit, 5836
Object within a group, 3982
Edit networking
Disconnecting from a network, 588
Editable(i) property (VBS), 5779
EditAlarm, 4665, 4875
EditEnabled property (VBS), 5780
Editing, (Tags)
Folder link, 3973
Function list, 4577
OPC XML DA server in the OPC XML Manager, 7164
Style sheet, 3929
System event, 4309
Editing a version of a type in testing, 513
Editing language, 454, 6823
Selecting, 6827
Editing networking
Copying a subnet, 588
Copying subnets and devices, 589
EditOnFocus property (VBS), 5320
Editor
Connection, 6026
Connections, 6146, 6147, 6153, 6324, 6327, 6328, 6381, 6383, 6449, 6451, 6458
Devices and networks, 6025
Graphics, 6836
Effective range, 5942, 5959, 5962, 5964
Calculating signal quality, 4188
Configure effective range object, 5972
Configuring, 5970
Configuring logoff, 5971
Configuring logon, 5971
Logoff, 4185
Logon, 4184
Override function, 4188
Principle of Operation, 5943
Runtime, 5962
Work area, 5960
Effective range (RFID), 5962
Configuring, 5973
Configuring logoff, 5974
Configuring logon, 5974
Overview, 5972
Runtime, 5964
Work area, 5962
Effective range name, 4184
Effective range name (RFID), 4186
Effective range signal, 4187
Effective range (RFID)
Logoff, 4187
Logon, 4186
Effective ranges editor, 5959, 5962
Effects in runtime
Audit Trail, 7066
Effects in Runtime
Recipe data change, 7071
Value changes to GMP-relevant tags, 7069
Electronic signature, 7042
Element list, 4440
Ellipse, 4107
Horizontal radius, 4107
Vertical radius, 4107
Ellipse object, 5065
EllipseSegment object, 5068
EllipticalArc object, 5070
ELSE, 2898, 2900
ELSIF, 2898
E-mail
Transferring e-mail with TM_MAIL, 3854
Transferring e-mail with TMAIL_C, 3648
E-Mail
Setting at the HMI device, 7089
E-mail notification, 4317, 7125
Configuring, 7126
Setting up the trigger, 7126
Empty box
Inserting a LAD element, 1589
Inserting an FBD element, 1631
EN/ENO mechanism
Basics, 1473
FBD example, 1476
LAD example, 1476
SCL example, 1477
STL example, 1478
EN_AIRT, 3245
EN_IRT, 3243
Enable autonegation, 1135, 6101, 6186, 6267, 6405
Enable output ENO, 227
Enable sorting
Alarm view, 4137
Enabled property (VBS), 5321
EnableEdit property (VBS), 5323
EnableNavigateKeys property (VBS), 5323
Enabling
Reporting, 4332
S7 diagnostic alarm, 4311
Enabling the firewall
SCALANCE S, 793, 794
Enabling tunneled communication
CP x43-1 Adv., 826
SCALANCE S, 793, 794
ENCO, 2430, 2711, 2930
Encode, 2430, 2711, 2930, 4670, 4876

EncodeEx, 4671, 4877
Encryption, 679
Encryption software, 120, 124
END key, 6914, 6974
End of detection of accessible devices, 1137
End of discovery of accessible devices, 6103, 6188, 6269, 6406
End of sync domain, 1137, 6103, 6188, 6269, 6406
End of topology discovery, 1137, 6103, 6188, 6269, 6406
END_CASE, 2900
END_FOR, 2902
END_IF, 2898
END_REPEAT, 2906
END_WHILE, 2905
EndAngle property (VBS), 5323
Endian, 2857, 2859, 2861, 2863
ENDIS_PW, 2408, 2688, 2912
EndLeft property (VBS), 5324
EndPoint property (VBS), 5780
EndPointLeft property (VBS), 5781
EndPointTop property (VBS), 5782
EndStyle property (VBS), 5324
EndTime(i) property (VBS), 5782
EndTop property (VBS), 5325
Energy-saving mode, 5976
Engineering station
    Starting Runtime, 6950
Engineering system
    Performance features, 7005
ENO, 102, 227
ENTER key, 6914, 6973
Entering
    Parameters in user-defined functions, 4578
EntryNameCaption property (VBS), 5325
Enumeration types, 870
Enumerations
    Web server, 871
EQ_ElemType, 2295, 2570
EQ_Type, 2292, 2568
Equal, 2297, 2573
EQUAL ARRAY, 2299, 2575, 2800
Error
    Logical error, 4593
    Runtime error, 4593
Error analysis with RET_VAL, 2195
Error handling, 1703
    Basics, 1699
    GetError, 2414, 2694, 2919
    GetErrorID, 2417, 2697, 2922
    Local error handling, 1701
Error information, 1701, 2195, 2414, 2417, 2694, 2697, 2919, 2922
Error message
    Downloading data to the PLC, 79
Error messages
    Download, 6904, 6952
Error status, 945
ESC key, 6914, 6973
ESP protocol, 747
Establishing a connection from a remote system ("PG_DIAL"), 7710
Establishing a remote connection, 7694
    Procedure, 7695
EstablishPROFIsafe, 4742, 4878
ET 200 CPU
    Communication, 6390
    Data types, 6449
ET 200eco, 1246
ET 200eco PN, 1246, 1262
ET 200iSP, 1245
ET 200iSP distributed I/O station
    Definition, 1248
ET 200L, 1245
ET 200M, 1245, 1336
    Definition, 1336
ET 200MP, 1324
ET 200pro, 1246
ET 200R, 1246
ET 200S, 1245
    DPV1 mode, 1105
    Option handling, 1102, 1104
    Positioning module, 94
    Reference junctions, 1098
    Slot rules, 1098
ET 200S COMPACT, 1245
ET 200SP, 1268
    Application area, 1268
ET 200AL, 1269, 1313
    Area of application, 1310
    Configuration example, 1311
ET 200M distributed IO device, 1336
ET 200SP, 1269
ET 200SP with ET 200AL mixed mode, 1269, 1313
ET-Connection, 1269
ET-Connection rack, 1269
Ethernet, 596, 598
    Parameters, 6152, 6242, 6332, 6456, 6471, 6505
ETHERNET
    Basic Panel, 6051
Ethernet interface
    Displaying parameters, 7607

Ethernet module
Removal/insertion, 91
Ethernet non IP frames, 701
Ethernet Statistics
Frame length, 954
Frame type, 955
Interface statistics, 953
Packet Errors, 955, 956
EtherNet/IP
Allen-Bradley, 6616
EvenRowBackColor property (VBS), 5326
Event
Acknowledge, 4853
activate, 4846
Alarm buffer overflow, 4853
Assigning organization block (OB) with ATTACH, 3216
Boundary reached, 4853
Canceling assignment to organization block (OB) with DETACH, 3218
Change, 4846
cleared, 4846
Click, 4851
Click when flashing, 4852
Configuring, 4442, 4447, 4450
Deactivate, 4849
double-click, 4849
Dynamize, 4017
Execute, 4847
For function lists, 4234
Free space critically low, 4856, 7053
Incoming, 4851
Input finished, 4850
Inspector window, 61
Loop-In-Alarm, 4852
Low free storage space, 4856, 7053
On exceeding, 4847
On falling below, 4848
Outgoing, 4851
Overflow, 4258, 4855
Press, 4850
Press ESC twice, 4850
Press key, 4854
release, 4852
Release key, 4854
Runtime Stop, 4854
Screen change, 4849
Selection changed, 4847
Switch OFF, 4855
Switch ON, 4855
Tags, 4234
Time expired, 4856
Toggle, 4855
User change, 4848
Value change, 4856
When dialog is closed, 4848
When dialog is opened, 4848
Event name, 1230
Event trigger, 5982, 5985, 5989
Event-driven output
Report, 4501
Event-driven tasks
Configuring:event-driven tasks, 4303
Events
Configuration, 976
Log table, 943
Severity filter, 978
EW, 1482
EX, 1482
Example
Application for alarm classes, 4287, 4288
Changing and displaying operating mode, 4608
Communication, 6578
Configuration ET 200AL, 1311
Controlling a conveyor belt, 3861, 3881
Controlling room temperature, 3868, 3879
Detecting the direction of a conveyor belt, 3863, 3872, 3882
Detecting the fill level of a storage area, 3864, 3874, 3884
For displaying bit memory in the assignment list, 1811
For displaying inputs and outputs in the assignment list, 1811
For entering force values in the force table, 1889
Modify values in the watch table, 1862
System alarm, 4284
User-defined function for temperature conversion, 4604
Example of homepage for user-defined documentation, 367
Example of user-defined documentation, 372
Example:
Discrete alarm, 4283
System event, 4284
Examples
Controlling a conveyor belt, 3871
Exchangeable medium, 1137
Exchanging, 7745
EXCLUSIVE OR, 2474, 2475
Bit logic operation, 2474
Execute, 4847
Executing
User-defined functions in Runtime, 4601

Exit
User program, 2413, 2693, 2918
EXIT, 2908
EXP, 2322, 2599, 2811
Expand ET 200SP, 1269
Expanded mode, 1855
Expanding and collapsing sections of code, 1673
Expanding ET 200SP with ET 200AL modules, 1269
Exponential value, 2322, 2599
Exponentiate, 2332, 2609
Export, 5836
Alarm, 6798
Excel format, 6794
Project texts, 6833
Recipe, 4681, 4881, 6794
Tag, 6805
Text list, 6812
User administration, 4553
Export of labeling strips, 438
ExportDataRecords, 4679, 4879
ExportDataRecordsWithChecksum, 4682, 4881
ExportDelimiter property (VBS), 5326
ExportDirectoryChangeable property (VBS), 5327
ExportDirectoryname property (VBS), 5327
ExportFileExtension property (VBS), 5328
ExportFilename property (VBS), 5328
ExportFilenameChangeable property (VBS), 5329
ExportFormatGuid property (VBS), 5330
ExportFormatName property (VBS), 5330
ExportImportUserAdministration, 4684, 4884
Exporting
Recipe, 4483
Recipe data record, 4431, 4483
Exporting an NTP server, 737
Exporting labeling data as XML, 441
ExportParameters property (VBS), 5331
ExportSelection property (VBS), 5331
ExportShowDialog property (VBS), 5332
Expression
Arithmetic expression, 1660
Basics, 1659
Logical expression, 1665
Relational expressions, 1663
EXPT, 2332, 2609
Extended download to device, 80
Extended status information, 1668
External, 99
External graphic
Edit folder, 3973
Link folder, 3973
Removing the folder link, 3973
Rename folder, 3973
External image file
add to graphics library, 6838
Managing, 3939
Storing in the image browser, 3974
External source file
Basics, 1744
Editing, 1749
Exporting blocks, 1746
inserting, 1748
Linking file types to an editor, 1748
Opening, 1749
Rules for programming, 1746
External sources, 99
External tags
Data exchange, 5998
ExtraSpaceForLabelDisplay property (VBS), 5783

F

f(x) trend view, 4108
Toolbar, 4109
F_TRIG, 2217, 2493, 2753
Faceplate, 4057
Cancel connection to faceplate type, 4079
Controlling properties dynamically, 4080
Create, 4081
Dynamically controlling included objects, 4085
Dynamize, 4079
Example, 4081
Faceplate type, 4057
Instance of the faceplate, 4057
Response to sizing, 4077
Faceplate type, 4057
Cancel connection, 4079
Create, 4062
Creating a tag, 4083
Defining properties, 4082
Deleting a category, 4067
Deleting a connection, 4065
Deleting a property, 4067
Deleting a property connection, 4065
Duplicating, 4076
Dynamic control, 4079
Edit, 4075
Enable, 4075
Event, 4061
Faceplate, 4077
Graphics list, 4062
Properties, 4061
Response to resizing, 4077
Script, 4061
Tag, 4061

Text list, 4061
Updating, 4075
Use, 4077
User text, 4062
Facility, 723
factory settings
Resetting to, 6994
Factory settings
Resetting to, 1386, 1387
FALSE, 2216, 2217, 2492, 2493, 2752, 2753
FAQs, 6017
Fault monitoring
Connection status change, 997
Power supply, 996
Redundancy, 997
Favorites
Adding, 1593, 1635, 1682
Hiding, 1538
Removing, 1594, 1636, 1684
Showing, 1538
Using, 1593, 1635, 1683
FB, 1415
FBD, 1619, 3871, 3872, 3874
Detailed comparison, 1759
FBD element
Copying, 1645
Cutting, 1646
Deleting, 1651
Inserting, 1630, 1631
Inserting an operand, 1652
pasting from the clipboard, 1646
Replacing, 1647
Rules for inserting, 1629
Selecting, 1644
FBD network
Branch, 1654
Deleting a branch, 1656
Inserting a block call, 1595, 1637
Inserting a branch, 1655
Program status display, 1847
Rules for branches, 1655
FC, 1414
F-CM AS-i Safety ST, 94
FDA, 7040
FETCH/WRITE, 1212
Field device, 1109, 1141, 1142
FieldLength property (VBS), 5333, 5334
FieldRead, 2371, 2649
FieldWrite, 2373, 2651
FIFO, 270
FIFOQueue, 270
File Browser, 858
File format
Audit Trail, 7062
Fill
Block, 2352, 2380, 2630, 2659, 2833, 2875
Block uninterruptible, 2353, 2632, 2835
FILL, 2380, 2659, 2875
Fill style, 4173
FILL_BLK, 2352, 2630, 2833
FillColorMode property (VBS), 5334
FillPatternColor property (VBS), 5334
FillStyle property (VBS), 5336
Filter
Defining for the assignment list, 1815
Delete, 1815
Hardware catalog, 557
In the assignment list, 1815
Selecting, 1816
Filter alarms
Alarm view, 4141
Filter property (VBS), 5336
Filtering
the alarm view, 4328
Filters
Assignment list , 1816
FilterTag property (VBS), 5336
FilterText property (VBS), 5337
FIND, 3053
Find and replace
Colors, 6888
Firewall
Creating service groups, 707, 710
Defining ICMP services, 706
Firewall rules, 701
Managing service groups, 708, 710
Firmware, 1235
Firmware card, 1169
Firmware update, 1384
Firmware update memory card, 1170
Firmware versions, 47
FirstConnectedObjectIndex property (VBS), 5337
FirstConnectedObjectName property (VBS), 5337
FitToLargest property (VBS), 5338
FitToSize property (VBS), 5338
Fixed aspect ratio, 4174
Fixed reference temperature, 1333
FixedAspectRatio property (VBS), 5338
Flash test, 1402
Flashing, 3952, 4019
FlashingColorOff property (VBS), 5338
FlashingColorOn property (VBS), 5340
FlashingEnabled property (VBS), 5341
FlashingOnLimitViolation property (VBS), 5342

FlashingRate property (VBS), 5342
FlashTransparentColor property (VBS), 5343
Flip, 4173
  Object, 3939
Flip property (VBS), 5344
Flip-flop
  Reset/set, 2208, 2484
  Set/reset, 2207, 2483
Floating-point number, 1925, 1926
  Check invalidity, 2292, 2568
  Check validity, 2291, 2567
Floating-point numbers, 1927
  Invalid, 1927
FLOOR, 2388, 2667, 2882
Flutter monitoring, 1338
FocusColor property (VBS), 5344
FocusWidth property (VBS), 5345
Folder link
  Editing, 3973
  Removing, 3974
  Renaming, 3974
Following error monitoring, 7394
Font property (VBS), 5346
Font settings
  Migrate, 175
Font size
  Asian languages, 6847
  Device-dependent, 3924
Font size adjustment
  Activating, 3923
FontBold property (VBS), 5347
FontItalic property (VBS), 5348
FontName property (VBS), 5349
FontSize property (VBS), 5349
FontUnderline property (VBS), 5350
Footer
  Report, 4496
FOR, 2902
Force
  Force all, 1904
  Stop forcing, 1906, 1907
Force job on SD card, 45
Force table
  "Monitor once and now" command for tags, 1897
  Basic mode, 1884
  Display, 1886
  Expanded mode, 1884
  Functionality, 1881
  Layout, 1883
  Meaning of the columns, 1883
  Meaning of the icons, 1885
  Monitoring and modifying modes, 1869
  Opening, 1886
  Overview of the display formats, 1890
  Overview of the test options, 1881
  Permitted operands, 1888
  Permitted operands for force values, 1889
  Saving, 1887
  Switching between basic mode and expanded mode, 1884
  Syntax check, 1887
  Test options, 1881
Force value
  Permitted operands, 1889
Forcing
  Audit, 7065
Forcing tags
  Safety precautions, 1883, 1899
ForeColor property (VBS), 5351
ForeColorTransparency property (VBS), 5352
Foreground color
  Dynamization, 4019
Form exponential value, 2811
Format, 4104
FormatPattern property (VBS), 5352
FormatPatternReadOnlySpecial property (VBS), 5785
Formatting
  Alarm text, 4306
FormatType property (VBS), 5692
FRAC, 2330, 2607, 2818
Fragment, 872, 877
Free property (VBS), 5352
Free space critically low, 4856, 7053
Free-form comments
  Deleting, 1602, 1644
  Editing, 1601, 1643
  Inserting, 1600, 1642
  Introduction, 1600, 1642
FreePercent property (VBS), 5353
FreezeProviderConnections property (VBS), 5786
Frequency meter, 1259, 1262
FTP, 695, 696, 1212, 3808
FTP_CMD, 3808
FTPS, 3808
FTPS certificates, 686
Function, 4569
  Assigning to a function key, 4041
  Function list, 4579
Function (FC)
  Creating, 1507
  Definition, 1414
  exporting to an external source file, 1746

Function block (FB)
Creating, 1507
Definition, 1415
exporting to an external source file, 1746
Instance data block, 1415
Function Block Diagram, 1619
Function key, 4036
Assigning a function, 4041
Assigning a graphic, 4045
Global assignment, 4039, 6916, 6975
Global screen, 3902
Local assignment, 3902, 4040, 6916, 6975
protect with password, 4043
Replacing devices, 6852
Use global assignment, 3902
used for screen navigation, 4049
Function list, 4234, 4574, 5980
Asynchronous execution, 4600
Configuring, 4575
Editing, 4577
Execution in Runtime, 4600
Synchronous execution, 4600
Function value, 101
Functional scope
ProSave, 6990
Functionality of the force table, 1881
Functions
Updating tag value, 4202, 6036
FunctionTrendControl object, 5077
FX1 series, 6666
FX2 series, 6666
FX3
PLC, 6651
FX3 series, 6655

G

GADR_LGC, 3370
Gateway, 596
Gauge, 4190
Color of individual ranges, 4192
Display peak value, 4191
GMP-relevant tag, 4190
Maximum value, 4191
Minimum value, 4191
Normal range visible, 4191
Gauge object, 5085
GBIT interface (CPU 1518-4 PN/DP), 1135
GE Fanuc SNP
Migrating data types, 198
GEN_DIAG, 3302
General information on troubleshooting, 7717
General rules, 1430
Generating diagnostics information, 3302
Generation of packed addresses, 1100
GEO_LOG, 3366
GEO2LOG, 3359
Geographic coordinates, 969
GET, 3612
Get_AlarmState, 3253
GET_DIAG, 3304
Get_IM_Data, 3277
GET_NAME, 3283
GetBlockName, 3061
GetBrightness, 4715, 4885
GetColumn, 5837, 5856, 5857, 5916, 5917, 5934
GetColumnCollection, 5838
GetDataRecordFromPLC, 4711, 4885
GetDataRecordName, 4712, 4713, 4887
GetDataRecordTagsFromPLC, 4714, 4889
GetError, 1701, 1702, 1703, 2414, 2694, 2919
GetErrorID, 1701, 1702, 1703, 2417, 2697, 2922
GetGroupNumber, 4715, 4889
GetHitlistColumn, 5840
GetHitlisteColumnCollection, 5839
GetInstanceName, 3058
GetInstancePath, 3060
GETIO, 3072
GETIO_PART, 3074
GetMessageBlock, 5841
GetMessageBlockCollection, 5842
GetMessageColumn, 5844
GetMessageColumnCollection, 5849
GetMessageColumnCollection , 5846
GetOperatorMessage, 5845
GetOperatorMessageCollection, 5847
GetPassword, 4716, 4890
GetPLCMode, 4717
GetRow, 5848
GetRulerBlock, 5851
GetRulerBlockCollection, 5852
GetRulerColumn, 5853
GetRulerColumnCollection, 5854
GetRulerData, 5855
GetSelectionText property (VBS), 5786
GetStationInfo, 3286
GetStatisticAreaColumn, 5859
GetStatisticAreaColumnCollection, 5860
GetStatisticResultColumn, 5861
GetStatisticResultColumnCollection, 5862
GetStatusbarElement, 5863
GetStatusbarElementCollection, 5864
GetSymbolName, 3055
GetSymbolPath, 3057

GetTimeAxis, 5866
GetTimeAxisCollection, 5867
GetTimeColumn, 5869
GetTimeColumnCollection, 5870
GetToolbarButton, 5871
GetToolbarButtonCollection, 5872
GetTrend, 5874
GetTrendCollection, 5875
GetTrendWindow, 5876
GetTrendWindowCollection, 5877
GetUserName, 4710, 4891
GetValue, 7150
GetValueAxis, 5878
GetValueAxisCollection, 5879
GetValueColumn, 5881
GetValueColumnCollection, 5882
GetXAxis, 5883
GetXAxisCollection, 5884
GetYAxis, 5886
GetYAxisCollection, 5887
Gigabit address, 689
Gigabit PROFINET interface, 1135
Global assignment
  Of a function key, 4036, 4039
Global data block, 1416
  Configuring, 6045
  Creating, 6045
  Retentive behavior, 1717
Global firewall rules, 702
  Assigning, 704
Global libraries, 471
  Archiving, 493
  Backward compatibility with older product versions, 485
  Compatibility mode, 485
  Corporate libraries, 471
  Retrieving, 494
  System libraries, 471
  User libraries, 471
Global library, 6770
  Adding types, 520
  Archiving, 493
  Cleaning up, 531
  Closing, 491
  Create, 6779
  Creating, 483
  Creating folders, 497
  Deleting, 492
  Displaying logs, 489, 6782
  Migrating, 188
  Open, 6781
  Opening, 486
  Save, 6780
  Saving, 490
  Showing properties, 489
  Updating the project, 523
  Upgrade, 6878
  Using filter view, 500
  Using the element view, 474
  Using types, 522
Global packet filter rules, 704
Global screen, 3902
  Function key, 3902
  Template, 3903
Global softkey, 4039
Global tag, (See PLC tag)
GMP, 7040
GMP settings, 7067, 7071
GMP-relevant tag, 7067
  Gauge, 4190
GMRP, 1045
Go online, 1400, 7591
  connecting several devices, 7593, 7594
  Multiple TIA Portal instances, 91
Going offline, 1402
Good Manufacturing Process, 7040
GOTO, 2909
GoToEnd, 4685, 4891
GoToHome, 4685, 4892
Gradation property (VBS), 5353
GRAPH
  Detailed comparison, 1769
  Program status display, 1851
GraphDirection property (VBS), 5354
Graphic
  Adapting, 4111
  add to graphics library, 6837
  Adjust, 4113
  Alignment, 3957
  Assigning to a function key, 4045
  Button, 4162
  Distance, 3957
  Graphic view, 4111, 4520
  Inserting, 3939, 4502
  managing, 3972
  Size adjustment, 3956
  Stretching, 4520
  Using from the graphic browser, 3972
  With transparent background, 3972
Graphic browser, 3972, 6838
Graphic I/O field, 4112
  Output graphics list, 4011
  Use in reports, 4521

Graphic view, 4110, 4520
transparent color, 4111
GraphicIOField object, 5088
Graphics, 3974
Editor, 6836
Graphics list
Application, 4001
Bit (0, 1), 4007
Bit number (0 - 31), 4009
Creating, 4002
Graphic I/O field, 4011
Outputting configuration data, 4010
Range (... - ...), 4006
Range (0 - 31), 4003
GraphicView object, 5091
GridBackColor property (VBS), 5693
GridLineColor property (VBS), 5354
GridlineFillColor property (VBS), 5355
GridlineStyle property (VBS), 5355
GridLineWidth property (VBS), 5355
group
Cancel, 3979
Edit, 3978
Ungroup, 3979
Group
Adding objects, 3980
Cancel, 3979
Creating, 3978
Editing, 3978
Removing an object, 3981
Ungroup, 3979
Group name, 705, 708
Group object, 5094
Group properties, 745
GSD files
Configuring devices (PROFIBUS), 1109
Delete, 1108
GSD revisions (PROFIBUS), 1107
Installation, 1107
GSD files (PROFINET), 1140
Changing revision, 1142
Delete, 1142
Installation, 1141
GSDML, (See GSD files (PROFINET))

H

Handle for slide-in screen
Configuring, 3913
Handshaking, 1201
Handwheel, 4114
Tag, 4114
Hardware
Configuring and assigning parameters, 554
Detection, 95
Edit parameters, 575
Edit properties, 575
Hardware acceleration
SmartServer, 7112
Hardware and network editor
Device view, 540, 6009
Network view, 537, 6002
Topology view, 542, 6011
Hardware and software limit switches
Function, 7344
Hardware catalog
Adding device, 566
Browsing, 557
DP slave, 1086
DP/DP coupler, 1086
I slave, 1086
Selecting the hardware component, 564
Task card, 549, 6015
Hardware configuration
Adding device, 565
Adding module, 568
Hardware configuration for Motion Control
S7-1200, 7329
Hardware data types, 1957
Hardware detection, 568
Hardware diagnostics, 1354
Hardware documentation, 7777
Hardware editor
Components, 536, 6000
Function, 535, 6000
Hardware catalog, 549, 6015
Inspector window, 547, 6013
Hardware identifier, 848, 1276, 1321, 1328, 1349, 1494, 3358, 3830
Determining with LOG2MOD, 3361
Hardware interrupt, 1221
How it works, 1182
Hardware interrupt OB
Description, 1221
Parameter assignment, 1230
Hardware requirements, 118, 119
Communication instruction "AS_MAIL", 7715
Communication instruction "PG_DIAL", 7710
Communication instruction "SMS_SEND", 7713
Hardware requirements for AS-AS remote link
Communication instruction "AS_DIAL", 7712
Hardware support package
Installing, 141
Hardware version, 942

Hardware-controlled data flow control, 1200
Harmonizing
Object size, 3944, 4509
HART, 1250
HART variables
Configuring, 1247
Structure, 1248
Header
Report, 4496
HeaderFont property (VBS), 5356
Height property (VBS), 5356
Help
Browsing, 356
Call topic from favorites, 357
Components of the information system, 351
Delete topic from favorites, 357
Find keywords, 355
Full-text search, 356
Help on alarms, 354
Identification of help topics, 352
Open tooltip cascades automatically, 358
Opening, 355
Preparing your own help contents, 361
Printing help topics, 357
Roll-out, 352
Save topics in favorites, 357
Settings for user-defined documentation, 364, 365
Tooltip, 353
Use index, 355
Help indicator, 4115
HelpText property (VBS), 5360
Hidden input, 4106
Hidden parameters, 1611, 1653
HiddenInput property (VBS), 5361
HideAlarm, 5888
HideTagNames property (VBS), 5786
Hiding
Report sections, 4500
Highlighting
Connection, 6025
Highlights, 37, 39
HighLimitColor property (VBS), 5361
High-speed analog modules
Isochronous mode, 1298
High-speed counter
Configuring, 1196
General, 1193
How it works, 1193
Hitlist property (VBS), 5362
HitlistColumnAdd property (VBS), 5362
HitlistColumnCount property (VBS), 5363
HitlistColumnIndex property (VBS), 5363
HitlistColumnName property (VBS), 5364
HitlistColumnRemove property (VBS), 5364
HitlistColumnRepos property (VBS), 5365
HitlistColumnSort property (VBS), 5365
HitlistColumnSortIndex property (VBS), 5366
HitlistColumnVisible property (VBS), 5366
HitlistDefaultSort property (VBS), 5367
HitlistMaxSourceItems property (VBS), 5367
HitlistMaxSourceItemsWarn property (VBS), 5368
HitlistRelTime property (VBS), 5369
HitlistRelTimeFactor property (VBS), 5368
HitlistRelTimeFactorType property (VBS), 5369
HMI backup
Deleting, 6883, 6958, 6983
Display, 6881, 6956, 6981
Rename, 6883, 6958, 6983
HMI connection, 214, 616, 1212, 5998
Configuring, 6085, 6088, 6168, 6171, 6254, 6257, 6343, 6346, 6394, 6397
Creating, 6081, 6163, 6250, 6340, 6391
MPI, 6285, 6289, 6291
MPI parameters, 6294
Password, 6100, 6119, 6185, 6205, 6353
PC, 6085, 6109, 6171, 6197, 6257, 6275, 6291, 6343, 6394, 6413
PROFIBUS, 6105, 6109, 6111, 6194, 6197, 6271, 6275, 6412, 6413
PROFIBUS parameters, 6114, 6200, 6280, 6418
PROFINET, 6080, 6081, 6083, 6085, 6088, 6162, 6163, 6166, 6168, 6171, 6249, 6250, 6252, 6254, 6257, 6339, 6340, 6341, 6343, 6346, 6390, 6391, 6392, 6394, 6397
PROFINET parameters, 6091, 6173, 6260, 6348, 6399
S7 1200, 6163
S7 1500, 6081
S7 300/400, 6250
SIMATIC ET 200 CPU, 6391
SIMATIC PC, 6088, 6111, 6168, 6194, 6254, 6289, 6346, 6397, 6412
SIMATIC S7-1500 Software Controller, 6340
WinCC RT Advanced, 6083, 6109, 6166, 6194, 6197, 6252, 6275, 6289, 6291, 6341, 6392, 6412, 6413
WinCC RT Professional, 6083, 6109, 6111, 6392
WinCC Runtime, 6083, 6166, 6252, 6341, 6392
WinCC RT Advanced, 6111
HMI connections
Devices & Networks, 6023, 6029
HMI device, 132
as OPC client, 7155

as OPC server, 7156
available area pointers, 6077
Changing the device type, 6906, 6954
Configuration, 6879
Data backup, 6988, 6989, 6991, 6993
Image, 6988, 6989
Load, 6944, 6988, 6989
MPI parameters, 6295
Performance features, 7007, 7011, 7014, 7018, 7024, 7027, 7032
PROFIBUS parameters, 6115, 6201, 6281, 6415
PROFINET parameters, 6092, 6175, 6261, 6349, 6400
Reloading the operating system, 6994
Replacing, 6863
Reset to factory settings, 6996
Restoring data, 6991, 6993
Server, 6558
Software, 6988, 6989
System limits, 7007, 7011, 7014, 7018, 7024, 7027
Transferring authorization, 6766
Transferring license key, 6997
Updating the operating system (Windows CE), 6995
WinAC MP, 6752, 6762
HMI device configuration, 6879
HMI device dependency
System function, 4583
HMI device image
Deleting, 6885, 6960, 6985
HMI device licensing
Non-PC-based, 132
HMI device on a PLC
Commissioning, 6618, 6636, 6658, 6667, 6687, 6699, 6719
HMI device replacement, 54
by the migration, 172
HMI device type
Changing, 6906, 6954
HMI device version, 6863, 6879, 6880, (Device version), (Device version)
HMI device wizard, 6766, 6768
HMI HTTP protocol, 6555
Basics, 6555
HMI style, 6791
HMI style sheet, 6792
HMI tag
Access to user-defined VB functions, 4580
HMI user data types
Interconnecting with faceplates, 4065
HMIRuntime, 4974
HMIRuntime object, 4974, 4993
HOME key, 6914, 6973
Homing
Homing modes, 7346
Horizontal radius, 4107
HorizontalAlignment property (VBS), 5370
HorizontalGridLines property (VBS), 5371
HorizontalScrollBarPosition property (VBS), 5371
Hotkey property (VBS), 5372
HourNeedleHeight property (VBS), 5372
HourNeedleWidth property (VBS), 5372
How the frequency meter works, 1259
HSC, 1193
HSP, (See support package)
HTA, 3037
HTML, 847
HTML Browser, 4116
Button, 4117
Connection to the FTP server, 4116
File explorer, 4116
HTML page
Displaying data type DATETIME, 7130
HTMLBrowerZoomIn, 4689
HTMLBrowserBack, 4694
HTMLBrowserForward, 4693, 4694
HTMLBrowserHome, 4693
HTMLBrowserPageDown, 4692
HTMLBrowserPageUp, 4692
HTMLBrowserRefresh, 4688
HTMLBrowserScrollDown, 4687
HTMLBrowserScrollLeft, 4690
HTMLBrowserScrollRight, 4691
HTMLBrowserScrollUp, 4688
HTMLBrowserStop, 4687
HTMLBrowserZoomOut, 4690
HTTP, 706, 6555
Load/save, 972
HTTP client, 7144
Configure HTTP-Connection, 7144
Configuring a tag, 6562
Configuring tags, 7145
Configuring the SIMATIC HMI HTTP protocol, 7144
Importing certificates, 6568, 7146
HTTP connection
Client, 6559
HTTP protocol
Panel, 6054
Timeout, 6561
HTTP server
Configure WinCC-Project, 7142
Configuring a tag, 6559

Setting at the HMI device, 7142
Setting up the WinCC Internet connections, 6565
HTTPS, 793, 857, 6555, 6557
HTTPS server only, 966
HW ID
See Hardware identifier, 848

I

I address, 848
I slave
Hardware catalog, 1086
I&M data: Read with Get_IM_Data, 3277
I/O
Direct access to, 1177
I/O access error, 1175
I/O address, 848, 1177
I/O field, 4104
application in reports, 4518
Data format, 4105
Decimal format, 62
Display format in reports, 4519
Format, 62
Hidden input, 4106
Lock/unlock, 6970
Mode, 4105
I/O input, 1172
I/O output, 1172
I/Os, 1455
IB, 1482
ICMP, 712
Icon
For comparison, 1360
For comparison status, 1407
for connection status, 1407
For hardware diagnostics, 1359
for operating state, 1361
For software diagnostics, 1360
Overlay icon, 1361
Icons
in TeleService, 7666
Icons in the force table, 1885
Icons in the watch table, 1857
IconSpace property (VBS), 5373
ID, 1482
Identify alarm classes
Alarm view, 4141
Identifying
Alarm class, 4137
I-device, 1111, 1124
Proxy device, 1154
IE/AS-i link PN IO, 1111
IE/PB Link, 95, 1138
IE/PB Link PN IO, 95
IEC check, 2091
Setting, 2093
IEC Check
Setting, 1961, 2160
IEC counters, 236
IEC timer, 1930, 1931
IEC timers, 236
IEEE 802.3, 701
IEEE tag, 1249
IF, 2898
IF1B
Basic Panel, 6051
IGMP, 1045
IGMP querier, 1045
IKE settings, 745
Illuminated pushbutton, 4126
Tag, 4127
IM0_Data, 3277
Image, 6879, 6900, 6901
HMI device, 6988, 6989
Image file
Storing in the image browser, 3974, 6838
Image memory, 907
IMC, 2460, 2741, 2959
Import
Alarm, 6799
Analog alarm structure, 6800
Project texts, 6835
Recipe, 6796
Structure of discrete alarms, 6803
Structure of recipe data, 6797
Tag, 6807, 6808
Text list, 6814
User administration, 4554
ImportDataRecords, 4695, 4893
ImportDataRecordsWithChecksum, 4697, 4895
Importing
Recipe, 4483
Recipe data record, 4431, 4483
Importing a recipe
Structure for the import, 6797
Importing analog alarms
Structure for the import, 6800
Importing discrete alarms
Structure for the import, 6803
Importing NTP servers, 737
in the screen
Arrange object, 3947, 4507
IN_RANGE, 2288, 2564
INC, 2310, 2586

Incoming, 4285, 4851
IncreaseFocusedValue, 4678
IncreaseTag, 4678, 4896
Increment, 2310, 2586
Indenting and outdenting
Lines, 1672
Index property (VBS), 5373
Index tag, 4232, 4233
Indirect addressing, 226, 251, 254, 1464, 1465, 1466, 1467, 1469, 1470, 4232
Industrial Ethernet, 6018
Information
ARP table, 943
Ethernet Statistics, 954
Event log, 943
Frame type, 955
LLDP, 958
Packet Errors, 955, 956
Spanning Tree, 946
Versions, 941
Information system
Components of the information system, 351
Help on alarms, 354
Roll-out, 352
Tooltip, 353
Infotext, 4294
Creating, 4302, 4305
display, 6915, 6919, 6975, 6979
Display, 4358
Key, 6915, 6975
INIT_RD, 2420, 2700, 2925
Initialization
IPE file, 7746
Project file, 7752
Initialization (user-defined web pages), 860
Initialize all retain data, 2420, 2700, 2925
Initializing
Controller data, 6816
IPE file, 7747
Project file, 7752
InnerBackColorOff property (VBS), 5374
InnerBackColorOn property (VBS), 5375
Input
Insert, 2476
Inserting, 1606
remove, 1607, 1649
Input (I), 1172
Input area plan
SmartClient, 7089
SmartServer, 7089
SmartService, 7089
Input byte (IB), 1172
Input field
Character mode, 6914, 6973
Standard mode, 6914, 6973
Input finished, 4850
Input on the HMI device
By means of function key, 6916, 6975
Input word (IW), 1172
InputValue property (VBS), 5375
INSERT, 3049
Insert a comment section, 2911
Insert empty line, 7421
Insert input, 1648
Insert separator line, 7421
InsertData(i) property (VBS), 5741
InsertEnable property (VBS), 5694
Inserting
Code template in user-defined functions, 4579
Graphic, 3939, 4502
Library object, 6788
Object, 3939, 4502
Rectangle, 3989
Inspector window, 575
Cross-reference, 60
Diagnostics tab, 1355
Event, 61
Hardware and network editor, 6013
Layout, 324
Reducing automatically, 318
Inspector window
Hardware and network editor, 547
Install update, 141
Installation
Displaying software, 142
Licenses, 112
Log, 136
Modifying products, 143
ProSave, 79
Repairing products, 145
run automatically, 111
Smartdrive, 55
Starting, 137
Support package, 141
System requirements, 113
Target directory, 111
Updates, 141
Updating products, 143
Installing, 128
Add-on, 128
Option, 6999
Powerpack, 136
Installing CA certificate, 7699
Installing license keys, 116

Installing support packages, 141
Installing the communications driver (WinAC MP), 6751
Installing the local modem, 7674
Instance
  Creating an instance, 510
  Displaying associated type, 511
Instance data block
  Creating, 1417
  Definition, 1417
  Modifying the data types of IEC timers and IEC counters, 1681
  Retentive behavior, 1716
Instruction
  Copying, 1696
  Cutting, 1696
  Deleting, 1696
  Inserting, 1696
  Rules, 1676
  Search, 1540
  Specify data type, 1590, 1592, 1632, 1634
  Versions, 1541
Instruction profile
  Activating and deactivating, 1545
  Basics, 1542
  Creating, 1543
  Deleting, 1546
  Editing, 1544
  Opening, 1544
Instructions
  Function list, 4579
Instructions for configuring a remote modem, 7676
INT, 1918, 1972, 2032, 2100, 2129, 2164, 2179
INT_TO_, 2032, 2129, 2179
Integer, 2389, 2668, 2884
  16-bit, 1918, 1919
  32-bit, 1920, 1921
  64-bit, 1922, 1923
  8-bit, 1917, 1918
IntegerDigits property (VBS), 5376
Integrated
  Configuring, 7764
  Connection, 6023, 6026
Integrated connection, 4202, 6036
Integrated project
  Converting an unspecified CPU, 213
  Creating an integrated HMI connection, 214
  Deleting an unspecified connection, 217
  Linking HMI tags, 216
  Migrating, 192, 209
  Post-editing, 212
Intelligent DP slave, (See I-slave)
Inter Project Engineering
  Creating "Device proxy data" in the source project, 7744
  Exchanging controller data across projects, 7741
  Exporting an IPE file by means of the "Device proxy data", 7745
  Procedure for exchanging controller data via a project file, 7742
  Procedure for exchanging controller data via an IPE file, 7742
  Procedure for updating already transferred controller data, 7743
  Requirements, 7741
  Software and hardware requirements, 7741
Inter Project Engineering (IPE)
  Exchanging controller data via IPE file, 7739
  Exchanging controller data via project file, 7739
  Introduction, 7738
Interconnecting, 598
Interconnecting of ports, 1130
Interconnecting ports
  Graphic view, 670, 672
  Table view, 671, 673
Interface, (See block interface)
  Adding, 7605
  Basic Panel, 6051
  Comfort Panel, 6062
  Displaying, 1367
  Mobile Panel, 6071
  Multi Panel, 6066
  Panel, 6055
  Renaming, 673
  WinCC RT Advanced, 6076
  WinCC RT Professional, 6076
  WinCC Runtime Advanced, 6076
  WinCC Runtime Professional, 6076
Interface properties, 1367
Interfaces, 47
Interference frequency suppression, 1331
Internal
  Ethernet port, 6654
Internal Ethernet port
  Open settings, 6654
Internal network nodes
  Configuring, 751
  Diagnostics, 763
Internal reference junction, 1333
Internal tags
  Data exchange, 5998
Internet Key Exchange (IKE), 746
Interpolator OB, 7340, 7341

Inter-Project_Engineering
IPE, 6816
Interrupt event
Delaying with DIS_AIRT, 3245
Disabling with DIS_IRT, 3242
Enabling with EN_AIRT, 3245
Enabling with EN_IRT, 3243
Interrupts
Activating time-of-day interrupt with ACT_TINT, 3227
Canceling time-delay interrupt with CAN_DINT, 3231
Canceling time-of-day interrupt with CAN_TINT, 3226
Querying time-delay interrupt with QRY_DINT, 3232
Querying time-of-day interrupt with QRY_TINT, 3228
Receiving from I/O module with RALRM, 3076
Setting time-of-day interrupt with SET_TINT, 3222
Setting time-of-day interrupt with SET_TINTL, 3224
Starting time-delay interrupt with SRT_DINT, 3230
With packed addresses, 1106
Interval property (VBS), 5376
Intrusion Detection System (IDS), 120, 124
Invalid license
With a time zone change, 117, 134
InverseLinearScaling, 4700, 4897
Invert, 2201, 2428, 2477, 2709
INVERT, 2428, 2709
InvertBit, 4698, 4898
InvertBitInTag, 4699, 4899
InvertLinearScaling, 4700, 4897
IO address, 3358
IO device
Networking, 1127
Update time, 1131
Watchdog time, 1132
IO system, 1127
Creating, 1127
IO2MOD, 3362
IOField object, 5098
IO-Link, 1145
IP access control list, 696
IP address, 46, 596, 598, 1115
Reading out, 6562
IP address parameters, 1116, 1124
IP configuration
Change parameters from the user program, 3700
IP packet filter rules, 712
IP parameters, 1367
IP protocol, 6177
IP rule sets, 702
IP services, 705
IP subnet, 598
IPE, 6816
Creating "Device proxy data" in the source project, 7744
Exchanging controller data across projects, 7741
Exporting controller data of the "Device proxy data" as an IPE file, 7745
File, 7746, 7747
Introduction, 7738
Software and hardware requirements, 7741
IPE data
Updating, 7750, 7755
IPE file, 7746
Initialization, 7746
Initializing, 7747
IPsec settings, 745
IPv4 addresses, 717
IPv6
E-mail, 1353
FETCH/WRITE, 1353
FTP client, 1353
FTP server, 1353
Notation, 1353
SNMP, 1353
Use with the CP 1543-1, 1353
IPv6 addresses, 718
IQ sense, 1339
iQ series, 6655
IRT (Isochronous Realtime Ethernet), 6584
IRT communication, 6590
IS_ARRAY, 2299, 2575, 2800
IS_NULL, 2297, 2573
ISAKMP, 755
I-slave, 1088, 1096
Configuring, 1097
Data access, 1097
Data exchange, 1088
Proxy device, 1152
ISO protocol, 809
Isochronous mode, 1111, 1147, 1152
Oversampling, 1298
Isochronous real-time communication (IRT), 1111
ISO-on-TCP
Characteristics, 639
TSAP, 648
Item methods, 5889
Item object, 4995

ItemBorderStyle property (VBS), 5377
IW, 1482
IX, 1482

J

Jerk limit
Function, 7345
JMP, 2399, 2679
JMP_LIST, 2402, 2682
JMPN, 2400, 2680
Job mailbox, 5999
Transferring data, 6130, 6220, 6310, 6366, 6434, 6486, 6740
JOIN, 3022
Jump, 2399, 2400, 2401, 2402, 2404, 2406, 2679, 2680, 2681, 2682, 2684, 2686
Jump distributor, 2404, 2684
Jump label, 2401, 2681, 2909
Jump list, 2402, 2682
JumpToLimitsAfterMouseClick property (VBS), 5377

K

K parameter, 101
Key
Backspace, 6914, 6973
Cursor, 6913, 6973
Deleting, 6914, 6973
END, 6914, 6974
ENTER, 6914, 6973
ESC, 6914, 6973
HOME, 6914, 6973
Infotext, 6915, 6975
Scrolling back, 6914, 6973
Scrolling up, 6914, 6974
TAB, 6913, 6973
Key assignment
Replacing devices, 6852
Key operation, 6913, 6972
Key switch
Application, 4164
Layout, 4164
Tag, 4165
Keyboard operation, 552
Customizing editors, 344
Editing objects, 345
Editing texts, 346
Navigation in the TIA Portal, 342
Online functions, 348
Project editing, 341
Selecting objects, 345
Tables, 347
Window, 341
Keyboard shortcuts
Basic functions of the TIA Portal, 339
KEY-PLUG, 999, 1001
Formatting, 1000
Keyword highlighting, 1668
Keywords, 1442
K-key
Replacing devices, 6853
Know-how protection
Changing a password, 1808, 4591
Introduction, 1802
opening a user-defined function, 4590
Opening blocks, 1806
Printing block, 1807
remove, 1808, 4591
Setting up, 1805, 4590

L

Label
Recipe view, 4159
LABEL, 2401, 2681
LabelColor property (VBS), 5378
LACP, 1037
LAD, 1575, 3863
Detailed comparison, 1759
LAD element
Copying, 1603
Cutting, 1604
Deleting, 1609
Inserting, 1588, 1589
Inserting an operand, 1609
pasting from the clipboard, 1604
Replacing, 1605
Rules for inserting, 1586
Selecting, 1602
LAD network
Branch, 1612
Closing a branch, 1614
Crossing, 1615
Deleting a branch, 1615
Deleting a crossing, 1617
Insert crossing, 1616
Inserting a block call, 1595, 1637
Inserting a branch, 1614
Program status display, 1847
Prohibited interconnections, 1588
Rearranging a crossing, 1616
Rules for simultaneous branches, 1613

Ladder Logic, 1575
Language
Activate project language, 6826
Asian languages, 6825
Asian operating system, 6825
Disabling the project language, 6826
Editing language, 6827
Language support, 6825
Language-dependent format, 6824
Language-specific graphic, 6836
Log, 4264, 4348, 6845
multilingual project, 6829
Reference language, 6827
Regional format of the date, time, currency, and numbers, 6824
Language abbreviation, 882
Language behavior
On-screen keyboard, 82
Language property (VBS), 5378
Language switching, 882, 6840
In Runtime, 4604
Log, 4264, 4348, 6845
Runtime language, 6843
Languages
Migrating, 185
LargeTicksBold property (VBS), 5379
LargeTicksSize property (VBS), 5379
LastConnectedObjectIndex property (VBS), 5380
LastConnectedObjectName property (VBS), 5380
Layer
Assigning objects to a layer, 4052
Layer 2, 701
Layer 3, 701, 1001
Layer object, 4995
Layers object (list), 4997
Layout
Alarm indicator, 4145, 4361
Alarm line, 4134
Alarm report, 4522
Alarm view, 4132, 4135, 4143, 4356, 4359
Bar, 4097
Button, 4161
Camera view, 4119
Charging condition, 4125
Circle, 4122
Clock, 4183
Date/time field, 4103
Date/time field in reports, 4518
Effective range name, 4184
Effective range name (RFID), 4186
Effective range signal, 4187
Ellipse, 4107
f(x) trend view, 4108
Gauge, 4190
Graphic I/O field, 4112
Graphic I/O field in reports, 4521
Graphic view, 4110, 4520
Handwheel, 4114
HTML Browser, 4116
I/O field, 4105
I/O field in reports, 4519
Illuminated pushbutton, 4126
Key switch, 4164
Line, 4129
Media Player, 4131
Page number in reports, 4525
Polygon, 4150
Polyline, 4151
Recipe report, 4524
Recipe view, 4156, 4158
Rectangle, 4153
Simple recipe view, 4470
Simple user view, 4099
Slider, 4163
Sm@rtClient view, 4168
Status/Force, 4171
Switch, 4160
Symbol library, 4172
Symbolic I/O field, 4174
Symbolic I/O field in reports, 4526
Text field, 4182, 4527
Trend view, 4123
User view, 4101, 4102
WLAN reception, 4189
Zone name, 4192
Zone signal, 4193
Layout of the force table, 1883
LDT, 1934, 1994, 2069
LDT_TO_, 2069
Lead and lag algorithm, 2464, 2745, 2964
LEAD_LAG, 2464, 2745, 2964
Learning functionality, 751
Learning mode, 809
LeaveMarginForBorder property (VBS), 5787
LeaveMarginForMarkers property (VBS), 5788
LED, 3274
Assignment, 6138, 6228, 6317, 6374, 6442, 6495, 6644, 6675, 6706, 6725
Function, 6138, 6228, 6317, 6374, 6442, 6495, 6644, 6675, 6706, 6725
Image, 6138, 6228, 6317, 6374, 6442, 6495, 6644, 6675, 6706, 6725
LED function key
Bit assignment, 6533

LED status
Reading out with LED, 3274
LEFT, 3042
Left property (VBS), 5386
LeftOffset property (VBS), 5387
LEN, 3039
Length
Area pointer, 6042
Area pointers, 6045
LG GLOFA GM
Migrating data types, 199
LGC_GADR, 3372
Library, 45, 6770
Adding a master copy, 499
Adding and using a block, 1510
Basics, 470
Buttons and Switches, 6774
Cleaning up, 531
Comparing library elements, 532
Copying a library object, 6773
Copying types to the clipboard, 526
Creating folders, 497
Cutting library elements, 526
Cutting master copies, 526
Cutting types, 526
Deleting instances, 527
Deleting types and versions, 527
Harmonizing names and path structure, 530
Master copies, 470
Master copy, 6773
Moving library elements, 527
Open, 6781
Opening a released type version, 504
Pasting master copies from the clipboard, 527
Pasting types from the clipboard, 526
PLC data types generated by the system, 100
Released type version, 503
Save, 6780
Storing an object, 6787
Style, 6791
Style sheet, 6792
System diagnostics indicator, 4409
Task card, 472
Type, 6773
Type version in progress, 503
Type version in test, 503
Types, 470
Using filter view, 500
Using master copies, 497, 500
Using the element view, 474
Using types, 501
Versioning of types, 502
Library management
Displaying cross references of an instance, 482
Displaying instances of type versions, 482
Displaying relations between library objects, 483
Filtering, 6778
Filtering by non-released types, 481
Opening, 480, 6777
Overview, 478, 6776
Library object, 6770, 6771
Inserting, 6788
Library view
Exit, 478
Opening, 478
Overview, 475, 6772
License
Defective, 117, 134
Defective license, 117, 134
Managing, 6997
Starting without a valid license, 128
Starting without valid license, 115
License key, 116, 133
Handling license keys, 116
Working with license keys, 133
License Keys, 6996
Transfer to an HMI device, 6997
License Manager Panel Plugin, 132
Licenses, 112
Life of certificates, 744
Light emitting diode
Acknowledge, 6915, 6974
Toggle, 6915, 6974
Limit, 4294
LIMIT, 2317, 2593, 2806
Limit and enable password legitimation, 2408, 2688, 2912
Limit value, 1289, 2317, 2593, 2806
Tag, 4219
Limit values
Tag, 4218
Limit4LowerLimit property (VBS), 5387
Limit4LowerLimitColor property (VBS), 5388
Limit4LowerLimitEnabled property (VBS), 5388
Limit4LowerLimitRelative property (VBS), 5389
Limit4UpperLimit property (VBS), 5390
Limit4UpperLimitColor property (VBS), 5390
Limit4UpperLimitEnabled property (VBS), 5391
Limit4UpperLimitRelative property (VBS), 5391
Limit5LowerLimit property (VBS), 5392
Limit5LowerLimitColor property (VBS), 5392
Limit5LowerLimitEnabled property (VBS), 5393
Limit5LowerLimitRelative property (VBS), 5394
Limit5UpperLimit property (VBS), 5394

Limit5UpperLimitColor property (VBS), 5395
Limit5UpperLimitEnabled property (VBS), 5395
Limit5UpperLimitRelative property (VBS), 5396
Line, 4129
  Design, 3952
  Line end, 3952, 4130
  Line start, 4130
Line break, 1265
Line end
  Line, 4130
  Polyline, 4151
Line object, 5102
Line start
  Line, 4130
  Polyline, 4151
Linear programming, 1412
LinearScaling, 4717, 4900
LineColor property (VBS), 5396
LineEndShapeStyle property (VBS), 5397
LineWidth property (VBS), 5398
Link folder
  External graphic, 3973
Linked objects
  Copying, 6869, 6870
Linking, 6821
Linking HMI tags, 216
LINT, 1922, 1979, 2046
LINT_TO_, 2046
Listbox, 5104
Lists, 5016
Lists in VBS
  Alarms object (list), 4984
Lists:DataSet, 4990
Lists:Layers, 4997
Lists:Tags Object (List), 5015
LLDP, 695, 696, 697, 958, 1137
LLDP (Link Layer Discovery Protocol), 1137, 6103, 6188, 6269, 6406
LN, 2321, 2597, 2810
load
  Recipe data record in Runtime, 4481
Load
  Extended download to device, 80
  HMI device, 6944, 6988, 6989
  Project, 6943
  SIMATIC PC station, 80
Load memory, 1171, 1830
  Displaying, 1834
Load window layout
  Loading additional window layouts, 336
  Loading via quick access, 336
LoadDataImmediately property (VBS), 5399
LoadDataRecord, 4709, 4902
Loading
  Configurations with PROFIBUS, 1152
  Configurations with PROFINET, 1153
  Configurations with Web server, 1151
  Device configurations, 1149
  Distributed I/O, 1150
  Downloading project data to the device, 396
  from a device, 91, 99
  General information, 394
  Going online, 1157
  GSDML, 1154
  HMI devices, 1156
  I-device, 1154
  IE/PB-Link, 1155
  I-slave, 1152
  Isochronous applications, 1147, 1152
  Module comments, 93
  Proxy device, 1152, 1154
  Shared Devices, 1156
  Subnets, 1151
  to PG/PC, 1148
  to the device, 91, 1147
  USB, 6947
Loading projects
  with connected HMI device, 6900, 6936
  Without connected HMI device, 6937
Loading with S7 routing, 6901
Local assignment
  Of a function key, 4036, 4040
Local data bit (L), 1172
Local data byte (LB), 1172
Local data double word (LD), 1172
Local data word (LW), 1172
Local error handling, 1701
  Error information, 1701
  Instructions, 1701
  Priorities, 1702
Local logging, 757
Local script, 4569
Local tag, 1551
  Access to user-defined VB functions, 4580
Local time, 1185
  Calculating with SET_TIMEZONE, 2989
  Reading out with RD_LOC_T, 2985
  Writing with WR_LOC_T, 2986
LocalCursor property (VBS), 5400
Localizable property (VBS), 5749
Lock
  Operator control, 6969, 6970
LockAlarm, 5891
LockSquaredExtent property (VBS), 5400

Log
Alarm log, 4418
Automatic entries, 4419
Checksum, 4262, 4346, 4423
Data log, 4418
Language switching, 4264, 4348, 6845
Level-dependent management, 4255, 4350
Runtime language, 4264, 4348, 6845
Log behavior
Control with system function, 4258
Level-dependent management, 4255, 4350
Managing upon system start, 4254, 4349
Log contents
Displaying, 4251, 4339
Log data
Migrating, 188
Log database
Direct access with ODBC, 4264, 4277, 4343
Log entries, 4419
Log file
Modem alarms, 7722
Log language, 4264, 4348, 6845
Log table
Event log, 943
Log tag value change
Audit Trail, 7066
Log type, 4339
Log with level-dependent execution of system functions, 4339
Log with level-dependent system alarm, 4339
LOG_GEO, 3368
LOG2GEO, 3361
LOG2MOD, 3361
Logarithm, 2321, 2597, 2810
Logging, 757
Change to a recipe data record in Audit Trail, 7069
Circular log, 4251
CP x43-1 Adv., 826
Log type, 4251, 4339
SCALANCE S, 793, 794
Segmented circular log, 4251
System functions in Audit Trail, 7075
Tag value, 4249
Tag value change in Audit Trail, 7066
Tags, 4251, 4256
Tolerance band, 4256
User actions in audit trail, 7072
Within/outside the limit values, 4256
Logging concept
Audit, 7043
Logging cycle, 4256
Tag, 4247
Logging method
Circular log, 4251
Level-dependent, 4251
Segmented circular log, 4251
Logging object, 4998
Logging recipe data changes
Audit Trail, 7069
Logging system functions
Audit Trail, 7075
Logging tag
Configuring, 4266
Logging user actions
Audit Trail, 7072
Logic analyzer function, 7620
Logic operation
AND, 2424, 2704
EXCLUSIVE OR, 2427, 2707
OR, 2425, 2706
Logic path
Deleting, 1658
Inserting, 1658
Use, 1657
Logical address
Determining a module with GADR_LGC, 3370
Determining a module with RD_LGADR, 3369
Determining associated slot with LGC_GADR, 3372
Determining associated slot with LOG_GEO, 3368
Logical error, 4593
Logoff, 4656, 4903
Configuring on the effective range, 5971
Configuring to effective range (RFID), 5974
Effective range, 4185
Effective range (RFID), 4187
Logoff time
Changing, 4541
Changing in runtime, 4552
Logon, 4662, 4903
Configuring on the effective range, 5971
Configuring to effective range (RFID), 5974
Effective range, 4184
Effective range (RFID), 4186
Logon failed, 4556
Reporting, 4560
User, 4555
Logon dialog
Configuring access protection, 4559
Logon for web server, 855
Logon web server, 855

LogOperation property (VBS), 5401
Logs
Deleting logs, 376
Displaying logs, 376
LogTag, 4664
Long word, 1915
LongTermArchiveConsistency property (VBS), 5401
LookupText, 4798, 4904
Loop, 2898, 2902, 2905, 2906, 2907, 2908
Loop-in alarm
trigger, 4361
LoopInAlarm, 5891
Loop-in-alarm
Configured, 4311
Loop-In-Alarm, 4852
Low free storage space, 4856, 7053
LowerLimit property (VBS), 5402
LowerLimitColor(i) property (VBS), 5698
LowerLimitValue(i) property (VBS), 5699
LowLimitColor property (VBS), 5402
LREAL, 290, 1926, 1985, 2055, 2105, 2144
LREAL_TO_, 2055, 2144
LTIME, 1931, 1990, 2063
LTIME_TO_, 2063
LTOD, 1932, 2002, 2079
LTOD_TO_, 2079
LWORD, 1915, 1967, 2022
LWORD_TO_, 2022

M

MAC packet filter rules, 715
MAC rule sets, 702
MAC services, 708
Machine tailoring, (See Reconfiguring IO systems)
MachineName property (VBS), 5403
Main, 1213
Main entry, 7792
Main mode, 746
managing
Graphic, 3972
Managing
License, 6997
User group, 4542
Users, 4541
Users in runtime, 4551
Manual fragment, 877
Manuals, 6017
MarginToBorder property (VBS), 5403
Master copy
Library, 6773
Math functions
CALCULATE, 1599, 1641
Mathematical functions
CALCULATE, 3867, 3877
MAX, 2315, 2591, 2804
MAX_LEN, 3021
Maximum, 2315, 2591, 2804
Maximum cycle time, 1168, 1189, 1190, 2413, 2693
Maximum length
Tag, 82
Maximum load memory available
Displaying, 1834
MaximumValue property (VBS), 5404
MB, 1482
MB_CLIENT, 3770, 3788
MB_COMM_LOAD, 3753
MB_MASTER, 3757
MB_SERVER, 3778, 3800
MB_SLAVE, 3765
MC_ChangeDynamic
Instruction, 3404
Parameter, 3404
MC_CommandTable
Instruction, 3402
Parameter, 3403
MC_Halt
Function chart, 3387
Instruction, 3385
Parameter, 3419
Parameters, 3386
MC_Home
Instruction, 3381
Parameter, 3382
MC_MoveAbsolute
Function chart, 3390
Instruction, 3388
Parameter, 3389
MC_MoveJog
Function chart, 3401
Instruction, 3399
Parameter, 3400
MC_MoveRelative
Function chart, 3394
Instruction, 3391
Parameter, 3392
MC_MoveVelocity
Function chart, 3398
Instruction, 3395
MC_Power
Function chart, 3378
Instruction, 3374
Parameters, 3374

MC_ReadParam
Instruction, 3406
Parameter, 3407
MC_Reset, 3379
MC_WriteParam
Instruction, 3408
Parameter, 3409
MCAT, 2456, 2738, 2955
MC-Interpolator OB, 7340, 7341
MC-Servo OB, 7340, 7341
MD, 1482
MD5, 741, 747
MDM, 360
Meaning of the columns in the force table, 1883
Measure program runtime, 2422, 2702, 2927
Measuring range, 1303
Measuring window (GATE), 1262
Media converter, 1130
Media Player, 4130
Media redundancy (MRP), 1111, 6591, 6593, 6594, 6597
MediaControl, 5107
Memory area
Load memory, 1171
Retentive memory areas, 1173
Work memory, 1172
Memory bit (M), 1172
Memory byte (MB), 1172
Memory card, 45, 1169, (See Memory Card)
Accessing, 468
Adding a card reader, 468
Formatting, 1388
Introduction, 467
Removal/insertion, 91
Showing properties, 469
Memory double word (MD), 1172
Memory medium
Audit Trail, 7064
Memory requirements
Recipe, 7038, 7039
Memory reserve, 100
Memory reset, 1169
Making, 1383
Memory size
Alarm buffer, 4337
Memory word (MW), 1172
Menu command
Simple recipe view, 4471
MenuToolBarConfig property (VBS), 5404
Message
Sending, 1204
Specifying the end, 1205
Specifying the start, 1204
MessageBlockAlignment property (VBS), 5405
MessageBlockAutoPrecisions property (VBS), 5405
MessageBlockCaption property (VBS), 5406
MessageBlockCount property (VBS), 5406, 5409
MessageBlockDateFormat property (VBS), 5407
MessageBlockExponentialFormat property (VBS), 5407
MessageBlockFlashOn property (VBS), 5408
MessageBlockHideText property (VBS), 5408
MessageBlockHideTitleText property (VBS), 5409
MessageBlockID property (VBS), 5409
MessageBlockLeadingZeros property (VBS), 5410
MessageBlockLength property (VBS), 5410
MessageBlockName property (VBS), 5411
MessageBlockPrecisions property (VBS), 5411
MessageBlockSelected property (VBS), 5412
MessageBlockShowDate property (VBS), 5412
MessageBlockShowIcon property (VBS), 5413
MessageBlockShowTitleIcon property (VBS), 5413
MessageBlockTextId property (VBS), 5414
MessageBlockTimeFormat property (VBS), 5414
MessageBlockType property (VBS), 5415
MessageColumnAdd property (VBS), 5415
MessageColumnCount property (VBS), 5416
MessageColumnIndex property (VBS), 5416
MessageColumnName property (VBS), 5417
MessageColumnRemove property (VBS), 5417
MessageColumnRepos property (VBS), 5418
MessageColumnSort property (VBS), 5418
MessageColumnSortIndex property (VBS), 5419
MessageColumnVisible property (VBS), 5419
MessageListType property (VBS), 5420
Methods, 5826, 5830, 5831, 5832, 5835, 5836, 5837, 5838, 5839, 5840, 5841, 5842, 5844, 5845, 5846, 5847, 5848, 5849, 5851, 5852, 5853, 5854, 5855, 5856, 5857, 5859, 5860, 5861, 5862, 5863, 5864, 5866, 5867, 5869, 5870, 5871, 5872, 5874, 5875, 5876, 5877, 5878, 5879, 5881, 5882, 5883, 5884, 5886, 5887, 5888, 5891, 5892, 5893, 5894, 5895, 5896, 5897, 5898, 5899, 5900, 5901, 5902, 5903, 5916, 5917, 5918, 5919, 5920, 5921, 5922, 5923, 5924, 5925, 5926, 5927, 5928, 5929, 5930, 5931, 5933, 5934, 5938, 5939, 5940, 5941
Methods (VBS), 5823, 5832
Activate, 5823
CreateTagSet, 5832
Item, 5889
Stop, 5932
Trace, 5932

Methods in VBS
Create, 5831
DeactivateDynamic, 5833
Methods:Add, 5828
Methods:Read, 5904
Methods:Refresh, 5907
Methods:Remove, 5908
Methods:RemoveAll, 5912
Methods:Restore, 5913
Methods:Write, 5935
MIB, 695
Micro, 6685
MicroLogix, 6610, 6634
Microsoft Script Debugger
Registering, 4599
Registering as default script debugger, 4599
Starting automatically, 4599
MID, 3045
Migrated project
Compiling, 166
Migrating projects
Procedure, 159
Requirements, 158
Migration, 44
Adaptations beforehand, 169
Alarm groups, 181
Allen-Bradley DF1 data types, 197
Allen-Bradley DH485 data types, 197
Allen-Bradley Ethernet IP data types, 198
Area pointer, 179
Basics, 162
Data types Modicon Modbus, 201
Data types Modicon Modbus TCP/IP, 201
Data types Omron Hostlink/Multilink, 202
Data types OPC, 202
Data types SIMATIC 500/505 DP, 203
Data types SIMATIC 500/505 serial, 203
Data types SIMATIC HMI HTTP Protocol, 204
Data types SIMATIC S5 AS511, 204
Data types SIMATIC S5 DP, 205
Data types SIMATIC S7 200, 205
Data types SIMATIC S7 300/400, 206
Data types Telemecanique Uni-Telway, 209
Displaying a log file, 160
Displaying history, 160
Font settings, 175
GE Fanuc SNP data types, 198
Global library, 188
HMI device replacement, 172
Including the hardware configuration, 154
Integrated project, 209
Introduction, 161
Introduction to migration, 152
LG GLOFA GM data types, 199
Migrating an integrated project, 192, 209
Migrating projects, 165, 194
Mitsubishi FX data types, 199
Mitsubishi Protocol 4 data types, 200
of external tags, 196
of languages, 185
of log data, 188
of project text, 185
Project library, 187
Recipe data, 188
Reconfigure connection, 178
Runtime data, 188
Script, 184
Supported HMI devices, 167
Supported products, 153
Text, 185
The migration process, 153
User administration, 188
VB-script, 184
WinCC V7.0 SP3, 58
Migration of IP access protection lists when activating Security, 96
Migration of projects with Ethernet CPs and Security, 96
Migration tool, 163
Creating a migration file, 158
Distribution and sources, 148
Including the hardware configuration, 156
Removing, 149
System requirements, 148
Using the migration tool, 155
Migration: Check migration readiness of the hardware, 154
MIN, 2314, 2590, 2802
Mini USB, 6947
Minimizing projects, 386
Minimum, 2314, 2590, 2802
Minimum cycle time, 1168, 1189
MinimumValue property (VBS), 5420
MinuteNeedleHeight property (VBS), 5421
MinuteNeedleWidth property (VBS), 5421
Mirroring, 1018, 1020
Missing supply voltage, 1330
Mitsubishi, 6599
Address, 6656
Analog alarm, 6672
Basic Panel, 6049
Communication drivers, 6645
Connection interruption, 75
FX, 6645

Panel, 6054
TCP/IP, 6645
Mitsubishi FX
Address areas, 6666
Configuring a connection, 6659
Connection, 6659, 6663
Connection parameters, 6661
CPU type, 6666
Data type, 6665
Migrating data types, 199
Mitsubishi MC TCP/IP, 6650, 6664
Mitsubishi MC TCP/IP
Configuring a connection, 6646
Connection, 6646, 6649
Connection parameters, 6647
CPU type, 6655
Data type, 6654
Mitsubishi Protocol 4
Migrating data types, 200
Mixed mode, 748
Mobile Panel
Area pointer, 6072
Display and operating element, 4094
Interface, 6071
OPC, 7154
Mobile Panels
Available system functions, 4643
Mobile Wireless, 5960, 5962
Work area, 5960
Zone ID / connection point ID, 5976
Mobile Wireless (RFID)
Work area, 5962
Mobile Wireless V2, 5962
MOD, 1660, 2308, 2584
Modbus communication
as a slave with MB_SLAVE, 3765
as Modbus master with MB_MASTER, 3757
Configuring a port with MB_COMM_LOAD, 3753
Mode, 4161
Graphic I/O field, 4113
I/O field, 4105
Mode property (VBS), 5422
Modem
Local, 7674
Problem, 7717
Remote, 7675
Modem alarms
Log file, 7722
Modem connection cannot be established, 7721
Modem connection is closed, 7722
Modem connection is interrupted, 7721
Causes and solutions, 7721
Modems without plug-and-play, 7675
Modicon
Approved communication with Modbus RTU, 6693
Connection, 6692
Connection cable, 6693
Restrictions with Modbus RTU, 6693
Modicon M340, 6685
Modicon Modbus, 6599
Basic Panel, 6049
Communication drivers, 6675
Migrating data types, 201
Panel, 6054
RTU, 6675
TCP, 6675
Modicon Modbus connection
Data types, 6704
Modicon Modbus RTU, 6698
Configuring a connection, 6688
Connection, 6688
Connection parameters, 6690
Data type, 6697
Modicon Modbus TCP, 6685
Configuring a connection, 6676
Connection, 6676, 6681
Connection parameters, 6678
Data type, 6684
Modicon MODBUS TCP/IP
Change word order, 6681
Modicon Modbus TCP/IP
Migrating data types, 201
Modify
High-speed counters, 3439, 3441
Modify value
Permitted operands, 1862
Modifying mode, 1869
Modifying recipe structures, 4485
Module
Addressing, 848
Asynchronously reading data record with RD_DPARA, 3210
Communication Status, 762
Copying, 572
Deleting, 571
Determine diagnostics status, 1375
Determining logical address of the slot with LOG_GEO, 3368
Determining logical address with GADR_LGC, 3370
Determining logical addresses with RD_LGADR, 3369

Determining slot of a logical address with LGC_GADR, 3372
Determining start address with GEO_LOG, 3366
Inserting, 91, 569, 570
Moving, 573
Reading data record with RD_DPAR, 3208
Removing, 91
Replacing, 574
Select, 560
Time of day on a module, 1383
Module addressing, 1176
Module arrangement, 557
Module comments, 93
Module replacement, 213
Module rights, 695
Module titles, 559
Module version
Updating, 578
ModuleStates, 3297
Momentum, 6685, 6698
Monitor, 1134, 6101, 6186, 6267, 6404
Monitoring
"Monitor all" command, 1872, 1896
"Monitor now" command, 1897
"Monitor once and now" command, 1873
Monitoring and modifying tags in the DB editor, 1725, 1726
Monitoring and modifying modes, 1869
Monitoring mode, 1869, 7101
Monitoring of the partner port..., 1130
Monitoring window, 1338
Mono-master system, 1087
Motion Control CPU S7-1200
Guidelines, 7347
Move
Block, 2344, 2375, 2621, 2653, 2825, 2870
Block uninterruptible, 2350, 2378, 2627, 2656, 2831, 2873
Character string in array with Strg_TO_Chars, 3015
Screen, 3897
Value, 2333, 2610
MOVE, 234, 2333, 2610
Move block, 2346, 2623, 2827
Move view
Keyboard operation, 553
Overview navigation, 538, 541, 543
MOVE_BLK, 2344, 2621, 2825
MOVE_BLK_VARIANT, 267, 2346, 2623, 2827
Moveable property (VBS), 5423
MoveAxis, 5892
MoveToFirst, 5892
MoveToFirstLine, 5893
MoveToFirstPage, 5893
MoveToLast, 5894
MoveToLastLine, 5894
MoveToLastPage, 5895
MoveToNext, 5895
MoveToNextPage, 5896
MoveToPrevious, 5897
MoveToPreviousLine, 5897
MoveToPreviousPage, 5898
MoveVelocity
Parameter, 3396
Moving
Hardware components, 573
Moving columns
User view, 4103
Moving in a group
Pop-up screen, 3909
Moving the view
Overview navigation, 6003, 6010, 6012
MPI, 6018
Addressing, 6298
Connection, 6328
HMI connection, 6285, 6289, 6291
Network, 6020
Network architecture, 6020
Parameters, 6294, 6295, 6297
S7 200, 6334, 6473
S7 300/400, 6285, 6287, 6289, 6291
WinCC RT Advanced, 6287
MPI address
S7 300, 6298
S7 400, 6298
MPI communication
S7 300/400, 6285, 6287
MPI parameters
S7 300/400, 6294, 6295, 6297
MRP (Media Redundancy Protocol), 6591, 6593
MRP domain, 1370, 6594, 6597
MRP domain properties, 1370
MsgFilterSQL property (VBS), 5423
MSK_FLT, 3239
MSTP, 1032
MUL, 2306, 2581
Multi Panel
Area pointer, 6067
Display and operating element, 4092
Interface, 6066
OPC, 7154
Multi Panels
Available system functions, 4630
Multicast, 725, 1043

Multi-instance
Declaring, 1563
Definition, 1427
Multi-key operation, 86
MultiLineEdit, 5109
Multiple instance, 1458
Correcting the call type, 1598, 1640
Multiple selection, 560, 3966, 4509
Multiple Spanning Tree, 1032
Multiplex, 2433, 2714, 2932
Multiplex/synchronous mode, 1344
Multiplexing, 4232
Address multiplexing, 4225
Address multiplexing Allen-Bradley Ethernet IP, 6614
with absolute addresses, 4225
with symbolic addresses, 4226
Multiply, 2306, 2581
Multi-point connection
Allen-Bradley DF1, 6627, 6628
Multi-touch devices
Gestures, 6967
Scrolling, 6966
Two-hand operation, 6969
Zoom in and out, 6967
Zooming, 6967
MUX, 101, 2433, 2714, 2932
MW, 1482
MX, 1482
My Documentation Manager, 360

N

N, 2210, 2213, 2486, 2489
N_TRIG, 2215, 2491
Name
Array element, 66
Name property (VBS), 5424
Names of alarm classes
Change through migration, 182
Naming conventions
Alarm log, 4422
Data log, 4422
NAT/NAPT
Routing, 724
NColumnTextDate property (VBS), 5300
NE_ElemType, 2296, 2572
NE_Type, 2294, 2569
NeedleBorderColor property (VBS), 5426
NeedleColor property (VBS), 5427
NeedleFillStyle property (VBS), 5428
NEG, 2309, 2585
Negate, 2309, 2585
Negotiation, 993
Nesting depth, 1425
Network, 98, 5994
Closing, 1582, 1625
Copying, 1581, 1624
Deleting, 1582, 1625
Entering a comment, 1584, 1627
Ethernet, 6019
Insert title, 1583, 1626
Inserting, 1579, 1581, 1623, 1624
MPI, 6020
Navigating blocks, 1585, 1628
Opening, 1582, 1625
PPI, 6021
PROFIBUS, 6019
PROFINET, 6019
Selecting, 1580, 1623
Using, 1579, 1622
Network access
Opening the properties, 7605
Network architecture
PPI, 6021
Network comments
Hiding, 1538
Showing, 1538
Network connection
Creating, 7766
Network drive, 44
Network editor
Components, 536, 6000
Function, 535, 6000
Hardware catalog, 549, 6015
Inspector window, 6013
Inspector window , 547
Network ID, 802
Network overview
Basic functions, 585
Basic functions for editing the network overview table, 585
Network Syslog, 757
Network view, 6022
Adding device, 566
Hardware and network editor, 537, 6002
Networking
Communication partners, 6022
Devices, 6022
Networking devices
Basics of configuring networks, 581
Editing interface parameters, 587
Editing network parameters, 587

Networking several interfaces at the same time, 583
Networking with an existing subnet, 584
Networking without an existing subnet, 582
Networks within a project, 581
Requirements, 586
Types of communication, 581
Networking in the device view
Networking options, 585
Procedure, 586
Networking mode, 583
New, 37, 39
New data types, 225
New features, 37, 39
NextColumn, 5898
NextTrend, 5899
No supply voltage, 1289
Nodes with an unknown IP address, 749
Non-integrated
Connection, 6026
Non-licensed mode
Engineering System, 128
ES, 115
HMI devices, 132
Runtime, 130
non-typed, 295
NORM_X, 2393, 2672, 2888
Normal range visible, 4191
Normalize, 2393, 2672, 2888
Normally closed contact, 2200
Normally open contact, 2199
NormalRangeColor property (VBS), 5428
NormalRangeVisible property (VBS), 5429
NOT, 1665, 2201
Not equal, 2298, 2574
NOT_NULL, 2298, 2574
NOT_OK, 2292, 2568
NotifyUserAction, 4676, 4905, 7074
NTP, 1043
NTP (secure), 738
NTP server, 738
NULL, 1946
Number of lines
User view, 4102
NumberOfValues(i) property (VBS), 5788
Numerical key assignment, 6915, 6974

O

O, 7588
OB
Events and OBs, 1177
Overview, 1177
OB 1, 1213
OB 80, 1223
OB 82, 1224
OB for manufacturer-specific interrupt, 1218
OB for profile-specific interrupt, 1218
OB 83, 1225
object
Group, 3978
Paste, 3964, 4504
Rotate, 3949, 4513
Object
Arrange, 3939, 3947, 4507
Assigning to a layer, 4052
Availability for Basic Panels, 4088
Availability for Comfort Panels, 4090
Availability for Mobile Panels, 4094
Availability for Multi Panels, 4092
Availability for Panels, 4089
Availability for WinCC Runtime Advanced, 4096
Creating a new OLE object, 3939
Creating an OLE object from a file, 3939
Deleting, 3941, 4503
Design, 3952
Designing the background color, 3954
Designing the fill pattern, 3955
Dynamization, 4012
Dynamization of the appearance, 4019
Dynamize direct movement, 4022
editing within a group, 3982
Evenly distributing, 3946, 4512
Flashing, 3952
Flip, 3939
Flush alignment, 3946, 4512
Grouping, 3978
Inserting, 3939, 4502
Inserting of the same type, 3939
Inserting several times, 3964, 4504
Multiplying, 3964, 4504
outside the area, 3948
Referencing, 4582
Repositioning and resizing multiple objects, 3939, 3968, 4511
Repositioning objects, 3942, 4505
Resize, 3944, 4508
Rotate, 3939, 3949, 4513
Select multiple objects, 3966, 4509
Selecting multiple objects, 3939
Storing in a library, 6787

Tab sequence, 3939
Transparency, 3955
Object group
Animation, 4028, 4029
Editing an object within a group, 3982
Removing an object, 3981
Object list
Tag, 4208
Object names
Renaming upon migration, 162
Uniqueness, 162
Object properties
Dynamization, 4012
Object property (VBS), 5429
Object size
Harmonizing, 3944, 4509
ObjectName property (VBS), 5704
Objects
Alarm report, 4521
Page number, 4525
Pop-up screen, 3908
Recipe report, 4524
Slide-in screen, 3911
Objects (VBS)
ActiveScreenItem, 5227, 5228
AlarmControl object, 5018
AlarmView, 5029
ApplicationWindow, 5157
Bar, 5033
Battery, 5038
BrowserView, 5096
Button, 5040
ChannelDiagnose, 5044
CheckBox, 5046
Circle, 5049
CircleSegment, 5051
CircularArc, 5054
Clock, 5056
Connector, 5058
DateTimeField, 5061
DiskSpaceView, 5063
Ellipse object, 5065
EllipseSegment, 5068
EllipticalArc, 5070
FunctionTrendControl, 5077
Gauge, 5085
GraphicIOField, 5088
GraphicView, 5091
Group object, 5094
HMIRuntime, 4974
IOField object, 5098
Line, 5102
Listbox, 5104
MultiLineEdit, 5109
OLEView object, 5112
OnlineTrend Control, 5122
OptionGroup, 5131
Polygon, 5134
Polyline, 5137
ProjectName object, 5139
ProtectedAreaName, 5141
RangeLabelView, 5142
RangeQualityView, 5144
Rectangle, 5151
RoundButton, 5152
Screen, 4976, 5000
Screen object (list), 4975
ScreenItem, 4978, 5003
ScreenItems (list), 4979, 5005
ScreenWindow, 5155
Slider, 5161
SmartClientView object, 5162
SmartTag, 4982, 5009
SmartTags (list), 4981, 5011
StatusForce, 5164
Switch, 5166
SymbolicIOField, 5170
SymbolLibrary, 5175
SystemDiagnoseView, 5177
SystemDiagnoseWindow, 5181
TableView object, 5114
TextField, 5184
TrendRulerControl, 5187
TrendView, 5195
TubeArcObject, 5198
TubeDoubleTeeObject, 5200
TubePolyline, 5202
TubeTeeObject, 5205
UserArchiveControl object, 5207
UserView, 5214
WindowSlider, 5217
WLanQualityView, 5219
ZoneLabelView, 5220
ZoneQualityView, 5221
Objects in VBS
Alarm object, 4983
Objects:AlarmLogs, 4986
Objects:DataItem, 4987
Objects:DataLogs, 4989
Objects:HMIRuntime, 4993
Objects:Item, 4995
Objects:Layer, 4995
Objects:Logging, 4998
Objects:Tag, 5012

Obtaining user rights
Logging on to the operating system with administrator privileges, 350
With Windows user account control, 350
OCXState property (VBS), 5432
OF, 2900
Off delay, 2223, 2231, 2249, 2258, 2499, 2525, 2534, 2761, 2781
Offline, 679
OK, 2291, 2567
OLE object
create from a file, 3939
Recreate, 3939
Save in Graphics, 3974
Storing in the image browser, 6838
OLEView object, 5112
Omron, 6599
Analog alarm, 6723
Basic Panel, 6049
Communication drivers, 6707
Data type, 6722
Host Link, 6707
Panel, 6054
Trend, 6722
Omron Host Link, 6717
Address, 6718
Configuring a connection, 6708
Connection, 6708, 6712
Connection cable, 6712
Connection parameters, 6710
Data type, 6716
Omron Hostlink/Multilink
Migrating data types, 202
On delay, 2221, 2229, 2244, 2255, 2496, 2520, 2530, 2758, 2776
Retentive, 2247, 2256, 2522, 2532, 2778
On demand, 4222
On exceeding, 4847
On falling below, 4848
Ones complement, 2428, 2709
OneToOneView, 5899
Online, 47, 679
Change connection path, 7591
Display of the online mode, 7592
Establish connection, 7591
Go offline, 7593
Hardware detection, 568
Recipe tag, 4433
Show accessible devices, 7585
Online access, 7602
Online and Diagnostics view, 1354
Online connection, 7588
Multiple TIA Portal instances, 91
Specifying a default connection path, 7590
Online device, 1355
Online diagnostics, 759
Online displays
Orange color, 1363
Online mode, 7587
Display of the online mode, 7589
Go online, 7591, 7593
Standby or hibernation of the programming device / PC, 7588
Online operation, 48, 91
Online property (VBS), 5432
Online Tools, 1354, 1364
OnlineTrendControl, 5122
OP 177B mono
Symbol library, 4173
OP 73
Baud rate on PROFIBUS, 6151, 6243, 6454
OP 77A
Baud rate on PROFIBUS, 6151, 6243, 6454
OP 73
Baud rate on PROFIBUS, 6333, 6335, 6472, 6474, 6475
OP 77A
Baud rate on PROFIBUS, 6333, 6335, 6472, 6474, 6475
OP 77A and TP 177A
Differences in the recipe, 4434
OP73
Loading a project, 80
OP77A
Loading a project, 80
OPC, 6571, 7153
Accessing tags, 7167
Basic Panel, 6049
Basics, 6571
Communication, 6571
Compatibility, 7154
Configuring a connection, 6572
DCOM user rights, 7158
HMI device as OPC client, 7155
HMI device as OPC server, 7156
HTTP connection, 7154
Migrating data types, 202
Mobile Panel, 7154
Multi Panel, 7154
OPC XML DA server, 7154
Panel, 6054
Permitted characters in tag names, 7167

Permitted data type, 7166
Specifications, 7153
OPC DA
Data types, 6574
OPC DA server
Configuring HMI device as ~, 7157
Creating a connection, 7160
ProgID, 7155
OPC DA Server
Configuring HMI device as ~, 7158
OPC Gateway
Proxy setting, 7162
OPC server
Accessibility, 7155
OPC UA
Data types, 6575
OPC UA server
Security concept, 7164
Security settings, 7166
OPC UA Server
Creating a connection, 7161
OPC XML
"Date/Time" data type, 7168
"STRING" data type, 7167
Configuring server, 7163
Permitted cycle time, 7167
OPC XML DA
Data types, 6574
Deleting a server, 7164
Editing a server, 7164
OPC XML DA client, 7154
OPC XML DA server
URL, 7155
Open
Global library, 6781
Selection list, 6914, 6974
Open force job, 45
Open log file
Packet filter events, 765, 767
Security events, 765, 767
System events, 764, 766
Open User Communication
Ability to read back, 647
Changing IP configuration parameters with T_CONFIG, 3700
Changing parameter values, 646
Connection configuration, 626, 628
Connection description, 640, 643, 644
Connection establishment, 626
Connection parameters, 630
Creating a connection, 634
Deleting an interconnection, 637
Establishing a communication connection with TCON, 3662, 3664
Establishing a connection and reading data with TRCV_C, 3636, 3641
Establishing a connection and sending data with TSEND_C, 3625, 3629
General, 625
Instructions, 626
Port numbers, 645
Protocols used, 638
Receiving data via UDP with TURCV, 3690
Receiving data with TRCV, 3677, 3681
Sending data via UDP with TUSEND, 3687
Sending data with TSEND, 3671, 3674
Starting connection parameter assignment, 633
TCON_IP_RFC, 644
TCON_IP_v4, 643
TCON_Param, 640
Terminating a communications connection with TDISCON, 3669
TSAP, 647
OpenAllLogs, 4727, 4906
OpenCommandPrompt, 4730, 4907
OpenControlPanel, 4732
OpenControlPanelDialog, 4729, 4908
OpenFileBrowser, 4728
Opening a force table, 1886
Opening a project, 379
Compatibility mode V12 SP1, 378
Opening a watch table, 1859
Opening phone books, 7666
Opening Siemens Industry Online Support, 551
OpenInternetExplorer, 4731, 4909
OpenScreenKeyboard, 4728, 4910
OpenTaskManager, 4732, 4910
Operability
Dynamization, 4024
Operand, 1441, 1442, 1447, 1449, 1450, 1451, 1452, 1454, 1455, 1456, 1458, 1459, 1461, 1462
Inserting, 1609, 1652
Operand area, 1172
Operate recipe
Change recipe data record, 4481
Operating behavior
Enabling system memory, 1206
Using clock memory, 1207
Using time-of-day functions, 1192
Operating element
Availability for Basic Panels, 4088
Availability for Comfort Panels, 4090
Availability for Mobile Panels, 4094
Availability for Multi Panels, 4092

Availability for WinCC Runtime Advanced, 4096
operation, 6914, 6973
Recipe view, 4478
Operating hours counters
Handling with RTM, 2999
Operating mode, 824, 6517
Introduction, 1163
RUN, 1168
STARTUP, 1165
STOP, 1168
Transitions, 1164
Operating object
Availability for Panels, 4089
Operating recipes
Change recipe data record, 4481
Copying a recipe data record, 4480
Creating a recipe data record, 4474
Creating recipe data records, 4474, 4475, 4480
Delete recipe data record, 4481
Load recipe data record, 4481
Modifying the recipe structure, 4485
Read recipe data record, 4475, 4482
Transfer data record, 4476, 4483
Operating system, 118, 1411
Asian language setting, 6824
Setting to Western, 6824
operation
Key, 6913, 6972
Operating element, 6914, 6973
Operation, (Two-hand)
Alarm indicator, 4361
Alarm view, 4356
Alarm window, 4356
Recipe view, 4477
Simple alarm view, 4359
Simple recipe view, 4470
Operation in runtime
Multi-key operation, 86
Operation using the keyboard
Alarm view, 4357
Alarm window, 4357
Simple alarm view, 4360
Operation using the mouse
Alarm indicator, 4362
Alarm view, 4357
Alarm window, 4357
Simple alarm view, 4360
Operation with the keyboard
Recipe view, 4479
Operation with the mouse
Recipe view, 4478
OperationSteps property (VBS), 5433
Operator control
Alarm view, 4133, 4136, 4143, 4356
Alarm window, 4356
f(x) trend view, 4109
HTML Browser, 4117
Lock/unlock, 6970
Locked, 6969
Media Player, 4131
PDF view, 4147
Recipe view, 4156
Trend view, 4124
Operator controls
Alarm view, 4139
Recipe view, 4158
Operator input in the recipe view
Transferring data, 6219, 6309, 6365, 6433, 6485, 6739
OperatorMessageID property (VBS), 5433
OperatorMessageIndex property (VBS), 5434
OperatorMessageName property (VBS), 5435
OperatorMessageNumber property (VBS), 5435
OperatorMessageSelected property (VBS), 5436
OperatorMessageSource1 property (VBS), 5436
OperatorMessageSource10 property (VBS), 5441
OperatorMessageSource2 property (VBS), 5437
OperatorMessageSource3 property (VBS), 5437
OperatorMessageSource4 property (VBS), 5438
OperatorMessageSource5 property (VBS), 5438
OperatorMessageSource6 property (VBS), 5439
OperatorMessageSource7 property (VBS), 5439
OperatorMessageSource8 property (VBS), 5440
OperatorMessageSource9 property (VBS), 5441
OperatorMessageType1 property (VBS), 5442
OperatorMessageType10 property (VBS), 5446
OperatorMessageType2 property (VBS), 5442
OperatorMessageType3 property (VBS), 5443
OperatorMessageType4 property (VBS), 5443
OperatorMessageType5 property (VBS), 5444
OperatorMessageType6 property (VBS), 5444
OperatorMessageType7 property (VBS), 5445
OperatorMessageType8 property (VBS), 5445
OperatorMessageType9 property (VBS), 5446
Optimize position controller, 7436
Optimized access, 1419, 1421
Optimized block access, 225
Optimizing the configuration
Allen-Bradley DF1, 6619, 6636, 6658, 6667, 6687, 6699, 6719
Option
Installing, 6999
Uninstalling, 6999

Option handling, 1102, 1104, (See Configuration control (ET 200SP)), (See Configuration control (ET 200MP))
Option handling (see Configuration control), 885, 1278, 1316, 1346
OptionGroup object, 5131
OR, 1665, 2425, 2473, 2474, 2706
Order number, 942, (See Article number)
Organization block (OB)
  Assigning event with ATTACH, 3216
  Canceling event assignment with DETACH, 3218
  Creating, 1506
  Cyclic interrupt OB, querying parameters with QRY_CINT, 3220
  Cyclic interrupt OB, setting parameters with SET_CINT, 3219
  exporting to an external source file, 1746
  Function, 1414
  Measuring runtime with RT_INFO, 3270
  Reading start information with RD_SINFO, 3262
  Start information, 1414
Orientation property (VBS), 5706
OSPF
  Area range, 1066
  Areas, 933, 1065
  Configuration, 1063
  Interfaces, 1067
  Link State Advertisement, 933
  OSPFv2 Interfaces, 960
  OSPFv2 LSDB (information), 964
  OSPFv2 neighbors, 961
  OSPFv2 virtual neighbors, 963
  Router, 933
  Router status, 933
  Virtual Links, 1069
Other PLCs
  Addressing, 6600
  Data types, 6600
  Special features, 6600
OUC, (See Open User Communication)
OUT_RANGE, 2289, 2565
Outgoing, 4285, 4851
Output
  Inserting, 1606, 1648
  remove, 1607, 1649
Output (O), 1172
Output byte (QB), 1172
Output double word (QD), 1172
Output fields
  Multiplex tag, 68
Output rate (high-speed analog modules), 1299
Output word (QW), 1172
Overflow, 1266, 1289, 4855
Overlay icon, 1361
Overlaying
  Tag, 1462
Overlaying tags, 1562
Override function, 4188
Oversampling, 1298
Overview
  using the display formats in the force table, 1890
  via the display formats in the watch table, 1864
Overview navigation, 538, 541, 543, 6003, 6010, 6012
Overview of the cross-reference list, 1837
Overview of TS Adapters that can be used, 7673
Overview window, 54, 331
  Comparing objects, 333
  Display all blocks, 334
  Showing and hiding columns, 334
  Sorting details view, 334

P

P, 2209, 2212, 2485, 2488
P_TRIG, 2214, 2490
Pack&Go file
  Create, 6944
  Download to HMI device, 6945
Packet filter log, 765, 767
Page
  Adding to report, 4500
  Deleting report page, 4500
  Maximum number of configurable in report, 4498
  Sorting report page, 4500
Page number
  Layout in reports, 4525
  Use in reports, 4525
PageDown, 4758, 4911
PageMode property (VBS), 5447
PageModeMessageNumber property (VBS), 5447
PageUp, 4759, 4912
Pane mode, 328
Panel
  Area pointer, 6056
  Communication drivers, 6054
  Display and operating object, 4089
  HTTP protocol, 6054
  Interface, 6055
  Mitsubishi, 6054
  Modicon Modbus, 6054
  Omron, 6054
  OPC, 6054
  S7 200, 6054

S7 300, 6054
S7 400, 6054
S7-1200, 6054
Panels
Available system functions, 4624
Panning, (See move view)
Parallel communication
Communication drivers, 6077, 6600, 6726
Parameter
For CPU, 1185
Hidden parameters, 1611, 1653
Predefined - writing with WR_DPARM, 3214
SIMATIC LOGO!, 6503
Parameter assignment, 1430
Hardware, 547
Parameter assignment parameters
Hardware, 554
Parameter transfer, 1429, 1431, 1433, 1437, 1438, 1439, 1440
Transfer parameter as copy or as pointer, 1434
User-defined function, 4584
Parameter types, 1953
Parameters
Client, 6560
Connection, 6605, 6621, 6647, 6661, 6678, 6690, 6710
MPI, 6294, 6295, 6297
PROFIBUS, 6114, 6115, 6117, 6200, 6201, 6203, 6280, 6281, 6283, 6415, 6417, 6418
S7 1200, 6241
S7 1500, 6149, 6457
S7 200, 6330, 6334, 6469, 6473, 6475
Tags, 6564
Parent property (VBS), 5789
Partner port
Information on monitoring, 1130
Password
Changing, 4541
For CPU access protection, 1185
for function key, 4043
Hierarchy level, 4533, 4535
HMI connection, 6100, 6119, 6185, 6205, 6353
Password aging, 4534
Password complexity, 4534, 4536
SmartServer, 7112
Password legitimation, 2408, 2688, 2912
Password property (VBS), 5448
Password protection, 855, 1185, 1208, 1209, 6097, 6182, 6184
Protection concept, 417
Revoking access rights, 418
Paste
Tag, 7772
PasteRows, 5900
Pasting
Adjusting screen size, 6868
Color, 6867
Font, 6867
Function key, 6868
Invalid object, 6867
Principle, 6865
PC
HMI connection, 6085, 6109, 6171, 6197, 6257, 6275, 6291, 6343, 6394, 6413
S7 1500, 6085
S7 300/400, 6257
SIMATIC ET 200 CPU, 6394
SIMATIC S7-1500 Software Controller, 6343
WinCC RT Advanced, 6085, 6343, 6394
WinCC RT Professional, 6085, 6109, 6394
WinCC RT Advanced, 6109
PC CP, 678
PDF view
Button, 4147
Configuring, 4146
Multi-touch operation, 4147
PDFFitToHeight, 4735
PDFFitToWidth, 4734
PDFGoToFirstPage, 4736
PDFGoToLastPage, 4736
PDFGoToNextPage, 4737
PDFGoToPage, 4737
PDFGoToPreviousPage, 4738
PDFScrollDown, 4733
PDFScrollLeft, 4740
PDFScrollRight, 4740
PDFScrollUp, 4734
PDFZoomIn, 4739
PDFZoomOriginal, 4741
PDFZoomOut, 4739
PE_CMD, 3159
PE_DS3_Write_ET200S, 3164
PE_End_RSP, 3196
PE_Error_RSP, 3193
PE_Get_Mode_RSP, 3198
PE_I_DEV, 3189
PE_List_Modes_RSP, 3197
PE_Measurement_List_RSP, 3204
PE_Measurement_Value_RSP, 3205
PE_PEM_Status_RSP, 3200
PE_Start_End, 3154
PE_Start_RSP, 3194
PE_WOL, 3165

PEEK, 2848
Read memory address, 2848
PEEK_BOOL, 2850
Read memory bit, 2850
Perfect Forward Secrecy, 747
Performance, 224
Performance features
Engineering system, 7005
HMI device, 7007, 7011, 7014, 7018, 7024, 7027, 7032
WinCC Runtime, 7032
Permanent connection, 899
Permanent window, 3906
Permit access with PUT/GET communication from remote partners, 1212
Permit overwriting of device name, 1137
Permitted data type
OPC, 7166
SIMATIC HMI HTTP Protocol, 7141
Permitted data types
Trends, 6135, 6225, 6314, 6371, 6439, 6492, 6509
Permitted file names, 369
Permitted operands for the force table, 1888
Permitted operands for the watch table, 1861
PersistentRTAuthorization property (VBS), 5707
PersistentRTCSAuthorization property (VBS), 5749
PersistentRuntime property (VBS), 5708
PersistentRuntimeAuthorization property (VBS), 5709
PersistentToDownload property (VBS), 5750
PersistentToDownloadAuthorization property (VBS), 5750
PG/PC interface
Go online, 7593
Modifiable MPI interface parameters, 7610
Modifiable PROFIBUS interface parameters, 7610
Picture property (VBS), 5449
PictureAlignment property (VBS), 5449
PictureDeactivated property (VBS), 5450
PictureOff property (VBS), 5450
PictureOn property (VBS), 5451
PID_3Step
In/out parameters, 3501
Input parameters, 3497, 3528
Instruction, 3488, 3520
Output parameters, 3499, 3529
Static tags, 3530
PID_Compact
In/out parameters, 3453
Input parameters, 3451, 3473
Instruction, 3470
Output parameters, 3452, 3474
Static tags, 3454, 3475
PID_Temp
ActivateRecoverMode tag, 3601
Cascade, 3564
Cascading, 7319
ErrorBits parameter, 3599
Functional description, 3554
In/out parameters, 3564
Input parameters, 3560
Mode, 3564
Multi-zone applications, 7326
Output parameters, 3561
PID_Temp state and mode parameters, 3591
PwmPeriode, 3605
Static tags, 3565
Tag Warning, 3604
Pin assignment
6XV1440 - 2P for Mitsubishi PG protocol, 6665
Allen-Bradley, 6628
Allen-Bradley cable 1747-CP3, 6630
Allen-Bradley cable 1761-CBL-PM02, 6631
Allen-Bradley cable 1784-CP10, 6629
Multipoint cable 1:MP/TP/PC, 6714
Multipoint cable 2:RS422, MP/TP/PC, 6715
Point-to-point cable 1,
Point-to-point cable 2, 6696
Point-to-point cable PP1 for Omron, 6715
Point-to-point cable PP2 for Omron, 6716
Ping, 1002
PLC
Connecting, 6607, 6649, 6663, 6681, 6712
Enabling system diagnostics, 4409
MPI parameters, 6297
PROFIBUS parameters, 6117, 6203, 6283, 6417
PROFINET parameters, 6094, 6177, 6263, 6351, 6402
System diagnostics view, 4409
Tag, 4225
PLC data, 7745
PLC data type, 254, 286
Addressing, 1460
Change the number, 1737
Comparing, 1780
Creating, 1736
Declaration in data blocks, 1713
Declaration in the block interface, 1562
Declaration table for PLC data types, 1735
Declaring, 1740
Declaring ARRAY, 1738
Declaring STRUCT, 1739

Definition, 1734, 1954
Deleting, 1737
Offline/offline comparison, 1753
PLC data types generated by the system in libraries, 100
Programming the structure, 1738
Tag properties, 1741
PLC data types, 98, 226, 4205
PLC tag
Automatically filling in cells, 1501
Comparing, 1779
Comparing a PLC tag table, 1753
Comparison, 1750
Copying, 1500
Declaring, 1487, 1490, 1492
Definition, 1448
Importing and exporting, 1502, 1503, 1504
Monitor, 1498
Offline/offline comparison, 1753
Permissible addresses and data types, 1484
PLC tag table, 1479, 1480, 1485, 1486
Properties, 1496
Reconnecting, 4216
Retentive behavior, 1488, 1489
Rules, 1484
Sorting rows, 1501
Updating comparison results, 413
PLC tags, 864, 867
PLC UDTs
WinCC, 4205
PLC user data types
Interconnecting with faceplates, 4065
PLC5, 6634
PLUG, 1001
C-PLUG, (C-PLUG)
KEY-PLUG, (KEY-PLUG)
PN/PN couplers
Grouping, 1144
Linking Ethernet subnets, 1144
Pointer
ANY, 1948
POINTER, 1946
VARIANT, 1951
POINTER, 1946
Pointer name
Area pointer, 6042
PointerColor property (VBS), 5452
PointsCount property (VBS), 5452
point-to-point, 931
Point-to-point communication (PtP), 1198
Freeport protocol, 1198
Point-to-point connection
Allen-Bradley DF1, 6626
Configuring communication parameters with PORT_CFG, 3722
Configuring receive parameters with RCV_CFG, 3727
Configuring serial transmission parameters with SEND_CFG, 3725
Deleting receive buffer with RCV_RST, 3738
Enable receiving with RCV_PTP, 3737
Initiating data transfer with SEND_PTP, 3735
Querying the signal state with SGN_GET, 3739
Setting output signals with SGN_SET, 3740
POKE, 2852
Write memory address, 2852
POKE_BLK, 2855
Write memory area, 2855
POKE_BOOL, 2853
Write memory bit, 2853
Polygon, 4150
Configuring corners, 4150
Layout, 4150
Radii, 4150
Polygon object, 5134
Polyline, 4151
Configuring corners, 4152
Line end, 4151
Line start, 4151
Radii, 4152
Polyline object, 5137
Pop-up screen, 3907
Copying, 3909
Creating, 3908
Deleting, 3910
Moving in a group, 3909
Objects, 3908
Port, 598
102 (S7 protocol - TCP), 706
123 (NTP), 725, 736
161 (SNMP), 725
20/21 (FTP), 706
443 (HTTPS), 725
4500 (IPsec), 725
500 (IPsec), 725
500 (ISAKMP - UDP), 755
514 (Syslog), 725
80 (HTTP), 706
8448 (security diagnostics), 838, 841
Configuration, 994
Interconnecting, 1129
Port configuration, 993, 996
Renaming, 673

Port configuration, 996
Port diagnostics
SFP diagnostics, 1006
Port numbers, 645
Port options, 1136, 6102, 6187, 6268, 6405
Enable autonegation, 1135, 6101, 6186, 6267, 6405
Monitor, 1134, 6101, 6186, 6267, 6404
Transmission medium/duplex, 1134
Transmission rate / duplex, 6101, 6186, 6267, 6404
PORT_CFG, 3722
Portal view, 307
Position
Editing multiple objects, 3968, 4511
Of objects, 3942, 4505
of objects in the report, 4498
Position control, 7395
Position monitoring, 7393
Positioning axis technology object
Active homing, 7390
Actor tags, 7499
Add new object, 7360
Axis name configuration, 7362, 7544
Basic parameters, 7362, 7543
Changing the configuration parameters for dynamics in the user program, 7385
Changing the configuration parameters for homing in the user program, 7392
Commissioning overview, 7359
Configuration overview, 7359
Configuration window icons, 7361
Configuring acceleration, 7380
Configuring active homing, 7386, 7548
Configuring approach/homing direction, 7387, 7549
Configuring deceleration, 7380
Configuring distance per motor revolution, 7373, 7546
Configuring end of homing switch, 7388, 7389
Configuring end of reference point switch, 7548, 7550
Configuring home position offset, 7388, 7550
Configuring homing switch input, 7387, 7389
Configuring invert direction signal, 7374, 7547
Configuring jerk limiter, 7381
Configuring maximum velocity / start/stop velocity, 7380
Configuring passive homing, 7388, 7547
Configuring permit auto reverse at the hardware limit switch, 7549
Configuring pulses per motor revolution, 7373, 7546
Configuring ramp-down time, 7380
Configuring ramp-up time, 7380
Configuring reference point position, 7388, 7390, 7548, 7550
Configuring reference point switch input, 7548, 7549
Configuring smoothing time, 7382
Configuring the homing speed, 7388, 7550
Configuring the velocity limiting unit configuration, 7380
Diagnostics overview, 7359
Drive enable configuration, 7546
Drive ready configuration, 7546
Drive signal configuration, 7365, 7546
DynamicDefaults tags, 7512
DynamicLimits tags, 7511
Emergency stop deceleration configuration, 7383
ErrorBits tags, 7528
Extended parameters, 7546
FollowingError tags, 7518
General dynamics configuration, 7379
Hardware and software components, 7355
Hardware interface configuration, 7364, 7544
Homing tags, 7516
Mechanics configuration, 7373, 7374, 7546
Mechanics tag, 7509
Modulo tags, 7510
Passive homing, 7391
Position tag, 7497
PositionControl tags, 7517
PositionLimitsHW tags, 7514
PositionLimitsSW tags, 7513
PositionMonitoring tags, 7519
PTO and HSC configuration, 7364, 7544
Response when jerk limiter is activated, 7384
Sensor[1] tags, 7502
Sensor[1].ActiveHoming tags, 7505
Sensor[1].Interface tags, 7503
Sensor[1].Parameter tags, 7504
Sensor[1].PassiveHoming tags, 7507
StandstillSignal tags, 7520
StatusBits tags, 7523
StatusDrive tags, 7522
StatusPositioning tags, 7521
StatusSensor tags, 7522
Tag ActualPosition, 7498
Tag ActualVelocity, 7498
Tools, 7358
Unit of measurement configuration, 7363
Units tag, 7508, 7509

Updating the tags, 7530
User unit configuration, 7545
Velocity tag, 7497
Possible cause of error
Transferring data, 6133, 6222, 6312, 6368, 6436, 6488, 6742
Power budget, 1325
Power Management, 5976
Powerpack, 135, 136
Engineering system, 135
Installing, 136
Runtime, 135
PPI, 6018
Network, 6021
Network architecture, 6021
S7 200, 6475
Precision property (VBS), 5453
Predefined alarm classes, 4287, 4288
Predefined firewall rules
CP 1543-1, 838, 841
CP 1628, 829
CP x43-1 Adv., 827
SCALANCE S, 793, 794
Predefined styles, 3963
Premium, 6685
Prerequisites for establishing a VPN connection, 7702
Preshared keys, 745
Press, 4850
Press ESC twice, 4850
Press key, 4854
PressButton, 4752
Pressed property (VBS), 5453
PreviousColumn, 5900
PreviousTrend, 5901
Print, 5901
Print alarm
Configuring print parameters, 7057
Print scope, 420
Print server, 7037
Printable contents, 420
PrintConfiguration property (VBS), 5752
Printing
Audit Trail, 7054
Changing the settings, 421
Creating a cover page, 426
Creating frames, 425
Documentation settings, 419
Elements in the library, 424
Elements in the project tree, 424
Non-printable contents, 420
Non-printable objects, 421
Print contents, 423
Print server, 7037
Printing device view, 544
Printing network view, 544
Process a cover page, 427
Process borders, 427
Report, 4501, 4516
Specify layout, 422
Specify print layout, 422
Specifying the print area, 431
Structure of printout, 419
Use borders, 424
Use borders and cover pages, 419
Use cover page, 422
Printing a project, 436
Printing labeling strips, 440
Determining the correction value for the print shift, 445
Printing of alarms
Configuring:printing of alarms, 4315
PrintJobName property (VBS), 5712
PrintReport, 4676, 4912
PrintScreen, 4675, 4913
PrintVisibleColumnsOnly property (VBS), 5712
Priorities
Local error handling, 1702
Prioritized startup, 1111
Process data area
Module addressing, 1176
Module start address, 1176
Process image, 1177
Basics, 1174
I/O access error, 1175
Reading inputs with GETIO, 3072
Reading the process image area with GETIO_PART, 3074
Synchronizing the inputs with SYNC_PI, 3066
Synchronizing the outputs with SYNC_PO, 3067
Transferring the process image area with SETIO_PART, 3075
Updating, 1175
Updating inputs with UPDAT_PI, 3062
Updating the outputs with UPDAT_PO, 3064
Writing outputs with SETIO, 3073
Process value logging, 4250
ProcessValue property (VBS), 5454
Product support, 6017
PROFIBUS, 593, 6018, 6204, 6284
Connection with PROFINET, 1111
ET 200S in DPV1 mode, 1105
HMI connection, 6105, 6109, 6111, 6190, 6194, 6197, 6271, 6275, 6408, 6412, 6413

Network, 6019
OP 73, 6151, 6243, 6454
OP 77A, 6151, 6243, 6454
OP 73, 6333, 6335, 6472, 6474, 6475
OP 77A, 6333, 6335, 6472, 6474, 6475
Parameters, 6114, 6115, 6117, 6150, 6200, 6201, 6203, 6243, 6280, 6281, 6283, 6333, 6415, 6417, 6418, 6454, 6472
S7 1200, 6189, 6192, 6194, 6197
S7 1500, 6104, 6107, 6109, 6111
S7 300/400, 6105, 6270, 6271, 6273, 6275
S7-1200, 6190, 6407, 6408, 6413
SIMATIC ET 200 CPU, 6409, 6412
Standard, 6204, 6284
Universal, 6204, 6284
WinCC RT Advanced, 6107, 6192, 6273, 6409
WinCC RT Professional, 6107
PROFIBUS cable configuration
Optical ring, 595
PROFIBUS communication
S7 1200, 6189, 6192
S7 1500, 6104, 6107
S7 300/400, 6270, 6273
SIMATIC ET 200 CPU, 6407, 6409
PROFIBUS DP, 1089, 6019
Configuring the Control Panel, 6753
Connection, 6147, 6327, 6451
Direct key, 6539
PROFIBUS parameters
S7 1200, 6200, 6201, 6203, 6415, 6417, 6418
S7 1500, 6114, 6115, 6117
S7 300/400, 6280, 6281, 6283
PROFIBUS profile, 593, 6204, 6284
Different profiles on the same subnet, 593, 6204, 6284
Effect on the transmission rate, 593, 6204, 6284
Meaning of profiles, 594, 6204, 6284
PROFIBUS DP
DirectKey, 4674
Network configuration, 6757
PROFIenergy, 1111
Control PROFIenergy commands in the I-Device, 3189
Description, 3152
Generate answer to command at end of pause, 3196
Generate answer to command at start of pause, 3194
Generate list of supported measured values as answer, 3204
Generate negative answer to command, 3193
Generate PEM status as answer, 3200
Generate queried energy data as answer, 3198
Generate queried energy savings modes as answer, 3197
Generate queried measured values as answer, 3205
Generate supported PROFIenergy commands as answer, 3202
PE_CMD, 3159
PE_DS3_Write_ET200S, 3164
PE_End_RSP, 3196
PE_Error_RSP, 3193
PE_Get_Mode_RSP, 3198
PE_I_DEV, 3189
PE_Identify_RSP, 3202
PE_List_Modes_RSP, 3197
PE_Measurement_List_RSP, 3204
PE_Measurement_Value_RSP, 3205
PE_PEM_Status_RSP, 3200
PE_Start_End, 3154
PE_Start_RSP, 3194
Reading out status information, 3159
Set power module switching behavior, 3164
Start and exit energy-saving mode, 3154, 3159
Wake on LAN, 3165
Profile, (See bus profile)
Profiles, 1218
PROFINET, 598, 809, 6018, 6584, 6594, 6597
Connection, 6146, 6324, 6381, 6449
Connection with PROFIBUS, 1111
Device replacement without exchangeable medium, 1137
HMI connection, 6080, 6081, 6083, 6085, 6088, 6162, 6163, 6166, 6168, 6249, 6250, 6252, 6254, 6257, 6339, 6340, 6341, 6343, 6346, 6390, 6391, 6392, 6394, 6397
S7 1200, 6162, 6166
S7 1500, 6080, 6083
S7 300/400, 6249, 6252
SIMATIC ET 200 CPU, 6390, 6392
SIMATIC S7-1500 Software Controller, 6339, 6341
PROFINET address, 689
PROFINET device name, 1121
PROFINET interface, 1124, 1185, 1367
PROFINET IO, 1109, 1111, 1115, 1116, 1126, 1127, 6582
Port options, 1134, 6101, 6186, 6267, 6404
PROFINET IO device
Assigning a name, 1391
Name assignment in the dialog, 1393
Name assignment in the online and diagnostics view, 1392

PROFINET IO devices
Reading a portion of the inputs with GETIO_PART, 3074
Reading consistent data with DPRD_DAT, 3124
Writing a part of the outputs with SETIO_PART, 3075
Writing all outputs with SETIO, 3073
Writing consistent data with DPWR_DAT, 3126
PROFINET IO system, 1127
PROFINET parameters
HMI connection, 6091, 6173, 6260, 6348, 6399
HMI device, 6092, 6175, 6261, 6349, 6400
PLC, 6094, 6177, 6263, 6351, 6402
S7 1200, 6173, 6177
S7 1500, 6091, 6094, 6348, 6349, 6351, 6399, 6402
S7 300/400, 6260, 6261, 6263
S7-1500, 6092, 6400
PROFINET RT class, 1110, 6583
PROFIsafe
Disconnecting, 5956
PROFIsafe address, 5975
Program alarm
generating with Program_Alarm, 3246
Program card, 1169
Program control operations, 2413, 2693, 2918
Program cycle OB
Description, 1213
Program editor
"Instructions" task card, 1531
Block interface, 1531
Enlarging the programming window, 1537
Favorites, 1531
Function, 1529
General settings, 302, 304, 305, 1549
Programming window, 1531
Structure, 1530
Testing task card, 1532
Toolbar, 1530
Using the keyboard, 1533
Program execution
Cyclic, 1213
Program information
Displaying, 1809
In the assignment list, 1810
In the call structure, 1818
In the dependency structure, 1824
in the tab, 1830
Views, 1809
Program name
Displaying, 7765
Program resources, 1506
Program status
Call environment, 1841, 1842
Editing blocks, 1845
FBD, 1847
Function, 1840
GRAPH, 1851
LAD, 1847
Modify tag, 1846
SCL, 1850
STL, 1848
Switching off, 1844
Switching on, 1844
Program_Alarm, 3246
Programming
Linear, 1412
Structured, 1413
Programming language
Changeover, 1699
FBD, 1619
LAD, 1575
Rules for changing, 1698
Programming recommendations, 226
Programming window
Customizing, 1671
SCL, 1670
Project
Basics, 375
Carrying out an online/offline comparison, 403
Carrying out offline/offline comparisons, 404
Closing, 384
Comparing data, 402
compile (overview), 6893, 6925
Compiling, 6894, 6927
Compiling project data, 393
Creating, 376
Deleting, 386
Detailed comparison, 413
Download, 6902, 6940
download (overview), 6893, 6925
Harmonizing, 530
Load, 6943
Migrate, 165, 194
multilingual, 6829
Multiple connections, 6125, 6213, 6304, 6360, 6427, 6481
Removing, 385
Saving, 383
Showing properties, 382
Simulation with a tag simulator, 6896, 6930
Upgrade, 6877
Project archives, 386

Project data
Compiling, 393
Downloading to a memory card, 397
Export alarm, 6798
Export text list, 6812
Exporting a recipe, 6794
Exporting tag, 6805
Importing a tag, 6806, 6808
Importing alarms, 6799
Information on compiling, 392
Information on loading, 394
IO-Link master module, 92
Loading, 396
Uploading from a device, 399
Writing to a memory card, 397
Project file
Initialization, 7752
Initializing, 7752
Project ID, 5999
Project language, 454, 6823
Activate, 6826
Disabling, 6826
Project languages
Changing the editing language, 457
Specifying project languages, 456
System texts, 6830
Use, 454
User texts, 6830
Project library, 55, 471, 6770, 7176
Adding types, 506
Assigning a common version to types, 525
Cleaning up, 531
Creating folders, 497
Discarding all versions, 516
Discarding the type version, 515
Duplicating types, 509
Migrating, 187
Performing a consistency check for a type version, 515
Release all versions, 518
Release the type versions, 516
Updating the project, 519
Using filter view, 500
Using the element view, 474
Using types, 510
Project text
Exporting, 6833
Project texts
Application example, 462
Changing the text of selected objects, 458
Displaying reference text, 458, 6832
Exporting all texts, 461
Exporting individual texts, 460
Exporting texts of a device, 460
Importing, 462, 6835
Migrating, 185
Translating individual texts, 4074, 6831
Translating project texts, 457
Translation into project languages, 454, 6830
Project tree
Adding device, 566
Function, 312
Hiding, 1537
Reducing automatically, 318
Showing, 1537
Showing additional columns, 316
Project version
Upgrade, 6877
Project view, 309
ProjectName object, 5139
ProjectPath property (VBS), 5752
Project-wide alarm class
Create, 4298
Use, 4298
Properties (CPU), 1185
Properties (VBS)
AboveUpperLimitColor, 5223
AcceptOnExit, 5223
AcceptOnFull, 5224
AccessPath, 5224
ActiveProject, 5225
ActiveScreen, 5226, 5227
ActualPointIndex, 5228
ActualPointLeft, 5229
ActualPointTop, 5229
AdaptBorder, 5230
AdaptFontSizeToLineHeight, 5761
AdaptPicture, 5231
AdaptScreenToWindow, 5231
AdaptWindowtoScreen, 5231
Address, 5232
AddressEnabled, 5232
AdjustBorder3DWithStyle, 5737
AdjustRulerWindow, 5761
AdressPreview, 5232
Alarm, 5232
AlarmClasses, 5233
AlarmColor, 5233
AlarmFilter, 5738
AlarmHigh, 5762
AlarmID, 5234
AlarmLog, 5234
AlarmLow, 5677
AlarmLowerLimit, 5234

AlarmLowerLimitColor, 5235
AlarmLowerLimitEnabled, 5236
AlarmLowerLimitRelative, 5236
AlarmSource, 5237
AlarmUpperLimit, 5237
AlarmUpperLimitColor, 5238
AlarmUpperLimitEnabled, 5238
AlarmUpperLimitRelative, 5239
Alignment, 5762
AlignmentLeft, 5763
AlignmentTop, 5678
AllFilters, 5239
AllFiltersForHitlist, 5239
AllowEdit, 5240
AllowMenu, 5240
AllowPersistence, 5763
Analog, 5240
AngleMax, 5240
AngleMin, 5241
Appearance, 5242
ApplyProjectSettings, 5242
ApplyProjectSettingsForDesignMode, 5242
AskOperationMotive, 5242
Assignments, 5243
AssociatedS7GraphDBTag, 5243
AssumeOnExit, 5679
AssumeOnFull, 5679
Authorization, 5244
AutoCompleteColumns, 5245
AutoCompleteRows, 5246
AutoScroll, 5246
AutoSelectionColors, 5247
AutoSelectionRectColors, 5247
AutoSizing, 5248
AverageLast15Values, 5248
AxisXBunchCount, 5249
AxisXMarkCount, 5249
AxisXShowBunchValues, 5249
AxisY1BunchCount, 5249
AxisY1MarkCount, 5249
AxisY1ShowBunchValues, 5249
AxisY2BunchCount, 5249
AxisY2MarkCount, 5250
AxisY2ShowBunchValues, 5250
BackColor, 5250
BackColorBottom, 5252
BackColorTop, 5252
BackFillStyle, 5253
BackFillStyleReadOnlySpecial, 5764
BackFlashingColorOff, 5254
BackFlashingColorOn, 5255
BackFlashingEnabled, 5256
BackFlashingRate, 5257
BackgroundColor, 5258
BackPictureName, 5711
BarBackColor, 5258
BarBackFillStyle, 5258
BarBackFlashingColorOff, 5260
BarBackFlashingColorOn, 5260
BarBackFlashingEnabled, 5260
BarBackFlashingRate, 5261
BarColor, 5261
BarEdgeStyle, 5261
BarStartValue, 5765
BaseScreenName, 5261
BeginTime(i), 5765
BelowLowerLimitColor, 5262
BitNumber, 5262
BlinkColor, 5263
BlinkMode, 5263
BlinkSpeed, 5264
Blocks, 5264
BorderBackColor, 5265
BorderBrightColor3D, 5266
BorderColor, 5267
BorderEnabled, 5268
BorderEndStyle, 5269
BorderFlashingColorOff, 5269
BorderFlashingColorOn, 5271
BorderFlashingEnabled, 5272
BorderFlashingRate, 5273
BorderInnerStyle3D, 5274
BorderInnerWidth3D, 5274
BorderOuterStyle3D, 5275
BorderOuterWidth3D, 5275
BorderShadeColor3D, 5276
BorderStyle, 5277
BorderStyle3D, 5278
BorderWidth, 5278
BorderWidth3D, 5280
BottomMargin, 5280
BusyText, 5280
ButtonBackColor, 5281
ButtonBackFillStyle, 5281
ButtonBarElements, 5281
ButtonBarStyle, 5281
ButtonBorderBackColor, 5281
ButtonBorderColor, 5281
ButtonBorderWidth, 5281
ButtonCommand, 5682
ButtonCornerRadius, 5282
ButtonEdgeStyle, 5282
ButtonFirstGradientColor, 5282
ButtonFirstGradientOffset, 5282

ButtonMiddleGradientColor, 5282
ButtonSecondGradientColor, 5282
ButtonSecondGradientOffset, 5282
BV_ColumnWidth_Date, 5283
BV_ColumnWidth_Event, 5283
BV_ColumnWidth_EventSeverity, 5283
BV_ColumnWidth_EventState, 5283
BV_ColumnWidth_Number, 5283
BV_ColumnWidth_Time, 5283
BV_ItemText_Date, 5283
BV_ItemText_Event, 5284
BV_ItemText_EventSeverity, 5284
BV_ItemText_EventState, 5284
BV_ItemText_Number, 5284
BV_ItemText_Time, 5284
BV_ShowItem_Date, 5284
BV_ShowItem_Event, 5284
BV_ShowItem_EventSeverity, 5285
BV_ShowItem_EventState, 5285
BV_ShowItem_Number, 5285
BV_ShowItem_Time, 5285
Caption, 5285
CaptionBackColor, 5286
CaptionColor, 5286
CaptionText, 5287
CaptionTop, 5288
CellCut, 5288
CellSpaceBottom, 5289
CellSpaceLeft, 5289
CellSpaceRight, 5290
CellSpaceTop, 5290
CenterColor, 5291
CenterSize, 5291
CentrePoint, 5766
CentrePointLeft, 5767
CentrePointTop, 5767
ChangeMouseCursor, 5292
CheckMarkAlignment, 5292
CheckMarkCount, 5293
ClearOnError, 5293
ClearOnFocus, 5294
Closeable, 5294
Color, 5295
ColorChangeHysteresis, 5295
ColorChangeHysteresisEnabled, 5296
ColumnAlignment, 5296
ColumnColor(i), 5769
ColumnDisplayName(i), 5769
ColumnResize, 5297
Columns, 5298
ColumnScrollbar, 5298
ColumnSettings, 5299
ColumnSettingsBufferView, 5299
ColumnSizingEnable, 5740
ColumnsMoveable, 5299
ColumnTextAckGroup, 5299
ColumnTextAlarmState, 5299
ColumnTextAlarmText, 5299
ColumnTextClassName, 5300
ColumnTextDate, 5300
ColumnTextDevice, 5300
ColumnTextDiagnosable, 5300
ColumnTextNumber, 5300
ColumnTextTime, 5300
ColumnTitleAlignment, 5301
ColumnTitles, 5302
ColumnUpdateEnabled(i), 5770
ComboBoxFont, 5302
Command, 5770
CommonTimeAxisColor, 5771
CompatibilityMode, 5302
ConfigureTimeAxis(i), 5771
ConnectionType, 5303
ConnectOnStart, 5303
ConnectTrendWindows, 5303
ContinousChange, 5304
ControlDesignMode, 5505
CornerRadius, 5304
CornerStyle, 5304
Count, 5306
CountDivisions, 5306
CountOfLinesPerAlarms, 5307
CountOfVisibleAlarms, 5307
CountSubDivisions, 5307
CountValueColumns, 5688
CountVisibleItems, 5307
CurrentColumnIndex, 5688
CurrentCurveIndex, 5772
CursorControl, 5308
CurveColor(i), 5772
CurveLineWidth(i), 5773
Curves, 5309
CurvesCount, 5774
CurveUpdateEnabled(i), 5774
DangerRangeColor, 5309
DangerRangeStart, 5310
DangerRangeVisible, 5310
DataFormat, 5311
DataIndex(i), 5775
DataLogTag, 5775
DataRecordNameCaption, 5312
DataRecordNrCaption, 5312
DataX(i), 5776
DataXY(i), 5776

DataY(i), 5777
DeleteDate(i), 5777
DeleteEnable, 5689
DialColor, 5312
DialFillStyle, 5313
DialSize, 5314
Direction, 5778
DisplayButton2Plc, 5314
DisplayButtonComparison, 5314
DisplayButtonDelete, 5314
DisplayButtonFromPlc, 5314
DisplayButtonHelp, 5315
DisplayButtonNew, 5315
DisplayButtonSave, 5315
DisplayButtonSaveAs, 5315
DisplayComboBox, 5315
DisplayGridlines, 5315
DisplayLabeling, 5315
DisplayName(i), 5778
DisplayNumbers, 5316
DisplayOptions, 5316
DisplayStatusBar, 5316
DisplayTable, 5316
DoubleClickAction, 5316
DrawAlwaysInsideFrame, 5689
DrawEnhancedHTMLBrowser, 5779
DrawInsideFrame, 5317
DrawStylishButton, 5690
Drive, 5318
EdgeStyle, 5319
Editable(i), 5779
EditEnabled, 5780
EditOnFocus, 5320
Enabled, 5321
EnableEdit, 5323
EnableNavigateKeys, 5323
EndAngle, 5323
EndLeft, 5324
EndPoint, 5780
EndPointLeft, 5781
EndPointTop, 5782
EndStyle, 5324
EndTime(i), 5782
EndTop, 5325
EntryNameCaption, 5325
EvenRowBackColor, 5326
ExportDelimiter, 5326
ExportDirectoryChangeable, 5327
ExportDirectoryname, 5327
ExportFileExtension, 5328
ExportFilename, 5328
ExportFilenameChangeable, 5329
ExportFormatGuid, 5330
ExportFormatName, 5330
ExportParameters, 5331
ExportSelection, 5331
ExportShowDialog, 5332
ExtraSpaceForLabelDisplay, 5783
FieldLength, 5333, 5334
FillColorMode, 5334
FillPatternColor, 5334
FillStyle, 5336
Filter, 5336
FilterTag, 5336
FilterText, 5337
FirstConnectedObjectIndex, 5337
FirstConnectedObjectName, 5337
FitToLargest, 5338
FitToSize, 5338
FixedAspectRatio, 5338
FlashingColorOff, 5338
FlashingColorOn, 5340
FlashingEnabled, 5341
FlashingOnLimitViolation, 5342
FlashingRate, 5342
FlashTransparentColor, 5343
Flip, 5344
FocusColor, 5344
FocusWidth, 5345
Font, 5346
FontBold, 5347
FontItalic, 5348
FontName, 5349
FontSize, 5349
FontUnderline, 5350
ForeColor, 5351
ForeColorTransparency, 5352
FormatPattern, 5352
FormatPatternReadOnlySpecial, 5785
FormatType, 5692
Free, 5352
FreePercent, 5353
FreezeProviderConnections, 5786
GetSelectionText, 5786
Gradation, 5353
GraphDirection, 5354
GridBackColor, 5693
GridLineColor, 5354
GridlineFillColor, 5355
GridlineStyle, 5355
GridLineWidth, 5355
HeaderFont, 5356
Height, 5356
Help text, 5360

HiddenInput, 5361
HideTagNames, 5786
HighLimitColor, 5361
Hitlist, 5362
HitlistColumnAdd, 5362
HitlistColumnCount, 5363
HitlistColumnIndex, 5363
HitlistColumnName, 5364
HitlistColumnRemove, 5364
HitlistColumnRepos, 5365
HitlistColumnSort, 5365
HitlistColumnSortIndex, 5366
HitlistColumnVisible, 5366
HitlistDefaultSort, 5367
HitlistMaxSourceItems, 5367
HitlistMaxSourceItemsWarn, 5368
HitlistRelTime, 5369
HitlistRelTimeFactor, 5368
HitlistRelTimeFactorType, 5369
HorizontalAlignment, 5370
HorizontalGridLines, 5371
HorizontalScrollBarPosition, 5371
Hotkey, 5372
HourNeedleHeight, 5372
HourNeedleWidth, 5372
IconSpace, 5373
Index, 5373
InnerBackColorOff, 5374
InnerBackColorOn, 5375
InputValue, 5375
InsertData(i), 5741
InsertEnable, 5694
IntegerDigits, 5376
Interval, 5376
ItemBorderStyle, 5377
JumpToLimitsAfterMouseClick, 5377
LabelColor, 5378
Language, 5378
LargeTicksBold, 5379
LargeTicksSize, 5379
LastConnectedObjectIndex, 5380
LastConnectedObjectName, 5380
LeaveMarginForBorder, 5787
LeaveMarginForMarkers, 5788
Left, 5386
LeftOffset, 5387
Limit4LowerLimit, 5387
Limit4LowerLimitEnabled, 5388
Limit4LowerLimitRelative, 5389
Limit4LowrLimitColor, 5388
Limit4UpperLimit, 5390
Limit4UpperLimitColor, 5390
Limit4UpperLimitEnabled, 5391
Limit4UpperLimitRelative, 5391
Limit5LowerLimit, 5392
Limit5LowerLimitColor, 5392
Limit5LowerLimitEnabled, 5393
Limit5LowerLimitRelative, 5394
Limit5UpperLimit, 5394
Limit5UpperLimitColor, 5395
Limit5UpperLimitEnabled, 5395
Limit5UpperLimitRelative, 5396
LineColor, 5396
LineEdShapeStyle, 5397
LineWidth, 5398
LoadDataImmediately, 5399
LocalCursor, 5400
Localizable, 5749
LockSquaredExtent, 5400
LogOperation, 5401
LongTermArchiveConsistency, 5401
LowerLimit, 5402
LowerLimitColor(i), 5698
LowerLimitValue(i), 5699
LowLimitColor, 5402
MachineName, 5403
MarginToBorder, 5403
MaximumValue, 5404
MenuToolBarConfig, 5404
MessageBlockAlignment, 5405
MessageBlockAutoPrecisions, 5405
MessageBlockCaption, 5406
MessageBlockCount, 5406, 5409
MessageBlockDateFormat, 5407
MessageBlockExponentialFormat, 5407
MessageBlockFlashOn, 5408
MessageBlockHideText, 5408
MessageBlockHideTitleText, 5409
MessageBlockID, 5409
MessageBlockLeadingZeros, 5410
MessageBlockLength, 5410
MessageBlockName, 5411
MessageBlockPrecisions, 5411
MessageBlockSelected, 5412
MessageBlockShowDate, 5412
MessageBlockShowIcon, 5413
MessageBlockShowTitleIcon, 5413
MessageBlockTextId, 5414
MessageBlockTimeFormat, 5414
MessageBlockType, 5415
MessageColumnAdd, 5415
MessageColumnCount, 5416
MessageColumnIndex, 5416
MessageColumnName, 5417

MessageColumnRemove, 5417
MessageColumnRepos, 5418
MessageColumnSort, 5418
MessageColumnSortIndex, 5419
MessageColumnVisible, 5419
MessageListType, 5420
MinimumValue, 5420
MinuteNeedleHeight, 5421
MinuteNeedleWidth, 5421
Mode, 5422
Moveable, 5423
MsgFilterSQL, 5423
NeedleBorderColor, 5426
NeedleColor, 5427
NeedleFillStyle, 5428
NormalRangeColor, 5428
NormalRangeVisible, 5429
NumberOfValues(i), 5788
ObjectName, 5704
OCXState, 5432
Online, 5432
OperationSteps, 5433
OperatorMessageID, 5433
OperatorMessageIndex, 5434
OperatorMessageName, 5435
OperatorMessageNumber, 5435
OperatorMessageSelected, 5436
OperatorMessageSource1, 5436
OperatorMessageSource10, 5441
OperatorMessageSource2, 5437
OperatorMessageSource3, 5437
OperatorMessageSource4, 5438
OperatorMessageSource5, 5438
OperatorMessageSource6, 5439
OperatorMessageSource7, 5439
OperatorMessageSource8, 5440
OperatorMessageSource9, 5441
OperatorMessageType1, 5442
OperatorMessageType10, 5446
OperatorMessageType2, 5442
OperatorMessageType3, 5443
OperatorMessageType4, 5443
OperatorMessageType5, 5444
OperatorMessageType6, 5444
OperatorMessageType7, 5445
OperatorMessageType8, 5445
OperatorMessageType9, 5446
Orientation, 5706
PageMode, 5447
PageModeMessageNumber, 5447
Parent, 5789
Password, 5448
PersistentRTAuthorization, 5707
PersistentRTCSAuthorization, 5749
PersistentRuntime, 5708
PersistentRuntimeAuthorization, 5709
PersistentToDownload, 5750
PersistentToDownloadAuthorization, 5750
Picture, 5449
PictureAlignment, 5449
PictureDeactivated, 5450
PictureName, 5711
PictureOff, 5450
PictureOn, 5451
PointerColor, 5452
PointsCount, 5452
Precision, 5453
Pressed, 5453
PrintConfiguration, 5752
PrintJobName, 5712
PrintVisibleColumnsOnly, 5712
ProcessValue, 5454
ProjectPath, 5752
Radius, 5456
RadiusHeight, 5457
RadiusWidth, 5457
ReadOnly, 5713
RecipeName, 5458
RecipeNumber, 5458
RecordName, 5459
RecordNumber, 5459
RelativeFillLevel, 5459
Rotation, 5460
RotationAngle, 5461
RotationCenterLeft, 5461
RotationCenterTop, 5462
RoundCornerHeight, 5463
RoundCornerWidth, 5463
RowNumber, 5791
RowScrollbar, 5463
RowSizingEnable, 5753
RowTitles, 5496
RTPersistence, 5464
RTPersistencePasswordLevel, 5714
RTPersistenceType, 5465
RulerColor, 5466
RulerPrecision(i), 5791
RulerXPrecision(i), 5792
RulerYPrecision(i), 5792
ScaleColor, 5467
ScaleDenominator, 5471
ScaleGradation, 5467
ScaleLabelColor, 5468
ScaleNumerator, 5468

ScalePosition, 5468
ScaleTickColor, 5469
ScaleTickLabelPosition, 5470
ScaleTickLength, 5470
ScaleTickPosition, 5471
Scaling, 5472
ScalingType, 5472
ScreenItems, 5473
ScreenName, 5473
Screens, 5474
Scrollable, 5753
SecondNeedleHeight, 5474
SecondNeedleWidth, 5475
SegmentColoring, 5475
SelectBackColor, 5476
SelectedCellColor, 5476
SelectedCellForeColor, 5477
SelectedIndex, 5478
SelectedRowColor, 5478
SelectedRowForeColor, 5479
SelectedTitleColor, 5480
SelectedTitleForeColor, 5480
SelectForeColor, 5481
SelectionBackColor, 5482
SelectionColoring, 5483
SelectionForeColor, 5483
SelectionMode, 5716
SelectionRect, 5484
SelectionRectColor, 5484
SelectionRectWidth, 5485
SelectionType, 5486
SeparatorBackColor, 5486
SeparatorColor, 5487
SeparatorStyle, 5488
SeparatorWidth, 5488
SeperatorCornerStyle, 5488
ServerNames, 5489
ServerPrefix, 5490
ServerScale, 5489
SetpointTrendArchiveStartId(i), 5793
SetpointTrendColor(i), 5794
SetpointTrendNumberOfValues(i), 5794
Shared, 5490
ShareTimeColumn, 5796
ShareValueAxis, 5796
ShareXAxis, 5797
ShareYAxis, 5797
ShiftDecimalPoint, 5490
ShortenCellText, 5717
ShortenColumnHeaderText, 5717
ShowAlarmsFromDate, 5490
ShowBadTagState, 5491
ShowBar, 5491
ShowBorder, 5798
ShowCaption, 5492
ShowColumn(i), 5798
ShowCurve(i), 5798
ShowDecimalPoint, 5492
ShowFillLevel, 5493
ShowFocusRectangle, 5494
ShowInputControls, 5799
ShowLargeTicksOnly, 5494, 5697
ShowLimitMarkers, 5494
ShowMainFrame, 5799
ShowMessagesAtDate, 5754
ShowOverflowIndicator, 5800
ShowPeakValuePointer, 5495
ShowPosition, 5495
ShowRuler, 5496
ShowRulerInAxis, 5497
ShowScale, 5497
ShowScrollbars, 5498
ShowSetpointTrend(i), 5800
ShowSortButton, 5498
ShowSortIcon, 5499
ShowStatusBar, 5500
ShowTableGridlines, 5500
ShowThumb, 5501
ShowTickLabels, 5501
ShowTicks, 5502
ShowTitle, 5502
ShowToolBar, 5503
ShowTrendIcon, 5503
ShowTrendIndicator, 5504
ShowVerticalGridlines, 5718
ShowXValuesExponential(i), 5718
ShowYValuesExponential(i), 5719
SmartTags, 5505
Sort, 5755
SortByTimeDirection, 5506
SortByTimeEnable, 5720
SortByTimeEnabled, 5506
SortOnColumnHeaderClick, 5720
SortSequence, 5506
SortTimeAscending, 5721
SortTimeEnable, 5721
StartAngle, 5507
StartLeft, 5722
StartStyle, 5508
StartTop, 5508
StartValue, 5508
StatusbarBackColor, 5509
StatusbarElementAdd, 5510
StatusbarElementAutoSize, 5510

StatusbarElementCount, 5511
StatusbarElementIconId, 5512
StatusbarElementID, 5512
StatusbarElementIndex, 5513
StatusbarElementName, 5514
StatusbarElementRemove, 5514
StatusbarElementRename, 5515
StatusbarElements, 5515
StatusbarElementTooltipText, 5516
StatusbarElementUserDefined, 5516
StatusbarElementVisible, 5517
StatusbarElementWidth, 5517
StatusbarFontColor, 5518
StatusbarShowArchiveName, 5801
StatusbarShowColumn, 5801
StatusbarShowRecord, 5802
StatusbarShowRow, 5802
StatusbarShowText, 5803
StatusbarShowTooltips, 5519
StatusbarText, 5519
StatusbarUseBackColor, 5520
StatusbarVisible, 5521
Style, 5521
StyleSettings, 5522, 5735
SwapDimensionsWithOrientation, 5803
SwapFirstWithLastConnection, 5522
TableBackColor, 5523
TableColor, 5524
TableColor2, 5524
TableFocusOnButtonCommand, 5755
TableForeColor, 5525
TableForeColor2, 5526
TableGridLineColor, 5526
TableHeaderBackColor, 5527
TableHeaderForeColor, 5528
TagPrefix, 5528
Template, 5530
Text, 5531
TextList, 5532
TextOff, 5532
TextOn, 5533
TextOrientation, 5534
ThumbBackColor, 5534
TicksColor, 5535
TickStyle, 5536
TimeAxisBeginTime(i), 5536
TimeAxisEndTime, 5537
TimeAxisLabel(i), 5537
TimeAxisRange, 5538
TimeAxisShowGridLines(i), 5722
TimeAxisShowLargeIncrements(i), 5804
TimeAxisShowSmallIncrements(i), 5804
TimeAxisTimeFormat(i), 5538
TimeBase, 5538
TimeColumnActualize, 5539
TimeColumnAdd, 5540
TimeColumnAlignment, 5540
TimeColumnAlignment(i), 5541
TimeColumnBackColor, 5541
TimeColumnBeginTime, 5542
TimeColumnCaption, 5542
TimeColumnCount, 5543
TimeColumnDateFormat, 5543
TimeColumnEndTime, 5544
TimeColumnForeColor, 5544
TimeColumnFormat(i), 5723
TimeColumnHideText, 5545
TimeColumnHideTitleText, 5545
TimeColumnIndex, 5546
TimeColumnLength, 5546
TimeColumnMeasurePoints, 5546
TimeColumnName, 5547
TimeColumnRangeType, 5547
TimeColumnRemove, 5548
TimeColumnRename, 5548
TimeColumnRepos, 5549
TimeColumnShowDate, 5549
TimeColumnShowIcon, 5550
TimeColumnShowTitleIcon, 5550
TimeColumnSort, 5551
TimeColumnSortIndex, 5551
TimeColumnTimeFormat, 5552
TimeColumnTimeRangeBase, 5552
TimeColumnTimeRangeFactor, 5553
TimeColumnUseValueColumnColors, 5553
TimeColumnVisible, 5554
TimeJumpColor(i), 5805
TimeJumpEnabled(i), 5805
TimeOverlapColor(i), 5806
TimeOverlapEnabled(i), 5806
TimeRangeBase(i), 5755
TimeRangeFactor(i), 5756
TitleColor, 5724
TitleCut, 5555
TitleDarkShadowColor, 5556
TitleForeColor, 5557
TitleGridLineColor, 5557
TitleLightShadowColor, 5558
TitleSort, 5559
TitleStyle, 5559
Toggle, 5560
Tolerance, 5560
ToleranceColor, 5561
ToleranceLowerLimit, 5561

ToleranceLowerLimitColor, 5562
ToleranceLowerLimitEnabled, 5563
ToleranceLowerLimitRelative, 5563
ToleranceUpperLimit, 5564
ToleranceUpperLimitColor, 5564
ToleranceUpperLimitEnabled, 5565
ToleranceUpperLimitRelative, 5565
ToolbarAlignment, 5566
ToolbarBackColor, 5567
ToolbarButtonActive, 5567
ToolbarButtonAdd, 5568
ToolbarButtonAuthorization, 5573
ToolbarButtonBeginGroup, 5568
ToolbarButtonCount, 5569
ToolbarButtonEnabled, 5570
ToolbarButtonHotKey, 5570
ToolbarButtonID, 5571
ToolbarButtonIndex, 5571
ToolbarButtonLocked, 5572
ToolbarButtonName, 5573
ToolbarButtonRemove, 5574
ToolbarButtonRename, 5574
ToolbarButtonRepos, 5575
ToolbarButtonTooltipText, 5575
ToolbarButtonUserDefined, 5576
ToolbarShowTooltips, 5577
ToolbarUseBackColor, 5577
ToolbarUseHotKeys, 5578
ToolbarVisible, 5578
ToolTipText, 5579
Top, 5580
TopOffset, 5582
Total, 5583
Transparency, 5583
TransparentColor, 5585
TransparentColorDeactivatedPicture, 5585
TransparentColorPictureOff, 5586
TransparentColorPictureOn, 5587
TrendActualize, 5587
TrendAdd, 5588
TrendBeginTime, 5588
TrendColor, 5589
TrendCount, 5589
TrendEndTime, 5590
TrendExtendedColorSet, 5590
TrendFill, 5591
TrendFillColor, 5591
TrendIndex, 5592
TrendIndicatorColor, 5593
TrendLabel, 5593
TrendLineStyle, 5594
TrendLineType, 5594
TrendLineWidth, 5595
TrendLowerLimit, 5595
TrendLowerLimitColor, 5596
TrendLowerLimitColoring, 5596
TrendMeasurePoints, 5597
TrendName, 5597
TrendPointColor, 5598
TrendPointStyle, 5598
TrendPointWidth, 5599
TrendProvider, 5599
TrendRangeType, 5600
TrendRemove, 5601
TrendRename, 5601
TrendRepos, 5602
TrendSelectTagNameX, 5602
TrendSelectTagNameY, 5603
TrendTag, 5725
TrendTagNameX, 5603
TrendTagNameY, 5604
TrendTimeRangeBase, 5604
TrendTimeRangeFactor, 5605
TrendTrendWindow, 5605
TrendUncertainColor, 5606
TrendUncertainColoring, 5606
TrendUpperLimit, 5607
TrendUpperLimitColor, 5607
TrendUpperLimitColoring, 5608
TrendVisible, 5608
TrendWindowAdd, 5609
TrendWindowCoarseGrid, 5609
TrendWindowCoarseGridColor, 5610
TrendWindowFineGrid, 5611
TrendWindowFineGridColor, 5611
TrendWindowForegroundTrendGrid, 5612
TrendWindowGridInTrendColor, 5612
TrendWindowHorizontalGrid, 5613
TrendWindowIndex, 5613
TrendWindowName, 5614
TrendWindowRemove, 5614
TrendWindowRename, 5614
TrendWindowRepos, 5615
TrendWindowRulerColor, 5615
TrendWindowRulerLayer, 5616
TrendWindowRulerStyle, 5617
TrendWindowRulerWidth, 5617
TrendWindowSpacePortion, 5618
TrendWindowVerticalGrid, 5618
TrendWindowVisible, 5619
TrendXAxis, 5619
TrendYAxis, 5620
Type, 5807
UncertainStateColor(i), 5727

UncertainStateEnabled(i), 5728
Unit, 5620
UnitColor, 5621
UnitText, 5621
UnitTop, 5622
UpdateEnable, 5729
UpperLimit, 5622
UpperLimitColor(i), 5729
UpperLimitEnabled(i), 5730
UpperLimitValue(i), 5730
UseAllServers, 5731
UseBarBorderConstraints, 5808
Used, 5623
UseDesignColorSchema, 5623
UseDesignShadowSettings, 5625
UsedPercent, 5626
UseEffectiveProcessValue, 5808
UseExponentialFormat, 5627
UseFlashTransparentColor, 5627
UseGDI, 5809
UseMeasurePoints(i), 5809
UseMessageColor, 5628
UseMultipleLimits, 5810
UserArchiveNumberOfValues(i), 5812
UserArchiveStartId(i), 5813
UseScaleConstraints, 5810
UseScaledBarBorder, 5811
UseSelectedTitleColor, 5628
UseSeparateDiagrams, 5811
UseSimplePresicionOffset, 5812
UseTableColor2, 5629
UseTagLimitColors, 5629
UseTimeRange(i), 5757
UseTransparentColor, 5630
UseTransparentColorDeactivatedPicture, 5630
UseTransparentColorPictureOff, 5631
UseTransparentColorPictureOn, 5631
UseTrendNameAsLabel, 5632
UV_ColumnWidth_AKZ, 5632
UV_ColumnWidth_Descriptor, 5632
UV_ColumnWidth_InstallationDate, 5633
UV_ColumnWidth_LADDR, 5633
UV_ColumnWidth_Name, 5633
UV_ColumnWidth_OKZ, 5633
UV_ColumnWidth_OperationState, 5633
UV_ColumnWidth_OrderID, 5633
UV_ColumnWidth_ProfileID, 5633
UV_ColumnWidth_Rack, 5634
UV_ColumnWidth_Slot, 5634
UV_ColumnWidth_SoftwareRevision, 5634
UV_ColumnWidth_SpecificProfileData, 5634
UV_ColumnWidth_State, 5634
UV_ColumnWidth_Station, 5634
UV_ColumnWidth_SubAddress, 5634
UV_ColumnWidth_SubSlot, 5635
UV_ColumnWidth_SubSystem, 5635
UV_ColumnWidth_Type, 5635
UV_ShowItem_AKZ, 5635
UV_ShowItem_Descriptor, 5635
UV_ShowItem_InstallationDate, 5635
UV_ShowItem_LADDR, 5635
UV_ShowItem_Name, 5636
UV_ShowItem_OKZ, 5636
UV_ShowItem_OperationState, 5636
UV_ShowItem_OrderID, 5636
UV_ShowItem_ProfileID, 5636
UV_ShowItem_Rack, 5636
UV_ShowItem_Slot, 5636
UV_ShowItem_SoftwareRevision, 5637
UV_ShowItem_SpecificProfileData, 5637
UV_ShowItem_State, 5637
UV_ShowItem_Station, 5637
UV_ShowItem_SubAddress, 5637
UV_ShowItem_SubSlot, 5637
UV_ShowItem_SubSystem, 5637
UV_ShowItem_Type, 5638
ValidateFormatPattern, 5758
ValueAxisAutorange(i), 5639
ValueAxisBegin(i), 5758
ValueAxisDecimalPrecision(i), 5732
ValueAxisEnd(i), 5759
ValueAxisGridLineInterval(i), 5814
ValueAxisLabel(i), 5639
ValueAxisLargeIncrementsSize(i), 5815
ValueAxisScalingType(i), 5640
ValueAxisShowGridLines(i), 5733
ValueAxisShowLargeIncrements(i), 5815
ValueAxisShowSmallIncrements(i), 5816
ValueAxisSmallIncrementSize(i), 5816
ValueColumnAlignment(i), 5640
ValueColumnPrecision(i), 5734
VerticalAlignment, 5641
VerticalGridLines, 5642
VerticalScrollBarPosition, 5642
ViewOnly, 5643
Visible, 5643
Warning, 5645
WarningColor, 5646
WarningLowerLimit, 5646
WarningLowerLimitColor, 5647
WarningLowerLimitEnabled, 5648
WarningLowerLimitRelative, 5648
WarningRangeColor, 5649
WarningRangeStart, 5649

WarningRangeVisible, 5650
WarningUpperLimit, 5651
WarningUpperLimitColor, 5651
WarningUpperLimitEnabled, 5652
WarningUpperLimitRelative, 5652
Width, 5653
WindowCloseEnabled, 5655
WindowMaximizeEnabled, 5655
WindowMovingEnabled, 5656
WindowOnTop, 5656
WindowSizingEnabled, 5657
WindowsStyle, 5657
XAxisAdd, 5658
XAxisAlignment, 5658
XAxisAutoPrecisions, 5659
XAxisAutoRange(i), 5659
XAxisBegin(i), 5735
XAxisBeginValue, 5660
XAxisColor, 5660
XAxisCount, 5661
XAxisDecimalPrecision(i), 5817
XAxisEnd(i), 5736
XAxisEndValue, 5661
XAxisExponentialFormat, 5662
XAxisGridLineInterval(i), 5818
XAxisIndex, 5662
XAxisInTrendColor, 5663
XAxisLabel(i), 5663
XAxisLargeIncrementSize(i), 5818
XAxisMode(i), 5819
XAxisName, 5664
XAxisPrecisions, 5664
XAxisRemove, 5664
XAxisRepos, 5665
XAxisScalingType(i), 5665
XAxisShowGridLines(i), 5736
XAxisShowLargeIncrements(i), 5819
XAxisShowSmallIncrements(i), 5820
XAxisSmallIncrementSize(i), 5820
XAxisTrendWindow, 5666
XAxisVisible, 5666
XDataLogTag(i), 5821
XOnlineTag(i), 5821
YAxisAdd, 5667
YAxisAlignment, 5667
YAxisAutoPrecisions, 5668
YAxisAutoRange, 5668
YAxisBeginValue, 5669
YAxisColor, 5669
YAxisCount, 5670
YAxisEndValue, 5670
YAxisExponentialFormat, 5670
YAxisIndex, 5671
YAxisInTrendColor, 5671
YAxisLabel, 5672
YAxisName, 5672
YAxisPrecisions, 5673
YAxisRemove, 5673
YAxisRename, 5674
YAxisRepos, 5674
YAxisScalingType, 5675
YAxisTrendWindow, 5675
YAxisVisible, 5676
YDataLogTag(i), 5822
YOnlineTag(i), 5823
ZeroPoint, 5676
Zoomable, 5760
ZoomFactor, 5677
Properties (VBS)","Name, 5424
Properties (VBS)";"Object, 5429
Properties for slide-in screen
Configuring, 3912
Properties in VBS
Comment, 5771
ComputerName, 5302
Context, 5303
Instance, 5376
SelText, 5476
State, 5509
UserName, 5632
Properties in VBS/ScrollPositionX, 5715
Properties in VBS/ScrollPositionY, 5715
Properties in VBS:AngleAlpha, 5738
Properties in VBS:AngleBeta, 5739
Properties in VBS:Application, 5657, 5678
Properties in VBS:Axe, 5739
Properties in VBS:AxisSection, 5739
Properties in VBS:BackBorderWidth, 5680
Properties in VBS:BackColor2, 5764
Properties in VBS:BackColor3, 5680
Properties in VBS:BackFlashColorOff, 5680
Properties in VBS:BackFlashColorOn, 5680
Properties in VBS:Background, 5739
Properties in VBS:BarDepth, 5739
Properties in VBS:BarHeight, 5739
Properties in VBS:BarWidth, 5765
Properties in VBS:BaseX, 5740
Properties in VBS:BaseY, 5740
Properties in VBS:BorderColorBottom, 5766
Properties in VBS:BorderColorTop, 5766
Properties in VBS:BorderFlashColorOff, 5680
Properties in VBS:BorderFlashColorOn, 5680
Properties in
VBS:BottomConnectedConnectionPointIndex, 5681

Properties in VBS:BottomConnectedObjectName, 5681
Properties in VBS:BoxAlignment, 5681
Properties in VBS:BoxCount, 5681
Properties in VBS:BoxType, 5681
Properties in VBS:Button1Width, 5681
Properties in VBS:Button2Width, 5682
Properties in VBS:Button3Width, 5682
Properties in VBS:Button4Width, 5682
Properties in VBS:ButtonColor, 5682
Properties in VBS:CheckAlarmHigh, 5683
Properties in VBS:CheckAlarmLow, 5684
Properties in VBS:CheckLimitHigh4, 5684
Properties in VBS:CheckLimitHigh5, 5684
Properties in VBS:CheckLimitLow4, 5684
Properties in VBS:CheckLimitLow5, 5684
Properties in VBS:CheckToleranceHigh, 5685
Properties in VBS:CheckToleranceLow, 5685
Properties in VBS:CheckWarningHigh, 5685
Properties in VBS:CheckWarningLow, 5685
Properties in VBS:ClearOnNew, 5685
Properties in VBS:CloseButton, 5740
Properties in VBS:ColorAlarmHigh, 5686
Properties in VBS:ColorAlarmLow, 5686
Properties in VBS:ColorBottom, 5768
Properties in VBS:ColorChangeType, 5768
Properties in VBS:ColorLimitHigh4, 5686
Properties in VBS:ColorLimitHigh5, 5686
Properties in VBS:ColorLimitLow4, 5686
Properties in VBS:ColorLimitLow5, 5687
Properties in VBS:ColorToleranceHigh, 5687
Properties in VBS:ColorToleranceLow, 5687
Properties in VBS:ColorTop, 5768
Properties in VBS:ColorWarningHigh, 5687
Properties in VBS:ColorWarningLow, 5687
Properties in VBS:CurrentContext, 5308
Properties in VBS:EditAtOnce, 5690
Properties in VBS:Exponent, 5690
Properties in VBS:ExtendedOperation, 5783
Properties in VBS:ExtendedZoomingEnable, 5332
Properties in VBS:FillColor, 5690
Properties in VBS:Filling, 5691
Properties in VBS:FillingIndex, 5691
Properties in VBS:FillStyle2, 5692
Properties in VBS:FlashBackColor, 5692
Properties in VBS:FlashBorderColor, 5692
Properties in VBS:FlashFlashPicture, 5784
Properties in VBS:FlashForeColor, 5784
Properties in VBS:FlashPicReferenced, 5784
Properties in VBS:FlashPicture, 5784
Properties in VBS:FlashRate, 5692
Properties in VBS:FlashRateBackColor, 5692
Properties in VBS:FlashRateBorderColor, 5784
Properties in VBS:FlashRateFlashPic, 5785
Properties in VBS:FlashRateForeColor, 5785
Properties in VBS:ForeFlashColorOff, 5741
Properties in VBS:ForeFlashColorOn, 5741
Properties in VBS:Hysteresis, 5694
Properties in VBS:HysteresisRange, 5694
Properties in VBS:ItemBorderBackColor, 5695
Properties in VBS:ItemBorderColor, 5695
Properties in VBS:ItemBorderWidth, 5695
Properties in VBS:LanguageSwitch, 5787
Properties in VBS:Layer, 5382
Properties in VBS:Layer00Checked, 5742
Properties in VBS:Layer00Color, 5742
Properties in VBS:Layer00Value, 5742
Properties in VBS:Layer01Checked, 5742
Properties in VBS:Layer01Color, 5742
Properties in VBS:Layer01Value, 5743
Properties in VBS:Layer02Checked, 5743
Properties in VBS:Layer02Color, 5743
Properties in VBS:Layer02Value, 5743
Properties in VBS:Layer03Checked, 5743
Properties in VBS:Layer03Color, 5744
Properties in VBS:Layer03Value, 5744
Properties in VBS:Layer04Checked, 5744
Properties in VBS:Layer04Color, 5744
Properties in VBS:Layer04Value, 5744
Properties in VBS:Layer05Checked, 5745
Properties in VBS:Layer05Color, 5745
Properties in VBS:Layer05Value, 5745
Properties in VBS:Layer06Checked, 5745
Properties in VBS:Layer06Color, 5745
Properties in VBS:Layer06Value, 5746
Properties in VBS:Layer07Checked, 5746
Properties in VBS:Layer07Color, 5746
Properties in VBS:Layer07Value, 5746
Properties in VBS:Layer08Checked, 5746
Properties in VBS:Layer08Color, 5747
Properties in VBS:Layer08Value, 5747
Properties in VBS:Layer09Checked, 5747
Properties in VBS:Layer09Color, 5747
Properties in VBS:Layer09Value, 5747
Properties in VBS:Layer10Checked, 5748
Properties in VBS:Layer10Color, 5748
Properties in VBS:Layer10Value, 5748
Properties in VBS:LayerDeclutteringEnable, 5385
Properties in VBS:Layers, 5386
Properties in VBS:LeftComma, 5695
Properties in VBS:LightEffect, 5748
Properties in VBS:LimitHigh4, 5695
Properties in VBS:LimitHigh5, 5696
Properties in VBS:LimitLow4, 5696

Properties in VBS:LimitLow5, 5696
Properties in VBS:LimitMax, 5696
Properties in VBS:LimitMin, 5696
Properties in VBS:ListType, 5748
Properties in VBS:LockStatus, 5697
Properties in VBS:LockText, 5697
Properties in VBS:LockTextColor, 5697
Properties in VBS:Logging, 5400
Properties in VBS:LongStrokesBold, 5697
Properties in VBS:LongStrokesOnly, 5697
Properties in VBS:LongStrokesTextEach, 5698
Properties in VBS:Marker, 5700
Properties in VBS:Max, 5749
Properties in VBS:MaximizeButton, 5700
Properties in VBS:MCGUBackColorOff, 5700
Properties in VBS:MCGUBackColorOn, 5701
Properties in VBS:MCGUBackFlash, 5701
Properties in VBS:MCGUTextColorOff, 5701
Properties in VBS:MCGUTextColorOn, 5701
Properties in VBS:MCGUTextFlash, 5701
Properties in VBS:MCKOBackColorOff, 5701
Properties in VBS:MCKOBackColorOn, 5702
Properties in VBS:MCKOBackFlash, 5702
Properties in VBS:MCKOTextColorOff, 5702
Properties in VBS:MCKOTextColorOn, 5702
Properties in VBS:MCKOTextFlash, 5702
Properties in VBS:MCKQBackColorOff, 5702
Properties in VBS:MCKQBackColorOn, 5703
Properties in VBS:MCKQBackFlash, 5703
Properties in VBS:MCKQTextColorOff, 5703
Properties in VBS:MCKQTextColorOn, 5703
Properties in VBS:MCKQTextFlash, 5703
Properties in VBS:MCText, 5703
Properties in VBS:MessageClass, 5704
Properties in VBS:Min, 5788
Properties in VBS:NumberLines, 5704
Properties in VBS:ObjectSizeDeclutteringEnable, 5430
Properties in VBS:ObjectSizeDeclutteringMax, 5431
Properties in VBS:ObjectSizeDeclutteringMin, 5432
Properties in VBS:OnTop, 5705
Properties in VBS:OperationMessage, 5706
Properties in VBS:OperationReport, 5706
Properties in VBS:OutputFormat, 5707
Properties in VBS:OutputValue, 5707
Properties in VBS:Path, 5448
Properties in VBS:PicDeactReferenced, 5709
Properties in VBS:PicDeactTransparent, 5710
Properties in VBS:PicDeactUseTransColor, 5710
Properties in VBS:PicDownReferenced, 5751
Properties in VBS:PicDownTransparent, 5710
Properties in VBS:PicDownUseTransColor, 5710
Properties in VBS:PicReferenced, 5751
Properties in VBS:PicTransColor, 5710
Properties in VBS:PictureUp, 5711
Properties in VBS:PicUpReferenced, 5791
Properties in VBS:PicUpTransparent, 5711
Properties in VBS:PicUpUseTransColor, 5711
Properties in VBS:PicUseTransColor, 5711
Properties in VBS:PointCount, 5712
Properties in VBS:Position, 5751
Properties in VBS:PredefinedAngles, 5752
Properties in VBS:Process, 5713
Properties in VBS:QualityCode, 5455
Properties in VBS:ReferenceRotationLeft, 5713
Properties in VBS:ReferenceRotationTop, 5713
Properties in VBS:Relevant, 5714
Properties in VBS:RightComma, 5714
Properties in VBS:SameSize, 5714
Properties in VBS:ScaleTicks, 5715
Properties in VBS:ScrollBars, 5715
Properties in VBS:SelBGColor, 5715
Properties in VBS:SelTextColor, 5717
Properties in VBS:SignificantMask, 5720
Properties in VBS:Sizeable, 5504
Properties in VBS:SmallChange, 5801
Properties in VBS:TagName, 5722
Properties in VBS:Tags, 5530
Properties in VBS:TimeStamp, 5554
Properties in VBS:Titleline, 5724
Properties in VBS:ToleranceHigh, 5724
Properties in VBS:ToleranceLow, 5724
Properties in VBS:TopConnectedConnectionPointIndex, 5724
Properties in VBS:TopConnectedObjectName, 5725
Properties in VBS:Trend, 5725
Properties in VBS:TypeAlarmHigh, 5726
Properties in VBS:TypeAlarmLow, 5726
Properties in VBS:TypeLimitHigh4, 5726
Properties in VBS:TypeLimitHigh5, 5726
Properties in VBS:TypeLimitLow4, 5726
Properties in VBS:TypeLimitLow5, 5726
Properties in VBS:TypeToleranceHigh, 5727
Properties in VBS:TypeToleranceLow, 5727
Properties in VBS:TypeWarningHigh, 5727
Properties in VBS:TypeWarningLow, 5727
Properties in VBS:UnselBGColor, 5807
Properties in VBS:UnselTextColor, 5807
Properties in VBS:UpdateCycle, 5808
Properties in VBS:UserValue1, 5732
Properties in VBS:UserValue2, 5732
Properties in VBS:UserValue3, 5732
Properties in VBS:UserValue4, 5732
Properties in VBS:Value, 5638

Properties in VBS:WarningHigh, 5734
Properties in VBS:WarningLow, 5734
Properties in VBS:WindowBorder, 5735
Properties in VBS:ZeroPointValue, 5759
Properties of the VPN group, 745
Properties window, (See Inspector window), (See Inspector window)
Property list
    Dynamize, 4015
ProSave, 6752, 6990
    Installation, 79
Protect
    Function key with password, 4043
ProtectedAreaName object, 5141
Protection concept, 1209, 1212, 6097, 6183
    Introduction, 417
    Revoking access rights, 418
Protection level, 1208, 1209, 6097, 6182, 6184
    Revoking access rights, 418
Protocol, 706
Proxy
    Routing, 7757
Proxy ARP, 820
Proxy device, 1152, 1154
PRVREC, 3130
PT, 2513
PTC resistor, 1300
PTO, 100
Pull/plug interrupt (ET 200M), 1336
Pull/plug interrupt OB, 1225
Pulse, 2218, 2227, 2238, 2252, 2494, 2514, 2527, 2755, 2770
    Extended, 2241, 2253, 2517, 2528, 2773
Pulse extension, 1288
Pulse generator
    Enabling/disabling with CTRL_PWM, 3310
Pulse interface
    Principle, 7334
Pulse stretching, 1338
PUT, 3615
PUT / GET, 1212
PWM, 100

Q

Q
    PLC, 6652
Q address, 848
Q series, 6655
Q0xUDEH CPU
    PLC, 6654
QB, 1482
QD, 1482
QRY_CINT, 3220
QRY_DINT, 3232
QRY_TINT, 3228
Quantum, 6685
QuitHorn, 5902
QuitSelected, 5902
QuitVisible, 5903
QW, 1482
QX, 1482

R

R, 593, 2203, 2479
R_TRIG, 2216, 2492, 2752
Rack error OB, 1226
Rack or station failure, 1226
Racks, 558
    Inserting a module, 569
Radius, 4122, 4150, 4152
RADIUS, 1074
Radius property (VBS), 5456
RadiusHeight property (VBS), 5457
RadiusWidth property (VBS), 5457
RALRM, 3076
Range
    Value outside range, 2289, 2565
    Value within range, 2288, 2564
Range (... - ...)
    Graphics list, 4006
RangeLabelView object, 5142
RangeQualityView object, 5144
RCV_CFG, 3727
RCV_PTP, 3737
RCV_RST, 3738
RCVREC, 3128
RD_ADDR, 3364
RD_DPAR, 3208
RD_DPARA, 3210
RD_DPARM, 3213
RD_LGADR, 3369
RD_LOC_T, 2985
RD_REC, 3118
RD_SINFO, 3262
RD_SYS_T, 2983
RDREC, 3069
RE_TRIGR, 2413, 2693, 2918
Read continuously
    Tag, 4222
Read field, 2371, 2649
Read memory address, 2848
Read memory bit, 2850

Read PLC tags
String or character tags in expressions, 866
Tags of the type String and Character, 866
Read system time with TIME_TCK, 2997
READ_BIG, 2861
READ_DBL, 106, 3351
READ_ERR, 3240
READ_LITTLE, 2857
Readback data record, 1278, 1316, 1346
Readback data record 197, 1274, 1278, 1316, 1326, 1346
ReadFromArrayDB, 2357, 2635, 2838
ReadFromArrayDBL, 2361, 2639, 2842
Reading
A data record with RD_REC, 3118
Asynchronously- data record of a module with RD_DPARA, 3210
Data record of a module with RD_DPAR, 3208
from a data block in load memory with READ_DBL, 3351
Reading data
From DP standard slaves/PROFINET IO devices with DPRD_DAT, 3124
From remote CPU with GET, 3612
Reading out
IP address, 6562
LED status with LED, 3274
Local time with RD_LOC_T, 2985
Reading out a diagnostics buffer, 1376
Reading tags, 864
Read-only, 1208, 6184
ReadOnly property (VBS), 5713
ReadPLCMode, 4717
ReadTags, 5903
REAL, 290, 1925, 1983, 2052, 2104, 2141, 2165, 2185
REAL_TO_, 2052, 2141, 2185
Receiving data
on I-device with RCVREC, 3128
Receiving SMS messages (TC_RECV), 3836
Recipe, 858, 4423, 4425, 4426, 6797
Application example: Machine parameter assignment, 4425
Application example:Batch production, 4425
Basics, 4423, 4425
Configuration option, 4432
Creating new, 4456
Data Flow, 4429
Data record, 4426
Differences in TP 177A and OP 77A, 4434
Entry, 4426
Export format, 4681, 4881
Exporting, 4483, 6794
Exporting to CSV file, 3312
GMP settings, 7071
Importing, 4483, 6796
Importing into data block, 3313
Memory requirements, 7038, 7039
Recipe screen, 4439
Recipe view, 4439
Synchronizing recipe tag, 4481
Use, 4425
Use of text lists, 4438
Recipe data
Migration, 188
Recipe data record
change, 4481
Copy, 4480
Create on the HMI device, 4474, 4475, 4480
Creating new, 4456
Creating on the HMI device, 4474
Deleting, 4481
Export format, 4681, 4881
Exporting, 4483
Importing, 4483
Importing and exporting, 4431
load, 4481
Synchronizing, 4481
Transfer option, 4429
Transferring the project, 87
Use of text lists, 4438
Recipe data record change
Effects in Runtime, 7071
Recipe data record name
Writing to a tag, 4442, 4446, 4449
Recipe data record number
Writing to a tag, 4442, 4446, 4449
Recipe element
Creating new, 4456
Recipe list, 4440
Recipe name
Writing to a tag, 4442, 4446, 4449
Recipe number
Writing to a tag, 4442, 4446, 4449
Recipe report
application in reports, 4524
Layout in reports, 4524
Recipe screen, 4453
Automatic transfer, 4453
Configuring, 4466
Recipe tag, 4428
Synchronizing recipe value, 4453
Visual machine simulation, 4452

Recipe tag
Synchronizing, 4432, 4481
Teach-in mode, 4433
Recipe value, 4453
Automatic transfer in the recipe screen, 4453
Synchronizing in the recipe screen, 4453
Recipe view, 4155, 4440, 4444, 4477
Advanced, 4448
Application, 4153, 4477
Behavior with screen change, 4451
Column headers, 4160
Configurable events, 4442, 4447, 4450
Configuring, 4463, 4465
Display number, 4156, 4159
Displaying one recipe only, 4445, 4449
Displaying values only, 4442
Dynamic properties, 4443
Expanded, 4444
Label, 4159
Layout, 4154, 4158
Operating element, 4478
Operation, 4477
Operation using the function key, 4451
Operation with the keyboard, 4479
Operation with the mouse, 4478
Operator control, 4156
Operator controls, 4158
Recipe data record, 4427, 4428, 4451
Simple, 4440
Simple recipe view, 4155
Toolbar, 4154
Update, 4448
Updating, 4441, 4444
Using as a drop-down list, 4446
Using as selection field, 4450
RecipeExport, 3312
RecipeImport, 3313
RecipeName property (VBS), 5458
RecipeNumber property (VBS), 5458
Recipes
Export CSV file from ES, 4469
Import CSV file into ES, 4468
RecipeViewBack, 4749
RecipeViewClearDataRecord, 4744
RecipeViewGetDataRecordFromPLC, 4744
RecipeViewMenu, 4745
RecipeViewNewDataRecord, 4743
RecipeViewOpen, 4745
RecipeViewRenameDataRecord, 4748
RecipeViewSaveAsDataRecord, 4747
RecipeViewSaveDataRecord, 4747
RecipeViewSetDataRecordToPLC, 4746
RecipeViewShowOperatorNotes, 4749
RecipeViewSynchronizeDataRecordWithTags, 4748
ReconfigIOSystem, 3113
Reconfiguration via user program, 885, 1274, 1278, 1316, 1326, 1346
Reconfiguring IO systems, 3113
Reconnecting
Tag, 4216
Reconnecting a PLC tag, 4216
Recording a log file for the modem, 7717
RecordName property (VBS), 5459
RecordNumber property (VBS), 5459
Rectangle
Configuring, 3987
Inserting, 3989
Inserting and configuring, 3989
Radius corners X, 4153
Radius corners Y, 4153
Rectangle object, 5151
Redoing actions
Basics of redoing actions, 446
Redoing actions, 449
Reference channel, 1289
Reference channel error, 1331
Reference channel of the module, 1333
Reference junction, 1099, 1251, 1266, 1289, 1292, 1332, 1333
Reference language, 454, 6823
Selecting, 6827
Reference object
Defining, 3966, 4510
Reference project
Basics, 390
Closing, 390
Comparing, 391
Opening, 390
Reference temperature, 1289, 1292, 1332, 1333
References, 226
Referencing
Object, 4582
Reinitialization of technology objects, 7456
RelativeFillLevel property (VBS), 5459
release, 4852
Release
User data type, 4244
Release button, 6969
Configuring, 6971
Release key, 4854
ReleaseButton, 4753
Remainder of division, 2308, 2584
Remote access user, 693

Remote connection
as CPU-controlled remote connection, 7709
as dial-up connection, 7672
As VPN connection, 7696
Remote connection cannot be established, 7695
Remote connection from the TS Adapter is not established, 7720
Remote connection to the TS Adapter is not established, 7718
Remote control
By means of Internet Explorer, 7114
By means of SmartClient application, 7115
Configure SmartClient, 7106
Configure SmartServer, 7102
Devices with keys, 7111
Direct keys, 7118
Monitoring mode, 7101
Session Management, 7101
Smart Options, 7113
SmartClient display, 7118
Remote desktop, 44
Remote monitoring
Smart Options, 7113
Remote system access to a programming device/ personal computer, 7710
Remove
Formatting in the alarm text, 4306
Object from the group, 3981
Removing, 128
Add-on, 128
Folder link, 3974
Rename
Customized VB function, 4588
Screen, 3897
Tag, 4212
Template, 3903
Renewing the CA group certificate, 750
REPEAT, 2906
Replace
Cross-references, 6821
HMI device, 6863
REPLACE, 3051
Replace modules during operation, 1336
Replacing
Module, 574
Replacing devices, 574
Function key, 6852
K-key, 6853
Limitations on connections, 6850
Principle, 6850
Report
Adding a detail page, 4500
Back page, 4495
Creating, 4499
creating (overview), 4497
Deleting page, 4500
Detail page, 4496
Device dependency, 6072
Event-driven output, 4501
Footer, 4496
Header, 4496
Maximum configurable number of pages, 4498
output to file, 4501
print, 4501
printing, 4516
Showing and hiding sections, 4500
Size and position of objects, 4498
Sorting pages, 4500
Steps for the output configuration, 4501
Time-driven output, 4501
Title page, 4495
Reporting
Audit Trail, 7054
enabling, 4332
Reports
Alarm report, 4522
Date/time field, 4518
Graphic I/O field, 4521
I/O field, 4518
Page number, 4525
Recipe report, 4524
Symbolic I/O field, 4526
Requirements for establishing a remote connection, 7694
Requirements for sending an SMS, 7713
Requirements for sending emails, 7714
Reset, 972
Bit field, 2206, 2482
IEC timer, 2235, 2767
Operand, 2203, 2208, 2479, 2484
to factory settings, 6994
To factory settings, 1386, 1387
Reset timer, 2511
RESET_BF, 2206, 2482
ResetBit, 4750, 4914
ResetBitInTag, 4751, 4915
ResetTagToHandWheel, 4799
Resetting
HMI device to factory settings, 6996
to factory settings, 6994
Resetting the connection, 3693
Resetting the MPI/PROFIBUS settings, 7617
Resetting the TCP/IP settings, 7609
Resetting the user interface layout, 338

Resistance thermometers, (see Thermal resistor)
Resizing
Faceplate, 4077
Resources, 1830
Code work memory, 1830
Data work memory, 1830
Introduction, 1830
Load memory, 1830
Retentive memory, 1830
Retentive memory data, 1830
Work memory, 1830
Resources tab
Structure, 1832
Response to resizing
Faceplate type, 4077
Restart, 972
Restart of technology objects, 7456
Restore
Data of the HMI device, 6991, 6993
Restore window layout, 336
Restoring a backup, 7597, 7601
Restoring a device, 7597, 7601
Restoring data
HMI device, 6991, 6993
Restriction
Modicon Modbus RTU, 6701
Modicon Modbus TCP/IP, 6701
Restrictions due to user rights, 349, 350
Recognizing, 349
RET, 2406, 2686
Retain, 1421
Retentive behavior
Bit memory, timers, counters, 1489
Retentive bit memories
Enabling the display, 1818
Retentive memory, 1830
Retentivity, 1173, 1419
Bit memory, timers, counters, 1488
Block interface, 1568
Data block, 1716, 1717
PLC tag, 1488, 1489
Retentivity of IP address parameters, 1124
Retrieving projects, 389
Return, 101
RETURN, 2910
RFID tag, 5946
RIGHT, 3043
Ring buffer, 270
Ring port, 6594, 6597
Ring Redundancy, 1025
RIP
RIPv2 statistics, 966
RIPv2
Configuration, 1071
Interfaces, 1072
RLO
Invert, 2201, 2477
RMON
History, 1050
Statistics, 1049
ROL, 2445, 2726, 2943
Role name, 694
Roles, 692
System-defined, 693
User-defined, 693
Root certification authorities, 688
Root, square, 2809
ROR, 2443, 2724, 2941
Rotate
Left, 2445, 2726, 2943
Object, 3939
Right, 2443, 2724, 2941
Rotation property (VBS), 5460
RotationAngle property (VBS), 5461
RotationCenterLeft property (VBS), 5461
RotationCenterTop property (VBS), 5462
Round, 2385, 2386, 2388, 2389, 2664, 2666, 2667, 2668, 2880, 2881, 2882
ROUND, 2385, 2664, 2880
RoundButton object, 5152
RoundCornerHeight property (VBS), 5463
RoundCornerWidth property (VBS), 5463
Routed
Connection, 6029
Routed connection
Configuring, 7761
Router, 596
Router IP address, 802
Routing, 932, 6745
Routing table, 959
Static routes, 932
VRRP, 932
Routing connection
Configuring, 6576, 7757
Routing mode, 723
RowNumber property (VBS), 5791
RowScrollbar property (VBS), 5463
RowSizingEnable property (VBS), 5753
RowTitles property (VBS), 5496
RS, 2208, 2484
RS-232/PIP multi-master cable, 1202
RT, 2235, 2511, 2767
RT class, 1110, 6583
RT_INFO, 3270

RTM, 1957, 2999
RTPersistence property (VBS), 5464
RTPersistencePasswordLevel property (VBS), 5714
RTPersistenceType property (VBS), 5465
RulerColor property (VBS), 5466
RulerPrecision(i) property (VBS), 5791
RulerXPrecision(i) property (VBS), 5792
RulerYPrecision(i) property (VBS), 5792
Rules for configuring MPI networks
    Rules for assigning the MPI address, 589
Rules for network configuration for PROFIBUS networks
    Assigning device addresses, 589, 590
RUN, 1168
Rung
    Deleting, 1619
    Inserting, 1619
    Use, 1617
Runtime, 5962, 5964
    Changing object property, 4604
    Effective range, 5962
    Effective range (RFID), 5964
    Executing user-defined functions, 4601
    Execution of the function list, 4600
    Language switching, 4604
    Measuring with RT_INFO, 3270
    Mouse wheel, 84
    Simulating, 6895
    Start screen, 3898
    starting on a panel, 6904, 6952
    starting on PC, 6952
RUNTIME, 2422, 2702, 2927
    Measure program runtime, 2422, 2702, 2927
Runtime data
    Migration, 188
Runtime error, 4593
Runtime language, 6823, 6840
    Font, 6844
    Log, 4264, 4348, 6845
    Order for language switching, 6843
    Selecting, 6841
Runtime scripting, 4569
    Function, 4569
    System function, 4569
    User-defined function, 4569
Runtime settings, 7085
    Communication, 6558
    User administration, 3899, 4533, 4534, 4547, 4553, 4555
Runtime start
    Basic Panel, 6903
Runtime Stop, 4854
Runtime version, 6894

S

S, 2204, 2480
S_AVERZ, 2249
S_CD, 2269, 2547
S_COMP, 3003
S_CONV, 101, 3005
S_CU, 2268, 2545, 2792
S_CUD, 2549
S_EVERZ, 2244
S_MOVE, 3001
S_ODT, 2244, 2520, 2776
S_ODTS, 2247, 2522, 2778
S_OFFDT, 2249, 2525, 2781
S_PEXT, 2241, 2517, 2773
S_PULSE, 2238, 2514, 2770
S_SEVERZ, 2247
S5TIME, 1929, 1986, 2058, 2167, 2187
S5TIME_TO_, 2058, 2187
S7 1200
    Communication, 6162
    Connection parameters, 6241
    HMI connection, 6163
    Parameters, 6241
    PROFIBUS, 6189, 6192, 6194, 6197
    PROFIBUS parameters, 6200, 6201, 6203, 6415, 6417, 6418
    PROFINET, 6162, 6166
S7 1500
    Communication, 6079
    Connection parameters, 6149, 6457
    Data types, 6145, 6380
    HMI connection, 6081
    Parameters, 6149, 6457
    PC, 6085
    PROFIBUS, 6104, 6105, 6107, 6109, 6111
    PROFIBUS parameters, 6114, 6115, 6117
    PROFINET, 6080, 6083
    PROFINET parameters, 6091, 6094, 6348, 6349, 6351, 6399, 6402
S7 200
    Communication, 6467
    Connection, 6467, 6502
    Connection parameters, 6469
    Data types, 6500
    MPI, 6334, 6473
    Parameters, 6334, 6469, 6473, 6475
    PPI, 6475
S7 300
    Communication, 6249

MPI address, 6298
Panel, 6054
S7 300/400
Connection parameters, 6330
Data types, 6324
HMI connection, 6250
MPI, 6285, 6287, 6289, 6291
MPI parameters, 6294, 6295, 6297
Parameters, 6330
PC, 6257
PROFIBUS, 6270, 6271, 6273, 6275
PROFIBUS parameters, 6280, 6281, 6283
PROFINET, 6249, 6252
PROFINET parameters, 6260, 6261, 6263
S7 400
Communication, 6249
MPI address, 6298
Panel, 6054
S7 communication
Parameter, 3609
Receive data from a remote partner instruction with BRCV, 3623
Receive data from a remote partner instruction with URCV, 3619
Send and receive parameters, 3611
Send data to a remote partner instruction with BSEND, 3621
Send data to a remote partner instruction with USEND, 3618
User data size, 3611
S7 connection, 603
TSAP, 614
S7 CPU
Load memory, 1171
Operand area, 1172
Work memory, 1172
S7 diagnostic alarm
enabling, 4311
S7 diagnostic alarms, 4285
S7 routing, 95
Configuring, 7761
Loading, 6901
via IE/PB Link, 95
S7-1200, 93
Panel, 6054
PROFIBUS, 6190, 6408, 6413
S7-1200 modules, 93
S7-1200 V2
Data types, 4205, 6235
S7-1500
PROFINET parameters, 6092, 6400
S7-CP, 678
S7-PCT, 1146
SA lifetime, 747
SafelyRemoveHardware, 4686, 4916
Safety guidelines, 359
Safety instruction
Changing the recipe data record in the background, 4441
Direct key, 6976
Recipe data record in background, 4444, 4448
Safety precautions when forcing tags, 1883, 1899
Sampling rate (high-speed analog modules), 1299
Save
Data of the HMI device, 6991, 6993
Global library, 6780
Save window layout, 335
SaveDataRecord, 4786, 4917
Saving a force table, 1887
Saving a watch table, 1860
Saving phone books, 7667
Saving the user interface
Saving the window layout, 335
SC, 2273, 2551
Scalable measuring range, 1303
SCALANCE S, 677
Scale, 2390, 2395, 2670, 2674, 2885, 2894
SCALE, 2395, 2674, 2894
SCALE_X, 2390, 2670, 2885
ScaleColor property (VBS), 5467
ScaleDenominator property (VBS), 5471
ScaleGradation property (VBS), 5467
ScaleLabelColor property (VBS), 5468
ScaleNumerator property (VBS), 5468
ScalePosition property (VBS), 5468
ScaleTickColor property (VBS), 5469
ScaleTickLabelPosition property (VBS), 5470
ScaleTickLength property (VBS), 5470
ScaleTickPosition property (VBS), 5471
Scaling
Applying linear scaling to a tag, 4222, 4224
Scaling property (VBS), 5472
ScalingType property (VBS), 5472
Scan operand for negative signal edge, 2210, 2486
Scan operand for positive signal edge, 2209, 2485
Scan RLO for negative signal edge, 2215, 2491
Scan RLO for positive signal edge, 2214, 2490
Scheduler, 5977
Acyclic triggers, 5982, 5983
Administer task, 5986
Cyclic triggers, 5982, 5984, 5991
Deactivated task, 5981
Event trigger, 5981, 5982, 5985, 5989
Function list, 5980

Timers for cyclic and acyclic triggers, 5983
Triggers, 5981
Work area, 5978
SCL, 99, 1659, 3881, 3882
Detailed comparison, 1766
Insert comment, 2911
Programming window, 1670
SCL instruction
Changing the data type, 1680, 1681
Data type basics, 1679
Inserting, 1677, 1678
Rules, 1676
Selecting, 1695
SCL program
Inserting a block call, 1684, 1685, 1687, 1688, 1689, 1690
Inserting an instruction, 1677, 1678
Inserting comments, 1694
Program status display, 1850
Selecting an instruction, 1695
Showing and hiding the parameter list, 1693
Screen
Availability for specific devices, 3891
Copy, 3897
Copying, 6868
Creating, 3896
Creating a type, 6790
Deleting, 3897
Font, 3892
Inserting, 3897
Library, 6790
Move, 3897
Rename, 3897
Use template, 3907
Working steps in creating, 3895
Zooming, 3895
Screen change, 4849
Displaying the screen on entering a zone, 5956, 5965, 5968
Screen keyboard
Basic Panels, 6909
Basic Panels 2nd Generation, 6910
Language behavior, 82
RT Advanced, 6965
Screen number, 5999
Screen object, 4976, 5000
Dynamic movement, 4021
Dynamization of the control enable state, 4024
Screen object (list), 4975
Screen template, 4039
ScreenItem, 5107
Control, 5072
ScreenItem object, 4978, 5003
ScreenItems object (list), 4979, 5005
ScreenItems property (VBS), 5473
ScreenName property (VBS), 5473
ScreenObjectCursorDown, 4666
ScreenObjectCursorLeft, 4667
ScreenObjectCursorRight, 4668
ScreenObjectCursorUp, 4666
ScreenObjectPageDown, 4668
ScreenObjectPageUp, 4669
Screens property (VBS), 5474
ScreenWindow object, 5155
Script
Creating a type, 6790
Library, 6790
Migrating, 184
Updating tag value, 4202, 6036
Scrollable property (VBS), 5753
Scrolling
Multi-touch devices, 6966
Scrolling back key, 6914, 6973
Scrolling up key, 6914, 6974
SD, 2253, 2255, 2530
SD card, 45, (See Memory card)
SE, 2253, 2528
Search
Additional options for searching in the editor, 450
Basics of searching, 450
Find and replace in the editor, 453
Find and replace palette, 451
Hardware catalog, 557
Replacing search terms in the editor, 453
SecondNeedleHeight property (VBS), 5474
SecondNeedleWidth property (VBS), 5475
Security, 98
Security Configuration Tool
Configuration views, 679
Security events, 765, 767
Security mode, 7165
Security module, 678
Security online diagnostics of S7 CPs, 97
Security program, 120, 124
Security settings, 981
SEG, 2467, 2748, 2967
Segment diagnostics
Graphic display, 1404
Icons, 1404
Text display, 1404
SegmentColoring property (VBS), 5475
Segmented circular log, 4339
SEL, 2431, 2712, 2931

Select, 2431, 2712, 2931
Multiple objects, 3966, 4509
Select alarm classes
Alarm view, 4140
SelectBackColor property (VBS), 5476
SelectedCellColor property (VBS), 5476
SelectedCellForeColor property (VBS), 5477
SelectedIndex property (VBS), 5478
SelectedRowColor property (VBS), 5478
SelectedRowForeColor property (VBS), 5479
SelectedStatisticArea, 5916
SelectedTitleColor property (VBS), 5480
SelectedTitleForeColor property (VBS), 5480
SelectForeColor property (VBS), 5481
Selecting
Multiple objects, 3939
Selection changed, 4847
Selection list
Open, 6914, 6974
SelectionBackColor property (VBS), 5482
SelectionColoring property (VBS), 5483
SelectionForeColor property (VBS), 5483
SelectionMode property (VBS), 5716
SelectionRect property (VBS), 5484
SelectionRectColor property (VBS), 5484
SelectionRectWidth property (VBS), 5485
SelectionType property (VBS), 5486
Send buffer, 907
Send clock cycle, 1131
SEND_CFG, 3725
SEND_PTP, 3735
SendEMail, 4760, 4918
Sending SMS messages (TC_SEND), 3832
SeparatorBackColor property (VBS), 5486
SeparatorColor property (VBS), 5487
SeparatorStyle property (VBS), 5488
SeparatorWidth property (VBS), 5488
SeperatorCornerStyle property (VBS), 5488
Sequence of columns, 4172
Sequencer, 2447, 2728, 2945
Serial number, 942
Serialize, 2341, 2618, 2822
Server
Configuring, 6558
HMI device, 6558
ServerExport, 5918
ServerImport, 5918
ServerNames property (VBS), 5489
ServerPrefix property (VBS), 5490
ServerScale property (VBS), 5489
Service & Support, 6017
Service data
Saving, 1381
Service groups, 707, 710
Service-pages, 7130
Create own, 7137
Display, 7133
remote control, 7133
Transfer, 7138
Servo motor, 7329
Servo OB, 7340, 7341
Session Management, 7101
Setting, 7102
Set
Bit field, 2205, 2481
Counter start value, 2273, 2551
Operand, 2204, 2207, 2480, 2483
Set limit value, 2317, 2593, 2806
Set operand on negative signal edge, 2213, 2489
Set operand on positive signal edge, 2212, 2488
Set the services
SmartClient, 7085
SmartServer, 7085
SET_BF, 2205, 2481
SET_CINT, 3219
SET_TIMEZONE, 2989
SET_TINT, 3222
SET_TINTL, 3224
SetAccessModeViaWeb, 4779
SetAcousticSignal, 4761, 4919
SetAlarmReportingMode, 4771, 4920
SetAlarmReportMode, 4771, 4920
SetAndGetBrightness, 4775, 4921
SetBacklightColor, 4767
SetBit, 4763, 4921
SetBitInTag, 4764, 4922
SetBitWhileKeyPressed, 4766
SetBrightness, 4768, 4924
SetConnectionMode, 4777, 4925
SetDataRecordTagsToPLC, 4758, 4926
SetDataRecordToPLC, 4757, 4927
SetDaylightSavingTime, 4773, 4928
SetDeviceMode, 4762, 4929
SetDisplayMode, 4761, 4930
SETIO, 3073
SETIO_PART, 3075
SetLanguage, 4774, 4930
SetPLCMode, 4769, 4770, 4931
SetpointTrendArchiveStartId(i) property (VBS), 5793
SetpointTrendColor(i) property (VBS), 5794
SetpointTrendNumberOfValues(i) property (VBS), 5794
SetRecipeTags, 4771, 4932

SetScreenKeyboardMode, 4769, 4933
SetStart at boot, 4828, 4949
SetTag, 4776, 4934
SetTagToHandWheel, 4799
Setting
  Deadband, 4305
  Delay time, 4305
  Language, 6920, 6980
  Languages in the operating system, 6824
  Style, 3964
  View options for the assignment list, 1814
  View options for the dependency structure, 1828
Setting AS-i network configuration parameters, 602
Setting at the HMI device
  E-Mail, 7089
  HTTP server, 7142
Setting at the HMI-device
  HTTP server, 7091
  SmartServer, 7092
  Web authorization, 7091
  Web server, 7091
Setting ENO automatically, 1668
Setting language, 6920, 6980
Setting parameters for the Ethernet interface, 7608
  Adding a dynamic IP address, 7608
  Connecting to a subnet, 7607
  Deleting dynamic IP addresses, 7609
  Modifiable parameters, 7606
  Options in the parameter settings, 7606
Setting the communication load, 596
Setting the font, size and color, 1671
Setting the Internet Settings
  HTTP server, 6565
Setting the mnemonics, 1539
Setting the tab spacing, 1671
Setting up
  DCOM user rights, 7158
Setting up DCOM user rights, 7158
Settings
  Changing, 306, 1550, 1579, 1622, 1669
  FBD, 1578, 1621
  General, 302, 304, 305, 1549
  LAD, 1578, 1621
  Online and diagnostic functions, 305
  SCL, 1668
Settings for the PG/PC interface, 7604
  Automatic bus parameter detection, 7611
  Setting parameters for the MPI interface, 7612
  Setting parameters for the PROFIBUS interface, 7614
Settings for user help, 365
SetValue, 7150
SetWebAccess, 4779
Seven-segment display, 2467, 2748, 2967
Severity, 723
SF, 2258, 2534
SFP diagnostics, 1006
SGN_GET, 3739
SGN_SET, 3740
SHA algorithm, 982
SHA1, 741, 747
Shared device, 1111
  Associated IO controllers, 1368
Shared property (VBS), 5490
ShareTimeColumn property (VBS), 5796
ShareValueAxis property (VBS), 5796
ShareXAxis property (VBS), 5797
ShareYAxis property (VBS), 5797
Shift
  Left, 2441, 2722, 2939
  Right, 2439, 2720, 2938
SHIFT
  Key, 6915, 6974
ShiftDecimalPoint property (VBS), 5490
SHL, 2441, 2722, 2939
Short-circuit to ground, 1287
Short-circuit to L+, 1288
ShortenCellText property (VBS), 5717
ShortenColumnHeaderText property (VBS), 5717
Show
  Limit lines on the bar, 4098
Show line numbers, 1671
Show/hide interfaces, 7604
ShowAlarmsFromDate property (VBS), 5490
ShowAlarmWindow, 4805, 4808, 4937, 4942
ShowBadTagState property (VBS), 5491
ShowBar property (VBS), 5491
ShowBorder property (VBS), 5798
ShowCaption property (VBS), 5492
ShowColumn(i) property (VBS), 5798
ShowColumnSelection, 5919
ShowComment, 5919
ShowCurve(i) property (VBS), 5798
ShowDecimalPoint property (VBS), 5492
ShowDisplayOptionsDialog, 5920
ShowEmergencyQuitDialog, 5920
ShowFillLevel property (VBS), 5493
ShowFocusRectangle property (VBS), 5494
ShowHelp, 5921
ShowHideList, 5921
ShowHitList, 5922
Showing
  Report sections, 4500
Showing and hiding absolute operands, 1671

Showing and hiding columns, 411, 1502, 1574, 1724, 1744
ShowInputControls property (VBS), 5799
ShowLargeTicksOnly property (VBS), 5494, 5697
ShowLimitMarkers property (VBS), 5494
ShowLockDialog, 5923
ShowLockList, 5923
ShowLogonDialog, 4803
ShowLongTermArchiveList, 5924
ShowMainFrame property (VBS), 5799
ShowMessagesAtDate property (VBS), 5754
ShowOperatorNotes, 4804, 4938
ShowOverflowIndicator property (VBS), 5800
ShowPeakValuePointer property (VBS), 5495
ShowPercentageAxis, 5925
ShowPopupScreen, 4806, 4939
ShowPosition property (VBS), 5495
ShowPropertyDialog, 5925
ShowRuler property (VBS), 5496
ShowRulerInAxis property (VBS), 5497
ShowScale property (VBS), 5497
ShowScrollbars property (VBS), 5498
ShowSelectArchive, 5926
ShowSelection, 5926
ShowSelectionDialog, 5927
ShowSelectTimeBase, 5927
ShowSetpointTrend(i) property (VBS), 5800
ShowShortTermArchiveList, 5928
ShowSlideInScreen, 4807, 4940
ShowSoftwareVersion, 4808, 4940
ShowSort, 5928
ShowSortButton property (VBS), 5498
ShowSortDialog, 5929
ShowSortIcon property (VBS), 5499
ShowStatusBar property (VBS), 5500
ShowSystemAlarm, 4809, 4941
ShowSystemDiagnosticsWindow, 4808, 4942
ShowTableGridlines property (VBS), 5500
ShowTagSelection, 5929
ShowThumb property (VBS), 5501
ShowTickLabels property (VBS), 5501
ShowTicks property (VBS), 5502
ShowTimebaseDialog, 5930
ShowTimeSelection, 5930
ShowTitle property (VBS), 5502
ShowToolBar property (VBS), 5503
ShowTrendIcon property (VBS), 5503
ShowTrendIndicator property (VBS), 5504
ShowTrendSelection, 5931
ShowVerticalGridlines property (VBS), 5718
ShowXValuesExponential(i) property (VBS), 5718
ShowYValuesExponential(i) property (VBS), 5719
SHR, 2439, 2720, 2938
SiClock, 710
Sign, 2312, 2588
Signal board, 852
  Inserting, 853
Signal quality
  Calculating the effective range, 4188
signature
  electronic, 7042
SIMATIC 500/505 DP
  Migrating data types, 203
SIMATIC 500/505 serial
  Migrating data types, 203
SIMATIC ET 200 CPU
  HMI connection, 6391
  PC, 6394
  PROFIBUS, 6409, 6412
  PROFINET, 6390, 6392
SIMATIC ET 200AL, 1269, 1309, 1313
SIMATIC HMI HTTP protocol
  Valid data type, 6570
SIMATIC HMI HTTP Protocol
  Configuring HTTP clients, 7144
  Configuring the connection, 7144
  HTTP Protocol, 6557
  HTTPS Protocol, 6557
  Migrating data types, 204
  Permitted data type, 7141
SIMATIC LOGO!
  Communication, 6501
  Connection parameters, 6503
  Parameter, 6503
SIMATIC Logon, 4543
SIMATIC Manager, 7764
SIMATIC memory card
  Formatting, 1388
SIMATIC PC
  HMI connection, 6088, 6111, 6168, 6194, 6254, 6289, 6346, 6397, 6412
  S7 1500, 6088
  S7 300/400, 6254
  SIMATIC ET 200 CPU, 6397
  SIMATIC S7-1500 Software Controller, 6346
  WinCC RT Advanced, 6088, 6111, 6346, 6397
  WinCC RT Professional, 6088, 6111, 6397
SIMATIC PC station
  Load, 80
SIMATIC S5 AS511
  Migrating data types, 204
SIMATIC S5 DP
  Migrating data types, 205

SIMATIC S7
Communication partners, 6151, 6244, 6334, 6455, 6473, 6476
SIMATIC S7 Embedded Controller, 94
SIMATIC S7-1500 Software Controller
Communication, 6336
HMI connection, 6340
PC, 6343
PROFINET, 6339, 6341
SIMATIC S7 200
Migrating data types, 205
SIMATIC S7 300/400
Migrating data types, 206
SIMATIC-ACC, 614
Simple alarm view, 4358
Application, 4358
Control element, 4359
Operation, 4359
Operation using the keyboard, 4360
Operation using the mouse, 4360
simple alarm window
Application, 4358
Simple recipe view, 4155, 4440
Behavior, 4471
Configuring, 4155
Constraints, 4443
Layout, 4470
Menu command, 4471
Operation, 4470
Simple user view
Application, 4099
Configuring, 4101, 4135
Display in Runtime, 4100
Layout, 4099
Number of lines, 4099
SIMTIC ET 200 CPU
PROFIBUS, 6407
Simulate hardware, 533
Simulate modules, 533
Simulate software, 533
SimulateSystemKey, 4780
SimulateTag, 4781
Simulating
Define temporary start screen, 6932
Project with tag simulator, 6896, 6930
Simulating devices, 533
Simulation, 86
PLC connection, 87
Runtime, 6895
SIN, 2323, 2600, 2812
Sine, 2323, 2600, 2812
Single instance
Correcting the call type, 1598, 1640
Definition, 1427
Example, 1427
SINT, 1917, 1969, 2026, 2098, 2123
SINT_TO_, 2026, 2123
Size
Editing multiple objects, 3968, 4511
of objects in the report, 4498
SLC, 6634
Slice, 250
Slice access, 250, 1461
Slicing, 226
Slide-in screen
Configuring, 3912
Configuring for devices without multi-touch function, 3914
Objects, 3911
Use, 3915
Slider, 4163
Display bar, 4164
Display current value, 4164
Lock/unlock, 6970
Maximum value, 4163
Minimum value, 4163
Slider object, 5161
Slot, 3358
Determining logical address with LGC_GADR, 3372
Determining logical address with LOG_GEO, 3368
Racks, 559
Select, 560
Slot rules, 556
ET 200S, 1098
Sm@rtClient view, 4168
Shared use, 4169
View only, 4169
Smart Objects
Control, 5072
Smart Options, 7113
HMI devices suitable for use, 7084
Remote control, 7113
Remote control by means of Internet Explorer, 7114
Remote monitoring, 7113
SmartAccess
Distributed operator stations, 7121
Editing tag values in MS Excel, 7150
SmartClient
Monitoring mode, 7106

Password input, 7119
SmartClient display, 7106
SmartClient display, 7118
SmartClientView object, 5162
SmartClientViewConnect, 4784
SmartClientViewDisconnect, 4784
SmartClientViewLeave, 4785
SmartClientViewReadOnlyOff, 4783
SmartClientViewReadOnlyOn, 4783
SmartClientViewRefresh, 4782
Smartdrive
Installation, 55
SmartServer, 4791, 4872
As a service, 7105
Control mode, 7102
Forced access, 7105
Hardware acceleration, 7112
Local operator control, 7105
Monitoring mode, 7102
Password, 7112
Password input, 7119
Setting at the HMI-device, 7092
SmartService
E-mail notification, 7125
Input area plan, 7089
Remote control by means of the SmartClient application, 7115
SmartTag object, 4982, 5009
SmartTags object (list), 4981, 5011
SmartTags property (VBS), 5505
SMC, 2462, 2743, 2961
Smoothing, 1265, 1331
SMS text (TC_RECV), 3838
SMS text (TC_SEND), 3834
SMS text (TCON_PHONE), 3844
SMTP Client, 979
SNC_RTCB, 2996
SNMP, 695, 928, 967, 981
Groups, 981
SNMP trap, 981
SNMPv1, 928
SNMPv2c, 928
SNMPv3, 928
Users, 983
SNMPv1, 741
SNMPv3, 741
SOAP
Access from Excel, 7149
GetValue, 7150
SetValue, 7150
Windows CE, 7150
Softbus, 6745
Softkey, 4036, 4040
global, 4039
SOFTNET Security Client
Configuring in the project, 769
Creating a configuration file, 769
Database, 770
Using, 768
SOFTNET Security Client, 677
Software
HMI device, 6988, 6989
Software controller
Configuring, 7197
Software requirements, 118
Software requirements for AS-AS remote link
Communication instruction "AS_DIAL", 7713
Software requirements on the programming device/ PC side
Communication instruction "PG_DIAL", 7711
Software requirements on the system side
Communication instruction "AS_MAIL", 7715
Communication instruction "PG_DIAL", 7711
Communication instruction "SMS_SEND", 7714
Software version, 942
Software-controlled data flow control, 1201
Sort property (VBS), 5755
SortByTimeDirection property (VBS), 5506
SortByTimeEnable property (VBS), 5720
SortByTimeEnabled property (VBS), 5506
Sorting
Alarm view, 4140
Report page, 4500
SortOnColumnHeaderClick property (VBS), 5720
SortSequence property (VBS), 5506
SortTimeAscending property (VBS), 5721
SortTimeEnable property (VBS), 5721
SP, 2252, 2527
Spanning Tree
Information, 946
Rapid Spanning Tree, 931
Special characters
Alarm text, 68
in tags for Web server, 864
Special considerations
Omron Host Link, 6720
Special features of CHM files, 370
Special tag (web server), 868
Specific .Net control
Add, 3970
Remove, 3971
Speedy Splitter, 537, 540, 542, 552
SPLIT, 3027
SQR, 2319, 2595, 2808

SQRT, 2320, 2596, 2809
Square, 2319, 2595, 2808
Square root, 2320, 2596, 2809
SR, 2207, 2483
SRT_DINT, 3230
SS, 2256, 2532
SSH server, 966
SSL certificate, 689
Stamping, 3964, 4504
Standard
PROFIBUS, 6204, 6284
Standard device, 1140
Standard router, 802
Standard slave, 1109
Standard user, 693
Standby module
Substitute value, 1103
Standstill signal, 7395
Start address, 848, 1176
Start information, 1414
Start off-delay timer, 2507
Start on-delay timer, 2505
Start pulse timer, 2503
Start screen, 3898
Start the online and diagnostics view, 1363
Start the simulation, 534
Start value, 99
Tag, 4220
Start value of danger range
Gauge, 4191
Start value of warning range
Gauge, 4191
StartAngle property (VBS), 5507
Starting
Debugger, 4599, 6934
Runtime at Engineering Station, 6950
Runtime on a panel, 6904, 6952
Runtime on PC, 6952
Runtime on the configuration PC, 6950
Starting removal, 146
Starting the migration tool, 157
Starting the topology view, 663
StartLeft property (VBS), 5722
StartLogging, 4787, 4942
StartNextLog, 4788, 4943
StartProgram, 4788, 4944
StartSequenceLog, 4788, 4943
StartStopUpdate, 5931
StartStyle property (VBS), 5508
StartTop property (VBS), 5508
Startup, 1213
Organization blocks, 1213
STARTUP
Function, 1165
STARTUP activities, 1166
Warm restart, 1166
Startup language
determine:Startup language, 4349
Startup OB
Description, 1213
Start-up parameters, 1167
Startup routine, 1213
Startup type, 1185
StartValue property (VBS), 5508
Stateful packet inspection, 701
Station
Copying, 572
Deleting, 571
Moving, 573
Read information with GetStationInfo, 3286
Renaming, 673
Station name, (See device name)
Status, 1217
Module defective, 1376
Status IDs - data points, 908
Status interrupt, 1217
Status interrupt OB, 1217
Status of the online connection, 1400
Status/Force
Button, 4172
Column header, 4172
Control element, 4171
Sequence of columns, 4172
Visible column, 4171
StatusbarBackColor property (VBS), 5509
StatusbarElementAdd property (VBS), 5510
StatusbarElementAutoSize property (VBS), 5510
StatusbarElementCount property (VBS), 5511
StatusbarElementIconId property (VBS), 5512
StatusbarElementID property (VBS), 5512
StatusbarElementIndex property (VBS), 5513
StatusbarElementName property (VBS), 5514
StatusbarElementRemove property (VBS), 5514
StatusbarElementRename property (VBS), 5515
StatusbarElements property (VBS), 5515
StatusbarElementTooltipText property (VBS), 5516
StatusbarElementUserDefined property (VBS), 5516
StatusbarElementVisible property (VBS), 5517
StatusbarElementWidth property (VBS), 5517
StatusbarFontColor property (VBS), 5518
StatusbarShowArchiveName property (VBS), 5801
StatusbarShowColumn property (VBS), 5801
StatusbarShowRecord property (VBS), 5802
StatusbarShowRow property (VBS), 5802

StatusbarShowText property (VBS), 5803
StatusbarShowTooltips property (VBS), 5519
StatusbarText property (VBS), 5519
StatusbarUseBackColor property (VBS), 5520
StatusbarVisible property (VBS), 5521
StatusForce object, 5164
StatusForceGetValues, 4790
StatusForceSetValues, 4791
Stepper motor, 7329
STL
  Detailed comparison, 1763
STL program
  Program status display, 1848
STL source
  Generating blocks, 1749
STOP, 1168
Stop forcing, 1906, 1907
Stop method, 5932
StopLogging, 4793, 4945
Stopping
  Debugger, 4600
StopRuntime, 4794, 4947
Storage location
  Audit Trail, 7064
  Database, 4423
  File - CSV (ASCII), 4422
  File - RDB, 4422
  File - TXT (Unicode), 4422
  Logs with checksum, 4423
Storage locations of the instance certificates, 7164
Storing
  External graphic, 3974
STP, 2413, 2693, 2918
Strg_TO_Chars, 3015
STRG_VAL, 3008
String
  Comparing tags with S_COMP, 3003
STRING, 102, 1469, 1937, 2007, 2086, 2111, 2112, 2154, 2156, 2170, 2192
  Addressing, 1460
STRING_TO_, 2086, 2154, 2156, 2192
STRUCT
  Addressing, 1460
  Declaration in global data blocks, 1712
  Declaration in the block interface, 1560
  Declaring in a PLC data type, 1739
  Structure, 1945
Structure, 1268, 1310, 1325
  Call structure, 1821
  of the cross-reference list, 1835, 6818
  Of the dependency structure, 1826
  Resources tab, 1832
  STRUCT, 1945
Structured programming, 1413
Structures, 254
Style
  Changing, 3922
  Creating a new type, 6791
  Define, 3919
  Deleting, 3921
  Duplicate, 3920
  Edit, 3921
  Setting, 3964
Style property (VBS), 5521
Style sheet
  Copying, 3930
  Creating, 3927
  Creating a new type, 6792
  Deleting, 3929
  Editing, 3929
  Use, 3931
StyleSettings property (VBS), 5522, 5735
SUB, 2305, 2580
Sub-cycle, 1298
Subnet, 598
Subnet mask, 596, 1116
Substitute value
  Standby module, 1103
Subtract, 2305, 2580
Subtracting
  Subtracting timers with T_SUB, 2976
Supply voltage, 1289
Supported file formats, 368
SWAP, 2355, 2634, 2837
Swap bytes, 2355
SwapDimensionsWithOrientation property (VBS), 5803
SwapFirstWithLastConnection property (VBS), 5522
Switch, 4160
  Type, 4160
SWITCH, 2404, 2684
Switch buffer, 6134, 6224, 6313, 6370, 6438, 6491, 6508, 6639, 6670, 6702, 6721
Switch object, 5166
Switch OFF, 4855
Switch ON, 4855
Switching between basic mode and expanded mode in the force table, 1884
Switching between basic mode and expanded mode in the watch table, 1856
Switching operating mode, 1382
Switchover time, 1203
Switchport, 1136, 6102, 6187, 6268, 6405

Symbol
Alarm classes, 4317
For value comparison, 7204
Symbol library, 4172
Fill style, 4173
Fixed aspect ratio, 4174
Flip, 4173
OP 177B mono, 4173
Rotate, 4173
TP 177B mono, 4173
Symbolic, 250
Symbolic addressing, 249
of a tag, 4203, 6037
Symbolic connection
Tag, 7771
Symbolic I/O field, 4174
application in reports, 4526
Mode, 4175
Text list, 3999, 4175
Symbolic programming
Displaying absolute addresses, 1539
SymbolicIOField object, 5170
Symbolism, 224
SymbolLibrary object, 5175
Symbols
In the assignment list, 1812
In the call structure, 1820
In the dependency structure, 1826
Sync domain, 1137, 1369, 6103, 6188, 6269, 6406
Sync domain properties, 1369
SYNC_PI, 3066
SYNC_PO, 3067
Synchronization
From DP slaves with DP_SYC_FR, 3133
Slave clocks with SNC_RTCB, 2996
Synchronization (user-defined web pages), 860
Synchronize, 4201, 6035
Synchronizing
Object in user-defined functions, 4579
Recipe data record, 4481
Recipe tag, 4432
Recipe view and recipe screen, 4428
Tag in user-defined functions, 4579
Synchronizing a PLC tag, 4216
Synchronizing user-defined web pages, 3721
Synchronous
Transferring data, 6129, 6218, 6308, 6364, 6432, 6490, 6738
Synchronous errors
Masking with MSK_FLT, 3239
Querying the error register with READ_ERR, 3240
Unmasking with DMSK_FLT, 3240
Syntax
Addressing, 6612
Syntax errors
Basics, 1697
Finding errors, 1697
Syntax for AWP commands, 863
Syntax highlighting
User-defined function, 4580
Syslog Client, 993
Syslog server, 757
System
General information, 968
System configuration, 966
System alarm, 4284
Meaning, 4369, 4371, 4379, 4384, 4387, 4399
System block
System blocks folder, 1506
System constant, 1494
System diagnostics, 4401, 4406
Button, 4413
Detail view, 4402, 4406
Device view, 4402, 4406
Diagnostic buffer view, 4402, 4407
Matrix view, 4405
Settings in the PLC, 4409
System diagnostics view, 4401, 4406
System diagnostics window, 4401, 4412
System diagnostics indicator
Button as system diagnostics indicator, 4413
Inserting, 4409
System diagnostics window, 4409
system diagnostics view
Layout, 4177
Show split view, 4179, 4416
System diagnostics indicator, 4409
System diagnostics view, 4401
Configuring, 4414, 4415, 4416
Detail view, 4177
Device view, 4177
Diagnostic buffer view, 4177
Settings in the PLC, 4409
Symbol, 4179
System diagnostics window, 4401, 4412
Configuring, 4412
Layout, 4180
System event, 4281, 4282, 4284
Editing, 4309

Meaning, 4369, 4370, 4371, 4372, 4375, 4377, 4379, 4380, 4381, 4383, 4385, 4387, 4388, 4389, 4391, 4392, 4394, 4395, 4400
Parameters, 4369
System events, 764, 766
Configuring, 4295, 4296
System function, 4569, 4823
Application, 4571
Applications, 4570
DirectKeyScreenNumber, 4674
HMI device dependency, 4583
HTMLBrowserScrollLeft, 4690
In a function list, 4570
in user-defined functions, 4601
In user-defined functions, 4570
Language dependency, 4571
NotifyUserAction, 7074
SafelyRemoveHardware, 4686, 4916
Sequence is not adhered to, 4602
SetStart at boot, 4828, 4949
WinACMPArchive, 4826
WinACMPClearCycleTimeBuffer, 4828
WinACMPControl, 4833
WinACMPGetStartMode, 4827
WinACMPGetVersion, 4827, 4948
WinACMPRestore, 4834
WinACMPSetHMIEnableTime, 4830
WinACMPSetKeySwitch, 4829
WinACMPSetRestartCharacteristics, 4830
WinACMPSetSleeptime, 4831
WinACMPSetStartMode, 4831, 4949
WinACMPStartHistogram, 4832
WinACMPUpdateAverageCycleTime, 4815
WinACMPUpdateAverageExecTime, 4814
WinACMPUpdateBUSF1LED, 4812
WinACMPUpdateBUSF2LED, 4813
WinACMPUpdateEXTFLED, 4816
WinACMPUpdateINTFLED, 4818
WinACMPUpdateKeySwitchSetting, 4811
WinACMPUpdateLastCycleTime, 4819
WinACMPUpdateMaximumCycleTime, 4820
WinACMPUpdateMinimumCycleTime, 4821
WinACMPUpdatePowerLED, 4822
WinACMPUpdateRUNLED, 4824
WinACMPUpdateSleepTime, 4823
WinACMPUpdateStartupCharacteristics, 4810
WinACMPUpdateSTOPLED, 4825
System functions
AcknowledgeAlarm, 4743, 4857
ActivateCleanScreen, 4659
ActivatePreviousScreen, 4660, 4857
ActivateScreen, 4657, 4858
ActivateScreenByNumber, 4658, 4859
ActivateSystemDiagnosticsView, 4795, 4860
AdjustContrast, 4661
AlarmViewAcknowledgeAlarm, 4725
AlarmViewEditAlarm, 4724
AlarmViewShowOperatorNotes, 4726
AlarmViewUpdate, 4724
ArchiveLogFile, 4663, 4861
Available for WinCC Runtime, 4651
Available on Comfort Panels, 4636
Available on mobile panels, 4643
Available on Multi Panels, 4630
Available on Panels, 4624
BackupRAMFileSystem, 4779, 4863
CalibrateTouchScreen, 4702, 4863
ChangeConnection, 4802, 4864
ClearAlarmBuffer, 4722, 4866
ClearAlarmBufferProTool, 4723, 4867
ClearDataRecordMemory, 4721, 4869
ClearLog, 4719, 4870
CloseAllLogs, 4756, 4871
ControlSmartServer, 4791, 4872
ControlWebServer, 4792, 4873
DecreaseFocusedValue, 4800
DecreaseTag, 4801, 4875
DeleteDataRecord, 4720, 4868
DirectKey, 4672
EditAlarm, 4665, 4875
Encode, 4670, 4876
EncodeEx, 4671, 4877
EstablishPROFIsafe, 4742, 4878
ExportDataRecords, 4679, 4879
ExportDataRecordsWithChecksum, 4682, 4881
ExportImportUserAdministration, 4684, 4884
GetBrightness, 4715, 4885
GetDataRecordFromPLC, 4711, 4885
GetDataRecordName, 4712, 4887
GetDataRecordTagsFromPLC, 4714, 4889
GetGroupNumber, 4715, 4889
GetPassword, 4716, 4890
GetUserName, 4710, 4891
GoToEnd, 4685, 4891
GoToHome, 4685, 4892
HTMLBrowerZoomIn, 4689
HTMLBrowserBack, 4694
HTMLBrowserForward, 4693, 4694
HTMLBrowserHome, 4693
HTMLBrowserPageDown, 4692
HTMLBrowserPageUp, 4692
HTMLBrowserRefresh, 4688
HTMLBrowserScrollDown, 4687
HTMLBrowserScrollRight, 4691

HTMLBrowserScrollUp, 4688
HTMLBrowserStop, 4687
HTMLBrowserZoomOut, 4690
ImportDataRecords, 4695, 4893
ImportDataRecordsWithChecksum, 4697, 4895
IncreaseFocusedValue, 4678
IncreaseTag, 4678, 4896
InvertBit, 4698, 4898
InvertBitInTag, 4699, 4899
InvertLinearScaling, 4700, 4897
LinearScaling, 4717, 4900
LoadDataRecord, 4709, 4902
Logoff, 4656, 4903
Logon, 4662, 4903
LogTag, 4664
LookupText, 4798, 4904
NotifyUserAction, 4676, 4905
OpenAllLogs, 4727, 4906
OpenCommandPrompt, 4730, 4907
OpenControlPanel, 4732
OpenControlPanelDialog, 4729, 4908
OpenInternetExplorer, 4731, 4909
OpenScreenKeyboard, 4728, 4910
OpenTaskManager, 4732, 4910
PageDown, 4758, 4911
PageUp, 4759, 4912
PDFFitToHeight, 4735
PDFFitToWidth, 4734
PDFGoToFirstPage, 4736
PDFGoToLastPage, 4736
PDFGoToNextPage, 4737
PDFGoToPage, 4737
PDFGoToPreviousPage, 4738
PDFScrollDown, 4733
PDFScrollLeft, 4740
PDFScrollRight, 4740
PDFScrollUp, 4734
PDFZoomIn, 4739
PDFZoomOriginal, 4741
PDFZoomOut, 4739
PressButton, 4752
PrintReport, 4676, 4912
PrintScreen, 4675, 4913
RecipeViewBack, 4749
RecipeViewClearDataRecord, 4744
RecipeViewGetDataRecordFromPLC, 4744
RecipeViewMenu, 4745
RecipeViewNewDataRecord, 4743
RecipeViewOpen, 4745
RecipeViewRenameDataRecord, 4748
RecipeViewSaveAsDataRecord, 4747
RecipeViewSaveDataRecord, 4747
RecipeViewSetDataRecordToPLC, 4746
RecipeViewShowOperatorNotes, 4749
RecipeViewSynchronizeDataRecordWithTags, 4748
ReleaseButton, 4753
ResetBit, 4750, 4914
ResetBitInTag, 4751, 4915
ResetTagToHandWheel, 4799
SaveDataRecord, 4786, 4917
ScreenObjectCursorDown, 4666
ScreenObjectCursorLeft, 4667
ScreenObjectCursorRight, 4668
ScreenObjectCursorUp, 4666
ScreenObjectPageDown, 4668
ScreenObjectPageUp, 4669
SendEMail, 4760, 4918
SetAcousticSignal, 4761, 4919
SetAlarmReportMode, 4771, 4920
SetAndGetBrightness, 4775, 4921
SetBit, 4763, 4921
SetBitInTag, 4764, 4922
SetBitWhileKeyPressed, 4766
SetBrightness, 4768, 4924
SetConnectionMode, 4777, 4925
SetDataRecordTagsToPLC, 4758, 4926
SetDataRecordToPLC, 4757, 4927
SetDaylightSavingTime, 4773, 4928
SetDeviceMode, 4762, 4929
SetDisplayMode, 4761, 4930
SetLanguage, 4774, 4930
SetRecipeTags, 4771, 4932
SetScreenKeyboardMode, 4769, 4933
SetTag, 4776, 4934
SetTagToHandWheel, 4799
SetWebAccess, 4779
ShowAlarmWindow, 4805, 4937
ShowLogonDialog, 4803
ShowOperatorNotes, 4804, 4938
ShowPopupScreen, 4806, 4939
ShowSlideInScreen, 4807, 4940
ShowSoftwareVersion, 4808, 4940
ShowSystemAlarm, 4809, 4941
ShowSystemDiagnosticsWindow, 4808, 4942
SimulateSystemKey, 4780
SimulateTag, 4781
SmartClientViewConnect, 4784
SmartClientViewDisconnect, 4784
SmartClientViewLeave, 4785
SmartClientViewReadOnlyOff, 4783
SmartClientViewReadOnlyOn, 4783
SmartClientViewRefresh, 4782
StartLogging, 4787, 4942

StartNextLog, 4788, 4943
StartProgram, 4788, 4944
StatusForceGetValues, 4790
StatusForceSetValues, 4791
StopLogging, 4793, 4945
StopRuntime, 4794, 4947
SystemDiagnosticsViewBack, 4797
SystemDiagnosticsViewDetailView, 4796
SystemDiagnosticsViewDeviceView, 4797
SystemDiagnosticsViewDiagnosticsBuffer, 4796
SystemDiagnosticsViewRefreshPLCBuffer, 4795
TerminatePROFIsafe, 4742, 4948
TraceUserChange, 4800
TrendViewBackToBeginning, 4708
TrendViewCompress, 4705
TrendViewExtend, 4705
TrendViewRulerLeft, 4707
TrendViewRulerRight, 4706
TrendViewScrollBack, 4704
TrendViewScrollForward, 4704
TrendViewSetRulerMode, 4707
TrendViewStartStop, 4708
UpdateTag, 4660
System libraries, 471
System limits
HMI device, 7007, 7011, 7014, 7018, 7024, 7027
System log, 764, 766
System memory, 1185, 1206
Diagnostics buffer, 1176, 1398
Operand areas, 1172
process image input/output, 1174
System power supply, 1325
System requirement for WinCC Advanced, 121
System requirements STEP 7 Basic, 118
System texts, 454
System Time
NTP Client, 988
PTP Client, 990
SNTP Client, 987
System-defined controller alarm, 4285
System-defined controller alarms, 4282
System-defined role
Administrator, 693
diagnostics, 693
Remote access, 693
standard, 693
System-defined text lists
Editing, 466
Modifying texts, 467
SystemDiagnoseView object, 5177
SystemDiagnoseWindow object, 5181
SystemDiagnosticsViewBack, 4797
SystemDiagnosticsViewDetailView, 4796
SystemDiagnosticsViewDeviceView, 4797
SystemDiagnosticsViewDiagnosticsBuffer, 4796
SystemDiagnosticsViewRefreshPLCBuffer, 4795
System-relevant information
Module information, 1365
Vendor information, 1365

T

T_ADD, 2974
T_COMBINE, 2979
T_COMP, 2971
T_CONFIG, 103, 3700
T_CONV, 2972
T_DIAG, 3694
T_DIFF, 2978
T_RESET, 3693
T_SUB, 2976
TAB key, 6913, 6973
Table
Connection, 6022
Table:Designing a border, 3961
TableBackColor property (VBS), 5523
TableColor property (VBS), 5524
TableColor2 property (VBS), 5524
TableFocusOnButtonCommand property (VBS), 5755
TableForeColor property (VBS), 5525
TableForeColor2 property (VBS), 5526
TableGridLineColor property (VBS), 5526
TableHeaderBackColor property (VBS), 5527
TableHeaderForeColor property (VBS), 5528
TableView object, 5114
Tag
absolute addressing, 4202, 6036
Acquisition cycle, 4221, 4235, 4247
Acquisition mode, 4221
Addressing, 4207
Changing the PLC, 4225
Comment, 4209
Configuration, 4213
Configuring several tags, 4213
Connection to PLC, 4207
Copy, 4212
Creating external tags, 4207
Creating internal tags, 4209
Creating the user data type, 4241
Data log, 4251, 4256
Data type, 4209
Deleting, 4212

Displaying or hiding tag information, 1611, 1653, 1681
Displaying values, 4268
Event, 4234
Exporting, 6805
External tag, 4200, 6034
Faceplate type, 4083
GMP settings, 7067
GMP-relevant tag, 7067
Handwheel, 4114
Illuminated pushbutton, 4127
Importing, 6807, 6808
in Runtime, 4221
Index tag, 4232, 4233
Indirect addressing, 4232, 4233
Insert, 7772
Internal tag, 4205, 6039
Key switch, 4165
Length, 4209
Limit value, 4219
Limit values, 4218
Linear scaling, 4222, 4224
Logging, 4251, 4256
Logging cycle, 4247
Maximum length, 82
Multiplexing, 4232
Name, 4209
Negative, 2217, 2493
Object list, 4208
Output in alarm, 4307
Overlaying, 1462
Overview, 1447
PLC tag, 1479
PLC tags and DB tags, 1448
Positive, 2216, 2492
Read continuously, 4222
Reconnecting, 4216
Rename, 4212
RFID tag, 5943
Start value, 4220
symbolic addressing, 4203, 6037
Symbolic connection, 7771
Tolerance band, 4251, 4256
Transponder, 5943
Update, 4222, 4235
User data type, 4240
User data type element, 4240
Tag array, 4236
Tag binding
Animation, 4026
Tag data
Structure for the import, 6809, 6814
Tag declaration
Automatically filling in cells, 1501, 1574, 1724, 1743
Based on a PLC data type, 1713
Block interface, 1551
Declaring ARRAY, 1559
Declaring PLC data type, 1562
Declaring STRUCT, 1560
Declaring tags, 1557, 1558, 1562
Deleting a tag, 1500, 1573, 1723, 1743
Importing and exporting tags, 1575, 1725
Inserting a table row, 1499, 1572, 1722, 1742
Inserting table rows at the end, 1573, 1723, 1742
Multi-instance, 1563
Overlaying tags, 1562
Purpose of tag declaration, 1551
Reserved key words, 1442
Retentivity, 1568
Showing and hiding columns, 1502, 1574, 1724, 1744
Sorting rows, 1501
Tag properties, 1567, 1569, 1718, 1720, 1721
Updating the block interface, 1564
Valid data types, 1554, 1556
Tag import
Tag data structure, 6809, 6814
Tag list
Indirect addressing, 4232, 4233
Tag log
Migrating, 188
Tag name
Web server (PLC), 864
Tag object, 5012
Tag simulator, 6898, 6932
Tag table
Default, 4198, 6033
for HMI devices, 4198, 6033
user-defined, 4198, 6033
Tag value
Output, 4268, 4272
TagPrefix property (VBS), 5528
Tags, 5015, 5016
Basics, 4198, 6032
Data exchange, 5998
Editing, 4213
Migration, 196
Monitor all, 1872, 1896
Monitor now, 1897
Monitor once and now, 1873
Tags object (list), 5016
Tan, 2814
TAN, 2325, 2602

Tangent, 2325, 2602, 2814
Task, 5978
Changing during Runtime, 5989
Changing the name, 5986
Deactivate, 5981, 5982
Deleting, 5987
Task card, 3892
Changing the pane mode, 328
Find instructions, 1540
Function, 326
Hardware catalog, 549, 6015
Reducing automatically, 318
Task Card
Online Tools, 1364
Tools, 3932, 3935
TCI (Tool Calling Interface), 1145
TCON, 3662, 3664
TCON_IP_RFC, 644
TCON_IP_v4, 643
TCON_Param, 640
TCP, 706
Characteristics, 638
Port numbers, 645
TDISCON, 3669
Teach-in mode, 4433
Recipe tag, 4433
Team Engineering
Creating a master project, 7726
Example of the program structure in the master project, 7733
Integrating project copies into the master project once again, 7727
Introduction to shared commissioning, 7725
Notes on compatibility, 7727
Procedure for creating the master project, 7729
Procedure for integrating project copies into the master project, 7731
Procedure for manual synchronization of competing changes, 7731
Procedure for modifying central objects within the master project, 7732
Procedure for shared commissioning, 7730
Requirements for shared, parallel working on the project, 7729
Rules for downloading to the CPU, 7735
Rules for editing shared central objects, 7735
Rules for online functions, 7736
Rules for shared working on one CPU, 7734
Rules for the master project, 7732
Software and hardware requirements, 7728
Working with project copies, 7726
Technology object command table: Shortcut menu commands, 7421
Technology objects
PID_3Step, 7256
PID_Compact, 7224
PID_Temp, 7289
Telemecanique Uni-Telway
Migrating data types, 209
TeleService
Access to phone books, 7664
AS_DIAL, 7711
Callback variants, 7677
Communications instruction:"AS_DIAL", 7663
Communications instruction:"AS_MAIL", 7663
Communications instruction:"PG_DIAL", 7663
Communications instruction:"SMS_SEND", 7663
Communications instruction:"TM_MAIL", 7663
Communications instruction:"TMAIL_C", 7663
Connection establishment options:S7-1200 CPUs, 7709
Connection establishment options:S7-1500 CPUs, 7709
Connection establishment options:S7-300/400 CPUs, 7709
Defining dialing rules, 7670
Establish connection to AS, 7711
Export phone book, 7669
Functionality, 7663
Gateways, 7674
Import phone book, 7667
Inserting rows in the phone book, 7667
Meaning of the icons, 7666
Modem support, 7673
Modem types / media, 7674
Open phone book, 7666
Performance in telephone networks, 7674
Phone book, 7663
Phone book properties, 7664
Printing the phone book, 7670
Save phone book, 7667
Showing or hiding columns in the phone book, 7667
Structure of the phone book, 7665
Transferring e-mail with TM_MAIL, 3854
Working with the phone book, 7664
TeleService callback options, 7679
TeleService phone book, 7663
TeleService via mobile wireless, 921
Temperature coefficient, 1264, 1331
Temperature compensation, 1251, 1292
template
Template, 3903

Template, 4036
Copy, 3903
Creating, 3904
Deleting, 3903
Global screen, 3903
Inserting, 3903
Move, 3903
Rename, 3903
Use in screen, 3907
Template property (VBS), 5530
Temporary connection, 899
Tens complement, 2469, 2750, 2968
Terminal modules and electronic modules, 1249
TerminatePROFIsafe, 4742, 4948
Terminating the remote connection, 7695
Test options in the force table, 1881
Testing
User-defined function, 4588
Text
Button, 4162
Text field, 4182, 4527
Text field, 4182, 4526
Size, 4182, 4527
Text field length
Asian languages, 6847
Text list
Application, 3990
Bit (0, 1), 3996
Bit number (0 - 31), 3997
Creating, 3992
Exporting, 6812
Importing, 6814
Output in alarm, 4308
Symbolic I/O field, 3999, 4175
Value/Range, 3994
Text lists
Introduction, 463
System-defined, 464
Text lists editor, 464
Use in recipe data records, 4438
User-defined, 464
Text property (VBS), 5531
TextField object, 5184
TextList property (VBS), 5532
TextOff property (VBS), 5532
TextOn property (VBS), 5533
TextOrientation property (VBS), 5534
Texts
Exporting all texts, 461
Exporting individual texts, 460
Exporting texts of a device, 460
Importing, 462
Migrating, 185
TFTP
Load/save, 974
THEN, 2898
Thermal resistor, 1292, 1303
Thermocouple, 1292, 1303
Third-party drivers
Communication, 6599
Special features, 6600
THIS, 1456
ThumbBackColor property (VBS), 5534, 5536
TIA Portal
Exiting, 301
Starting, 301
TicksColor property (VBS), 5535
Time, 967, 1185, 1929, 1931
TIME, 1930, 1988, 2060, 2106, 2147, 2166, 2186
Time accumulator, 2225, 2501, 2509, 2764
Time delay, 2421, 2702, 2926
Time delay interrupt, 1219
Time duration, 2237, 2513, 2768
Time error interrupt, 1223
Time error OB, 1223
Time expired, 4856
Time of day
Precision Time Protocol, 990
Setting the time of day in the Online and Diagnostics view, 1383
SNTP (Simple Network Time Protocol), 987
System Time, 984
Time-of-day synchronization, 987
Time stamp, 4294
Time stamping, 1252
Time synchronization, 737, 1185, 6158, 6244, 6386, 6463
Configuring, 6159, 6160, 6387, 6388, 6464, 6465
integrated connection, 6159, 6387, 6464
Non-integrated connection, 6160, 6247, 6388, 6465
Restriction, 6158, 6245, 6386, 6463
System limits, 6158, 6245, 6386, 6463
TIME_TCK, 2997
TIME_TO_, 2060, 2147, 2186
TimeAxisBeginTime(i) property (VBS), 5536
TimeAxisEndTime property (VBS), 5537
TimeAxisLabel(i) property (VBS), 5537
TimeAxisRange property (VBS), 5538
TimeAxisShowGridLines(i) property (VBS), 5722
TimeAxisShowLargeIncrements(i) property (VBS), 5804

TimeAxisShowSmallIncrements(i) property (VBS), 5804
TimeAxisTimeFormat(i) property (VBS), 5538
TimeBase property (VBS), 5538
TimeColumnActualize property (VBS), 5539
TimeColumnAdd property (VBS), 5540
TimeColumnAlignment property (VBS), 5540
TimeColumnAlignment(i) property (VBS), 5541
TimeColumnBackColor property (VBS), 5541
TimeColumnBeginTime property (VBS), 5542
TimeColumnCaption property (VBS), 5542
TimeColumnCount property (VBS), 5543
TimeColumnDateFormat property (VBS), 5543
TimeColumnEndTime property (VBS), 5544
TimeColumnForeColor property (VBS), 5544
TimeColumnFormat(i) property (VBS), 5723
TimeColumnHideText property (VBS), 5545
TimeColumnHideTitleText property (VBS), 5545
TimeColumnIndex property (VBS), 5546
TimeColumnLength property (VBS), 5546
TimeColumnMeasurePoints property (VBS), 5546
TimeColumnName property (VBS), 5547
TimeColumnRangeType property (VBS), 5547
TimeColumnRemove property (VBS), 5548
TimeColumnRename property (VBS), 5548
TimeColumnRepos property (VBS), 5549
TimeColumnShowDate property (VBS), 5549
TimeColumnShowIcon property (VBS), 5550
TimeColumnShowTitleIcon property (VBS), 5550
TimeColumnSort property (VBS), 5551
TimeColumnSortIndex property (VBS), 5551
TimeColumnTimeFormat property (VBS), 5552
TimeColumnTimeRangeBase property (VBS), 5552
TimeColumnTimeRangeFactor property (VBS), 5553
TimeColumnUseValueColumnColors property (VBS), 5553
TimeColumnVisible property (VBS), 5554
Time-delay interrupt
  Canceling with CAN_DINT, 3231
  Instructions, 3229
  Querying with QRY_DINT, 3232
  Starting with SRT_DINT, 3230
Time-delay interrupt OB, 1219
Time-driven output
  Report, 4501
TimeJumpColor(i) property (VBS), 5805
TimeJumpEnabled(i) property (VBS), 5805
Time-of-day, 1932, 1933, 1934, 1935
  Calculating local time with SET_TIMEZONE, 2989
  Reading the system time of the CPU with TIME_TCK, 2997
  Reading time-of-day and date of CPU with RD_SYS_T, 2983
  Setting for CPU with WR_SYS_T, 2981
Time-of-day function
  Basics, 1192
  Clock parameters, 1193
  Reading the time-of-day, 1192
  Setting the time-of-day, 1192
  Time-of-day format, 1192
Time-of-day interrupt
  Activating with ACT_TINT, 3227
  Cancel, 1215
  Canceling with CAN_TINT, 3226
  Function, 1215
  Querying with QRY_TINT, 3228
  Rules, 1215
  Set and activate, 1215
  Setting with SET_TINT, 3222, 3224
  Status query, 1215
Time-of-day interrupt OB
  Parameter assignment, 1227
Timeout
  HTTP protocol, 6561
TimeOverlapColor(i) property (VBS), 5806
TimeOverlapEnabled(i) property (VBS), 5806
Timer, 1930, 2238, 2514, 5983
  Acyclic triggers, 5983
  Cyclic triggers, 5983
  Recording, 2233
TIMER, 1953
TimeRangeBase(i) property (VBS), 5755
TimeRangeFactor(i) property (VBS), 5756
Timers
  Adding with the instruction T_ADD, 2974
  Combine date and time with T_COMBINE, 2979
  Comparing time tags with T_COMP, 2971
  Converting with T_CONV, 2972
  Determining difference with T_DIFF, 2978
  Subtracting with T_SUB, 2976
Time-triggered trends, 6133, 6223, 6313, 6369, 6437, 6491, 6507, 6638, 6669, 6701, 6721
Title page
  Report, 4495
TitleColor property (VBS), 5724
TitleCut property (VBS), 5555
TitleDarkShadowColor property (VBS), 5556
TitleForeColor property (VBS), 5557
TitleGridLineColor property (VBS), 5557
TitleLightShadowColor property (VBS), 5558
TitleSort property (VBS), 5559
TitleStyle property (VBS), 5559
TM_MAIL, 3854

TMAIL_C, 3648
TO, 2902
TO_PositioningAxis, 7360
TOD, 1932, 2000, 2077, 2108, 2150, 2169, 2189
TOD_TO_, 2077, 2150, 2189
TOF, 2223, 2231, 2499, 2507, 2761
Toggle, 4855
  Between Runtime languages, 6840
  Key, 6915, 6974
Toggle property (VBS), 5560
Tolerance band
  Tags, 4251, 4256
Tolerance property (VBS), 5560
ToleranceColor property (VBS), 5561
ToleranceLowerLimit property (VBS), 5561
ToleranceLowerLimitColor property (VBS), 5562
ToleranceLowerLimitEnabled property (VBS), 5563
ToleranceLowerLimitRelative property (VBS), 5563
ToleranceUpperLimit property (VBS), 5564
ToleranceUpperLimitColor property (VBS), 5564
ToleranceUpperLimitEnabled property (VBS), 5565
ToleranceUpperLimitRelative property (VBS), 5565
TON, 2221, 2229, 2496, 2505, 2758
TONR, 2225, 2233, 2501, 2509, 2764
Tool Calling Interface (TCI), 1145
Toolbar, 4109
  Order, 3939, 3947, 4507
ToolbarAlignment property (VBS), 5566
ToolbarBackColor property (VBS), 5567
ToolbarButtonActive property (VBS), 5567
ToolbarButtonAdd property (VBS), 5568
ToolbarButtonAuthorization property (VBS), 5573
ToolbarButtonBeginGroup property (VBS), 5568
ToolbarButtonCount property (VBS), 5569
ToolbarButtonEnabled property (VBS), 5570
ToolbarButtonHotKey property (VBS), 5570
ToolbarButtonID property (VBS), 5571
ToolbarButtonIndex property (VBS), 5571
ToolbarButtonLocked property (VBS), 5572
ToolbarButtonName property (VBS), 5573
ToolbarButtonRemove property (VBS), 5574
ToolbarButtonRename property (VBS), 5574
ToolbarButtonRepos property (VBS), 5575
ToolbarButtonTooltipText property (VBS), 5575
ToolbarButtonUserDefined property (VBS), 5576
ToolbarShowTooltips property (VBS), 5577
ToolbarUseBackColor property (VBS), 5577
ToolbarUseHotKeys property (VBS), 5578
ToolbarVisible property (VBS), 5578
Tools, 3932, 3935
ToolTipText property (VBS), 5579
Top property (VBS), 5580
TopOffset property (VBS), 5582
Topology discovery, 1137, 6103, 6188, 6269, 6406
Topology editor, 598
Topology view
  Adding device, 566
  Adopt devices identified online, 677
  Adopt port interconnections identified online, 676
  Compare offline/online, 667
  Configured topology, 663, 664
  Diagnostics status in the graphic view, 665
  Diagnostics status in the table view, 666
  Differences compared with the network view, 662
  Functions, 661
  Hardware and network editor, 542, 6011
  Interconnecting ports, 669
Total property (VBS), 5583
TP, 2218, 2227, 2494, 2503, 2755
TP 177B mono
  Symbol library, 4173
TP177A
  Loading a project, 80
Trace, 7619, 7620
  Bit track, 7627
  CPU load, 7648
  Creating a trace configuration, 7636
  Curve diagram, 7627, 7642
  Data storage, 7622
  Displaying a diagram, 7637
  Displaying a trace configuration, 7637
  Installed trace, 7621
  Lifetime of the values, 7647
  Measurement, 7621, 7622, 7640
  Measurement cursor, 7642
  Pre-trigger, 7653
  Printing, 7646
  Project navigator, 7624
  Quantity structure, 7648
  Quick start, 7631
  Recordable tags, 7646
  Recording, 7621, 7638, 7639
  Recording conditions, 7650, 7651, 7653
  Recording cycle, 7633, 7657
  Recording duration, 7657
  Recording levels, 7647
  Reduction, 7651
  Sampling, 7633
  Saving the trace configuration, 7642
  Signal group, 7645
  Signal table, 7629, 7644
  Signals, 7649, 7657
  Status, 7625
  Supported devices, 7619

Trace configuration, 7621, 7622, 7638, 7641, 7642, 7656
Trigger, 7658
Trigger mode, 7652
Trigger tag, 7650, 7652
User interface, 7623, 7631, 7648
Trace function, 7620
Trace methods, 5932
Trace S7-1200/1500, 7646
TraceUserChange, 4800
Transfer
License key to HMI device, 6997
Transfer card, 1169
Transfer card; see Transfer card, 1169
Transfer to the HMI device
WinAC MP, 6763
Transferring authorization
HMI device, 6766
Transferring data
Area pointer, 6039, 6119, 6206, 6299, 6354, 6420, 6476, 6728
Area pointer screen number, 6120, 6212, 6299, 6355, 6426, 6477, 6730
Control panel, 6753, 6754
Coordination area pointer, 6124, 6210, 6302, 6358, 6424, 6480, 6733
Data record are pointer, 6737
Data record area pointer, 6128, 6216, 6306, 6363, 6430, 6484
Date/time area pointer, 6121, 6207, 6208, 6300, 6356, 6421, 6422, 6478, 6731
Date/time PLC area pointer, 6122, 6301, 6357, 6479, 6732
Job mailbox, 6130, 6220, 6310, 6366, 6434, 6486, 6740
Job mailbox area pointer, 6126, 6214, 6304, 6360, 6428, 6482, 6734
Operator input in the recipe view, 6219, 6309, 6365, 6433, 6485, 6739
Possible cause of error, 6133, 6222, 6312, 6368, 6436, 6488, 6742
Project ID area pointer, 6125, 6213, 6303, 6359, 6427, 6481, 6734
Triggering by means of a configured function, 6132, 6221, 6311, 6368, 6435, 6487, 6742
With synchronization, 6129, 6218, 6308, 6364, 6432, 6490, 6738
Without synchronization, 6128, 6217, 6307, 6363, 6431, 6489, 6737
Transferring the project
HMI device, 87
Recipe data record, 87
Translate
Editor, 6829
Translating texts, 454
Transmission medium/duplex, 1134
Transmission protocol, 4119
Transmission rate / duplex, 6101, 6186, 6267, 6404
Transparency
In graphic, 3972
Transparency property (VBS), 5583
transparent color, 4111
Display on panels, 4111
TransparentColor property (VBS), 5585
TransparentColorDeactivatedPicture property (VBS), 5585
TransparentColorPictureOff property (VBS), 5586
TransparentColorPictureOn property (VBS), 5587
Transponder, 5943
TRCV, 3677, 3681
TRCV_C, 103, 3636, 3641
Trend, 4270
Data type, 6641, 6722
Trend control, 6703
Trend request
Trend transfer, 6134, 6224, 6313, 6370, 6438, 6491, 6508, 6639, 6670, 6702, 6721
Trend request area, 6134, 6224, 6313, 6370, 6438, 6491, 6508, 6639, 6670, 6702, 6721
Trend transfer area, 6134, 6224, 6313, 6370, 6438, 6491, 6508, 6639, 6670, 6702, 6721
Trend transfer area 1
Trend transfer area 2, 6134, 6224, 6313, 6370, 6438, 6491, 6508, 6639, 6670, 6702, 6721
Trend view, 4123, 4270
Button, 4124
Configuring for logging, 4275
Configuring for values from the PLC, 4268, 4273
TrendActualize property (VBS), 5587
TrendAdd property (VBS), 5588
TrendBeginTime property (VBS), 5588
TrendColor property (VBS), 5589
TrendCount property (VBS), 5589
TrendEndTime property (VBS), 5590
TrendExtendedColorSet property (VBS), 5590
TrendFill property (VBS), 5591
TrendFillColor property (VBS), 5591
TrendIndex property (VBS), 5592
TrendIndicatorColor property (VBS), 5593
TrendLabel property (VBS), 5593
TrendLineStyle property (VBS), 5594

TrendLineType property (VBS), 5594
TrendLineWidth property (VBS), 5595
TrendLowerLimit property (VBS), 5595
TrendLowerLimitColor property (VBS), 5596
TrendLowerLimitColoring property (VBS), 5596
TrendMeasurePoints property (VBS), 5597
TrendName property (VBS), 5597
TrendPointColor property (VBS), 5598
TrendPointStyle property (VBS), 5598
TrendPointWidth property (VBS), 5599
TrendProvider property (VBS), 5599
TrendRangeType property (VBS), 5600
TrendRemove property (VBS), 5601
TrendRename property (VBS), 5601
TrendRepos property (VBS), 5602
TrendRulerControl, 5187
TrendSelectTagNameX property (VBS), 5602
TrendSelectTagNameY property (VBS), 5603
TrendTag property (VBS), 5725
TrendTagNameX property (VBS), 5603
TrendTagNameY property (VBS), 5604
TrendTimeRangeBase property (VBS), 5604
TrendTimeRangeFactor property (VBS), 5605
TrendTrendWindow property (VBS), 5605
TrendUncertainColor property (VBS), 5606
TrendUncertainColoring property (VBS), 5606
TrendUpperLimit property (VBS), 5607
TrendUpperLimitColor property (VBS), 5607
TrendUpperLimitColoring property (VBS), 5608
TrendView object, 5195
TrendViewBackToBeginning, 4708
TrendViewCompress, 4705
TrendViewExtend, 4705
TrendViewRulerLeft, 4707
TrendViewRulerRight, 4706
TrendViewScrollBack, 4704
TrendViewScrollForward, 4704
TrendViewSetRulerMode, 4707
TrendViewStartStop, 4708
TrendVisible property (VBS), 5608
TrendWindowAdd property (VBS), 5609
TrendWindowCoarseGrid property (VBS), 5609
TrendWindowCoarseGridColor property (VBS), 5610
TrendWindowFineGrid property (VBS), 5611
TrendWindowFineGridColor property (VBS), 5611
TrendWindowForegroundTrendGrid property (VBS), 5612
TrendWindowGridInTrendColor property (VBS), 5612
TrendWindowHorizontalGrid property (VBS), 5613
TrendWindowIndex property (VBS), 5613
TrendWindowName property (VBS), 5614
TrendWindowRemove property (VBS), 5614
TrendWindowRename property (VBS), 5614
TrendWindowRepos property (VBS), 5615
TrendWindowRulerColor property (VBS), 5615
TrendWindowRulerLayer property (VBS), 5616
TrendWindowRulerStyle property (VBS), 5617
TrendWindowRulerWidth property (VBS), 5617
TrendWindowSpacePortion property (VBS), 5618
TrendWindowVerticalGrid property (VBS), 5618
TrendWindowVisible property (VBS), 5619
TrendXAxis property (VBS, 5619
TrendYAxis property (VBS, 5620
Trigger tag, 4294
Triggering by means of a configured function
  Transferring data, 6132, 6221, 6311, 6368, 6435, 6487, 6742
Triggers, 5981
  Acyclic, 5982, 5983
  Changing, 5987
  Cyclic, 5982, 5991
  Event trigger, 5981, 5982, 5985, 5989
  Standard cycle, 5982
TRUE, 2216, 2217, 2492, 2493, 2752, 2753
TRUNC, 2389, 2668, 2884
TS adapter, 86
TS Adapter IE Advanced
  Brief description, 7704
  Connection types, 7705
  Parameter assignment, 7707
  Parameter assignment options, 7706
TS Adapter MPI
  Brief description, 7681
  Default parameter assignment, 7682
  Direct connection, 7682
  Establishing a modem connection, 7683
  Establishing the direct connection, 7682
  Exporting adapter parameters, 7687
  Importing adapter parameters, 7688
  Modem connection, 7683
  parameter Assignment, 7685
  Parameter assignment, 7685
  Parameter assignment options, 7684
  Principle of operation, 7681
  Restoring the default parameter assignments, 7686
  Setting up access protection, 7678
TS Adapter IE
  Brief description, 7688
  Default parameter assignment, 7690
  Parameter assignment, 7693, 7707
  Principle of operation, 7689

TS Adapter IE Basic
Connection to the GSM network, 7691
Connection to the telephone network, 7691
Connection to the telephone network through an external modem, 7692
Connection types, 7690
TSAP
ASCII code table, 650
Structure, 614, 648
TSAP assignment
Examples, 650
TSEND, 3671, 3674
TSEND_C, 3625, 3629
TubeArcObject, 5198
TubeDoubleTeeObject, 5200
TubePolyline, 5202
TubeTeeObject, 5205
Tuning, 7436
TURCV, 3690
TUSEND, 3687
Two-hand operation
Multi-touch devices, 6969
Twos complement, 2309, 2585
Type
Changeable properties of a type, 505
Changeable properties of a version, 505
Creating an instance, 522
Displaying the properties of a type, 505
Displaying the properties of a version, 505
Library, 6773
Type property (VBS), 5807
typed, 295
TypeOf, 2798
TypeOfElements, 2799
Types
Copying instances, 510
Creating an instance, 510
Pasting instances, 510
Using types, 510
Types of DP slave, 1094

U

UART data transmission, 1197
UBLKMOV, 2378, 2656, 2873
UDINT, 1921, 1977, 2043, 2103, 2138
UDINT_TO_, 2043, 2138
UDP, 706, 712, 736
Characteristics, 639
Port numbers, 645
UDT, 98, 226, 254, 286
UFILL_BLK, 2353, 2632, 2835
UINT, 1919, 1973, 2036, 2101, 2132
UINT_TO_, 2036, 2132
ULINT, 1923, 1981, 2049
ULINT_TO_, 2049
UMOVE_BLK, 2350, 2627, 2831
UncertainStateColor(i) property (VBS), 5727
UncertainStateEnabled(i) property (VBS), 5728
Underflow, 1266, 1289
Undoing actions
Basics of undoing, 446
Undoing last action, 448
Undoing multiple actions, 448
UnhideAlarm, 5933
Uninstalling
Option, 6999
Uninstalling license keys, 116
Uniqueness
of object names, 162
Unit property (VBS), 5620
UnitColor property (VBS), 5621
UnitText property (VBS), 5621
UnitTop property (VBS), 5622
Universal
PROFIBUS, 6204, 6284
Universal Serial Interface Protocol, (See USS protocol)
Universal symbolism, 224
Unknown peers, 749
Unlock
Operator control, 6970
User, 4553
UnlockAlarm, 5933
Unplugged module, 563
Unscale, 2397, 2677, 2896
UNSCALE, 2397, 2677, 2896
Unspecified CPU, 567
UNTIL, 2906
UPDAT_PI, 3062
UPDAT_PO, 3064
Update, 1218
Operating system of the HMI device (Windows CE), 6995
Tag, 4222, 4235
Update cycle, 4247
Update interrupt, 1218
Update interrupt OB, 1218
Update time, 1131
UpdateEnable property (VBS), 5729
UpdateTag, 4660
Updating
Device proxy, 7750, 7755

Faceplate type, 4075
IPE data, 7750, 7755
Updating a library, 528, 6783
Updating the device version, 6994
Updating the firmware, 6994
Updating the operating system, 6994
Updating types to the latest version, 528, 6783
Upgrade
Global library, 6878
Project, 6877
Project version, 6877
Upgrading a global library, 488
Upload, 1148
Uploading
from a device, 399
UpperLimit property (VBS), 5622
UpperLimitColor(i) property (VBS), 5729
UpperLimitEnabled(i) property (VBS), 5730
UpperLimitValue(i) property (VBS), 5730
URCV, 3619
USB
Download, 6946
Loading, 6947, 6948
USB card readers, 46
USB driver, 6947, 6948
Installation under Windows 7, 6948
Installation under Windows XP, 6947
USB port, 6947, 6948
Use, 4425
Faceplate type, 4077
Of recipes, 4425
Project-wide alarm class, 4298
Screen navigation function keys, 4049
Slide-in screen, 3915
Style sheet, 3931
Use global assignment
Function key, 3902
UseAllServers property (VBS), 5731
UseBarBorderConstraints property (VBS), 5808
Used property (VBS), 5623
UseDesignColorSchema property (VBS), 5623
UseDesignShadowSettings property (VBS), 5625
UsedPercent property (VBS), 5626
UseEffectiveProcessValue property (VBS), 5808
UseEyponentialFormat property (VBS), 5627
UseFlashTransparentColor property (VBS), 5627
Useful information on configuring the TS Adapter MPI, 7684
UseGDI property (VBS), 5809
UseMeasurePoints(i) property (VBS), 5809
UseMessageColor property (VBS), 5628
UseMultipleLimits property (VBS), 5810
USEND, 3618
User
Assigning roles, 694
Changing, 4552
Creating roles, 693
Deleting in runtime, 4552
Logon, 4555
Setting up, 691
Unlock, 4553
User administration, 4527
Central user administration, 4543
exporting, 4553
importing, 4554
Migration, 188
Object with access protection, 4558
Runtime settings, 3899, 4533, 4534, 4547, 4553, 4555
Setting up, 4561
SIMATIC Logon, 4543
User change, 4848, 5988
User data
Area, 1177
Backup, 4530
Restoring, 4530
User data type, 4240
Create, 4240, 4241
Creating a user data type element, 4241
Deleting, 4245
Editing, 4244
Release, 4244
Tags, 4240
User data type element
Create, 4241
User data types
Interconnecting with faceplates, 4065
User group
Administer authorizations, 4542
Assigning, 4567
Assigning users, 4540
Change displayed name, 4542
Changing in runtime, 4553
Changing the name, 4542
Creating, 4565
Deleting, 4543
Managing, 4542
Unauthorized, 4556
User interface
"Reference projects" palette, 328
Details view, 330
Inspector window, 324
Maximizing the work area, 318
Minimizing the work area, 318

Overview window, 331
Portal view, 307
Project tree, 312
Project view, 309
Task card, 326
Views, 307
Work area, 316
User interface language, 6823
Selecting, 6826
User libraries, 471
User name, 692
User pages, (See user-defined web pages)
User program
Finding errors, 1697
Function, 1411
Testing, 1839
User texts, 454
User view, 4548
Application, 4102
Column width, 4103
Columns moveable, 4101
Complex user view, 4100, 4548
Configuring, 4549
Layout, 4102
Moving columns, 4103
Number of lines, 4101, 4102
Simple user view, 4101, 4135
UserArchiveControl object, 5207
UserArchiveNumberOfValues(i) property (VBS), 5812
UserArchiveStartId(i) property (VBS), 5813
User-defined documentation, 361
Enabling the call log, 364, 365
Specifying settings, 364, 365
Specifying the central file directory, 364, 365
User-defined function, 4569
Adapting display properties, 4580
Applications, 4572
Autocompletion, 4578
Changing object property, 4604
Code templates, 4579
configuring with VBS, 4585
Entering parameters, 4578
Executing in Runtime, 4601
In a function list, 4572
In user-defined functions, 4572
Opening know-how protected functions, 4590
Parameter transfer, 4584
Properties, 4572
Return value in VBS, 4585
Sequence is not adhered to, 4602
Synchronizing object, 4579
Synchronizing tag, 4579
Syntax highlighting, 4580
Temperature conversion, 4604
Testing, 4588
Usage, 4572
Using a system function, 4601
User-defined roles, 693
User-defined text lists
Creating, 465
Editing, 466
Editing value ranges and texts, 466
User-defined web pages, 859, 862, 871, 872, 874, 875
Users
Assigning a user group, 4540
Changing the name, 4541
Create, 4539
Creating, 4567
Creating in runtime, 4550
Deleting, 4542
Logging logons, 4560
Managing, 4541, 4551
Updating after change of user, 5988
UserView object, 5214
UseScaleConstraints property (VBS), 5810
UseScaledBarBorder property (VBS), 5811
UseSelectedTitleColor property (VBS), 5628
UseSeparateDiagrams property (VBS), 5811
UseSimplePresicionOffset property (VBS), 5812
UseTableColor2 property (VBS), 5629
UseTagLimitColors property (VBS), 5629
UseTimeRange(i) property (VBS), 5757
UseTransparentColor property (VBS), 5630
UseTransparentColorDeactivatedPicture property (VBS), 5630
UseTransparentColorPictureOff property (VBS), 5631
UseTransparentColorPictureOn property (VBS), 5631
UseTrendNameAsLabel property (VBS), 5632
Using a TS Adapter for TeleService, 7672, 7696
Using frames and cover pages from the library, 426
Using tags, 4104
Using the keyboard
Editing texts, 1533
FBD, 1533
GRAPH, 1533
LAD, 1533
Program editor, 1533
SCL, 1533
STL, 1533
Using the on-screen keyboard, 349

Using UDTs, 286
USINT, 1918, 1970, 2029, 2099, 2126
USINT_TO_, 2029, 2126
USS communication
Controlling a transmission to a drive with USS_PORT, 3746
Data exchange with drives via USS_DRIVE, 3747
Modifying parameters in the drive with USS_WPM, 3751
Reading parameters from the drive with USS_RPM, 3749
USS protocol
Instructions, 3742
USS status codes, 3752
USS_DRIVE, 3747
USS_PORT, 3746
USS_RPM, 3749
USS_WPM, 3751
UV_ColumnWidth_AKZ property (VBS), 5632
UV_ColumnWidth_Descriptor property (VBS), 5632
UV_ColumnWidth_InstallationDate property (VBS), 5633
UV_ColumnWidth_LADDR property (VBS), 5633
UV_ColumnWidth_Name property (VBS), 5633
UV_ColumnWidth_OKZ property (VBS), 5633
UV_ColumnWidth_OperationState property (VBS), 5633
UV_ColumnWidth_OrderID property (VBS), 5633
UV_ColumnWidth_ProfileID property (VBS), 5633
UV_ColumnWidth_Rack property (VBS), 5634
UV_ColumnWidth_Slot property (VBS), 5634
UV_ColumnWidth_SoftwareRevision property (VBS), 5634
UV_ColumnWidth_SpecificProfileData property (VBS), 5634
UV_ColumnWidth_State property (VBS), 5634
UV_ColumnWidth_Station property (VBS), 5634
UV_ColumnWidth_SubAddress property (VBS), 5634
UV_ColumnWidth_SubSlot property (VBS), 5635
UV_ColumnWidth_SubSystem property (VBS), 5635
UV_ColumnWidth_Type property (VBS), 5635
UV_ShowItem_AKZ property (VBS), 5635
UV_ShowItem_Descriptor property (VBS), 5635
UV_ShowItem_InstallationDate property (VBS), 5635
UV_ShowItem_LADDR property (VBS), 5635
UV_ShowItem_Name property (VBS), 5636
UV_ShowItem_OKZ property (VBS), 5636
UV_ShowItem_OperationState property (VBS), 5636
UV_ShowItem_OrderID property (VBS), 5636
UV_ShowItem_ProfileID property (VBS), 5636
UV_ShowItem_Rack property (VBS), 5636
UV_ShowItem_Slot property (VBS), 5636
UV_ShowItem_SoftwareRevision property (VBS), 5637
UV_ShowItem_SpecificProfileData property (VBS), 5637
UV_ShowItem_State property (VBS), 5637
UV_ShowItem_Station property (VBS), 5637
UV_ShowItem_SubAddress property (VBS), 5637
UV_ShowItem_SubSlot property (VBS), 5637
UV_ShowItem_SubSystem property (VBS), 5637
UV_ShowItem_Type property (VBS), 5638

V

VAL_STRG, 3011
Valid
Data type, 6608, 6654, 6665, 6684, 6697, 6716
Data types, 6145, 6235, 6324, 6380, 6449, 6500, 6574, 6575
Valid data type
Allen-Bradley DF1, 6632
SIMATIC HMI HTTP protocol, 6570
ValidateFormatPattern property (VBS), 5758
Value, 2312, 2588
Value assignment, 1666
Value change, 4856
Value changes to GMP-relevant tags
Effects in Runtime, 7069
Value/Range
Text list, 3994
ValueAxisAutorange(i) property (VBS), 5639
ValueAxisBegin(i) property (VBS), 5758
ValueAxisDecimalPrecision(i) property (VBS), 5732
ValueAxisEnd(i) property (VBS), 5759
ValueAxisGridLineInterval(i) property (VBS), 5814
ValueAxisLabel(i) property (VBS), 5639
ValueAxisLargeIncrementSize(i) property (VBS), 5815
ValueAxisScalingType(i) property (VBS), 5640
ValueAxisShowGridLines(i) property (VBS), 5733
ValueAxisShowLargeIncrements(i) property (VBS), 5815
ValueAxisShowSmallIncrements(i) property (VBS), 5816
ValueColumnAlignment(i) property (VBS), 5640
ValueColumnPrecision(i) property (VBS), 5734
Values
Comparing, 7204
Variable index, 251, 254

VARIANT, 260, 263, 264, 267, 270, 1951, 2292, 2294, 2295, 2296, 2367, 2368, 2369, 2568, 2569, 2570, 2572, 2645, 2646, 2648, 2798, 2799, 2866, 2867, 2868, 2890, 2892
VARIANT_TO_DB_ANY, 2890
VariantGet, 2367, 2645, 2866
VariantPut, 2368, 2646, 2867
VBS
  Object model, 4971
  Reference, 4971
VB-script
  Migrating, 184
Vendor ID, 942
Version overview, 1235
Versioning of types
  Released version, 503
  Version in progress, 503
  Version in test, 503
Versions in WinCC, 6874
Vertical movement
  Animation, 4022
Vertical radius, 4107
VerticalAlignment property (VBS), 5641
VerticalGridLines property (VBS), 5642
VerticalScrollBarPosition property (VBS), 5642
View options
  For the assignment list, 1814
  Setting the call structure, 1822
  Setting the dependency structure, 1828
ViewOnly property (VBS), 5643
Views of the cross-reference list, 1835, 6818
Virtual Machine (VM)
  Supported virtualization platforms, 119, 123
Virus scanners, 124
Visible column, 4171
Visible property (VBS), 5643
VLAN, 925
  Day, 1016
  Port VID, 1016
  Priority, 1016
  VLAN ID, 928
  VLAN tag, 926
VLAN operation, 745
VLAN tagging, 745
VOID, 1953
VPN connection
  Deleting CA certificates, 7702
  Establishing, 7702
  Installing CA certificate, 7699
  Terminating, 7704
VPN connection is not established
  Check and remedy, 7723
VPN group, 743
VRRP
  Address overview, 1062
  Configuration, 1061
  Router, 1060
  VRRP Statistics, 948

W

WAIT, 2421, 2702, 2926
Wake on LAN, 3165
WAN IP address
  Specifying, 755
Warm restart, 1166
Warning property (VBS), 5645
WarningColor property (VBS), 5646
WarningLowerLimit property (VBS), 5646
WarningLowerLimitColor property (VBS), 5647
WarningLowerLimitEnabled property (VBS), 5648
WarningLowerLimitRelative property (VBS), 5648
WarningRangeColor property (VBS), 5649
WarningRangeStart property (VBS), 5649
WarningRangeVisible property (VBS), 5650
WarningUpperLimit property (VBS), 5651
WarningUpperLimitColor property (VBS), 5651
WarningUpperLimitEnabled property (VBS), 5652
WarningUpperLimitRelative property (VBS), 5652
Watch table
  Basic mode, 1855
  Copying, 1859
  Creating, 1858
  Example of filling out a watch table, 1860
  Expanded mode, 1855
  Layout, 1855
  Loading data blocks during an active control job, 106
  Meaning of the columns, 1855
  Meaning of the icons, 1857
  Monitoring and modifying modes, 1869
  Multiple access to the same CPU, 106
  Opening, 1859
  Overview of the display formats, 1864
  Overview of the test options, 1854
  Permitted operands, 1861
  Permitted operands for modify values, 1862
  Possible applications, 1854
  Saving, 1860
  Switching between basic mode and expanded mode, 1856
  Syntax check, 1860
  Testing wiring, 1854
Watchdog time, 1132

WCHAR, 1937, 2005, 2084
WCHAR_TO_, 2084
Web address, 4116
Web application, 859
Web authorization, 7100
  Setting at the HMI-device, 7091
Web Control DB, 876
Web data blocks, 874
Web pages in browser, 881
Web server, 847, 853, 855, 859, 862, 874, 7129, (See Web server), (See Web server)
  Configure WinCC-Project, 7132
  Enable, 856
  Enumerations, 871
  Fragment, 872
  HTTPS, 857
  Rules PLC tag names, 863
  Service-pages, 7130
  User administration, 7099
  User-defined web pages, 859
  Web authorization, 7099, 7100
When dialog is closed, 4848
When dialog is opened, 4848
WHILE, 2905
Width property (VBS), 5653
WinAC MP, 6743
  HMI device, 6762
  Transfer, 6761
  Transfer to the HMI device, 6763
  Transferring, 6762
WinACMPArchive, 4826
WinACMPClearCycleTimeBuffer, 4828
WinACMPControl, 4833
WinACMPGetStartMode, 4827
WinACMPGetVersion, 4827, 4948
WinACMPRestore, 4834
WinACMPSetHMIEnableTime, 4830
WinACMPSetKeySwitch, 4829
WinACMPSetSleeptime, 4831
WinACMPSetStart at boot, 4828, 4949
WinACMPSetStartMode, 4831, 4949
WinACMPStartHistogram, 4832
WinACMPStopHistogram, 4833
WinACMPUpdateAverageCycleTime, 4815
WinACMPUpdateAverageExecTime, 4814
WinACMPUpdateBUSF1LED, 4812
WinACMPUpdateBUSF2LED, 4813
WinACMPUpdateEXTFLED, 4816
WinACMPUpdateHMIEnableTime, 4817
WinACMPUpdateINTFLED, 4818
WinACMPUpdateKeySwitchSetting, 4811
WinACMPUpdateLastCycleTime, 4819
WinACMPUpdateMaximumCycleTime, 4820
WinACMPUpdateMinimumCycleTime, 4821
WinACMPUpdatePowerLED, 4822
WinACMPUpdateRUNLED, 4824
WinACMPUpdateStartupCharacteristics, 4810
WinACMPUpdateSTOPLED, 4825
WinACSetStartMode, 4831, 4949
WinCC
  WinCC MediaControl, 5107
WinCC flexible
  Configuring, 6749
WinCC flexible project
  Migrate, 165, 194
WinCC OPC Client
  Configuring, 7162
WinCC RT Advanced, 6168, 6254
  HMI connection, 6109, 6194, 6197, 6275, 6289, 6291, 6412, 6413
  Interface, 6076
  MPI, 6287
  PC, 6085, 6343, 6394
  PROFIBUS, 6107, 6192, 6273, 6409
  SIMATIC PC, 6088, 6111, 6346, 6397
WinCC RT Professional
  HMI connection, 6109, 6111
  Interface, 6076
  PC, 6085, 6109, 6394
  PROFIBUS, 6107
  SIMATIC PC, 6088, 6111, 6397
WinCC Runtime
  Available system functions, 4651
WinCC Runtime Advanced
  Display and operating element, 4096
  Interface, 6076
WinCC Runtime Advanced Internet, 7088
WinCC Runtime Professional
  Interface, 6076
WinCC V7.0 SP3, 58
WinCC version
  Compatibility, 6874
WinCC flexible, 6752
WinCC RT Advanced, 6171, 6257
  HMI connection, 6111
  PC, 6109
Window layouts
  Changing order, 337
  Deleting window layouts, 337
WindowCloseEnabled property (VBS), 5655
WindowMaximizeEnabled property (VBS), 5655
WindowMovingEnabled property (VBS), 5656
WindowOnTop property (VBS), 5656
Windows, 118

WindowSizingEnabled property (VBS), 5657
WindowSlider object, 5217
WindowsStyle property (VBS), 5657
Wire break, 1265, 1330
Wireless LAN/PB link, 1111
Wiring tests, 1854
Wizard
Device wizard, 6766, 6768
WLAN area, 5942
WLAN reception, 4189
Layout, 4189
WLanQualityView object, 5219
WORD, 1914, 1964, 2015, 2096, 2116, 2162, 2174
WORD_TO_, 2015, 2116, 2174
WORD_TO_BLOCK_DB, 2193
Work area
Effective range, 5960
Effective range (RFID), 5962
Embedding floating elements, 320
Floating elements, 320
Function, 316
Maximizing, 318
Maximizing elements, 323
Minimizing, 318
Minimizing elements, 322
Mobile Wireless, 5960
Mobile Wireless (RFID), 5962
Saving a layout of editors and tables, 337
Splitting, 319
Switching between elements, 323
Using grouped elements, 321
Zone, 5958, 5959
Work memory, 1172, 1830
Working step
to create screens, 3895
WR_DPARM, 3214
WR_LOC_T, 2986
WR_REC, 3123
WR_SYS_T, 2981
WRIT_DBL, 106, 3353
Write field, 2373, 2651
Write memory address, 2852
Write memory area, 2855
Write memory bit, 2853
WRITE_BIG, 2863
WRITE_LITTLE, 2859
WriteTag, 5938
WriteToArrayDB, 2359, 2637, 2840
WriteToArrayDBL, 2364, 2642, 2845
Writing
A data record with WR_REC, 3123
Writing data
in remote CPU with PUT, 3615
To DP standard slaves/PROFINET IO devices with DPWR_DAT, 3126
Writing tags, 867
WRREC, 3071
WSTRING, 1469, 1939, 2008, 2089
Addressing, 1460
WSTRING_TO_, 2089
WWW (instruction), 875
WWW Synchronizing user-defined web pages, 3721

X

XAxisAdd property (VBS), 5658
XAxisAlignment(VBS), 5658
XAxisAutoPrecisions property (VBS), 5659
XAxisAutoRange(i) property (VBS), 5659
XAxisBegin(i) property (VBS), 5735
XAxisBeginValue property (VBS), 5660
XAxisColor property (VBS), 5660
XAxisCount property (VBS), 5661
XAxisDecimalPrecision(i) property (VBS), 5817
XAxisEnd(i) property (VBS), 5736
XAxisEndValue property (VBS), 5661
XAxisExponentialFormat property (VBS), 5662
XAxisGridLineInterval(i) property (VBS), 5818
XAxisIndex property (VBS), 5662
XAxisInTrendColor property (VBS), 5663
XAxisLabel(i) property (VBS), 5663
XAxisLargeIncrementSize(i) property (VBS), 5818
XAxisMode(i) property (VBS), 5819
XAxisName property (VBS), 5664
XAxisPrecisions property (VBS), 5664
XAxisRemove property (VBS), 5664
XAxisRepos property (VBS), 5665
XAxisScalingType(i) property (VBS), 5665
XAxisShowGridLines(i) property (VBS), 5736
XAxisShowLargeIncrements(i) property (VBS), 5819
XAxisShowSmallIncrements(i) property (VBS), 5820
XAxisSmallIncrementSize(i) property (VBS), 5820
XAxisTrendWindow property (VBS), 5666
XAxisVisible property (VBS), 5666
XDataLogTag(i) property (VBS), 5821
xlsx file, 6799, 6806, 6808, 6812, 6813
Importing tags, 6806, 6808
Preparing for the import, 6806, 6808
XOnlineTag(i) property (VBS), 5821
XOR, 1665, 2427, 2474, 2475, 2707

Y

YAxisAdd property (VBS), 5667
YAxisAlignment property (VBS), 5667
YAxisAutoPrecisions property (VBS), 5668
YAxisAutoRange property (VBS), 5668
YAxisBeginValue property (VBS), 5669
YAxisColor property (VBS), 5669
YAxisCount property (VBS), 5670
YAxisEndValue property (VBS), 5670
YAxisExponentialFormat property (VBS), 5670
YAxisIndex property (VBS), 5671
YAxisInTrendColor property (VBS), 5671
YAxisLabel property (VBS), 5672
YAxisName property (VBS), 5672
YAxisPrecisions property (VBS), 5673
YAxisRemove property (VBS), 5673
YAxisRename property (VBS), 5674
YAxisRepos property (VBS), 5674
YAxisScalingType property (VBS), 5675
YAxisTrendWindow property (VBS), 5675
YAxisVisible property (VBS), 5676
YDataLogTag(i) property (VBS), 5822
YOnlineTag(i) property (VBS), 5823

Z

ZeroPoint property (VBS), 5676
Zone, 5943, 5957
  Calculating quality, 4193
  Configuring a screen change, 5965, 5968
  Displaying an object in relation to the zone, 5966
  ID Zone, 5966
  On entry, 4192
  Work area, 5958, 5959
  Zone ID, 4192
  Zone ID / connection point ID, 5976
Zone name, 4192
Zone signal, 4193
ZoneLabelView object, 5220
ZoneQualityView object, 5221
Zones editor, 5957
Zoom
  Adjusting the zoom setting, 538, 540, 543, 6003, 6010, 6012
  Keyboard operation, 553
Zoom in/out, (See Zoom)
Zoomable property (VBS), 5760
ZoomArea, 5938
ZoomFactor property (VBS), 5677
Zooming
  Multi-touch devices, 6967
  Screen, 3895
ZoomInOut, 5939
ZoomInOutTime, 5939
ZoomInOutValues, 5940
ZoomInOutX, 5940
ZoomInOutY, 5941
ZoomMove, 5941